BMC Remedy Action Request System 7.6.04

Workflow Objects Guide

January 2011

Contacting BMC Software

You can access the BMC Software website at http://www.bmc.com. From this website, you can obtain information about the company, its products, corporate offices, special events, and career opportunities.

United States and Canada

Address	BMC SOFTWARE INC 2101 CITYWEST BLVD HOUSTON TX 77042-2827 USA	**Telephone**	713 918 8800 or 800 841 2031	**Fax**	713 918 8000

Outside United States and Canada

Telephone	(01) 713 918 8800	**Fax**	(01) 713 918 8000

If you have comments or suggestions about this documentation, contact Information Design and Development by email at doc_feedback@bmc.com.

Restricted rights legend

Customer Support

You can obtain technical support by using the Support page on the BMC Software website or by contacting Customer Support by telephone or email. To expedite your inquiry, please see “Before Contacting BMC Software.”

Support website

You can obtain technical support from BMC Software 24 hours a day, 7 days a week at `http://www.bmc.com/support`. From this website, you can:

- Read overviews about support services and programs that BMC Software offers.
- Find the most current information about BMC Software products.
- Search a database for problems similar to yours and possible solutions.
- Order or download product documentation.
- Report a problem or ask a question.
- Subscribe to receive email notices when new product versions are released.
- Find worldwide BMC Software support center locations and contact information, including email addresses, fax numbers, and telephone numbers.

Support by telephone or email

In the United States and Canada, if you need technical support and do not have access to the Web, call 800 537 1813 or send an email message to `customer_support@bmc.com`. (In the Subject line, enter `SupID:`*`<yourSupportContractID>`*, such as `SupID:12345`.) Outside the United States and Canada, contact your local support center for assistance.

Before contacting BMC Software

Have the following information available so that Customer Support can begin working on your issue immediately:

- Product information
  - Product name
  - Product version (release number)
  - License number and password (trial or permanent)
- Operating system and environment information
  - Machine type
  - Operating system type, version, and service pack
  - System hardware configuration
  - Serial numbers
  - Related software (database, application, and communication) including type, version, and service pack or maintenance level
- Sequence of events leading to the problem
- Commands and options that you used
- Messages received (and the time and date that you received them)
  - Product error messages
  - Messages from the operating system, such as `file system full`
  - Messages from related software

License key and password information

If you have a question about your license key or password, contact Customer Support through one of the following methods:

- E-mail `customer_support@bmc.com`. (In the Subject line, enter `SupID:<yourSupportContractID>`, such as `SupID:12345`.)
- In the United States and Canada, call 800 537 1813. Outside the United States and Canada, contact your local support center for assistance.
- Submit a new issue at `http://www.bmc.com/support`.

Contents

Preface

IMPORTANT

The compatibility information listed in the product documentation is subject to change. See the compatibility matrix at http://www.bmc.com/support for the latest, most complete information about what is officially supported.

Carefully read the system requirements for your particular operating system and database, especially the necessary patch requirements.

Audience

This manual contains reference information and procedures for creating, modifying, and maintaining AR System workflow objects, including active links, filters, and escalations. It is written for developers and administrators who create, customize, and maintain applications based on BMC Remedy Action Request System (AR System).

To follow the procedures in this guide, you will need to be able to log in to AR System with BMC Remedy Developer Studio as an administrator or subadministrator, and you must also be able to use BMC Remedy User or the web client. You should be familiar with BMC Remedy Developer Studio basics before you begin. For an introduction to BMC Remedy Developer Studio, see the *Introduction to Application Development with BMC Remedy Developer Studio* guide.

AR System documents

The following table lists documentation available for AR System 7.6.04.

Unless otherwise noted, online documentation in Adobe Acrobat (PDF) format is available on AR System product installation DVDs, on the Customer Support website (http://www.bmc.com/support), or both.

You can access product help through each product's Help menu or by clicking Help links.

NOTE

The AR System product help has not been updated for version 7.6.04. The help topics still apply to version 7.6.03. For the most recent content, refer to the PDF documentation.

Title	Description	Audience
Concepts Guide[1]	Overview of AR System architecture and features; includes information about add-on products that extend AR System functionality and a comprehensive glossary for the entire AR System documentation set.	Everyone
Installation Guide	Instructions for installing AR System.	Administrators
Introduction to Application Development with BMC Remedy Developer Studio	Information about the development of AR System applications, including an introduction to using BMC Remedy Developer Studio.	Developers[2]
Form and Application Objects Guide	Information about AR System applications and their user interface components, including forms, fields, views, menus, and images.	Developers
Workflow Objects Guide	Information about the AR System workflow objects (active links, filters, and escalations) and how to use them to create processes that enforce business rules.	Developers
Configuration Guide	Information about configuring AR System servers and clients, localizing, importing and exporting data, and archiving data.	Administrators
BMC Remedy Mid Tier Guide	Information about configuring the mid tier, setting up applications for the mid tier, and using applications in browsers.	Administrators
Integration Guide	Instructions for integrating AR System with external systems by using web services, plug-ins, and other products, including LDAP, OLE, and ARDBC.	Administrators/ Developers/ Programmers[3]
Optimizing and Troubleshooting Guide	Information about monitoring and maintaining AR System and AR System applications to optimize performance and solve problems.	Administrators/ Developers/ Programmers
Database Reference	Database administration topics and rules related to how AR System interacts with specific databases; includes an overview of the data dictionary tables.	Administrators/ Developers/ Programmers
BMC Remedy Distributed Server Option Guide	Information about implementing a distributed AR System server environment with BMC Remedy Distributed Server Option (DSO).	Administrators
BMC Remedy Flashboards Guide	Instructions for creating, modifying, and administering flashboards to display and monitor AR System information.	Administrators/ Developers
C API Reference	Information about AR System data structures, C API function calls, and OLE support.	Programmers
C API Quick Reference	Quick reference to C API function calls.	Programmers

Title	Description	Audience
Java API	Information about Oracle Java classes, methods, and variables that integrate with AR System. For the location of the JAR file containing this online documentation, see the information about the Java API in the *Integration Guide.*	Programmers
Java Plug-in API	Information about Java classes, methods, and variables used to write plug-ins for AR System. For the location of the JAR file containing this online documentation, see the information about plug-ins in the *Integration Guide.*	Programmers
BMC Remedy Email Engine Guide	Instructions for configuring and using BMC Remedy Email Engine.	Administrators
Error Messages Guide	Descriptions of AR System error messages.	Administrators / Developers / Programmers
Master Index	Combined index of all books.	Everyone
BMC Remedy Approval Server Guide	Instructions for using BMC Remedy Approval Server to automate approval and signature processes in your organization.	Administrators
Release Notes	Information about new features, compatibility, and international issues.	Everyone
Release Notes with Known Issues	Information about new features, compatibility, international issues, installation planning, and open issues.	Everyone
BMC Remedy User Help	Instructions for using BMC Remedy User.	Everyone
BMC Remedy Developer Studio Help	Instructions for using BMC Remedy Developer Studio to develop AR System forms, workflow objects, and applications.	Developers
BMC Remedy Data Import Help	Instructions for using BMC Remedy Data Import.	Administrators
BMC Remedy Alert Help	Instructions for using BMC Remedy Alert.	Everyone
BMC Remedy Mid Tier Configuration Tool Help	Instructions for configuring BMC Remedy Mid Tier.	Administrators
BMC Remedy Browser Help	Instructions for using AR System forms in browsers.	Everyone
BMC Remedy Migrator 7.6.04 BMC Remedy Migrator Guide	Outlines procedures for installing BMC Remedy Migrator, setting options, and performing migration tasks.	Administrators / Developers
BMC Remedy Migrator online help	Procedures for setting BMC Remedy Migrator options and performing migration tasks.	Administrators / Developers
BMC Remedy Encryption Security 7.6.04 BMC Remedy Encryption Security Guide	Provides an overview of the BMC Remedy Encryption Security products and explains how to install and configure them.	Administrators

[1] The full title of each guide includes *BMC Remedy Action Request System 7.6.04* (for example, *BMC Remedy Action Request System 7.6.04 Concepts Guide*), except

the *BMC Remedy Migrator Guide* and *BMC Remedy Encryption Security Guide*.
[2] Application developers who use BMC Remedy Developer Studio.
[3] C and Java programmers who write plug-ins and clients for AR System.

Terminology note

In this guide, the term *button* refers to a button that you place in a form to execute an active link; the term *toolbar button* refers to a button in a toolbar, which, in the case of active links, is the toolbar in BMC Remedy User. A *menu bar item* refers to the items displayed from a top-level menu in BMC Remedy User. Do not confuse it with a menu attached to a field.

Chapter

1 Introducing workflow

This section introduces the three AR System workflow objects: active links, filters, and escalations. By using workflow, you can define a set of processes to enforce business rules such as approval and service level requirements, tracking defects or assets, and so on.

For background information about workflow, see the *Concepts Guide*, "Workflow," page 37.

The following topics are provided:

Workflow basics

AR System workflow consists of active links, filters, and escalations that carry out business processes. For information about what these three main types of workflow, see "Workflow objects" on page 16.

All workflow objects include the following elements:

- **An associated form** is the basis for every workflow action. Sometimes a workflow object has more than one associated form, but one form is defined as the primary form and acts as the reference for fields and data used by the workflow. See "Configuring workflow forms and execution options" on page 35 and "Shared workflow" on page 32.
- **Workflow execution options** determine *when* the workflow runs. See "Configuring workflow forms and execution options" on page 35.
- **Run If qualifications** (optional) determine whether the workflow's If Actions or Else Actions are carried out. See "Building qualifications and expressions" on page 49.
- **Workflow actions** determine *what* an active link, filter, or escalation does when it runs. See "Specifying workflow actions" on page 63.

You can use active link guides and filter guides to control the order of workflow actions and organize a related set of workflow objects. See "Defining guides and guide actions" on page 137.

Active links allow you to create workflow designed for user interaction. You can use buttons and field menus with active links to assist the user. See "Using buttons and menu bar items to execute active links" on page 165.

For information about how AR System processes active links, filters, and escalations, see "Workflow processing" on page 175.

Workflow objects

Workflow objects automate your organization's business processes. You can create active links, filters, and escalations to perform actions on one form or several forms. When workflow is attached to multiple forms, it is considered shared workflow. See "Shared workflow" on page 32 for more information.

Active links, filters, and escalations share many similarities, but also have several differences that are described in this section.

About active links

An active link is an action or a series of actions that are triggered when a user performs an operation and are conditionally interpreted. The interpretation occurs on the AR System *client* in the current form window, and run with the permission of the user. For example, you might define an active link that displays a list of all problems reported for the current workstation whenever a user presses Enter in the Workstation field.

When an active link is executed, it can trigger varying actions as the result of a single user action. When you design an active link, you specify the conditions under which the active link executes, and further conditions to determine which action it will take. For example, an active link could attach one menu to the Short Description field if the user is a member of the Human Resources department, or else attach a different menu if the user is a member of the Shipping department.

Active links can be grouped into execution groupings called active link guides that allow procedural processing.

NOTE

Active links cannot be triggered through the use of an API program.

About filters

Filters implement and enforce your organization’s business rules. A filter tests every request transaction to see if certain conditions are met, and then responds to the conditions by taking specific actions. For example, a filter can notify support staff members when they are assigned responsibility for a new request.

Filters can act on virtually any condition that arises in a request. For example, filters can restrict how users create or modify a request. As another example, a filter can check for conditions in requests that are submitted by a network management system for a device that the system is monitoring. Then, the filter can automatically call a program to control that device.

Filters execute on the AR System *server* and run with administrator permissions. This means that filters can access any field in the AR System database, even if the field is not available to the user (no view or change access).

Filters can be grouped into filter guides to control the order of processing. For more information, see Chapter 5, “Defining guides and guide actions.”

About escalations

An escalation causes a condition to be checked on a regular basis and, depending on whether and how the condition is met, performs one or more actions. For example, an escalation can set the priority of a request to Urgent if the request is not closed within 24 hours, or send a page to a support staff member if a critical request has not been addressed in one hour.

Escalations execute on the AR System *server* and run with administrator permissions. Unlike filters, which run in response to a specific operation, escalations run at a specific time or after a defined time interval. Also, when they run, escalations find and act on all requests that meet a qualification, while filters act on the current request if it meets a qualification.

At the time specified in the escalation, AR System searches for requests that match the escalation qualification and performs the specified escalation actions on those requests that match.

Escalations can be assigned to pools so the escalations from each pool run in parallel on separate threads within the escalation queue. To use escalation pools, you must first configure multiple threads for the escalation queue as described in the *Configuration Guide*, "Queues," page 27. If you assign an escalation to a pool that has no thread configured, the escalation is run by the first thread.

All escalations in a particular pool run on the same thread, so the execution of escalations within a pool is serialized. Escalations run in the order of their firing times, but an escalation is delayed if an escalation from the same pool is currently running. If two or more escalations have dependencies and must not run at the same time, put them into the same pool to make sure they run in sequence.

Workflow comparison

Table 1-1 summarizes the functionality of active links, filters, and escalations.

Table 1-1: Workflow comparison

	Active Links	Filters	Escalations
Where performed	Client, current form window	Server	Server
Executed by	A user performing an action	A specific operation	A specific time
Purpose	Help the user to do something	Implement and enforce business rules	Initiate and ensure timely actions
What they do	Start a series of actions that are conditionally interpreted	Test every server transaction on the form to see if conditions are met in the current request, and respond by taking specific actions	At a specified time or time interval, check whether conditions are met for requests existing in the form, and if so, perform actions on requests that meet conditions
Run with permissions of	User	Administrator	Administrator
Example	Populate all name and address fields when user presses Enter in Last Name field	Notify support staff members when they are assigned a new request	Page a support staff member if a Critical request has not been addressed in one hour

General procedures for workflow objects

You create and manage workflow objects in BMC Remedy Developer Studio. Although active links, filters, and escalations have different purposes and properties, many procedures for working with them are the same.

This guide describes some common procedures for working with workflow objects. Other common procedures are described in other documents, as shown in the following table.

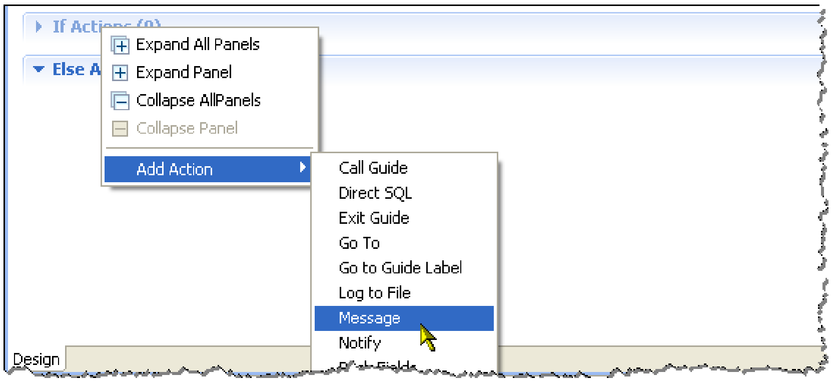
Table 1-2: Where to find information about workflow objects

For Information about	See
Creating, copying, renaming, and deleting AR System objects	The *Introduction to Application Development with BMC Remedy Developer Studio* guide.
Assigning permissions (enables you to specify users and groups who can access an object)	"Workflow object properties" on page 47 in this guide and *Form and Application Objects Guide*, "Assigning permissions," page 58.
Entering change history (a record of an object's owner, the user who last modified the object, and the date of modification)	*Introduction to Application Development with BMC Remedy Developer Studio*, "Updating change history," page 53.
Creating and modifying help text for objects	*Introduction to Application Development with BMC Remedy Developer Studio*, "Providing help text," page 54.
Creating a log file for tracing object activity (You can create a record of an object's operation, including what executed and whether it was successful.)	"Tracing active link, filter, or escalation activity" on page 191.

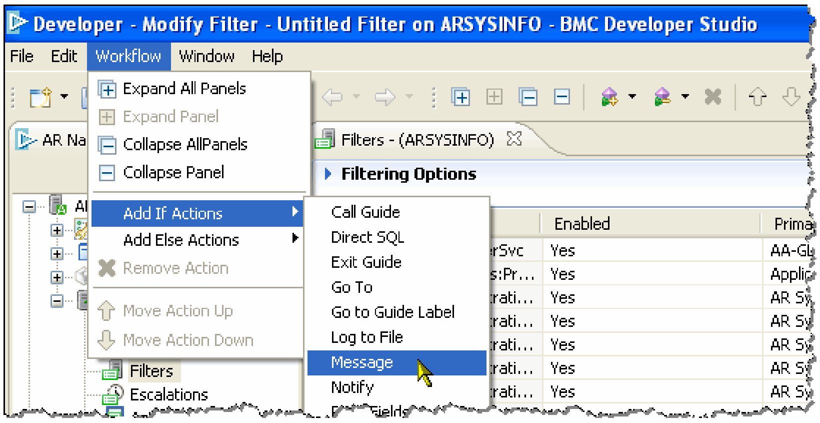
Using menu options in BMC Remedy Developer Studio

In BMC Remedy Developer Studio, you can access most menu options in multiple ways, including the use of right-click pop-up menus, main menu options, and menu bar icons. For example, to add an If Action to an active link, filter, or escalation, you can access the menu choices by any of these three methods:

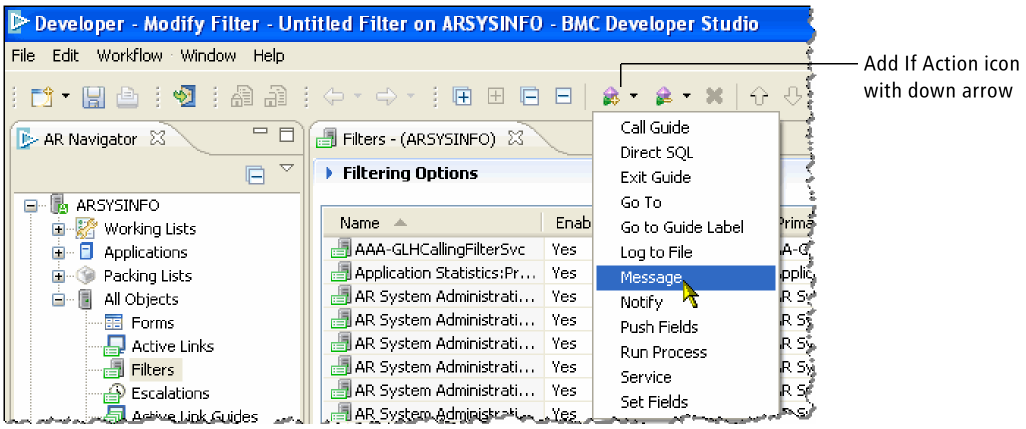
- Right-click the If Actions panel heading, and then select Add Action.



- Select the main menu option Workflow > Add If Actions.



- Click the down arrow next to the Add If Action icon in the menu bar.

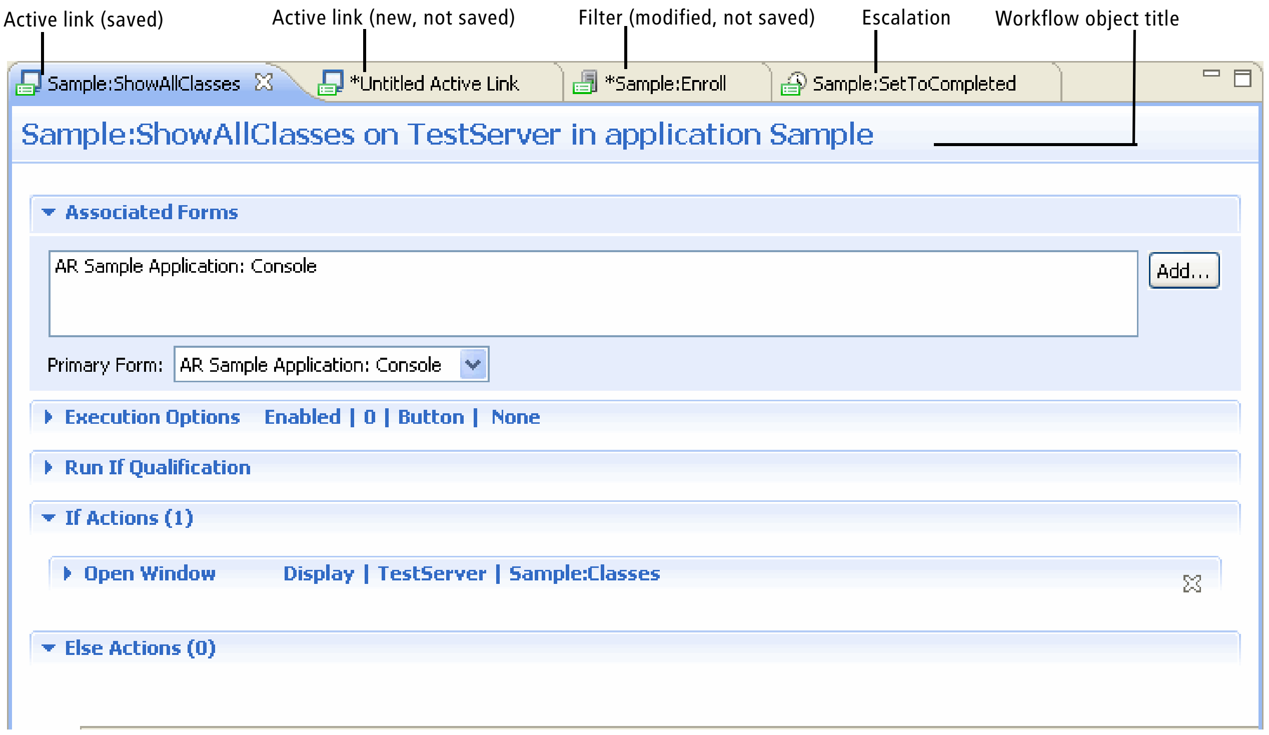


The procedures in this document describe using the right-click menu whenever it is available.

Using the workflow editor in BMC Remedy Developer Studio

When you open an active link, filter, or escalation, it appears as the active tab in the editor area of BMC Remedy Developer Studio. The information that defines the workflow object is organized into collapsible panels that you open to modify the object. Figure 1-1 shows the workflow editor, with the active link Sample:ShowAllClasses open in the active tab.

Figure 1-1: The workflow editor with open workflow objects


Active link (saved)
Active link (new, not saved)
Filter (modified, not saved)
Escalation
Workflow object title
Sample:ShowAllClasses
*Untitled Active Link
*Sample:Enroll
Sample:SetToCompleted
Sample:ShowAllClasses on TestServer in application Sample
Associated Forms
AR Sample Application: Console
Add...
Primary Form: AR Sample Application: Console
Execution Options Enabled | 0 | Button | None
Run If Qualification
If Actions (1)
Open Window Display | TestServer | Sample:Classes
Else Actions (0)

You can open multiple editing panels to create or modify the active links, filters, and escalations that you have permission to administer. In Figure 1-1, a new active link, the filter Sample:Enroll, and the escalation Sample:SetToCompleted are also open.

TIP

To maximize the editor, double-click the tab. You can also use the Editor perspective in BMC Remedy Developer Studio to make the editor area larger. See *Introduction to Application Development with BMC Remedy Developer Studio*, "Opening an object for editing," page 48.

The tab label for each workflow object displays the following information:

- When you create a new object, the tab label indicates whether you are working on an active link, filter, or escalation.
- The tab's icon also indicates whether the open object is an active link, filter, or escalation.
- After you save the workflow object, the tab label displays the name of the object.
- When you make changes to a workflow object, an asterisk indicates that the changes have not been saved. When you save the object, the asterisk disappears.

When you save a new active link, filter, or escalation, the workflow object title changes to include the name of the object and the server it is located on. If the object is part of an application, the application name also appears in the object title.

Table 1-3 provides an overview of the settings located in each of the expanding panels in a workflow object. For specific information about configuring these settings, see the referenced sections of this guide.

Table 1-3: Workflow editor panels

Panel label	Active links	Filters	Escalations
Associated forms See "Associating workflow objects with forms" on page 36.	■ Select the form or forms associated with the active link, filter, or escalation. ■ Identify the primary form, if there is more than one associated form.		
Execution options See "Defining workflow execution options" on page 37.	■ Set the state to Enabled or Disabled. ■ Select the user actions and associated fields (including buttons and menu options) that will execute the active link. ■ Select the execution order and interval, if any.	■ Set the state to Enabled or Disabled. ■ Select the actions that will cause the server to execute the filter. ■ Select the execution order.	■ Set the state to Enabled or Disabled. ■ Set the escalation pool number, if any. ■ Define the escalation time criteria, including whether the escalation runs at the specified days and times, or at an time interval.
Run If See Chapter 3, "Building qualifications and expressions."	■ Define the Run If conditions that determine whether the active link, filter, or escalation will run.		
Error handler See "Error handling filters" on page 187.	Not applicable	■ Enable or disable the use of an error handler. ■ Select the error-handling filter, if any.	Not applicable.
If Actions See Chapter 4, "Specifying workflow actions."	Define the actions that execute when the Run If conditions are met.		
Else Actions See Chapter 4, "Specifying workflow actions."	Define the actions, if any, that execute when the Run If conditions are not met. The Else action is optional.		

Graphical representation of workflow events

Beginning with release 7.6.04, the Event Navigator and Workflow Execution Viewer interfaces have been introduced in BMC Remedy Developer Studio.

These interfaces are supported for version 7.6.04 or later of the AR System server.

Context menus that enable you to launch the Workflow Execution Viewer and Event Navigator are available only if you select Record Object Relationships on the Configuration tab of the AR System Administration: Server Information form.

For more information, see the *Configuration Guide*, “Server Information—Configuration tab,” page 128.

This section provides the following information about these interfaces:

- About the Event Navigator (page 23)
- Launching the Event Navigator (page 25)
- About the Workflow Execution Viewer (page 26)
- Launching the Workflow Execution Viewer (page 32)

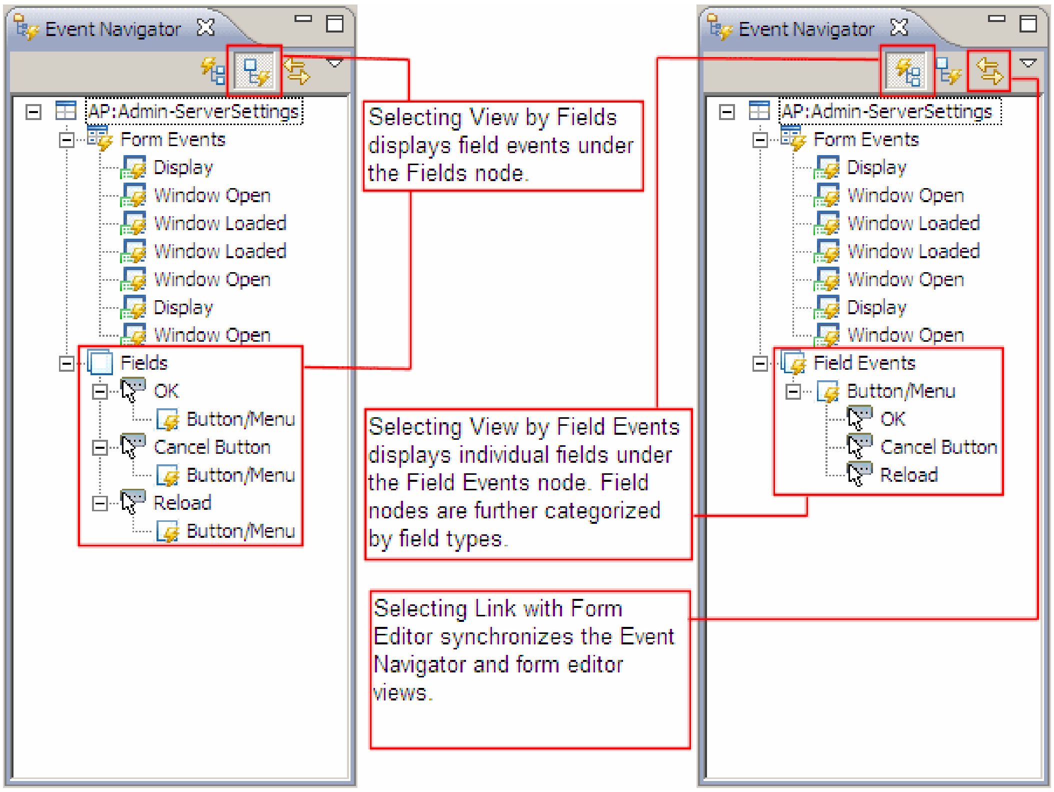
About the Event Navigator

BMC Remedy Developer Studio provides an Event Navigator view for a form, which helps you in the following ways:

- Displays in a tree structure the form events and field events that have workflow associated with them, thus outlining the events related to the form and its fields.
- Synchronizes itself with the active Form editor when the Link with Form Editor option is enabled.
- Enables you to view the workflow associated with an event easily.

The Event Navigator view can only be launched for one form at a time. See “Launching the Event Navigator.”

Figure 1-2: The Event Navigator view



The top-most parent node in the tree structure is the form node, which represents the form whose event navigation is being viewed. *Only those events that have workflow associated with them are listed.* When you open this view for the first time or switch to a different form's view, all the nodes are collapsed. You must expand each non-leaf node to view its details.

NOTE

The Event Navigator lists only those events that have workflow *in the Enabled state* associated with them.

The first child node is Form Events, which lists the events and escalations associated with the form. The time-based event nodes under Form Events might vary based on the number and type of escalations.

The second child node is either Field Events or Fields, depending on whether the View by Events or View by Fields option is set. Use the first two icons in the tab group toolbar to specify your choice; they are mutually exclusive. For more information, see "Tab group toolbar buttons in the Event Navigator" on page 25.

NOTE

When viewing by events, a field can appear under multiple event nodes depending on the events on which the workflow is fired.

The Show Workflow context menu is enabled only for the leaf nodes, whether it is a field or an event.

Relevant changes to the form, like addition or deletion of fields or workflow associated with the form or its fields, are reflected immediately in the Event Navigator.

Tab group toolbar buttons in the Event Navigator

The following buttons are available in the Event Navigator:

- View by Events ( )—Click to display the list of fields sorted by events. Expand the individual event nodes to view the fields to which they belong.
- View by Fields ( )—Click to display the list of events sorted by fields. Expand the individual field nodes to view the related events.
- Link with Form Editor ( )—The Link with Form Editor button on the tab group toolbar works as follows:
    - When selected, it links the Event Navigator with the active Form editor. This is useful when you want to watch the event navigation for a form being edited. When you select a field or event node in the Event Navigator, the appropriate field in the Form editor gets selected.

      If you switch a different form in the Form editor, the Event Navigator displays the events of the currently active form.

      If you reload the Event Navigator for a different form by using the Show Event Navigator menu in the Object List view, then the events for the new form are displayed. However, because Link with Form Editor is selected, switching to the previous form in the Form editor displays its events again.
- When deselected, the automatic synchronization is switched off, and you need to manually launch the Event Navigator from a form being edited. See "Launching the Workflow Execution Viewer."

  If the Form editor view is not open, then the state of the Link with Form Editor toggle button has no effect. The Event Navigator displays the events of the form from which you trigger the command.

Launching the Event Navigator

You can launch the Event Navigator by using the Window menu, but to use the view, you should open in the context of a form or field.

▶ To launch the Event Navigator regardless of a form

1. Choose Window > Show View > Other.
2. In the Show View dialog, either type `Event Navigator` as the filter text, or expand BMC Remedy Developer Studio, select Event Navigator, and click OK.

▶ To launch the Event Navigator for a particular form

1. Select the form in the Object List view.
2. From the context menu, select Show Event Navigator.

> **NOTE**
> The Show Event Navigator menu is only available in the Object List view for forms. This menu is disabled if you select multiple forms in the Object List view.

About the Workflow Execution Viewer

Application developers often need to understand how workflows associated with a single form are linked to each other, or how one workflow triggers some other workflow of a different form. In this regard, the Workflow Execution Viewer enables developers to do the following, without having to open forms and workflow objects:

- Generate a graphical representation of a workflow process, known as a workflow diagram.
- Document the flows and objects in a workflow.

See "Launching the Workflow Execution Viewer."

The workflow diagram depicts a series of activities, with an event as the starting point followed by sequential flow of control from one workflow object to another. Workflow objects appear in their order of execution. The workflow diagram ends when the last workflow object completes its execution.

Developers can expand each workflow object to view the actions and conditions involved. Workflow actions are executed in sequence.

The Workflow Execution Viewer is an editor that is *always* in the read-only state, and appears at the center of the workbench. It can utilize the maximum available space, which helps to view workflows that expand horizontally and vertically, when a developer drills down into a workflow.

This section describes the following aspects of the Workflow Execution Viewer:

- Events depicted in the Workflow Execution Viewer (page 27)
- Elements of the Workflow Execution Viewer (page 28)
- Operations in the Workflow Execution Viewer (page 31)

Events depicted in the Workflow Execution Viewer

The Workflow Execution Viewer depicts Form events and Side-effect events.

Form events

A form event could be one of many events that occur in sequence, in which case previous events may affect the current event. For example:

- When a form is opened, depending on the mode, the On Window Loaded event occurs, followed by On Display or On Window Opened.
- When an Apply action is performed, depending on the mode, the active link Modify or active link Submit, active link After Modify or active link After Submit, and Filter On Submit events occur sequentially.

For more information about form operations, see "Active link processing" on page 176.

Table 1-4 lists the form events (excluding those triggered by user actions) that can be depicted in the Workflow Execution Viewer.

Table 1-4: Form operations depicted in the Workflow Execution Viewer

Super event	Component events	Description
Launch Form for New	■ AL: On Window Open ■ AL: On Window Loaded ■ AL: On Set Default	This super event is triggered if the Open Window action is defined on the form to open the same form or another form; the On Set Default active link is triggered only if the option to set fields to default values is specified in the Open Window action for the Search or Submit window types.
Launch Form for Search	■ AL: On Window Open ■ AL: On Window Loaded ■ AL: On Set Default	This super event is triggered if the Open Window action is defined on the form to open the same form or another form; the On Set Default active link is triggered only if the option to set fields to default values is specified in the Open Window action for the Search or Submit window types.
Launch Form for Modify	■ AL: On Window Open ■ AL: On Window Loaded ■ AL: On Set Default ■ AL: On Search ■ AL: On Window Close ■ AL: On Window Open! ■ Filter: On Get Entry ■ AL: On Display	This super event is triggered if the Open Window action is defined on the form to open the same form or another form; the On Set Default active link is triggered only if the option to set fields to default values is specified in the Open Window action for the Search or Submit window types. Note: The active link is invoked twice in the second instance of On Window Open.
Save Modified	■ AL-On Modify ■ Filter- On Modify ■ Filter-On Get Entry ■ AL-On After Modify	This super event can occur when a user invokes the Save action (or the workflow performs a commit action) in Modify mode.

Table 1-4: Form operations depicted in the Workflow Execution Viewer

Super event	Component events	Description
Save New	■ AL-On Submit ■ Filter-On Submit ■ Filter-On Get Entry ■ AL-On After Submit ■ AL-On Set Default	Filter-On Get Entry is invoked only if the AL-On After Submit handler is defined. AL-On Set Default depends on the user preference. Set Default processing occurs only if the Set Default option has been specified for an Open Window action.
Search	■ AL-On Search ■ AL-On Window Close ■ AL-On Window Open ■ Filter-On Get Entry ■ AL-On Display	This super event is triggered when a user performs a search action on the form.

Side-effect events

Actions in the workflow objects can trigger some other events, for example:

- The Gain Focus event on a field can cause a Lose Focus event on some other field, which is difficult to capture.
- The Change Field action can trigger many events like Table Refresh, Gain Focus, and so on, which might have further workflow associated with them.
- The Push Fields action can trigger Modify, Submit, or Create events on the server, which in turn may have many filters associated with them.
- A Service Action almost always causes filters to fire.
- Some special Run Process commands in active links can trigger events.

When creating a workflow, application developers can benefit from viewing events (and the related workflow objects) that could possibly be triggered. To do so, developers can introduce related diagrams into the Workflow Execution Viewer on demand.

Elements of the Workflow Execution Viewer

The Workflow Execution Viewer contains items that depict the various elements in workflow like the form and object for which the workflow is defined, the qualifications and actions in a workflow, and so on.

Figure 1-3 depicts the following elements in the Workflow Execution Viewer:

1 Form node—Appears as rectangle that contains the form type icon (regular, join, display-only, vendor, or view) on the left and the form name on the right. It represents a form object, and is included to show the source of an event. In most cases, a form and its fields are the sources of events.

2 Transition—Appears as a simple, directed line. The arrow head in a transition indicates the direction of flow of control between actions within a workflow object or between workflow objects.

Some transitions have labels associated with them. A label could be an event name, which indicates that the following workflow is triggered on that event. A label could also indicate the condition on which the transition occurs. For example, a transition from a Run-If qualification to the first action in the workflow can be labeled “Y,” indicating that the transition occurs if the Run-If condition is satisfied.

If accompanied by a plus sign (+), it indicates that super events are associated with the parent node. Click the plus sign to launch the super events.

3 Workflow object node—Appears as a rectangle that contains the object type icon (active link, filter, or escalation) on the left and the object name on the right. The number on the node represents the execution order of that workflow object. If accompanied by a plus sign (+), it indicates that the node can be expanded to views its qualifications and actions.

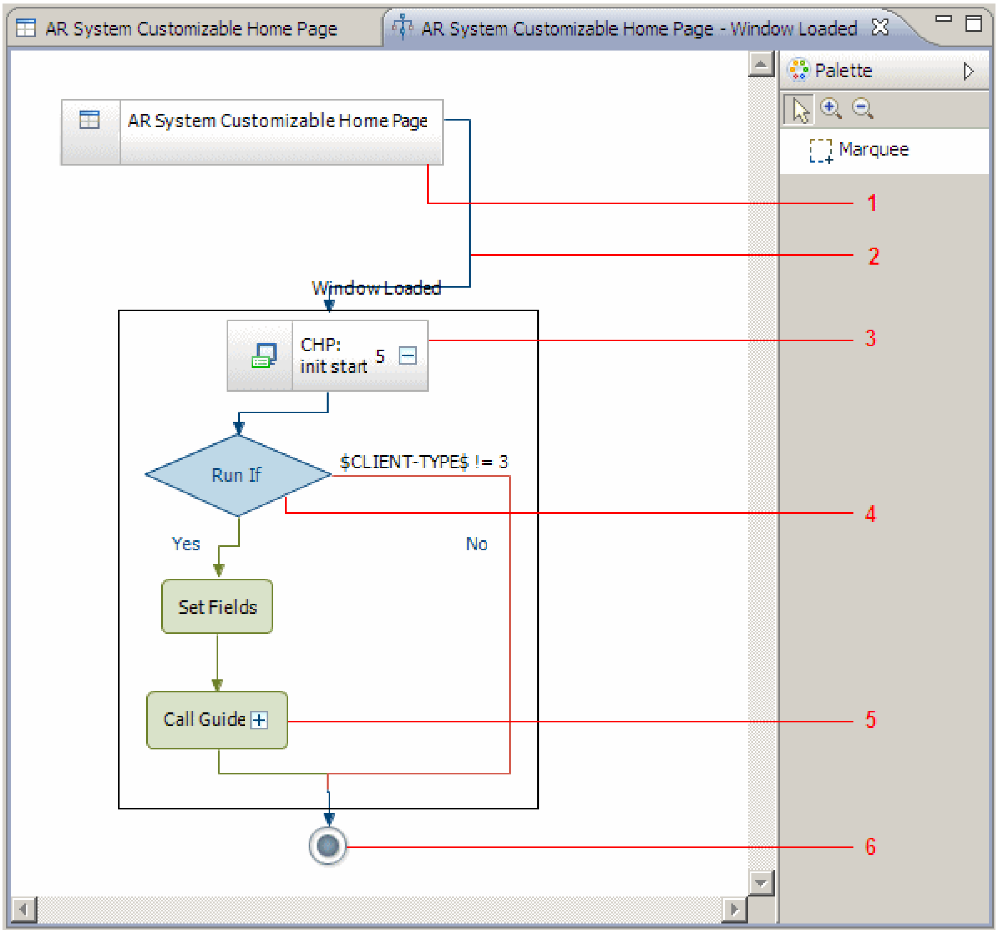
4 Qualification node—Appears as a rhombus that contains the qualification name. Depending on the possible outcomes, one or more flows can originate from a qualification node.

5 Action node—Appears as a rounded rectangle that contains the action name. If accompanied by a plus sign (+), it indicates one of the following:

- The action node has side-effect events associated with it; click the plus sign to launch the side-effect events.
- The action node indicates a Call Guide action; click the plus sign to expand the details-the guide name and its associated actions are displayed.

6 End of workflow—Appears as two concentric circles (with a filled inner circle), indicating that the workflow actions have been completed and no further activity will take place for this workflow.

Figure 1-3: A form event displayed in the Workflow Execution Viewer


AR System Customizable Home Page
AR System Customizable Home Page - Window Loaded
Palette
Marquee
AR System Customizable Home Page
Window Loaded
CHP: init start
5
Run If
$CLIENT-TYPE$!= 3
Yes
No
Set Fields
Call Guide
1
2
3
4
5
6

Generally, after the execution of a workflow action is completed the flow proceeds to the next action. However, in some cases, the flow might not proceed sequentially. For example:

- If the workflow being executed is contained in a guide, then actions such as Exit Guide and Goto Label can cause the flow to proceed either to the end of the guide execution or to some other node in the guide.
- For a Push Fields or Set Fields action, if a side-effect event is defined for the records that match the Run-If qualification, a plus sign appears next to the node. You can click the plus sign to expand and view the side-effect event.

  If a developer specifies the Display ‘No Match’ Error option, then the workflow diagram depicts that:

  - The flow associated with the failure of this qualification proceeds directly to the end of the workflow node.
  - The flow associated with the success of this qualification proceeds to the next action.
- For a Message action, if the message type is Error, then the flow proceeds directly to the end of the workflow node.

Operations in the Workflow Execution Viewer

You can perform the following operations in the Workflow Execution Viewer:

- Open—Right-click a node and select Open from the context menu to open the object in an editor. For example, an action node opens in the workflow editor.
- Document—Select one or more nodes and choose Document from the context menu. The Document Objects dialog appears with the selected nodes already added to the Select Objects panel. You can then proceed to document the selected objects by using the Documenter tool.
- Save As Image File—Select a node or right-click anywhere in the blank editor area and select File > Save As Image File from the context menu. The Save As Image File dialog box prompts you to provide a file name, location, and format. The diagram is saved in the current state—nodes are neither expanded nor collapsed when saving. You can also export the entire workflow diagram or the selection to HTML by using this menu.
- Print—Select one or more nodes and click File > Print. On the Print dialog, select an option from the Diagram Print Range. Alternatively, right-click an empty region in the Workflow Execution Viewer and use the File > Print context menu.
- Expand or Collapse—Choose Expand or Collapse from the context menu of a node. The Expand action expands a node and all its children; it expands the current activity but not its related activities. The Collapse action collapses all the child nodes of the selected node.

  If you select the Expand or Collapse context menu from a *blank area* in the workflow diagram, all the nodes in the diagram are expanded or collapsed, except side-effect events, super events, and active link or filter guides.
- Zoom—Use the Zoom In or Zoom Out toolbar buttons to zoom in or zoom out of the current diagram.
- Refresh—Click Refresh to reload the entire workflow diagram. This operation reloads the workflow only if it was modified after being opened in the Workflow Execution Viewer.
- Show workflow details—Hover over an action node to see its workflow details as a tool tip. The tooltip displays details similar to those seen in the active link, filter, and escalation editors when their actions are in the collapsed state.
- Marquee—Use the Marquee tool to select one or more nodes in the Workflow Execution Viewer. A dotted rectangle appears when you click and drag the mouse in an empty region, and all the nodes located in that region are selected. To select a node, you need to cover the complete area of that node; it does not get selected if you drag the mouse partially over the node.

  The Marquee tool is useful when you want to perform the Save As Image File, Open, Document, Expand, or Collapse operations on multiple nodes (those selected by the marquee and not all the nodes or a single node).

Launching the Workflow Execution Viewer

The Workflow Execution Viewer is associated with events because workflows are executed on:

- Field events like Gain Focus, Button Click, Menu Choice, and so on
- Form events like Submit, Merge, On Display, and so on
- Time-based events like at certain intervals or on certain calendar days

▶ To launch the Workflow Execution Viewer from the Form editor

If you know the exact field with which the workflow is associated:

1 Open the appropriate form in the Form editor.

2 From the context menu of the field, choose Show Workflow > *eventName*.

If the field does not have any workflow associated with it, the context menu appears as Show Workflow > No Workflows.

The Show Workflow menu is also available in the Outline view.

NOTE

The Show Workflow menu is not available for newly created forms and fields unless you commit the changes so that they are saved on the server.

▶ To launch the Workflow Execution Viewer from the Event Navigator

If you know the exact form and event to which the workflow is related:

1 From the context menu of the form in the Object List view, select Show Event Navigator.

The Show Event Navigator menu is also available for fields listed in the Outline view.

2 From the context menu of an event in the Event Navigator, select Show Workflow.

NOTE

The Show Workflow menu works only with a single event. If you select multiple events in the Event Navigator, this menu is disabled.

Shared workflow

In AR System, all workflow (active links, filters, and escalations) is based on forms. Workflow can be attached to one or multiple forms. For example, you can create an employee information active link that populates generic identification and address fields anytime a user enters a name or use this on multiple forms.

Shared workflow lets you efficiently build, maintain, and troubleshoot versions of forms and applications. Fewer workflow objects need to be stored on the server because any changes you make only need to be made once for all forms that use the objects.

WARNING

Use caution when sharing active links among forms in different deployable applications. Role permissions are resolved based on which application has ownership. The deployable application that contains the active link's primary form owns that active link or active link guide. If the non-owner application has identical roles mapped to different groups, these mappings are ignored. If only implicit groups have permission (no role permissions), there are no conflicts. For more information, see the *Form and Application Objects Guide*, "Defining access control," page 21.

The way you define shared active links, filters, or escalations is similar to the way you define workflow for an individual form. The main difference is that instead of attaching the workflow to one form, you attach it to multiple forms. If you do not want the workflow to be shared, select only one form. See "Associating workflow objects with forms" on page 36.

Workflow actions interact with fields based on field ID (not the field name). Plan carefully how you will use shared workflow before attaching it to multiple forms. To make it easier to administer shared workflow, create fields with the same ID *and* the same field name on each form. Otherwise, the workflow might not fire or the shared workflow actions might still be triggered but might not use the expected field. If fields have matching IDs but are different data types, AR System attempts to convert them appropriately.

After you have created a form with which you want to share workflow, you can:

- Create a new workflow object and then attach it to the forms.
- Select an existing workflow object and then attach it to the forms.

When exporting definitions, you can choose whether to maintain an association between the selected workflow and all related forms. For more information, see the *Form and Application Objects Guide*, "Importing and exporting object definitions and locking objects," page 557.

When you delete a form that uses non-shared workflow, the workflow is deleted along with the form. However, if workflow is shared by multiple forms, it is not deleted until the last form that uses it is deleted.

Sample uses of shared workflow

Possible ways to use shared workflow include:

- Populating common fields used to store employee data.
- Automatically assigning cases (based on tasks and applications).
- Accessing common forms such as bulletin boards, reporting, preferences, reminders, and so on.
- Navigating, for example, in a configuration tool.
- Displaying a message or warning dialog box that appears whenever a user submits a form or enters invalid information.

Chapter

2 Configuring workflow forms and execution options

This section describes how to associate a workflow object with a form, how to select the execution options for the workflow object, and how to set the time criteria for an escalation. It also briefly describes the properties of workflow objects.

The following topics are provided:

Associating workflow objects with forms

Because AR System workflow enforces business rules based on data that is stored in forms, all workflow objects must be associated with at least one form. To associate a form with a workflow object, use the Associated Forms panel.

▶ To associate a workflow object with a form

1 Open an existing active link, filter, or escalation, or create a new one.

2 Expand the Associated Forms panel if it is not already expanded.

3 Click Add.

4 In the Form Selector, select the form to associate with the workflow object, and click OK.

TIP

To locate a form quickly in a long list, you can use the Filtering Options or the Locate field in the Form Selector dialog box. See the *Introduction to Application Development with BMC Remedy Developer Studio* guide, "To filter the contents in an object list," page 38.

5 To associate the workflow object with an additional form, repeat steps 3 and 4.

The workflow object is attached to all of the forms you select. The first form you select automatically becomes the primary, or reference, form for the active link, filter, or escalation. See "About the primary form."

6 To change the primary form in cases where you have associated more than one form to the workflow object, select a different form from the Primary Form drop-down list.

About the primary form

All field references in the workflow refer to the primary form (also called the "reference form"), and these fields appear in the other panels as you define the workflow object. If you associate more than one form to a workflow object, make sure that any fields appearing on both forms have the same field ID on each form.

Shared workflow can use fields from the other associated forms that are not on the reference form, but you must define them by entering the field ID rather than the name. BMC Software recommends that you avoid doing this, however, because it makes applications more difficult to maintain.

Defining workflow execution options

The Execution Options panel contains settings that determine when the active link, filter, or escalation runs. For active links and filters, you specify what event or events trigger the workflow; for escalations, you specify the execution schedule for the workflow.

For all three workflow components, you can refine the execution options by adding a qualifying statement in the Run If panel. For information about creating a Run If qualification, see Chapter 3, "Building qualifications and expressions."

Examples of how you might use execution options include:

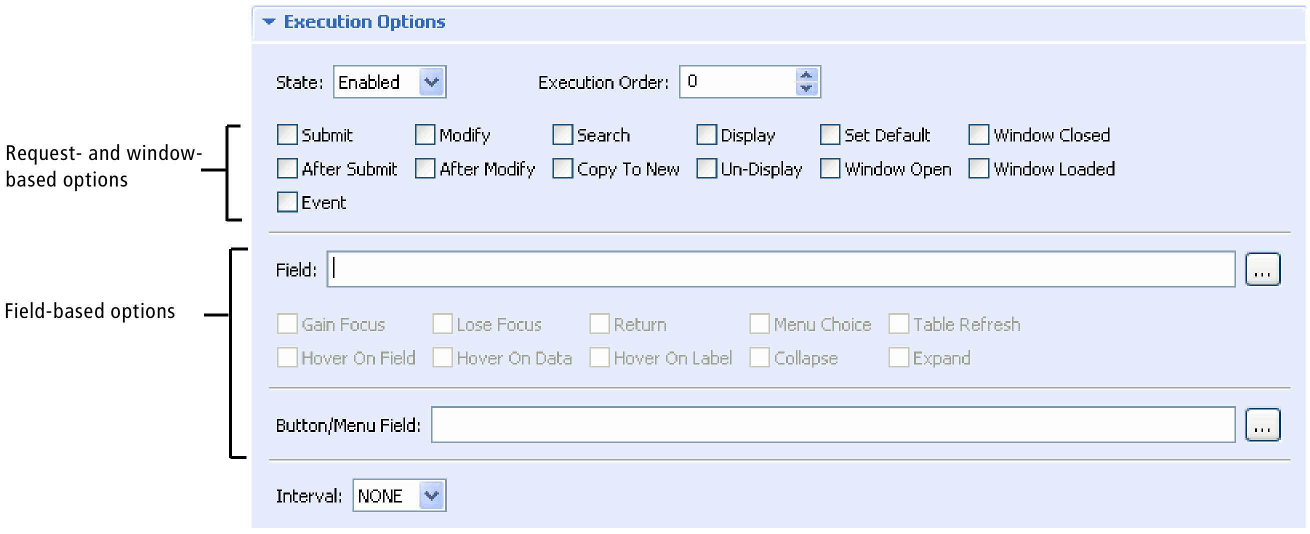
- An active link executes when the user presses Enter in a specified field or clicks a button on the form.
- A filter executes when a user submits a request.
- An escalation executes when a request more than eight hours old has not been closed by support personnel.

Creating active links

Active link execution options cause the active link to execute based on actions taken by the user. Some actions are specific to the open request or open window, while other actions are associated with fields in the referenced form. Figure 2-1 shows the execution options panel for an active link.

If an active link is part of a guide, you do not need to select an execution option.

Figure 2-1: Active link execution options



▶ To define the execution options for active links

1. Open an existing active link or create a new one.
2. Make sure there is an associated form selected. Select one in the Associated Forms panel if necessary. See "Associating workflow objects with forms" on page 36.
3. Expand the Execution Options panel.
4. Set the State field to Enabled or Disabled.

   When the state is Enabled, the active link becomes active as soon as it is saved. You might want to set the state to Disabled during development or when troubleshooting.
5. If necessary, enter a number in the Execution Order field.

   The value that you enter in the Execution Order field determines the order in which this active link is executed relative to other active links with the same triggering conditions. Numbers between 0 and 1000 are valid execution order values; lower numbers are processed first. The default value is 0.
6. Select the execution options for request and window actions, if any.

   The execution options for request and window actions that trigger active links are described in Table 2-1 on page 39. You can select any combination of these execution conditions, or none of them, as appropriate. If you select multiple options, the active link or filter executes when any one of the selected operations occurs.
7. Define the execution options for field actions, if any. To do so:
   a. Click the Field ellipsis button.

      The fields that appear in the field list are taken from the Primary Form defined in the Associated Forms panel.

      To locate a field quickly in a long list, use the Filtering Options or the Locate field in the Field Selector dialog box. See the *Introduction to Application Development with BMC Remedy Developer Studio* guide, "To filter the contents in an object list," page 38.

   TIP

   Instead of clicking the ellipsis button, you can also press Ctrl+Space with the cursor in the Field field. This brings up a list of the available fields from the form. Begin to type the field name to narrow the list, and then select the field you want.

   b. Select the appropriate field from the Field Selector dialog box and then click OK.

      When you select a field, the field execution options appropriate to the field type become active. The field-based, button, and menu execution options for active links are described in Table 2-2 on page 41.
   c. Select one or more field execution options for the field.

8 To cause the active link to execute when the user clicks a button:

a Click the Button / Menu Field ellipsis button.

b Select the appropriate button field and then click OK.

TIP

You can also associate an active link with a button by selecting the active link in the button's field properties.

9 In the Interval field, select an execution interval, if any. The minimum interval is three seconds, and the maximum interval is 7200 seconds.

When this is selected, the active link executes when the form is open at the specified time interval. (If two or more active links on a form have the same interval, they execute at the same time unless the execution order is set.)

WARNING

In workflow triggered by the Interval condition, avoid the use of Message actions and Open Window (of type Dialog) actions. This is to prevent an uncontrolled loop of messages or opened windows, which could consume resources on the client computer and make it difficult for the user to close the form.

Table 2-1 describes active link execution options relevant to request and window actions.

Table 2-1: Active links: Execution options for request and window actions (Sheet 1 of 3)

Execution option	Description
After Modify	Executes when a user modifies an existing request and *after* the request is written to the database. If the modification fails, the active link is not executed. (The active link does *not* execute during a Modify All operation.)
After Submit	Executes after a user submits a new request and *after* the request is written to the database. If the submission fails, the active link is not executed. If you use this condition for a set fields action on the current entry, the set value is not stored in the database.
Copy To New	Executes when a user chooses Edit > Copy to New in BMC Remedy User. The active link is executed after the data has been sent to the new window, so the application can access this data in the workflow. Example uses for Copy To New are: ■ Validating data when it is copied to a new submit form. ■ Disallowing the Copy To New function based on workflow. If an error is returned, no copy is performed and the form remains on the original window. This option is not supported in the browser client.
Display	Executes after an existing request is loaded into a form, but before the request appears in the Details Pane.

Table 2-1: Active links: Execution options for request and window actions (Sheet 2 of 3)

Execution option	Description
Event	Executes when a window has changed its application state, for example, when one window wants to send an event to one or more windows; when one window has caused data to change and another window references that data; or when one window is closed and other windows must be notified. This capability allows one part of an application in a client environment to notify other parts that an “event” has occurred. Other parts of the application can then react by performing actions such as refreshing table fields. The Event condition provides a mechanism in the context of a single client environment (for example, in a web client or in BMC Remedy User) for a parent window to be notified when a child window has been closed, so that workflow can refresh related data on the parent window. Events are constrained to the client environment. Windows in two separate client environments cannot send messages to each other. See “Sending events between windows” on page 295. **WARNING** In some cases, event driven workflow can fail when executed on the mid tier. To avoid this issue, explicitly define the parent-child relationship.
Modify	Executes when a user modifies an existing request. The active link is executed *before* the request is sent to the server. (The active link does *not* execute during a Modify All operation.)
Search	Executes when a user performs a search operation. The active link executes *before* the search operation so that, if the active link criteria is not met, the If actions are not performed. (If Else actions exist, they are performed.) To prevent a specific search from occurring (such as an unqualified search), you can have the active link return a message (such as an error); otherwise, the search is performed. You can also use active links to set fields to modify the search. Note: The active link can access and assign values in the Search window, including the Search Bar if the reserved Search Bar field is present on the form.
Set Default	Executes when the user chooses Edit > Set to Defaults from the menu bar. It can also happen after Window Open if default field values have been set in the form through user preferences.
Submit	Executes when a user submits a new request. The active link is executed *before* the request is sent to the server.
Un-Display	Executes when a request is removed from the Details pane because a new request was selected in the Results pane or because the window is closing. The workflow actions execute *before* the request is removed from the form.
Window Closed	Executes when a user closes a window.

Table 2-1: Active links: Execution options for request and window actions (Sheet 3 of 3)

Execution option	Description
Window Loaded	Executes after all the data values have been loaded into a Submit or Search window (from defaults, from a Copy to New action, or from an Window Open action).
Window Open	Executes when any of the following actions occur: ■ A form window opens in New, Search, Modify, or Modify All operation mode. ■ The mode of the Detail pane switches to New, Search, Modify, or Modify All mode. ■ The form is opened by using the Open Window action. This is useful for establishing the initial environment when a user opens a new window or changes modes. The active link is executed *before* any data is loaded into the window, except when the form is opened in Dialog mode by the Open Window action. Note: If you use a Set Fields action with the Window Open condition, the set fields values might be deleted if the user preference is set to Clear All Fields On Search or On New. To avoid this when using a Set Fields action, use the Window Loaded execution option, which executes after preferences are loaded, instead.

Table 2-2 describes active link execution options relevant to field actions.

Table 2-2: Active links: Field-based execution options (Sheet 1 of 3)

Execution option	Description
Button / Menu Field	Executes when a user selects a button or a menu button.
Collapse	Appears when the selected field is a Panel field type. ■ For Accordion and Stacked panel fields, executes when the user collapses a panel field. ■ For tabbed panel fields, executes with the user leaves the tab. ■ For a panel field of the Splitter type, this option causes no action.
Drag	Executes when the mouse moves over a field, and the user presses the left mouse button and drags the mouse. See "Allowing data to be dragged and dropped" on page 290.
Drop	Executes when the left mouse button is released over a "droppable" field after being dragged from a "draggable" field. See "Allowing data to be dragged and dropped" on page 290.
Expand	Appears when the selected field is a panel field type. ■ For Accordion and Stacked panel fields, executes when the user expands a panel field. ■ For Tabbed panel fields, executes when the user selects the tab. ■ For a panel field of the Splitter type, this option causes no action.

Table 2-2: Active links: Field-based execution options (Sheet 2 of 3)

Execution option	Description
Gain Focus	Executes when the specified field receives the focus. If you choose this option, the Field list is enabled so that you can specify the field that causes the active link to execute. Note: If you use the Gain Focus condition to execute an active link on a view field, the active link might not execute as expected. This is because the HTML page in the view field is taking the focus.
Hover On Field Hover On Data Hover On Label	Executes when the user hovers the mouse pointer over a field, field data, or a field label in the Web client. Along with the Message active link action, enables the use of tooltips to display a brief informational message. If the selected field is a data field, all three options are enabled. If the selected field does not have distinguishable label and data areas, only the "Hover on field" option is enabled. For information about creating tooltips, see "Message action" on page 88.
Level Choice	Appears when the selected field is a tree-view table field type. Executes when a user selects a level in the tree.
Level Double Click or Return	Appears when the selected field is a tree-view table field type. Executes when a user double-clicks a leaf in a tree view table or selects a leaf and then presses Enter to drill down to the source form. (If a user double-clicks on a parent node with child nodes under it, the node collapses or expands or no workflow is executed.) For more information, see the *Form and Application Objects Guide,* "Tree view tables," page 236.
Lose Focus	Executes as the focus is changed, for example, by clicking in another field. If you choose this option, the Field list is enabled so that you can specify the field that causes the active link to execute. An existing panel field loses focus when a new panel of the set appears at the front. Panels can gain focus whenever a user clicks a tab, an active link sets focus to a panel or a field on a panel, or another panel is hidden. Panel focus is independent of data field focus. Note: If you use the Lose Focus condition to execute an active link on a view field, the active link might not execute as expected. This is because the HTML page in the view field is taking the focus.
Menu Choice	Appears as an execution option when the selected field has an associated menu. Executes when a user makes a selection from a character menu attached to the field or selects a node in a tree view table field.

Table 2-2: Active links: Field-based execution options (Sheet 3 of 3)

Execution option	Description
Return	Executes when any of the following actions occur: ■ A user presses Enter on the keyboard. ■ A user presses Shift-Enter in a multirow character field. ■ A user selects a check box, drop-down menu item, or radio button.
Row Choice	Appears as an execution option when the selected field is a table field. Executes when a user selects a row in the table.
Row Double Click or Return	Appears when the selected field is a table field type. Executes when a user double-clicks a row in a table field or selects a row and then presses Enter to drill down to the source form. This execution condition works independently of whether the table drill-down option has been selected for the table field. This execution option works only if the user has been granted permission to fields on the supporting form, the column fields, and the table field.
Table Refresh	Appears as an execution option when the selected field is a table field. Executes when the user updates the table contents by loading the field, sorting, refreshing, or displaying the previous or next chunk.

1 Expand a Run If Qualification panel, and enter a qualification, if needed.

For more information, see Chapter 3, “Building qualifications and expressions.”

2 Enter If and Else actions, as needed.

For more information, see Chapter 4, “Specifying workflow actions.”

3 Save the active link.

Creating filters

Filter execution options cause the filter to execute based on actions that occur on the AR System server for requests in the reference form.

If a filter is part of a guide, you do not need to select an execution option. If an action generates an error during the processing of a filter, no further actions occur.

▶ To define the execution options for filters

1 Open an existing filter or create a new one.

2 Make sure there is an associated form selected, or select one in the Associated Forms panel if necessary. See “Associating workflow objects with forms” on page 36.

3 Expand the Execution Options panel.

4 Set the State field to Enabled or Disabled.

When the state is Enabled, the filter becomes active as soon as it is saved. You might want to set the state to Disabled during development or when troubleshooting.

5 If necessary, enter a number in the Execution Order field.

The value that you enter in the Execution Order field determines the order in which this filter is executed relative to other filters with the same triggering conditions. Numbers between 0 and 1000 are valid execution order values; lower numbers are processed first. The default value for filters is 500.

NOTE

Although filters are processed in execution order, some filter actions are queued up to be performed at a later time. For information about filter phases, see "Filter processing in the AR System server" on page 177.

6 Select the filter execution options, if any.

Filter execution options are described in Table 2-3. You can select any combination of these execution conditions, or none of them, as appropriate. If you select multiple options, the active link or filter executes when any one of the selected operations occurs.

Table 2-3: Filter execution options

Execution option	Description
Modify	Executes when a request is modified.
Submit	Executes when a request is submitted to the server.
Delete	Executes when a request is deleted.
Get Entry	Executes when a request is retrieved.
Merge	Executes when a request is merged into the database by using BMC Remedy Data Import, a DSO independent copy transfer, or the data import command line interface. For more information, see: ■ *Configuration Guide*, "Importing data into AR System forms," page 279 ■ *Integration Guide*, "Exporting and importing data and definitions," page 233 ■ *BMC Remedy Distributed Server Option Guide*, "Overwriting all fields in duplicate requests," page 40
Service	Executes when a Service active link action is performed. The filter accesses a request with the field values passed by the active link action and those retrieved from the database, if any, and returns output values to the calling request. No other database operation is performed. See "Service action" on page 127. If the filter runs a Push Fields action, the output field value list is *not* affected. If the filter runs a Set Fields action, the output field value list uses any modified values.

7 Expand a Run If Qualification panel, and enter a qualification, if needed.

For more information, see Chapter 3, "Building qualifications and expressions."

8 Expand the Error Handler panel, and complete the fields as needed.

For more information, see "Error processing" on page 187.

9 Enter If and Else actions as needed.

For more information, see Chapter 4, "Specifying workflow actions."

10 Save the filter.

Creating escalations

An escalation uses a schedule to determine when it is executed, using either a set time or a time interval. When you are working with an escalation, the settings in the Execution Options panel change depending on whether you select Time or Interval in the Run By field.

- **Time**—Time escalations run at the specified times, for example, every Monday, Wednesday, and Friday at 10:00 a.m., 2:00 p.m., and 4:00 p.m. The time specification permits recurring execution on multiple days of the month and days of the week at a fixed number of minutes after multiple hours in the day.
- **Interval**—Interval escalations run after the defined time interval has passed, for example, every 30 minutes. You can select any combination of days, hours, and minutes.

NOTE

Because an escalation can apply to many requests, it can require significant computer resources to perform its Run If test and its actions. Therefore, consider the performance impact when constructing the Run If test and scheduling escalations. To distribute the load, vary escalation times or use escalation pools.

NOTE

There might be irregularities the first time escalations execute after Daylight Saving Time (DST) transitions. For example, an escalation is scheduled to run at 12:00 noon every Monday. On the first Monday after clocks are set ahead, the escalation runs at 1:00 p.m. instead of at noon. On the first Monday after the clocks are set back, it runs at 11:00 a.m. and again at noon.

▶ To define an escalation that runs by time

1 Open an existing escalation or create a new one.

2 Make sure there is an associated form selected, or select one in the Associated Forms panel if necessary. See "Associating workflow objects with forms" on page 36.

3 Expand the Execution Options panel.

4 Set the State field to Enabled or Disabled.

When the state is Enabled, the active link becomes active as soon as it is saved. You might want to set the state to Disabled during development or when troubleshooting.

5 To assign an escalation to an escalation pool, enter the pool number in the Pool Number field.

The Pool Number should be between 1 and the number of threads configured for the escalation queue. If the Pool Number is blank or outside the valid range, the escalation is assigned to pool number 1 and is run by the first escalation thread.

See "About escalations" on page 17 for how to use escalation pools.

6 In the Run By field, select Time.

7 Define the execution run times by selecting the appropriate options from the Days of Month, Days of Week, and Hours of Day menus.

For example, to run an escalation every Friday at 10:00 p.m., click Friday in the Days of Week menu, and 10PM in the Hours of Day menu.

- You must select at least one day of the month or one weekday from Days of Month or Days of Week, and at least one time from Hours of Day.
- In the Days of Month menu:
  - Selecting **All** selects every day of the month.
  - Selecting **None** clears every day of the month.
  - Selecting **31** causes the escalation to run on the last day of the month for *all* months.
- To run the escalation at a certain number of minutes after the hour, select the hour and then type the number of minutes in the "Minutes after Hour" field. For example, for 5:15PM, select 5:00PM, and type "15" in the "Minutes after Hour" field.
- You can combine selections from the Days of Month and Days of Week menus. For example, if you select 15 and 31 from the Days of Month menu *and* Fridays from the Days of Week list with 10AM from the Hours of Day menu, the server executes the escalation on the 15th and 31st and all Fridays during the month. If a Friday occurs on a 15th or 31st, the server executes the escalation only once.

TIP

Clicking the time criteria menus has a toggle action. For example, click Friday to add Friday to the time criteria; click Friday again to remove it.

8 Expand a Run If Qualification panel, and enter a qualification, if needed.

For more information, see Chapter 3, "Building qualifications and expressions."

9 Enter If and Else actions, as needed.

For more information, see Chapter 4, "Specifying workflow actions."

10 Save the escalation.

▶ To define an escalation that runs by interval

1. Repeat steps 1 through 5 of the procedure “To define an escalation that runs by time” on page 45.
2. In the Run By field, select Interval.
3. Use the Days, Hours, and Minutes selection fields to set the duration of the interval.

   The interval begins when you create, modify, or enable the escalation, when the server is restarted, or when the escalation terminates. For example, if you enable an escalation with a 30 minute interval, it executes 30 minutes after you enable it. If the escalation takes five minutes to complete, it executes again 65 minutes after you enabled it (which is 30 minutes after it terminated).
4. Expand a Run If Qualification panel, and enter a qualification, if needed.

   For more information, see Chapter 3, “Building qualifications and expressions.”
5. Enter If and Else actions, as needed.

   For more information, see Chapter 4, “Specifying workflow actions.”
6. Save the escalation.

Workflow object properties

When you open an active link, filter, or escalation, the properties appear in the Properties view for the active workflow object. Table 2-4 describes the properties of workflow objects.

Table 2-4: Workflow object properties

Property type	Active links	Filters	Escalations
Permissions	■ Grant or prevent access to the active link for users and groups	Not applicable	Not applicable
Change history	■ This object property automatically records the owner, the user who last modified the object, and the date of the modification. You can also enter a description of your changes.		
Help text	■ Enter help text to describe what the workflow object does or how it is used.		

You can modify some of these properties, including the permissions for active links and the help text for all workflow objects. For information about setting and changing object properties, see the *Introduction to Application Development with BMC Remedy Developer Studio* guide, “Working with existing objects,” page 48.

Saving and copying workflow objects

To copy an active link, filter, escalation, or form in BMC Remedy Developer Studio, select File > Save As from the menu. The Save *objectType* As dialog box appears. Type the new name for the object and click OK.

Chapter

3 Building qualifications and expressions

This section discusses how to build qualifications for workflow actions.

The following topics are provided:

About Run If qualifications and expressions

In *active links* and *filters*, the Run If qualification is an expression that determines whether the If actions or Else actions are executed for the current request. In *escalations*, the Run If qualification first determines whether there are any requests in the database that match the qualification. If so, the escalation if actions are executed for each matching request. If not, the escalation else actions are executed once. See “If Actions and Else Actions” on page 64.

Other AR System features also use expressions, including Set Fields, Push Fields, and certain Open Window actions, as described in Chapter 4, “Specifying workflow actions.” Table fields, alert lists, search menus, join forms, defined searches, data archiving, auditing, and export (described in the *Form and Application Objects Guide*) also use expressions.

This chapter describes how to build an expression, focusing on Run If qualifications. For specific examples of building expressions in workflow actions, see Chapter 4, “Specifying workflow actions.”

Using the expression editor and content assist

You can define any expression in BMC Remedy Developer Studio by using either one of the following techniques:

- Click the ellipsis button on the appropriate field to open the Expression Editor dialog box, and then build the Run If qualification or other expression by selecting fields, keywords, and operators.
- In the workflow editor for the object, type the expression into the field with the help of the content assist feature.

About the expression editor

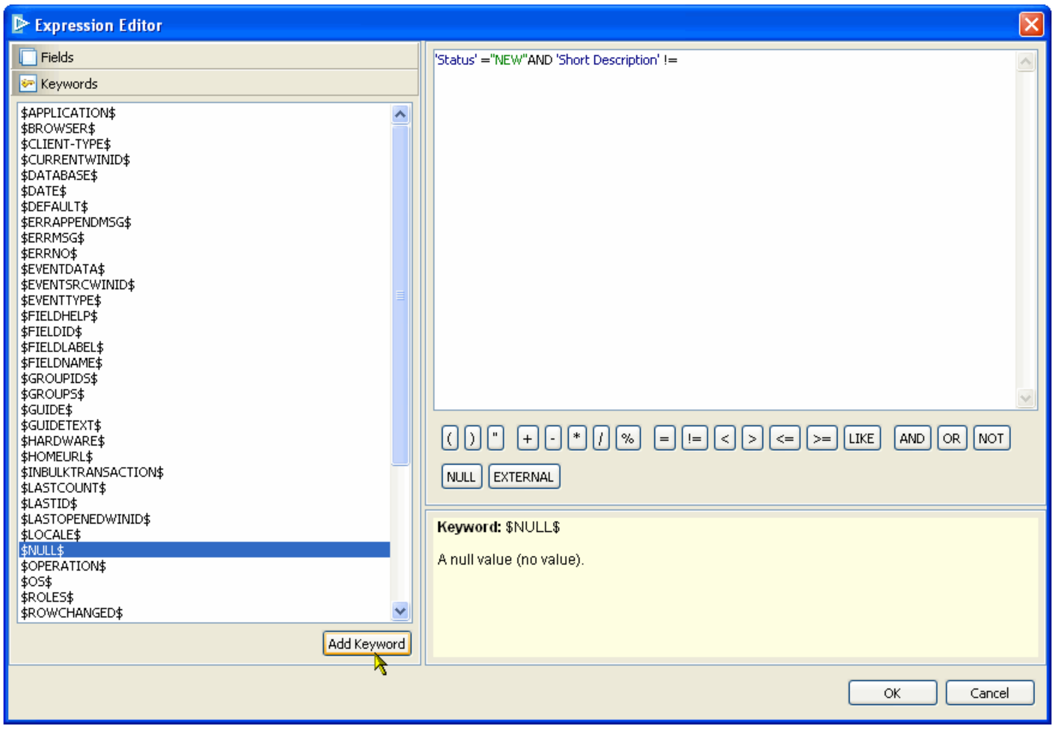
The expression editor is a dialog box that appears in BMC Remedy Developer Studio whenever you click the ellipsis button in a field that requires an expression. It presents the data types allowed for the expression in the current context, such as the field names from the primary form, AR System keywords, the results of a function, or static values. Additional data types depend on the workflow action type. The expression editor also includes buttons for the most commonly used operators.

To build an expression, you select from the options in the expression editor. For example, Figure 3-1 shows a Run If qualification in progress in the expression editor. In this case the developer has added the fields ‘Status’ and ‘Short Description’, along with the operators `AND` and `!=`, and is about to add the keyword `$NULL$` to the end of the expression.

When you highlight an entry in the lists of fields, keywords, and other data types, BMC Remedy Developer Studio provides information about the selection in the information area of the expression editor.

For an example procedure that describes building a Run If qualification in a filter, see "Filter qualification examples" on page 59. For an example of using an expression to map fields in a workflow action, see "Mapping fields to values" on page 68.

Figure 3-1: The Expression Editor dialog box



```
'Status' ="NEW"AND 'Short Description' !=
```

Keyword: $NULL$

A null value (no value).

Run If qualifications and other expressions can contain any valid sequence of operators, wildcards, and keywords, as shown in the following examples:

```
'Submitter' LIKE "Jackson%"
'Create Date' < $TIMETSTAMP$ - (60*60*24)
```

Appendix A, "Operators, wildcards, keywords, and NULL values," describes how each of these expression components is used and evaluated in AR System.

When entering an expression, make sure that it is acceptable to the database on which AR System is running. Avoid creating comparisons with fields of unlimited length, because the database might not support that capability. For more information about fields of unlimited length, see the *Form and Application Objects Guide*, "Input Length," page 515.

NOTE

Microsoft SQL Server and Sybase RDBMSs remove hard returns at the end of character strings. This can result in incorrect qualification syntax. To resolve this problem, AR System adds a space before any hard return at the end of a string.

About content assist

You can also type the expression directly into the appropriate field of the workflow editor. If you are already familiar with AR System expression syntax, using content assist is a quick way to build an expression manually, but still ensure that field names, keywords, and other data types are spelled correctly and have correct punctuation.

The content assist feature pops up when you begin to type an expression. For example:

- If you type a single quote (‘) in the field where are building an expression, content assist provides the list of available fields from the associated form. Figure 3-2 shows an example of this when building a Run If qualification.
- If you type a dollar sign ($) in the field where are building an expression, content assist provides the list of AR System keywords.
- If you press Ctrl+Space in the field where are building an expression, content assist provides a list of all the available data types.

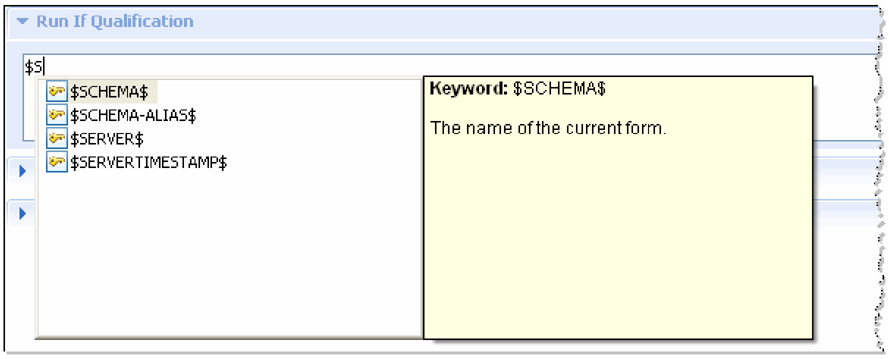
The content assist list is grouped by data type, such as fields, keywords, and functions. Within each data type the list is sorted alphabetically.

The list is filtered according to any characters you have already typed so far. For example, if you enter `'S`, the list is immediately filtered to show only the field names beginning with the letter “s” (such as Short Description, Status, and Submitter).

NOTE

If you type an incorrect character or click elsewhere on the screen, the content assist list might disappear. If this occurs, delete what you have typed so far and retype the quote, or dollar sign, or press Ctrl+Space, to reactivate it.

Figure 3-2: The content assist feature



Keyword: $SCHEMA$

The name of the current form.

Using fields in qualifications

This section describes the formatting conventions for fields used in a Run If qualification or expression.

Single quotation marks

Enclose field names or field IDs in single quotation marks, for example, `'Short Description'` or `'7'`. Single quotation marks are automatically added when you select fields from the Available Fields list in the Expression Editor.

NOTE

If the field name includes a single quotation mark, enter two single quotation marks in it's place. For example, if the field name is Doug's Requests, enter it as `'Doug''s Requests'`.

Currency fields

For currency fields, you must have one of the following enclosed within single quotation marks:

- The name or Field ID of the currency field, for example:

  ```
  'currencyField' = $NULL$
  ```

- The name of the currency field, followed by a period and a specific portion of the currency field's value, such as the date or a functional currency value, for example:

  ```
  'currencyField.VALUE' < 5000
  ```

  If the qualification for the currency field includes a calculation, you must include the currency type to avoid an error. For example, if you have a field called "Tax Savings" that displays the percentage of tax savings of a field that calculates the difference of two billing fields, determine the percentage by using a qualification like this:

  ```
  ($Billing Totals Difference.VALUE$ * $Current Tax Rate$) /
  ("100" + $Billing Assessment Total Difference.TYPE$)
  ```

  If you use field references in the calculation, you do not need to supply the currency type; it is determined at run time. For example:

  ```
  ($Billing Totals Difference.VALUE$ * $Current Tax Rate$) /
  $Integer field that stores the division number$
  ```

Attachment fields

The value of an attachment field used in a qualification is a character string containing the fully qualified file name of the attachment file.

Field values and filters

For filters, you can specify whether the qualification should reference field values in the transaction only, in the database only, or in both. See “Checking transaction versus database values” on page 58.

Field ID 112 and dynamic group fields

If referencing field ID 112 (for assignee group access) or dynamic group fields, enclose the name of the field in single quotation marks. Enclose the value statement in double quotes. The value statement can contain a group ID, group name, role ID, role name, or user name, and each value must be followed by a semicolon within the quotes. For example, if the name of field 112 is Assignee Group and the group ID of Sales Staff is 50, enter the qualification as follows:

```
'Assignee Group' = "50;"
```

For user Mary Manager, use single quotation marks:

```
'Assignee Group' = "'Mary Manager';"
```

To enter multiple groups, roles, and user names in your qualification, enter a semicolon at the beginning of the list and include no spaces. For example, the name of field 112 is Assignee Group. To create a qualification that includes the Sales Staff group (ID 50), the Marketing Staff group (ID 51), the Managers role (ID -90), and user Mary Manager, enter the qualification using the following format:

```
'Assignee Group' = ";50;51;-90;'Mary Manager';"
```

To use multiple groups for field ID 112 and for dynamic group fields, select Enable Multiple Assign Groups on the Configuration tab of the AR System Administration: Server Information form. See the *Configuration Guide*, “Server Information—Configuration tab,” page 128.

Field names containing special characters

If the field name contains the character that it will be delimited with in the expression, you must double that character in the field name within the expression.

For example, when the delimiter is single quotation marks:

- For *field’Name* use ‘*field’ ’Name*’
- For *field$Name* use ‘*field$Name*’
- For ‘*fieldName* use ‘’’*fieldName*’

When the delimiter is dollar signs:

- For *field$Name* use *$field$$Name$*
- For *field’Name* use *$field’Name$*
- For *$fieldName* use $*$$fieldName*$

In BMC Remedy Developer Studio, the auto-complete feature and the field selector dialog box manage this syntax convention for you.

Double quotation marks

Some values must be enclosed in double quotation marks (“) in a qualification. The following table outlines when to use double quotation marks.

Table 3-1: Using double quotation marks

Value	Use Quotation Marks?	Notes

Nonnumeric, such as time, character, and diary	Yes	Enclose a date in double quotations:
		`"04/01/04"`
		For example, to search for all requests created after a certain date, use:
		`'Create-date' > "04/01/04"`

Nonnumeric that include quotation marks	Yes	In addition to the double quotes contained in the value, include double quotes around the full entry. For example, if you want to search for a string such as:
		`Robert "Bobby" Smith`
		enter:
		`"Robert ""Bobby"" Smith"`

Integers (for time fields)	No	Integers without quotes are interpreted as a number of seconds. To specify a number of hours, multiply 60 (= 1 minute) by 60 (= 1 hour). For example, to specify 10 hours, enter: `60 * 60 * 10`

Keywords	No	Enter keywords with dollar signs ($) as shown in this example:
		`'Submitter' = $USER$`
		Using no quotation marks is also enforced for using keywords with calculations. Valid qualifications include:
		■ `'Create-date' > $TIMESTAMP$`
		■ `'Create-date' > ($TIMESTAMP$ - (60 * 60 * 24))`
		Use the keyword `$NULL$` to find requests that have no value in a field (for example, `'Assigned To' = $NULL$`).
		For qualifications passed to an `EXTERNAL()` operator, you can set a character field with a *keyword*, rather than with the *expanded value* of the keyword. To do so, add an escape character (`\`) to the qualification, like this:
		`‘Character Field’ = $\USER$`
		The character field is populated with `$USER$` instead of with the user’s name. See “Operator types” on page 216 for more information about `EXTERNAL()`.
		See “Keywords” on page 221 for available keywords.

Status history in filter qualifications

Status history references in a filter's Run If qualification must have the following information and format:

- The name or ID of the Status History field (followed by a period).
- The name or index of the status value that you want to match (followed by a period).
- The keyword `USER` (for the user who is changing the status) or `TIME` (for the time that the status changed).

For example:

```
'Status History.Fixed.TIME' < "07/01/99"
```

This syntax is created automatically when you select the Status History reference from the Run If list.

In a filter qualification, you can use a Status History reference if you are checking values in the database; Status History is not meaningful if you are trying to check the value of the current transaction. (For more information, see "Checking transaction versus database values" on page 58.) You cannot use Status History references in active link Run If qualifications or you cannot set the Status History value to a field in an active link set fields action.

Creating efficient qualifications

When the AR System server checks the Run If qualification of an active link or filter, it checks only the current request. However, when the server checks the qualifications for escalations, Set Fields actions, Push Fields actions, certain Open Window actions, table fields, alert lists, search menus, and defined searches, it must evaluate the qualifications against all requests in the database for the selected forms.

You can improve system performance dramatically if you follow these recommendations:

- Use indexed fields. Tests of indexed fields use the index to access the requests instead of scanning all requests. For information about indexing, see:
    - The *Form and Application Objects Guide*, "Defining indexes," page 181
    - The *Optimizing and Troubleshooting Guide*, "Creating effective indexes," page 33

- Avoid the following searches, which cause a full scan of a database table:
  - **The != Operator**

    Searches using the `!=` operator check every record to see if the value is *not* contained. If you have created indexes on a field, they are not used. Instead, design your qualifications to retrieve what you *are* looking for, instead of what you are *not* looking for. For example, you can rewrite the search qualification `'Status' != "Closed"` to `'Status' < "Closed"` to improve the use of an index.

    The `!=` operator does not match entries in which the value for the field is `NULL`. You must explicitly include a test for `NULL` to find `NULL` values. For example, `'Status' != "Closed"` does not search for cases where the Status field is empty. You must use the following syntax:

    ```
    'Status' != "Closed" OR 'Status' = $NULL$
    ```

NOTE

On Sybase databases, you cannot use the != operator against unlimited character fields where the database input length equals 0.

  - **Wildcards in front of search terms**

    Searches that begin with a wildcard (for example, `'Submitter' LIKE "%Jackson%"`) does not use the index but scans the database for every record containing the word `Jackson` that have any characters preceding the `J`.

    Searches with trailing wildcards are valid and use indexes. For example, `'Submitter' LIKE "Jackson%"` uses the index and finds all entries starting with `Jackson`.

  - **Poorly written arithmetic operations**

    Searches that use the field in an arithmetic operation can cause a table scan. For example, a qualification to find all requests greater than 24 hours (`$TIMESTAMP$ - 'Create Date' > 60*60*24`) searches the entire database for records with `$TIMESTAMP$` (the current date and time). Instead, rewrite the search qualification to place the indexed field on the *left* side of the equation, as in the following qualification: `'Create Date' < $TIMESTAMP$ -60*60*24`.

- Avoid unqualified Run If statements for escalations. An unqualified Run If statement for an escalation performs an unqualified query to the database table and execute the If actions for *every* request found in the form. Server performance is especially degraded for unqualified Run If statements in escalations that are set to run at frequent intervals, because the server must search the database every time the escalation is run.

NOTE

If the database is case-sensitive, any queries run by active link workflow are case-sensitive. For an Oracle® database, you can set the `Db-Case-Insensitive` option in the `ar.cfg (ar.conf)` file to support case-insensitive searches. However, this option can have a negative impact on performance.) See the *Configuration Guide*, "`Db-Case-Insensitive`[1]," page 361.

For more information about optimizing AR System searches, see the *Optimizing and Troubleshooting Guide*., "Defining effective searches," page 35.

Checking transaction versus database values

In filter Run If qualifications only (*not* Set Field or Push Field If qualifications), the qualification can access values for the current record from the current transaction or from the database. This makes it possible for you to check state transitions in filters. For example, you can define a filter that watches for transactions where the status of a request is changed to Closed. It can then check the database to verify that the current status of the request is Fixed, and reject the operation if the status is anything other than Fixed.

You can specify a field reference in the filter qualification in three ways:

- **Transaction Only**—References the value of the field in the current transaction only. If the value is not changed in the transaction, it is considered to be `$NULL$`. If the operation is a delete, it is considered to be `$NULL$`. To specify a check of the transaction only, use the format `'TR.`*`field`*`'` when you enter the field name in the Run If field.
- **Database Only**—References the value of the field in the database only. No check is made of the value in the current transaction. If the operation is a create operation, it is considered to be `$NULL$`. To specify a reference of the database only, use the format `'DB.`*`field`*`'` when you enter the field name in the Run If field.
- **Transaction and Database**—References the value of the field in the current transaction and uses that value if changed. If not changed in the current transaction, references the value of the field in the database. To specify a reference of both the transaction and the database (in this order), use the format `'`*`field`*`'` when you enter the field name in the Run If field.

Filter qualification examples

The procedures in this section outline the steps for creating three example filter qualifications.

Example 1—Filter execution based on transaction status

This example qualification causes the filter to execute when the status of the transaction is set to Fixed or Closed:

```
'TR.Status' = "Fixed" OR 'TR.Status' = "Closed"
```

The `TR.` syntax causes the filter to use the value from the current transaction, rather than the database value.

▶ To enter this qualification using the Expression Editor

1. In the filter editor, click the ellipsis button on the Run If Qualification panel.
2. In the Expression Editor, select `Status` from the Available Fields list, and then click Add Field.
3. In the expression editing area, insert `TR.` after the first quotation mark:

   ```
   'TR.Status'
   ```
4. Click the equal sign (=) operator to add it to the end of the expression:

   ```
   'TR.Status' =
   ```
5. Type `Fixed` enclosed in double quotation marks:

   ```
   'TR.Status' = "Fixed"
   ```
6. Click the OR operator:

   ```
   'TR.Status' = "Fixed" OR
   ```
7. Repeat steps 2 through 4.

   The Run If field now contains:

   ```
   'TR.Status' = "Fixed" OR 'TR.Status' =
   ```
8. Type `Closed` enclosed in double quotation marks to complete the qualification:

   ```
   'TR.Status' = "Fixed" OR 'TR.Status' = "Closed"
   ```
9. Click OK to save the qualification.

TIP

To manually type the expression, start by typing a single quote. BMC Remedy Developer Studio pops up a list of available fields. You can also type ‘TR, and then select from the list of available fields. If you enter an expression by hand, make sure to use correct spelling, punctuation, keywords, and quotations.

Example 2—Filter execution based on elapsed time

This example qualification causes the filter to act on requests that are older than 10 hours:

```
'Status History.New.TIME' < $TIMESTAMP$ - (60 * 60 * 10)
```

The qualification searches for requests where the time that the request was created is less than the current time minus 10 hours. This example uses a status history reference as one of its values.

▶ To enter this qualification using the Expression Editor

1. In the filter editor, click the ellipsis button on the Run If Qualification panel.
2. In the Expression Editor, select `Status History.New.TIME` from the Available Fields list, and then click Add Field:

   ```
   'Status History.New.TIME'
   ```

   In this case, New is the Status value that you want to match and TIME is the keyword for the time that the request was entered.
3. Click the less than (<) operator to add it to the end of the expression:

   ```
   'Status History.New.TIME' <
   ```
4. Expand the Keywords list, select $TIMESTAMP$, and then click Add Keyword:

   ```
   'Status History.New.TIME' < $TIMESTAMP$
   ```
5. Click the minus sign (–) operator:

   ```
   'Status History.New.TIME' < $TIMESTAMP$ -
   ```
6. Enter an hour value, for example `(60 * 60 * 10)`:

   ```
   'Status History.New.TIME' < $TIMESTAMP$ - (60 * 60 * 10)
   ```

   See "Double quotation marks" on page 55 for information about entering time values.
7. Click OK to save the qualification.

Example 3—Filter execution based on a changed field value

This example qualification causes the filter to test whether a value has changed in a field, in this case, the Status field:

```
'Status' != 'DB.Status'
```

▶ To enter this qualification using the Expression Editor

1. In the filter editor, click the ellipsis button on the Run If Qualification panel.
2. In the Expression Editor, select `Status` from the Available Fields list and then click Add Field:

   ```
   'Status'
   ```
3. Click the not equal (!=) operator to add it to the end of the expression:

   ```
   'Status' !=
   ```
4. Position the cursor at the end of the line, and then type `‘DB.` (including the period):

   ```
   'Status' != 'DB.
   ```

   Content assist presents the list of available fields, prefixed with `‘DB.`
5. Double-click `‘DB.Status’` to select it from the list:

   ```
   'Status' != 'DB.Status'
   ```
6. Click OK to save the qualification.

Chapter

4 Specifying workflow actions

This section describes how to use BMC Remedy Developer Studio to create and modify all types of workflow actions, including elements common to creating most or all workflow objects, and specific procedures for defining each type of workflow action.

The following topics are provided:

If Actions and Else Actions

In addition to the Execution Options of a workflow action, which determine when the workflow starts, you can use a Run If Qualification to further determine whether the workflow should run, and if so, whether a certain set of actions will occur, based on the conditions tested by the qualification.

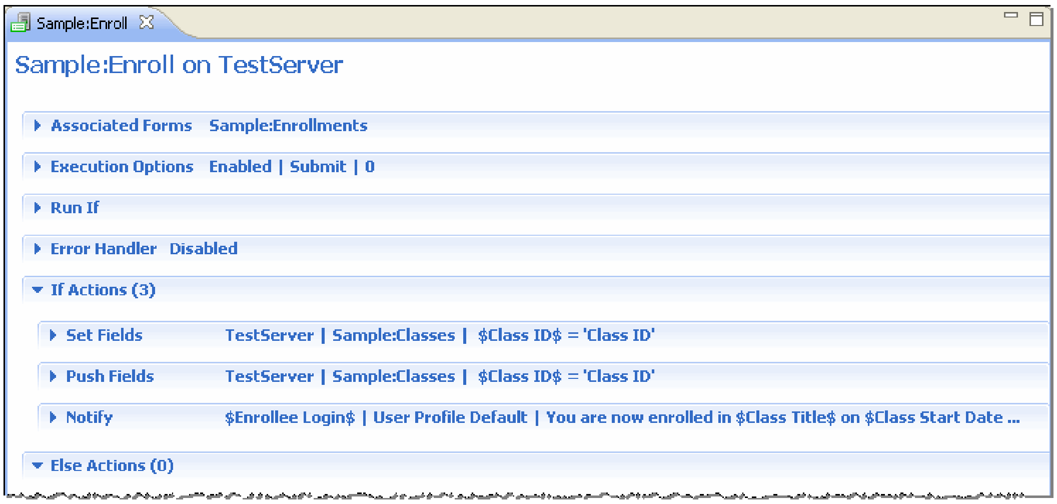
When the workflow starts, AR System evaluates the Run If Qualification. If the qualification is true, the If Actions occur. If the qualification is not true, the Else Actions, if any, occur. If there is no qualification, the If Actions occur. If the qualification is not true and there are no Else Actions, the workflow stops.

You can include any action type in the If Actions or Else Actions of a workflow object. For active links and filters, the If Actions and Else Actions act on the current request. For escalations, the If Actions run on all the matching requests in the form. Escalation Else Actions run once with no associated request. In this case all fields are NULL when the escalation starts.

For any active link, filter, or escalation, you must define at least one If Action. Else Actions are optional. You can define a maximum of 25 If Actions and 25 Else Actions in any workflow object. On the If Actions and Else Actions panel headings, the number in parentheses shows how many actions are defined. For example, Figure 4-1 shows three If Actions and no Else Actions defined in an active link.

Execution conditions and qualifications are always either true or false, so you cannot run a combination of If and Else actions together in the same active link, filter, or escalation.

Figure 4-1: If Actions and Else Actions panel headings



Associated Forms Sample:Enrollments
Execution Options Enabled | Submit | 0
Run If
Error Handler Disabled
If Actions (3)
Set Fields TestServer | Sample:Classes | $Class ID$ = 'Class ID'
Push Fields TestServer | Sample:Classes | $Class ID$ = 'Class ID'
Notify $Enrollee Login$ | User Profile Default | You are now enrolled in $Class Title$ on $Class Start Date ...
Else Actions (0)

TIP

To maximize the workflow editor window when working on an active link, filter, or escalation, double-click the tab for the workflow item. To return to the main BMC Remedy Developer Studio view, double-click the tab again.

Examples of If Actions and Else Actions

If actions execute when the qualification is met. For example:

- An active link qualification checks for membership in an access control group for the user name in the Submitter field. If the user is a member of the group, the If action allows access to a field.
- A filter qualification looks for "hardware" in the Problem Type field and, if found, the If action assigns the request to a hardware support person.
- An escalation qualification looks for any requests more than two hours old that have "New" in the Status field and if found, the If action assigns the matching requests to a manager.

Else actions are optional. They execute when the qualification is not met. For example:

- An active link qualification checks for membership in an access control group for the user name in the Submitter field and, if not found, the Else action denies access to a field.
- A filter qualification looks for "hardware" in the Problem Type field, and, if not present, the Else action assigns the request to a software support person.
- An escalation qualification looks for any requests more than two hours old that have "New" in the Status field. If none are found, the Else action sends an email notification reporting this fact to a shared status log account.

Developing workflow actions in BMC Remedy Developer Studio

You use the workflow editors in BMC Remedy Developer Studio to create, modify, and delete active links, filters, and escalations, as well as active link guides and filter guides. For an overview of the screen areas in BMC Remedy Developer Studio, including the workflow editors, see *Introduction to Application Development with BMC Remedy Developer Studio* guide, "Building applications with BMC Remedy Developer Studio," page 21. For an introduction to the workflow editors, see "Using the workflow editor in BMC Remedy Developer Studio" on page 20.

This section describes the general procedures and common tasks to create, modify, delete, or reorder workflow actions. For specific procedures for each type of workflow action, see the workflow action sections referenced in the table.

▶ To create any workflow action

1 Open a new or existing active link, filter, or escalation.

See the *Introduction to Application Development with BMC Remedy Developer Studio* guide, "Creating objects," page 46, and *Introduction to Application Development with BMC Remedy Developer Studio* guide, "Working with existing objects," page 48.

2 Specify the associated form, the execution options, and the Run If qualification if necessary. See Chapter 2, "Configuring workflow forms and execution options" and Chapter 3, "Building qualifications and expressions."

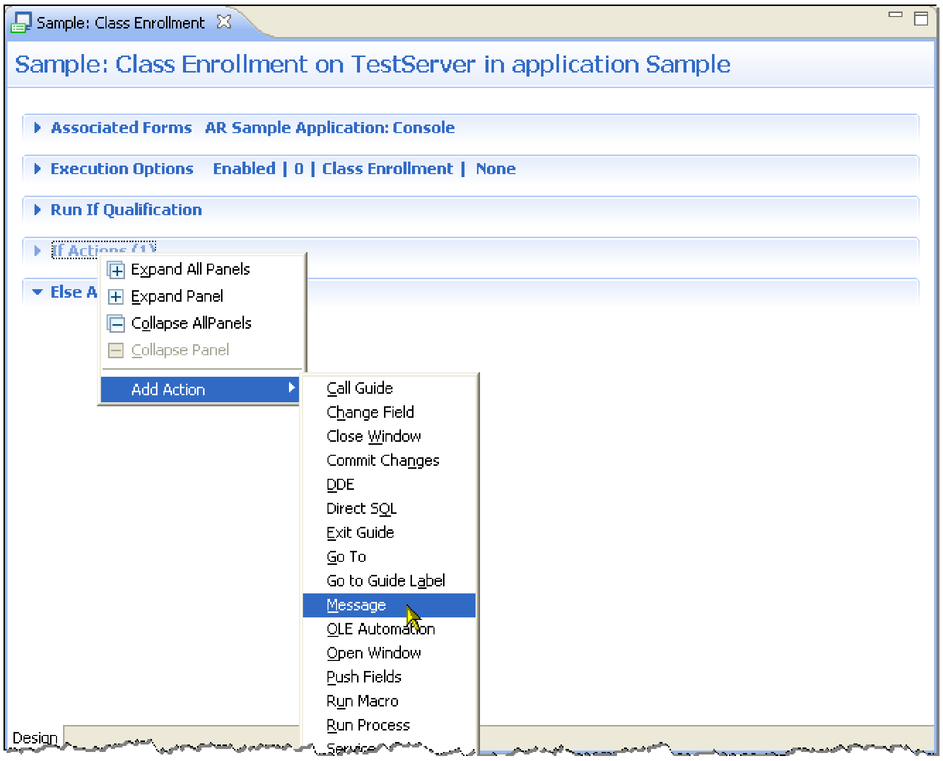
3 Right-click the appropriate action panel:

- **If Action**—Executes when the Run If qualification is met.
- **Else Action**—Executes when the Run If qualification is not met.

4 Choose Add Action > *actionType*.

See "Types of workflow actions" on page 75 for a description of the available action types for active links, filters, and escalations.

Figure 4-2: Adding active link actions


Sample: Class Enrollment on TestServer in application Sample

Associated Forms AR Sample Application: Console

Execution Options Enabled | 0 | Class Enrollment | None

Run If Qualification

BMC Remedy Developer Studio adds the new action below any existing actions.

5 In the new action, define the settings for the action as necessary. See the appropriate section describing each workflow action in this document.

6 To create additional actions, repeat steps 3 through 5 for each new action.

7 Save the active link, filter, or escalation.

   To save an active link, filter, or escalation, use either Ctrl+S, the menu option File > Save, or the Save icon in the icon bar.

▶ To modify a workflow action

1 Open the active link, filter, or escalation.

2 In the Active Link editor, click If Actions or the Else Actions to display the list of existing actions.

3 In the list of actions, click the appropriate action to expand it.

4 Make changes to the action.

5 Save the active link, filter, or escalation.

▶ To delete a workflow action

You can delete all the If Actions of a workflow object, provided you create at least one If Action before attempting to save it. Else Actions are optional, so you can delete all of them if necessary.

1 Open the active link, filter, or escalation.

2 In the Active Link editor, click If Actions or Else Actions to display the list of existing actions.

3 In the list of actions, right-click the action to be deleted, and then select Remove Action from the pop-up menu.

   A confirmation message appears if your preferences are set to display a message.

4 To delete additional actions, repeat step 3 for each action.

5 Save the active link, filter, or escalation.

▶ To change the order of workflow actions

1 Open the active link, filter, or escalation.

2 Click the If Actions or the Else Actions panel to expand the list of actions.

3 Right-click the action to be moved up or down in the list.

4 In the pop-up menu, click Move Action Up or Move Action Down.

5 Repeat steps 3 and 4 until the actions are in the correct order.

6 Save the active link, filter, or escalation.

Using field names, field values, and expressions in workflow

When you are creating workflow, field references almost always refer to the Primary Form defined in the Associated Forms panel. This includes fields used in Execute On options and Run If actions. Fields to be used by workflow must exist on the Primary Form. Fields found on other associated forms are not displayed as selectable values in the field selection dialog for most workflow actions, or as part of the auto complete feature in qualification or field selector dialogs.

An exception to this is when a workflow action references a second form, such as Set Fields and Push Fields actions, where you can select fields from both the primary form and the secondary form, or an Open Window search action, where you might search for data in another form.

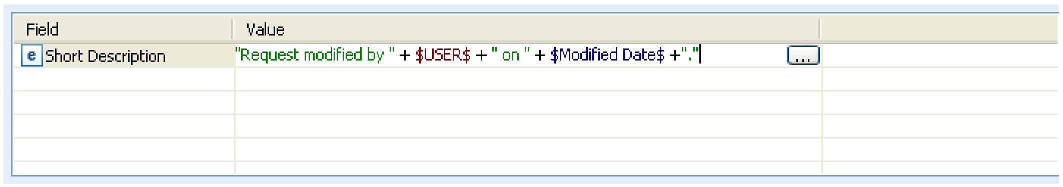
Mapping fields to values

Several workflow action types, including Open Window, Push Fields, Service, and Set Fields, allow you to transfer data between forms by using a field and value mapping. To do this, you use the field mapping table to select the appropriate fields and define the values to be assigned.

Although the purpose of the field mapping varies according to the type of workflow action, the way you enter fields and values in the field mapping table is the same in each case. This section describes the general steps for mapping values to fields by using the field mapping table.

Figure 4-3 shows an example of the field mapping table from a Set Fields action. In this example, the expression in the Value column defines the value to be set in the Short Description field by this Set Fields action.

Figure 4-3: Example field mapping table



Field	Value
Short Description	"Request modified by " + $USER$ + " on " + $Modified Date$ + "."

TIP

When editing a field mapping table, you can use the Tab key and arrow keys to move between cells in the table.

Selecting a field with the Field Selector dialog box

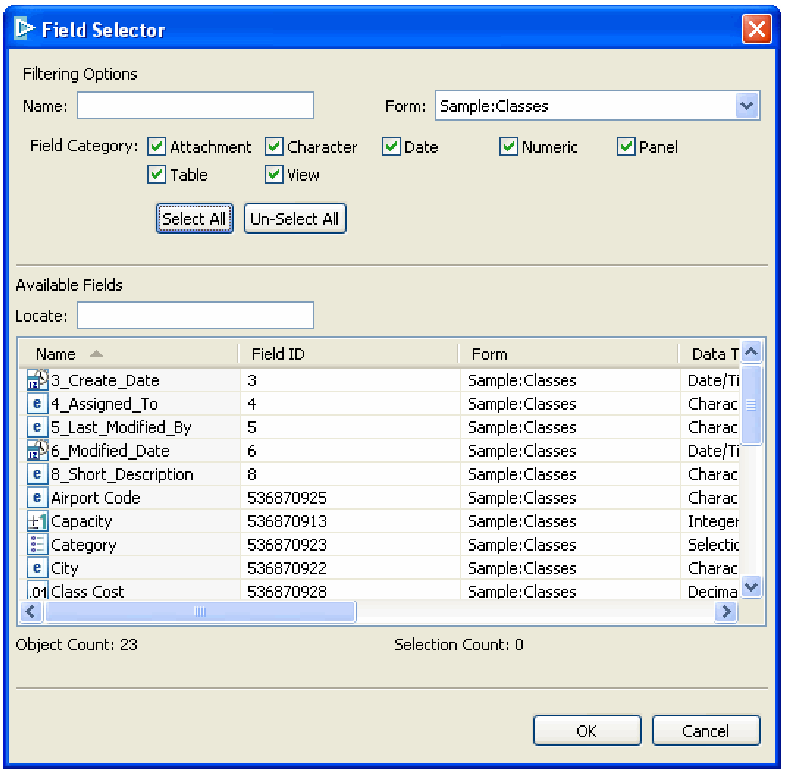
To select a field from the list of all fields contained in the associated form, use the Field Selector dialog box. This dialog box is available in BMC Remedy Developer Studio any time you need to select a field while defining workflow.

Figure 4-4 shows the Field Selector dialog box. In this case it was opened from the active link `Sample: Class Search` in the AR System sample application, so the associated form is the Sample:Classes form.

For a form that contains a large number of fields, you can use one of the methods in the Filtering Options or Available Fields areas to locate the appropriate field:

- **Name**—To locate a field by name, begin to type the name in the Name field of the Field Selector dialog box. The list narrows to display only field names beginning with the characters you type.
- **Field Category**—To locate a field by field type, select the checkbox for that type.
- **Locate**—To locate the field name in the list while still displaying all the field names, begin to type the field name in Locate field.

Figure 4-4: Field Selector dialog box



Filtering Options

Name:

Form: Sample:Classes

Field Category: ✓ Attachment ✓ Character ✓ Date ✓ Numeric ✓ Panel ✓ Table ✓ View

Available Fields

Locate:

Name	Field ID	Form	Data T
3_Create_Date	3	Sample:Classes	Date/Ti
4_Assigned_To	4	Sample:Classes	Charac
5_Last_Modified_By	5	Sample:Classes	Charac
6_Modified_Date	6	Sample:Classes	Date/Ti
8_Short_Description	8	Sample:Classes	Charac
Airport Code	536870925	Sample:Classes	Charac
Capacity	536870913	Sample:Classes	Integer
Category	536870923	Sample:Classes	Selectic
City	536870922	Sample:Classes	Charac
Class Cost	536870928	Sample:Classes	Decima

Object Count: 23

Selection Count: 0

▶ To add a field to the table with the Field Selector dialog box

1 Click in an empty row of the Field column.

An ellipsis button appears.

2 Click the ellipsis button.

3 In the Field Selector dialog box, select the field to add to the field mapping table.

The fields listed depend on the associated forms for the action and the type of mapping appropriate for the action.

Adding a value to the field mapping table

In the value column, you can enter a field name, keyword, static value, or expression. Use the expression editor select field names and keywords, or to create an expression. You can also type the expression in the field using the content assist feature.

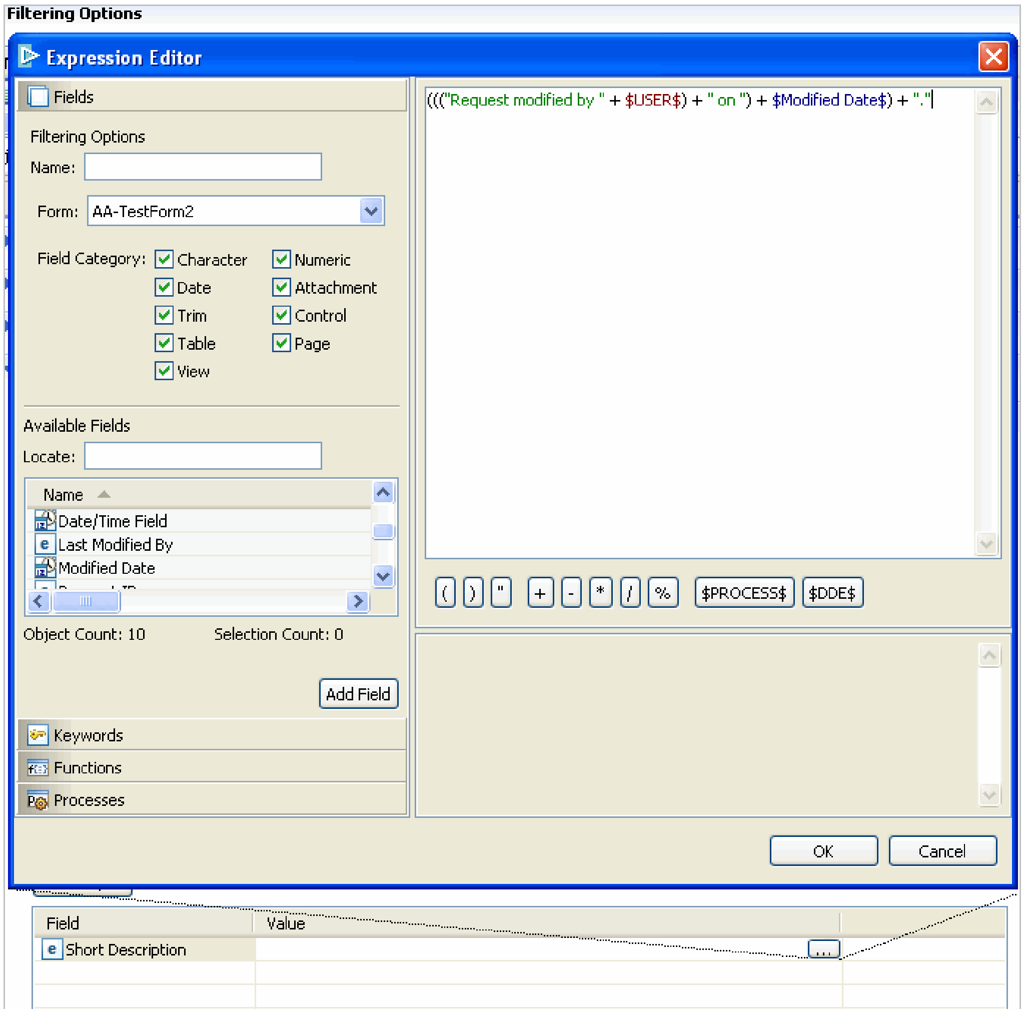
▶ To define a value for the field by using the expression editor

1 After entering a field in the Field column, click the same row in the Value column.

2 Click the ellipsis button that appears in the Value cell.

The Expression Editor dialog box opens. Figure 4-5 shows an example of the Expression Editor dialog box in a Set Fields action, with a value being defined for the Short Description field.

Figure 4-5: Expression Editor dialog box, Set Fields example


(("Request modified by " + $USER$) + " on ") + $Modified Date$) + "."

Filtering Options
Name:
Form: AA-TestForm2
Field Category: ✓ Character ✓ Numeric ✓ Date ✓ Attachment ✓ Trim ✓ Control ✓ Table ✓ Page ✓ View

Available Fields
Locate:

Name
Date/Time Field
Last Modified By
Modified Date

Object Count: 10 Selection Count: 0

Field	Value
Short Description	

3 In the Expression Editor dialog box, build an expression to define the value:

- To use a value from another field in the current request, select the field from the Available Fields list.
- To use the current value of a keyword, select it from the Keywords list.
- To set a static value in the field, enclose the value in quotes in the expression.

  For example, to set a field value that tracks what user modified the form and when, create an expression like the following:

  ```
  "Request modified by " + $USER$ + " on " + $Modified Date$ +"."
  ```

  In this example expression, "`Request modified by` " and " `on` " are static values, `$USER$` is an AR System keyword that contains the current user's login ID, and `$Modified Date$` is a field in the current request. The "+" signs are operators that concatenate the parts of the expression.
- In Set Fields, Service, and Push Fields actions, you can also use the results of data operations in the expression. For more information, see the descriptions of those workflow types in this document.

4 When the expression is complete, click OK to close the Expression Editor dialog box and enter the expression in the field mapping table.

▶ To define a value for the table by using the content assist feature

1 After entering a field in the Field column, click the same row in the Value column.

2 Begin typing the expression. For example, to select a field name, type a single quote (') or press Ctrl+Space.

NOTE

When you map fields, BMC Remedy Developer Studio does not automatically validate the field types or prevent you from mapping fields of different types. If you map fields of different types to each other, AR System converts the values at runtime.

Using a dynamic data source or destination in workflow

For several action types, instead of hard-coding a source or destination form when you create the workflow (at design time), you can configure the workflow to use a form determined when the workflow runs (at runtime). For active links, you can also configure the workflow to determine the server where the form resides at runtime. To do this, use the SAMPLE DATA data source or data destination when you create the workflow.

To identify the form and server to use at runtime, you use fields in the form associated with the workflow or keywords. In the workflow object, these are called *runtime values*. To map fields and values in the workflow object, you use a *sample* form and server.

The following actions can use a dynamic data source or destination:

Action	SAMPLE DATA data type
Call Guide	Source
Open Window	Source
Push Fields	Destination
Service	Source
Set Fields	Source

Configuring a dynamic data source

This section provides general information about configuring a dynamic data source. For information about configuring a dynamic data source for a particular action, see the section in this guide that describes that action.

NOTE

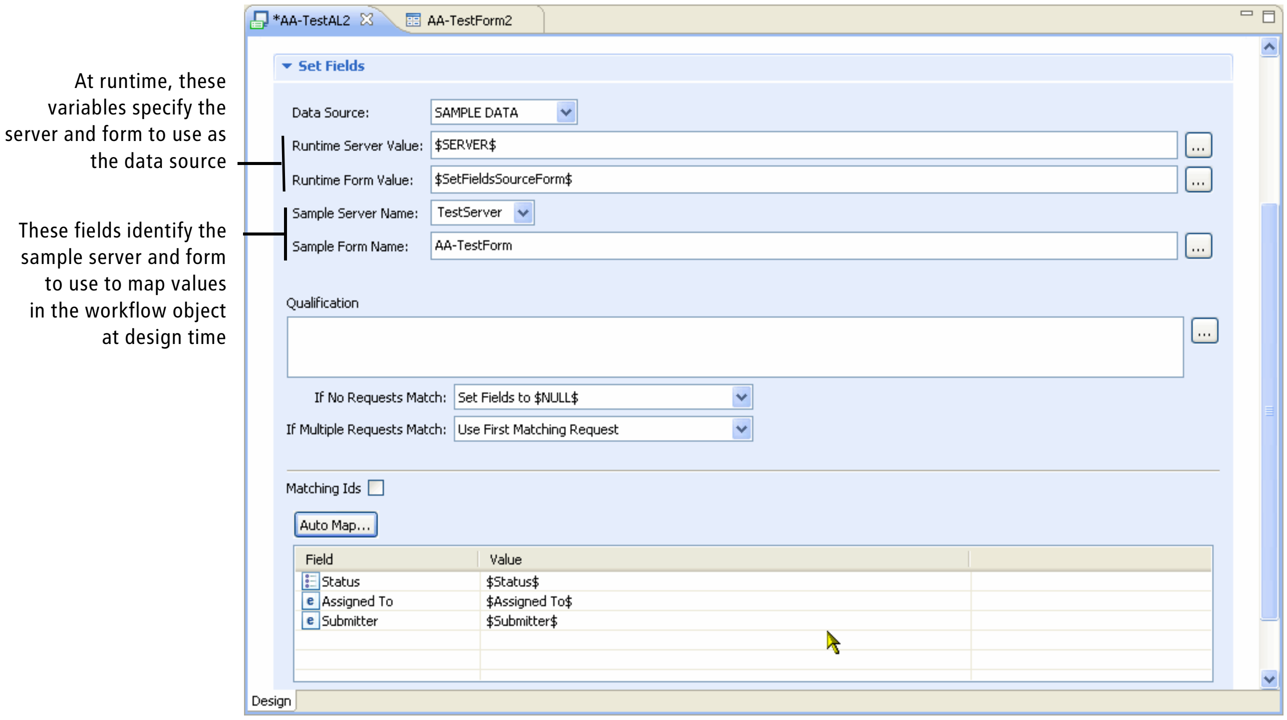
For information about configuring a dynamic data *destination*, see "Push Fields action" on page 119.

When you select the SAMPLE DATA data source in the workflow editor, the fields in the editor change to enable you to specify the following items:

- Variables that identify a runtime server and form
- A sample server and form

Figure 4-6 shows an example of a Set Fields action being configured to use SAMPLE DATA.

Figure 4-6: A Set Fields action with SAMPLE DATA as the data source


At runtime, these variables specify the server and form to use as the data source
These fields identify the sample server and form to use to map values in the workflow object at design time
*AA-TestAL2
AA-TestForm2
Set Fields
Auto Map...
Design

Data Source: SAMPLE DATA
Runtime Server Value: $SERVER$
Runtime Form Value: $SetFieldsSourceForm$
Sample Server Name: TestServer
Sample Form Name: AA-TestForm
Qualification
If No Requests Match: Set Fields to $NULL$
If Multiple Requests Match: Use First Matching Request
Matching Ids ☐

Field	Value
Status	$Status$
Assigned To	$Assigned To$
Submitter	$Submitter$

▶ To configure a SAMPLE DATA data source in a workflow action

1. Right-click the If Actions or Else Actions panel header.
2. Choose Add Action > *actionType*.
3. In the Data Source list, select SAMPLE DATA.
4. (Active links only) In the Runtime Server Value field, click the ellipsis button.
5. In the Field / Keyword selector dialog box, select one of these items:
   - The field on the primary form that will supply the source server name, for example, `$WorkflowServer$`.
   - If the source server is to be the user's current server, the `$SERVER$` keyword.
6. Click OK.
7. In the Runtime Form Value field, click the ellipsis button.
8. In the Field / Keyword selector dialog box, select one of these items:
   - The field on the primary form that will supply the source form name, for example, `$WorkflowForm$`.
   - A keyword to supply the source form name, such as `$SCHEMA$` or `$SCHEMA-ALIAS$`.

9 Click OK.

TIP

You can also use content assist to enter a field name in the Runtime Server Value and Runtime Form Value fields. To do so, enter a dollar sign ($) or press Ctrl+Space in the field. A list of fields in the primary form appears; select the appropriate field. If you enter field names manually, use variable format (for example, `$WorkflowServer$`).

10 (Active links only) In the Sample Server Name field, select the server that has the sample form to use at design time.

11 In the Sample Form Name field, select the sample form that contains the fields to map in the field mapping table.

12 Continue designing the workflow by completing the remaining fields in the workflow editor.

For workflow that uses a Run If Qualification, you can use fields from both the primary form and the sample form in the qualification. The field types and field IDs on the sample form and the corresponding runtime form must match.

When mapping fields to values in the field mapping table, the fields you can select in the Values column come from the form identified in the Sample Form Name field. If fields with the same IDs on the sample and runtime forms have different data types, the general conversion rules apply. See “Server-side table field implementation issues” on page 164.

Example of workflow with a dynamic data source

This example describes a Set Fields action in an active link that uses the SAMPLE DATA data source. The SetFldTarget form is the form associated with the active link. It contains two fields, RunTimeServer and RunTimeForm. The values in these fields, entered by the user or set by workflow, determine at runtime which server and form are used as sources in the Set Fields action. (When these values are set by workflow, these fields can be hidden or read-only.)

In this example, a record exists in the SourceFormA form on the Calbro1 server. The Item Description field in this form contains the value `Test Data`. When a user clicks a button on the SetFldTarget form that triggers the active link, the active link determines that the source server is Calbro1 and the source form is SourceFormA. Based on the field mappings in the active link, the workflow obtains the value `Test Data` from the Item Description field in the appropriate record in SourceFormA on Calbro1 and sets that value into the Short Description field on the SetFldTarget form.

Types of workflow actions

Table 4-1 lists the workflow actions, shows which workflow types can use the action, and provides a cross-reference to more information about the action.

Some active link actions are only for use in BMC Remedy User and are not supported by the web client. These are identified in the table with "BMC Remedy User only." (For information about client compatibility for active links that use the Run Process action, see "Run Process action" on page 124.)

Table 4-1: Using workflow actions in active links, filters, and escalations

Action type	Active links	Filters	Escalations	Page or book
Call Guide	+	+		page 76
Change Field	+			page 77
Close Window	+			page 81
Commit Changes	+			page 82
DDE (BMC Remedy User only)	+			*Integration Guide*
Distributed Server Option		+	+	*BMC Remedy Distributed Server Option Guide*
Direct SQL	+	+	+	page 83
Exit Guide	+	+		page 86
Go to Guide Label	+	+		page 86
Goto	+	+		page 87
Log to File		+	+	page 88
Message	+	+		page 88
Notify		+	+	page 95
OLE Automation (BMC Remedy User only)	+			*Integration Guide*
Open Window	+			page 102
Push Fields	+	+	+	page 119
Run Macro (BMC Remedy User only)	+			page 123
Run Process	+	+	+	page 124
Service	+	+	+	page 127
Set Fields	+	+	+	page 129
Wait (BMC Remedy User only)	+			page 136

The following sections describe how to define workflow actions. These procedures are the same whether you create If Actions or Else Actions.

Call Guide action

Use the Call Guide action in:

- Active links
- Filters

The Call Guide action starts a filter guide or an active link guide that is not an entry point. For example, you might have a guide that runs a series of active links to get information from the user, or a guide that walks a table field when the user clicks a button on a form. To start that guide, you create an active link that contains a Call Guide action. For more information about active link guides and filter guides, see "Defining guides and guide actions" on page 137.

Active links and filters that execute in the context of guides are triggered by the guide and not by their Execute On conditions. When a guide executes an active link or filter, any Execute On conditions are ignored. The Call Guide action triggers the first or the specified active link or filter in the guide and every active link or filter that follows.

You can use the Call Guide action to "loop" through the rows of a table field (also known as "table walk"). For more information, see "Using active link guides in client-side table fields" on page 151.

To define a Call Guide action

1. Right-click the If Action or the Else Action panel header.
2. Choose Add Action > Call Guide.
3. From the Data Source list, select SERVER or SAMPLE DATA:
   - To call a guide that exists on a server to which you are logged in, select SERVER, and go to step a.

     For filters, you cannot select a different server. The called guide must exist on the current server.
   - To enable the server (active links only) and guide to be specified at runtime, select SAMPLE DATA, and go to step b.

   a (*Data Source is SERVER*) Select the Server (active links only) where the guide exists and the Guide Name to call.
   - Servers that appear in the Server Name list are those to which you are currently logged in.
   - To select the Guide Name, click the ellipsis, and then select from the list of available guides.

   For active links, the Available Active Link Guides list includes all active link guides on the server you selected. For filters, the Available Filters List includes all filters on the current server.

   For active link guides, if the guide being called is not connected to the form where the active link is running, a new window opens.

b (*Data Source is SAMPLE DATA*) Define the variables that will determine the server and guide name at runtime:

- For active links, enter an expression in the Runtime Server Value field to determine the server name at runtime.
- In the Runtime Guide Value field, enter an expression to determine the guide name at runtime.

For example, enter the keyword variable `$SERVER$` in Runtime Server Value and the field variable `$GuideName$` in Runtime Guide Value. Use the typing assistant or click the ellipsis and use the Field / Keyword Selector dialog box.

For more information about using the SAMPLE DATA data source, see "Using a dynamic data source or destination in workflow" on page 71.

4 In the Table Loop field, select the appropriate option:

- If the Call Guide action is not a table-loop action, select None.
- If this Call Guide action will loop through the rows of a table, select All Rows or Selected Rows Only (active links only).

  For active link guides, the Selected Rows Only option can significantly improve the performance of table loop guides because the looping action is performed only on the selected rows, and not on every row in the table field.

  For information about using table loop guides, including additional performance tips, see "Using active link guides in client-side table fields" on page 151 and "Using a filter guide to loop through a table field" on page 160.

5 Save the active link or filter.

WARNING

AR System does not allow a Call Guide action that calls the guide itself. However, AR System allows recursive call guides; for example, CallGuide1 calls CallGuide2, and then CallGuide2 calls CallGuide1. You can create this scenario, but be aware that an infinite loop can occur. If this happens, BMC Remedy User hangs and the mid tier generates a Sun™ JavaScript™ error.

Change Field action

Use the Change Field action in:

- Active links

The Change Field action changes specified characteristics of fields in the current window. (This action does not change field properties in the database.)

Based on conditions when the active link executes and on the field type, you can:

- Change the field access type.
- Change the label font.
- Hide or display the field.

- Expand or collapse tree levels.
- Expand or collapse a panel field.
- Specify a different menu for a character field.
- Change the field label.
- Change the color of the field label. (Not supported for tree view table fields.)
- Change the field entry mode to Required.
- Move the focus (cursor) to the field.
- Refresh a list view or tree view table field.

▶ To define a Change Field active link action

1. Right-click the If Action or the Else Action panel header.
2. Choose Add Action > Change Field.
3. In the Field list, select Field Name or Field Value:
   - **Field Name**—Click the ellipsis button. In the Field Selector dialog box, select the field whose characteristics you want to change.
   - **Field Value**—Select a field whose runtime value will be the name of the field to change. This option enables you to select the field to change dynamically by using workflow. See "Creating a dynamic Change Field action" on page 79
4. Set the following field attributes as necessary.

   If you selected Field Name in step 3, the list of attributes is determined by the selected field's type.

   If you selected Field Value, all field characteristics appear. Attributes that are not applicable to the field type selected at runtime are ignored.

   To leave an attribute unchanged, select Unchanged (the default).

   - **Field Access**—Changes the Field Access field property to Read Only, Read / Write, or Disabled (for data fields) or to Enabled or Disabled (for trim, control, and tree view table fields).
   - **Field Font**—Changes the field label font to the specified AR Fonts style, such as Edit Field or Header Text (I). Not supported for tree view table fields.
   - **Field Visibility**—Changes the visibility of a field. To show a hidden field, select Visible. To hide a visible field, select Hidden.

     To be made visible, the field must be in the form view. If it is not in the current view, changing visibility has no effect. Not supported for tree view table fields.
   - **Expand Collapse Tree Levels**—Expands (Expand All Levels) or collapses (Collapse All Levels) all levels of a tree view table field.

     Collapse All Levels closes all levels of the tree view so that only the root node and the first level are displayed. If the root node is a level (that is, it contains non-collapsible items), the tree is not collapsed.

- **Expand Collapse Panel**—Opens or closes a panel field (a panel in a panel holder.)
- **Menu**—Changes the menu attached to a field: Select Changed, and either enter the name of the new menu or click the ellipsis button to select the new menu from the Menu / Keyword Selector.

  If a menu is not attached to the field, you cannot change to a different menu.

  To hide the menu icon, select the `$NULL$` keyword from the Menu / Keyword Selector. This allocates space on the form for the icon. If you later use the Change Field action to attach a different menu, the menu icon reappears without affecting the form layout. For information about creating menus, see the *Form and Application Objects Guide*, "Creating menus," page 299.
- **Field Label**—Changes the field label: Select Changed, and either enter a new label in the text box or click the ellipsis button and select a field reference from Field Selector. If you leave the text box empty, the field label appears blank.

WARNING

Changing field labels can affect the layout of the form. For example, long labels might be truncated or the field might change position on the form. Leave enough room around the field to handle the changed label size.

- **Label Color**—Changes the field label color: Select Custom, and select a color from the color palette. To reset the color to its default setting, select Default.
- **Field Drag**—Enables or disables a field configured to be dragged (the source field). See "Allowing data to be dragged and dropped" on page 290.
- **Field Drop**—Enables or disables a field configured to be dropped on (the target field). If you use this action to disable a field, the field is not highlighted when users perform drag operations. See "Allowing data to be dragged and dropped" on page 290.
- **Process Entry Mode**—Changes a nonrequired field to a required field (Required) or reinstates the field's Optional entry mode (Not Required).
- **Set Focus to Field**—When selected, moves the focus (cursor) to the field.
- **Refresh Tree/Table**—When selected, refreshes a list view or tree view table field.

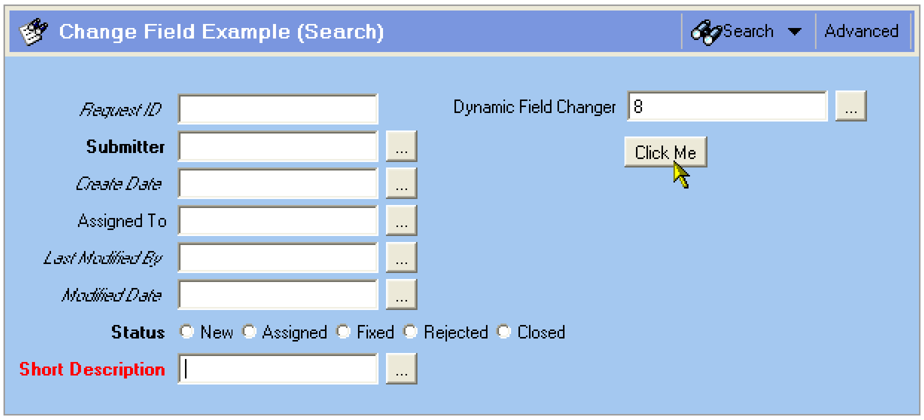
5 Save the active link.

Creating a dynamic Change Field action

To determine the field to be changed at runtime, use the Field Value option when creating the Change Field action, and supply the field or keyword that will identify the field ID to be changed at runtime. To allow the workflow to obtain this value from a field, add the field to the associated form. (If workflow will set the value in this field, it can be a hidden or display only field.)

For example, the Change Field Example form shown in Figure 4-7 contains a field named “Dynamic Field Changer.” At runtime, the user or the workflow enters a field ID in this field, in this case, field ID 8 (the Short Description field). When the user clicks the Click Me button, the label on the Short Description field turns red and the field receives focus.

Figure 4-7: Using dynamic workflow to change field attributes



Change Field Example (Search)

Request ID	
Submitter	
Create Date	
Assigned To	
Last Modified By	
Modified Date	
Status	○ New ○ Assigned ○ Fixed ○ Rejected ○ Closed
Short Description	
Dynamic Field Changer	8

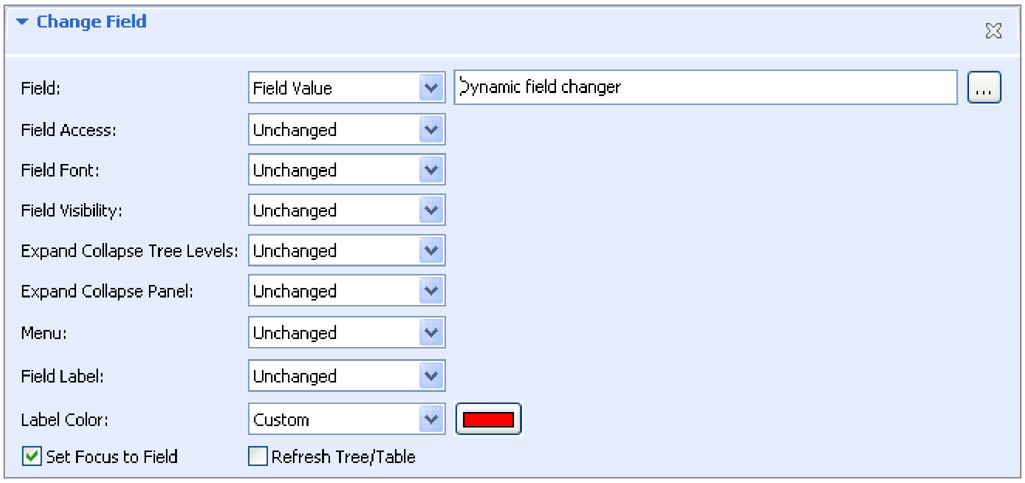
IMPORTANT

For forms viewed in the web client, do not use a dynamic Change Field action that makes a field visible; otherwise, performance is degraded significantly.

▶ To define a dynamic Change Field action

1 Select Field Value in the Change Field action, and then click the ellipsis button.

Figure 4-8: Defining the field to change by a using field value



Change Field

Field:	Field Value — Dynamic field changer
Field Access:	Unchanged
Field Font:	Unchanged
Field Visibility:	Unchanged
Expand Collapse Tree Levels:	Unchanged
Expand Collapse Panel:	Unchanged
Menu:	Unchanged
Field Label:	Unchanged
Label Color:	Custom 🟥

[✓] Set Focus to Field   [ ] Refresh Tree/Table

2 In the Field / Keyword Selector dialog box, select the field or keyword (such as `$FIELDNAME$`) that will identify the field holding the field ID to change at runtime.

3. Specify the field attributes to change with this action.

   In this example, the field label color is changed to red and the focus is set when the workflow is executed. All other attributes are unchanged.

4. Save the active link.

Close Window action

Use the Close Window action in:

- Active links

The Close Window action closes the current or all open windows. This action only closes windows, and does not save updated values or push them to the parent window. If the window to be closed is a dialog box and you want to write the data from the dialog box to the parent form, use the Close Window action in combination with the Open Window and Commit Changes action, as follows:

- In the Open Window active link action (page 102) specify the data to be transferred to the parent form with the Field Mapping Mode set to On Close.
- Call the Commit Changes active link action (page 82) before the Close Window action to save the specified data to the parent form.

You can create buttons to implement these actions. See "Using buttons to control window close actions" on page 82.

▶ To define the Close Window active link action

1. Right-click the If Action or the Else Action panel header.
2. Choose Add Action > Close Window.
3. From the State list, select Close Current or Close All:
   - **Close Current**—Only the current window the active link is running from closes when the active link executes.
   - **Close All**—All open windows close when the active link executes.
4. Save the active link.

Using buttons to control window close actions

When developing a dialog box, create buttons that predictably exit the dialog box and accomplish the actions that you want to occur. To do so, add a button field to the display-only dialog box form, and then create active links that execute when the button is clicked. For example, create any or all of the following buttons:

- **OK**—Use the Commit Changes action followed by the Close Window action to transfer field values from the dialog box to the parent form, and then close the dialog box.
- **Cancel**—Use the Close Window action to close the dialog box without transferring any data to the parent form.
- **Apply**—Use the Commit Changes action to transfer field values from the dialog box to the parent form, but leave the dialog box open.

If you do not create these buttons, the user can only exit the dialog box by clicking the X icon in the top right corner of the dialog box. Clicking the X icon results in unsaved changes and might bypass other workflow actions that should occur.

For more information about using a display-only form as a dialog box, see the *Form and Application Objects Guide*, "Using a display-only form as a dialog box," page 158.

Commit Changes action

Use the Commit Changes action in:

- Active links

The Commit Changes action two has functions:

- When used with a dialog box, Commit Changes works with the Open Window and Close Window actions to capture the data entered in the dialog box. In this case, the Commit Changes action pushes predetermined values from the dialog box to fields on the parent form, but does not initiate a save to the database.
- When used with a regular form, join form, view form, or vendor form, the Commit Changes action applies the changes in the form and performs the major form action (for example, Submit, Search, or Modify). In this case, the Commit Changes action does submit data to the database.

▶ To define a Commit Changes action

1. Right-click the If Action or the Else Action panel.
2. Choose Add Action > Commit Changes.

   The Commit Changes action is added to the If Action or Else Action list.
3. Save the active link.

Using Commit Changes with a dialog box

In workflow, you can use a display-only form as a dialog box to capture user input. To do this, you use the Commit Changes action to transfer the data back to the parent form, in combination with the Open Window and Close Window actions, as follows:

- In an Open Window action with the Dialog window type, map the data to be written in the field mapping for the On Dialog Close Action. See "Open Window action" on page 102.
- Use the Commit Changes action before the Close Window action to write the data entered in a dialog box to the parent form.
- Use the Close Window action to close the dialog box. See "Close Window action" on page 81.

DDE action

Use the DDE action in:

- Active links

The DDE active link action allows data to be exchanged between a client and server using DDE. See the *Integration Guide*, "Using active links with DDE," page 290.

Direct SQL action

Use the Direct SQL action in:

- Active links
- Filters
- Escalations

Use the Direct SQL action to submit any legal SQL command to a non-AR System database. Use this command only if it is required for integration with another database.

The Direct SQL action has a different result than the SQL command you use to set a field value in the Set Fields action. In the Set Fields action, you use SQL to query the database for information and then use the returned value to set a field. With the Direct SQL action, the SQL command is not expected to return data. For example, you can define a Direct SQL action to perform inserts or updates to a non-AR System database.

WARNING

BMC does not support or recommend using a Direct SQL action to modify data in the tables created by the AR System server to store object definitions or form data. It can result in data corruption. Use this action to push data *only* to non-AR System database tables.

You must know SQL syntax and concepts to create a Direct SQL action. For the most effective use of direct SQL commands, you must also have a general understanding of relational databases and a specific understanding of the relational database underlying your AR System.

AR System passes direct SQL commands to the database without checking the syntax. You must make sure that all submitted commands achieve the needed result. Your SQL commands should comply with ANSI SQL standards, so that single quotes are reserved for strings and double quotes are reserved for use with database object names only.

SQL commands can be generic or specific to the DBMS. If an SQL command is specific to one DBMS, you might not be able to move the workflow object that contains it to another environment.

Making sure your SQL commands are secure

Use care when configuring the Direct SQL action. The AR System server uses double quotation marks to substitute single quotation marks unless the entire command is a substitution parameter where you leave the value as is. The following rules explain this concept:

- If parameters are used when constructing the SQL command, the AR System server substitutes the values and escapes single quotation marks by doubling the quotation marks.
- If you enter the entire SQL command in a field and access the command with the `$Field$` reference, the AR System server does not examine or alter the contents of the command, and any single quotes in the SQL command are not escaped. If you use this technique, make sure the command can safely run on the database.

WARNING

BMC does not recommend the second method as this can leave your system vulnerable to hackers with malicious and harmful intent.

▶ To define the Direct SQL active link, filter, or escalation action

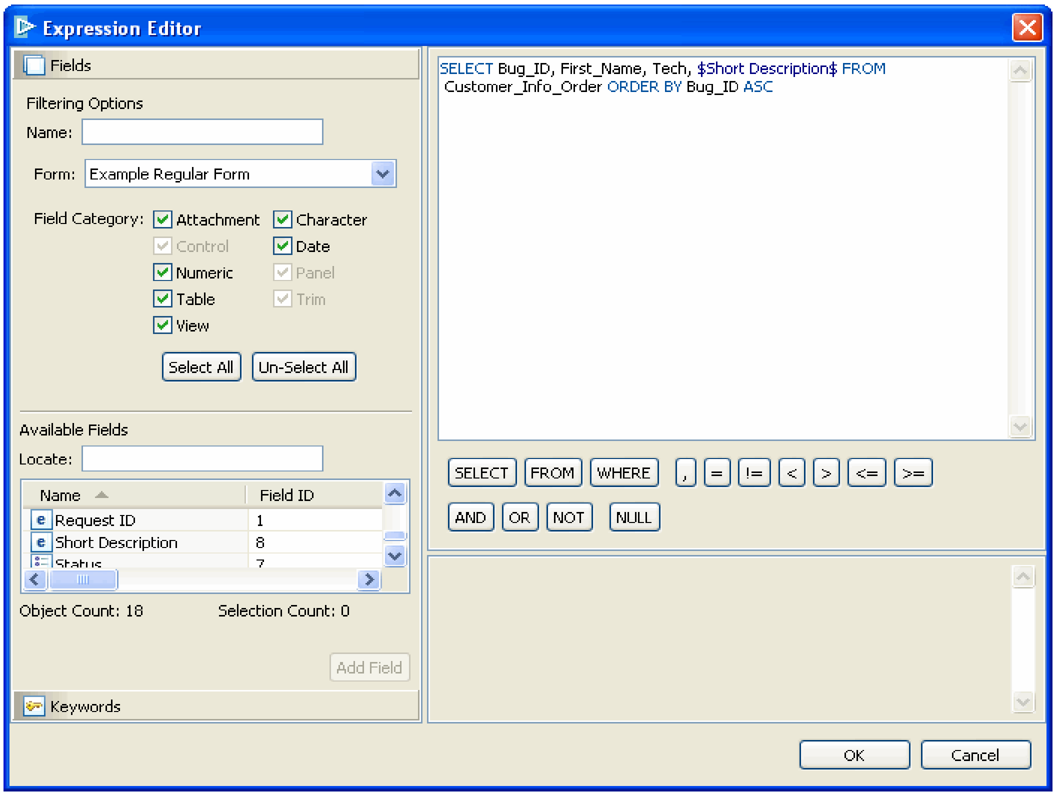
1. Right-click the If Action or the Else Action panel header.
2. Choose Add Action > Direct SQL.
3. From the Server list, select the AR System server where you want to perform the SQL command.

4 In the SQL Command field, enter the SQL command that you want to submit to the database.

To use the Expression Editor dialog box to build the SQL command, click the ellipsis button. As shown in Figure 4-9, the Expression Editor enables you to choose SQL commands, fields from the form, and AR System keywords.

5 Save the active link, filter, or escalation.

Figure 4-9: Expression Editor dialog box for Direct SQL action



```
SELECT Bug_ID, First_Name, Tech, $Short Description$ FROM
Customer_Info_Order ORDER BY Bug_ID ASC
```

In this example expression, the elements `Bug_ID`, `First_Name`, and `Tech` are columns in the table `Customer_Info_Order`. The contents of the `$Short Description$` field will be used to determine the name of the last column.

DSO action

Use the DSO action in:

- Filters
- Escalations

The DSO action enables you to configure DSO Transfers, DSO Returns, and DSO Deletes in workflow. See the *BMC Remedy Distributed Server Option Guide*, "Creating workflow to perform DSO operations," page 72.

Exit Guide action

Use the Exit Guide action in:

- Active links
- Filters

The Exit Guide action terminates the current guide and, optionally, all of its calling guides. Both active link and filter guides exit by default when they reach they have executed all the active links or filters contained in the guide. However, you use the Exit Guide action to terminate the guide under a specific circumstance.

For example, in a guide with two or more mutually exclusive workflow paths, use the Exit Guide action to end the guide at the end of each possible path.

For more information about active link guides and filter guides, see "Defining guides and guide actions" on page 137.

▶ To define an Exit Guide active link or filter action

1. Right-click the If Action or the Else Action panel header.
2. Choose Add Action > Exit Guide.
3. Select or clear the Close All Guides on Exit check box. If this check box is:
   - **Selected**—All running guides exit when the active link or filter is executed.
   - **Cleared**—Only the current guide exits when the active link or filter is executed. (Default)
4. Save the active link or filter.

Go to Guide Label action

Use the Go to Guide Label action in:

- Active links
- Filters

The Go to Guide Label action redirects the flow of execution within a guide to a specific location in the list of active links or filters. You mark these locations in the guide by inserting a guide "label" before the active link or filter that begins each unit of workflow. The Go to Guide Label action directs the workflow to the specified label, and the guide then executes the active links or filters that follow the label.

This action allows you to create modular workflow, so that you can repeat a series of workflow steps or create workflow can take any of several paths, controlled by qualifications.

For more information about active link guides and filter guides, see "Defining guides and guide actions" on page 137.

▶ To define the Go to Guide Label active link or filter action

1. Right-click the If Action or the Else Action panel header.
2. Choose Add Action > Go to Guide Label.
3. In the Guide Label field, type the name of the guide label.
4. Save the active link or filter.

Goto action

Use the Goto action in:

- Active links
- Filters

Use the Goto action to execute an active link or filter in an order other than the normal sequence determined by the active link or filter execution order. When a Goto action executes, the first item with an execution order greater than or equal to the number that you specify executes next. You can specify a static or variable execution order value.

Like the Goto Guide Label action, the Goto action allows you to control the order of workflow. However, the Goto action does not require the workflow to be organized into a guide. The Goto action can be useful when you want to execute a sequence multiple times. For each time that you want to execute an active link or filter, a Goto action returns you to the active link or filter with which you want to begin.

▶ To define the Goto active link or filter action

1. Right-click the If Action or the Else Action panel header.
2. Choose Add Action > Goto.
3. In the Type field, select one of the following options:
   - **Execution Order**— Specify a static execution order.

     Enter the execution order in the Goto Execution Order field. You can specify an execution order up to 1000. If you specify an execution order of 0, the first active link that meets the execution conditions executes.

   NOTE

   BMC Remedy Developer Studio does not prevent you from entering an execution order higher than 1000. However, any execution order of 1001 or more is interpreted as 1000.

   - **Execution Order from Field**—Use an execution order that is set at runtime.

     In the Field With Execution Order field, type or select the field that will determine the execution order. This field must be an integer field.
4. Save the active link or filter.

Log to File action

Use the Log to File action in:

- Filters
- Escalations

Use the Log to File action to append all operations meeting the filter or escalation conditions to a text file on the AR System server. Each log entry includes:

- A date and time stamp.
- The name of the user whose action triggered the filter.
- The name of the form.
- The request ID number.
- The fields included in the transaction.

Escalations support Log to File If Actions, but do not support Log to File Else Actions.

▶ To define the Log to File filter or escalation action

1. Right-click the If Action or the Else Action (filters only) panel header.
2. Choose Add Action > Log to File.
3. In the File Name field, enter the full directory path of the log file. To browse to an existing location, click Browse.
4. Save the filter or escalation.

NOTE

If a failure occurs when writing to the log file that you specify, AR System records a warning message in the `arerror.log` file. This warning message appears only once, not for every time logging fails. The next time the system successfully writes to the log file that you specified, a message is also written to the `arerror.log` file.

Message action

Use the Message action in:

- Active links
- Filters

Use the Message action to display an error, warning, note, or message for any operation that meets the active link or filter conditions. You define the content of the message and set options to control where the message appears.

▶ To define the Message active link or filter action

1 Right-click the If Action or the Else Action panel header.

2 Choose Add Action > Message.

3 From the Type list, select the appropriate message type.

The message type is displayed as a label with the message. The available message types are:

- **Note**—Displays a note and continues the operation. The message is labeled ARNOTE.
- **Warning**—Displays a warning and continues the operation. The message is labeled ARWARN.
- **Error**—Displays an error and terminates the operation. No actions are performed after an error message. The message is labeled ARERR.

  For example, you can use an error message to disallow requests with no value in a specific field, even if the field is not normally a required field.

TIP

In a filter, you can use an error handling filter instead of displaying a message and terminating the filter. See "Error processing" on page 160.

- **Prompt**—Active links only. In BMC Remedy User, displays a message in the Prompt Bar and continues. See "Using the Prompt Bar" on page 93.

  This option is not supported in the web client.
- **Accessible**—Active links only. Displays a message that can be interpreted by a screen reader for a visually impaired user.

  To support this message type, the Accessible Mode user preference must be set to Screen Reader / No Vision. For more information about accessibility, see:
  - The *Configuration Guide*, "Accessibility tab," page 88
  - BMC Remedy User help
- **Tooltip**—Active links only. In web clients, the message appears in a tooltip. For information about tooltips, see "Using the message action to create tooltips" on page 90.

4 In the Text field, enter the text of the message.

You can enter an unlimited number of characters for active links and a maximum of 255 characters for filters. To insert field values from the current form or keywords in the message, click the ellipsis and use the Expression Editor dialog box.

5 In the Number field, enter a message number.

The message number is displayed with the message. The number that you specify must be greater than or equal to 10000. (Numbers less than 10000 are reserved for AR System messages.) Message numbers are not displayed with Prompt messages.

Define a unique number for each message. You can add the messages you define to the Error Messages form to create a complete list of AR System messages and messages for your application. To allow for localization of the message text, create an entry for the message in the Message Catalog form.

6 (Active links only). To cause a note or warning to appear in the Prompt Bar instead of a dialog box, select the Show Message in Prompt Bar check box.

7 Save the active link or filter.

Using the message action to create tooltips

Tooltips are brief informational messages that are displayed in response to a user action with an object on the screen. Tooltips are commonly used to provide descriptions of menu items, toolbar buttons, or other objects.

In AR System, tooltips can be applied to tables, attachments, field labels, or field data.

A tooltip can be displayed either by hovering the mouse over an area in a form or by clicking an object such as a button. For table fields, a tooltip can be displayed by hovering over a row. The tooltip displays the values for the row being hovered over, even if that row is not the currently selected row.

When a tooltip appears by hovering, it is closed when the mouse is moved outside the tooltip's border. When a tooltip appears by clicking (for example, when a tooltip appears when you click a button), it is closed when you click anywhere outside the tooltip's border.

A tooltip also can be dismissed by pressing the Escape key.

Tooltips can include URL links, which can be added through a Set Fields action.

Implementing tooltips

Tooltips are implemented through two types of AR System features:

- `HOVER` event
- Active link Message action

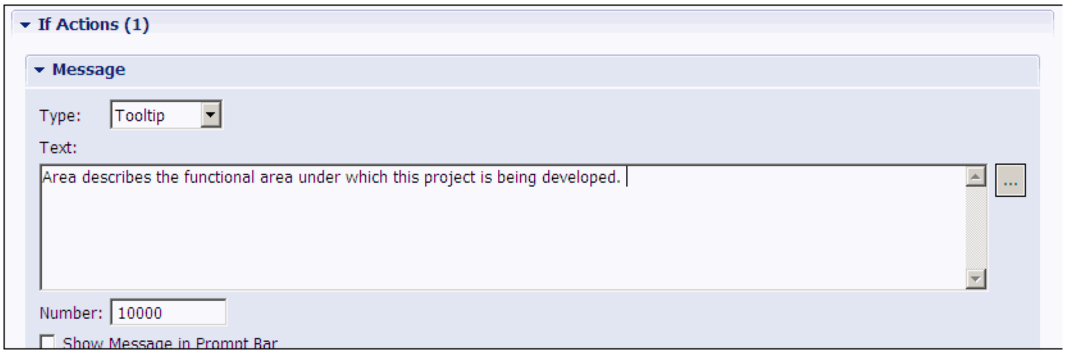
The `HOVER` event triggers the active link action that displays a tooltip. The `HOVER` event is an execution option in an active link. The `HOVER` event does not fire on fields that are disabled or hidden.

Depending on the field type you are working with, you can use the option buttons next to the Hover check box to select the area of the field where the `HOVER` event for the tooltip will be triggered. The options are Field, Label, and Data.

- If the applicable field is a data field, all three options are enabled.
- If the applicable field does not have distinguishable label and data areas, only the Field option is enabled.

A ToolTip selection is available as a message action option for an active link.

Figure 4-10: Message action for a tooltip



Type: Tooltip

Text:

Area describes the functional area under which this project is being developed.

Number: 10000

Show Message in Prompt Bar

Tooltip message actions are not available for filters.

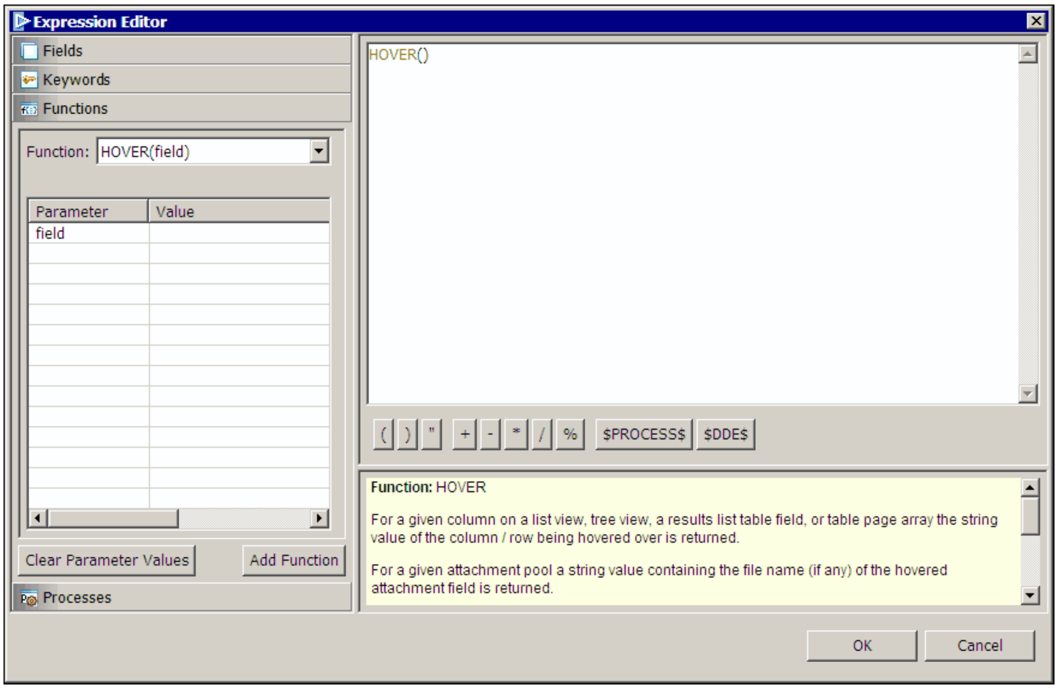
HOVER function

To add tooltips to table fields and attachment pools, use the HOVER function, which, when used with a field ID as an argument, returns the value of the field being hovered over. For example:

- `HOVER(colfield ID)` returns the value of the column field for the row being hovered over.
- `HOVER(tablefield ID)` returns a row value for the row being hovered over.
- `HOVER(attachpool)` returns the name of the file that is attached in the row of the attachment pool being hovered over.

Be sure to use the actual field ID to return the correct value. For example, if a column has an ID of 636880912, that is the value you must enter to have the correct value returned for the column.

Figure 4-11: Using the HOVER function for a tooltip



Function: HOVER

For a given column on a list view, tree view, a results list table field, or table page array the string value of the column / row being hovered over is returned.

For a given attachment pool a string value containing the file name (if any) of the hovered attachment field is returned.

Wait time for tooltip display

The wait time for detecting a `HOVER` event can be configured in two ways—by editing the `config.properties` file, or by specifying a wait time in the AR System User Preference form.

config.properties file

In the `config.properties` file, add a line that specifies the wait time in milliseconds, for example:

```
Arsystemhover_wait_time = 1500
```

where `1500` is 1500 milliseconds.

AR System User Preference form

In the AR System User Preference Form (Web tab), set a hover wait time (in milliseconds).

Formatting tooltips

The appearance of tooltips can be enhanced through the use of templates. For example, a template can be used to set the color and appearance of the tooltip's text and background.

A template is applied to a tooltip using the `TEMPLATE` function. A Set Field action sets the template result into a field. The field is referenced in the tooltip message.

Tooltips also can be formatted manually using HTML. However, the HTML is not validated.

Accessibility support (section 508)

For those using assistive devices, a tooltip is rendered as a dialog box, which can be dismissed by clicking its OK button. The following keyboard actions correspond to the areas on a form where a tooltip can be displayed:

- ALT+F9—label
- ALT+F10—data
- ALT+F12—field

Using the Prompt Bar

If you select the Show Message in Prompt Bar check box for a Message active link action, Note, Warning, and Error messages will be displayed in the prompt bar. (The check box is ignored for Prompt, Accessible, and Tooltip messages.)

In a *browser*, the prompt bar is displayed below the web toolbar.

In *BMC Remedy User*, the Prompt Bar is a message area at the top or bottom of the window (choose View > Prompt Bar > Top *or* Bottom). The user can control whether the Prompt Bar is hidden or visible (choose View > Prompt Bar > Visible).

If the Prompt Bar is *hidden* in BMC Remedy User, and a Message or Wait action directs a message to the Prompt Bar, one of the following events occurs:

- When a guide is running, the Prompt Bar becomes visible for the duration of the guide, assuming you use the Wait action.
- When no guide is running, messages directed to the Prompt Bar appear in a dialog box.

When the Prompt Bar is *visible* in BMC Remedy User, a note, warning, or error that is sent to the Prompt Bar can be overwritten by other prompt messages, and by the field help when the field focus changes. For this reason, select the Show Message in Prompt Bar check box only if you are including the message in a guide.

NOTE

System messages (from filters or from the server) appear in a prompt bar or a dialog box, depending on how they are configured in the AR System Administration: Server Information form.

Behavior of messages for prompt bar in a browser

If you select Note, Warning, or Error when creating a Message action link action *and* you select the Show Message in Prompt Bar check box, the message will usually appear in the prompt bar. Table 4-2 describes where messages appear in relation to where they are launched from. (In some cases, the message will appear in a pop-up box instead.)

Table 4-2: Where messages appear in relation to where they are launched from

If the Message active link action is configured for a:	The message appears in the:
Regular form	Prompt bar
Dialog box	Pop-up box
Modeless floating panel	Prompt bar
Dialog floating panel	Pop-up box
Tooltip floating panel	Prompt bar
Inline view field	Prompt bar of containing form
Dialog floating panel in an inline form	Pop-up box
Modeless floating panel in an inline form	Prompt bar of containing form

For more information about floating panels, see the *Form and Application Objects Guide*, "About floating panels," page 333.

NOTE

If a form is opened in a view field through the Open Window active link action, the Inline Form option is available. If Inline Form is selected, content in the view field appears as if it is part of the containing form. If a form is inside an inline view field, the form will never have a prompt bar. Any messages destined for a prompt bar are displayed in the containing form's prompt bar.

Workflow and prompt bar messages

After displaying a Note, Warning, or Error message, workflow resumes, so multiple messages might be displayed. Because Error messages cause workflow execution to stop, only one Error message is displayed at any time, and it is always the final message in the list.

By default, when a new set of workflow begins, the prompt bar is cleared for most conditions, except:

- Menu Choice / Row Choice (when Row Choice is used)
- Table Refresh
- Collapse (when not in the middle of a workflow set)
- Expand (when not in the middle of a workflow set)
- Close (for inline forms only)

When workflow is executed with the Window Closed execution option (when a window is closed or the mode is changed), messages are redirected to a pop-up box. This allows the user to see the message before the window closes.

However, if the message comes from an inline view field (configured through the Window Closed execution option), the message is displayed in the containing form's prompt bar. No redirection is needed.

Notify action

Use the Notify action in:

- Filters
- Escalations

Use the Notify action to send specified users or groups an email, an alert, or an alternate method outside of AR System. For example, a filter can notify support staff that they have been assigned a new request, or an escalation can notify a manager that a request has not been handled within the specified time.

Alerts are stored in the database, and users can view them in the alert list. Email messages are sent through the BMC Remedy Email Engine, which supports the SMTP and MAPI mail protocols for outgoing messages. You can also write notifications to a file for use by another application.

For information about BMC Remedy Alert, see BMC Remedy User Help and BMC Remedy Alert Help. For information about using alerts on the Web, see the *Configuration Guide*, “Using the alert list in a browser,” page 258. For information about sending notifications by using an external application, see “Using an external delivery mechanism” on page 101.

Creating a notify action

To configure a Notify action, you define basic information, such as the delivery mechanism, the recipients, and the contents of the notification message. You can configure the notify action to determine any of these options at runtime.

For notifications that will (or might) be delivered by email, you also specify email-related information, such as a subject, additional message header information, attachments and data content, and a template, if any. These options can also be determined at runtime. See "Entering additional email information" on page 99.

▶ To define the delivery mechanism, recipients, and contents

1 Right-click the If Action or the Else Action panel header.

2 Choose Add Action > Notify.

3 In the Mechanism field, select the notification mechanism:

- **Alert**—The specified users are notified with entries in the Alert Events form. Users can view their list of alerts and open the originating requests.
- **Email**—The notification is sent to the specified recipients by email. You can identify AR System recipients by their AR System user or group names, or enter email addresses to notify any recipient inside or outside of AR System.
- **User Profile Default**—The specified users are notified using the default method specified in their User form record. If the User record does not contain a default mechanism, the user is notified through BMC Remedy Alert.
- **Cross Reference**—The specified users are notified with the mechanism specified by an integer or selection field in the current request. Use this option to determine the notification method at runtime. See "Using a dynamic delivery method" on page 101.
- **Other**—The specified users are notified using the mechanism specified in the Other Code field. In this case, the notification is written to a file. Use this option to create your own delivery method. For example, you can set up your system to page users when they receive a notification. See "Using an external delivery mechanism" on page 101.

4 In the Priority field, enter a number from 0 through 10 to set the priority.

The priority appears in the Priority field of the Alert Events form to allow the user to sort the alert list by priority.

5 In the User field, enter information to identify the recipients of the notification.

The user field is limited to 255 characters. If you enter more than one recipient, separate each recipient by entering a hard return (press Enter). The server evaluates each line separately.

To specify one or more recipients, enter the recipient information in any of the following formats:

- **AR System user names**
  - Alert notifications are entered in the Alert Events form, with the user name appearing in the User field.
  - Email notifications are sent to the email address specified in the User form entry for the user.
- **AR System group names**—AR System takes the group name from the entry in the Group form, searches the User form for all users belonging to this group, and delivers the notification for each member.
  - For Alert notifications, an entry is made in the Alert Events form for each group member.
  - Email notifications are sent to the email address specified in the User form entry for each user.

For more information about users, groups, and the User and Group forms, see:

- The *Configuration Guide*, "Adding and modifying user information," page 57
- The *Form and Application Objects Guide*, "Groups in AR System," page 22

TIP

To broadcast a notification to a large group, BMC recommends that you send the notification to a mailing list or alias on the email server, instead of using an AR System group. This avoids the need to create all the notifications on the AR System server, which can impact performance.

- **Email addresses**—In addition to obtaining the email address from the User form, in filters you can enter email addresses directly or by using a keyword, such as `$USER$`. This allows the Notify action to send messages to users outside of AR System, including aliases or an email address that represents a program.

  Include the email domain name in the address, for example, `Joe.User@calbro.com` or `$USER$@calbro.com`.

- **A field reference, AR System keyword, or expression**—To allow the notification recipients to be determined at runtime, enter an expression using field names or keywords:
  - Type a dollar sign ($) and select the field name or keyword from the content assist list, or click the ellipsis to the open the Expression Editor dialog box. For example, to send a message to the user who created the request, enter $Submitter$.

    Entering a field name indicates that the name of the user or group to notify is in that field in the current request. You must create workflow to populate the field. The field can contain one or more recipients in the form of a user name, group name, or email address, separated by hard returns. The field cannot contain another field reference.

    Make sure that the data entered in the User field at runtime does not exceed 255 characters. If it does, the notification might not be delivered to some recipients.

    When you select a keyword, the current value of the keyword is substituted at runtime.

To resolve the contents of the User field, AR System first checks for a matching request in the User form. If found, the notification is sent to that user. If not found, the server checks for a matching request in the Group form. If the name corresponds to a group name, the notification is sent to all members of that group.

If the contents of the User field do not match an existing User or Group definition, AR System interprets the field contents as a literal address and sends the notification to that address, using the SMTP or MAPI mail protocols.

6 In the Text field, enter the content of the notification message.

The size limit for this field depends on the notification type. You can include up to 32 KB of message content in an email notification. For an Alert notification, enter no more than 4000 characters.

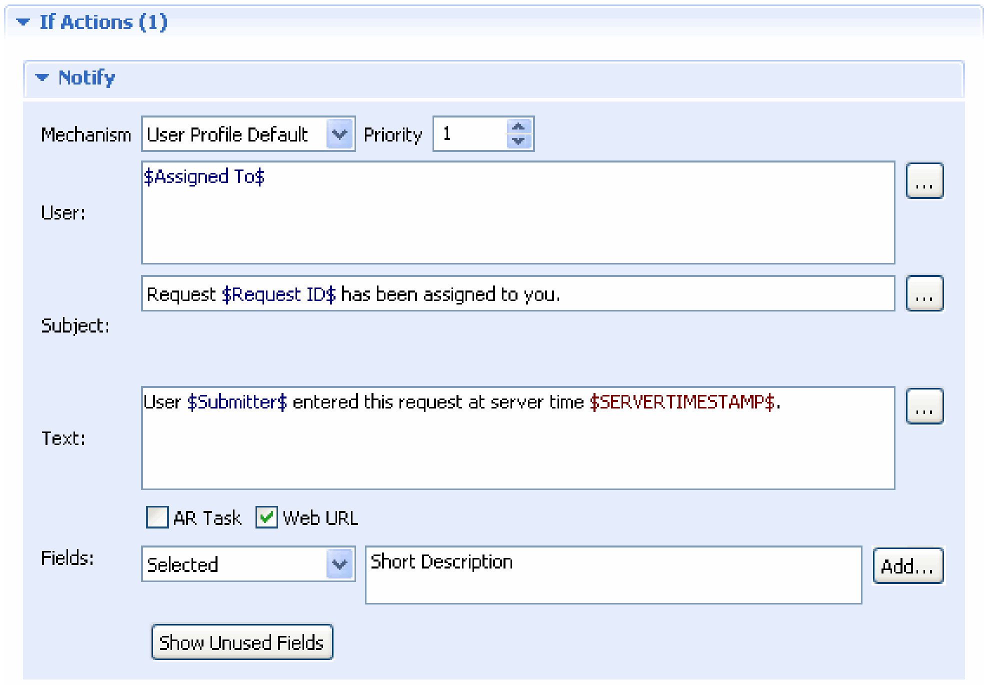
To include field values or keyword values in the notification text, use the Expression Editor dialog box or the content assist feature to select fields and keywords from the list. You must enter field names and keywords in the variable format, for example, `$Short Description$`. For information about using the expression editor, see Chapter 3, "Building qualifications and expressions."

Example Notify action

Figure 4-12 shows an example of a Notify action. In this example, the user in the Assigned To field of the current request will receive a priority 1 notification by either Alert or an email message, depending on the default set the recipient's User form record.

If the notification is an Alert, the contents of the Text field will appear in the Alert Text column of the Alert List, including the name of the submitter and the time stamp. If the notification is an email, the subject line will include the request ID, and the contents will include the contents of the Text field, along with a URL to the request, and the contents of the request's Short Description field.

Figure 4-12: Example Notify action



Mechanism: User Profile Default
Priority: 1
User: $Assigned To$
Subject: Request $Request ID$ has been assigned to you.
Text: User $Submitter$ entered this request at server time $SERVERTIMESTAMP$.
☐ AR Task ✓ Web URL
Fields: Selected
Short Description

Entering additional email information

For all notification mechanisms except Alert, optional fields appear in the Notify action for email-specific information. In addition actions set for Email in the Mechanism field, Notify actions configured for User Profile Default, Cross-Reference, or Other can also generate email notifications. The fields described in this section are ignored for Alert notifications.

▶ To enter additional email information

1 In the Subject field, enter a subject for the message.

You can include field and keyword values in the subject. The maximum length for the subject is 255 characters. If the value replacing a variable at runtime causes the Subject field data to exceed 255 characters, the subject is truncated at 255 characters. If you enter a field name in the Subject Line field, the notification contains the value of the field in the database, up to 255 characters.

NOTE

For some UNIX® email systems, you can include extra header lines in your email message by inserting hard returns followed by formatted mail headers in the Subject field. The resulting notification contains a Subject Line and your header lines. However, the total of all data in the Subject Line field must be 255 or fewer characters.

2 (Optional) Select a shortcut option to add a link to the request in the email message:

- **AR Task**—Adds an AR Task attachment to the email notification. The AR Task attachment allows the user to open the request in BMC Remedy User.
- **Web URL**—Includes a URL to the request in the email notification. The URL allows the user to open the request in the web client.

  To use a Web URL, you must define the web path in the Email Notifications Web Path field on the Advanced tab of the AR System Administration: Server Information form. See the *Configuration Guide*, "Server Information—Advanced tab," page 123.

3 From the Fields list, select an option to control the addition of request data in the message body of the email notification:

- **None**—None of the fields is included in the notification.
- **All**—All of the fields in the request are included in the notification.
- **Selected**—Selected fields are included in the notification. Click Add to open the Field Selector dialog box and select the fields to include.
- **Changed**—Only fields that have changed in the current transaction are included in the notification.

If you include field contents, AR System will check the field permissions to the notification's subject and body.

By default, the order of fields included in an email notification is based on their arrangement in the form view, as follows:

- Fields are taken from the default form view. If there is no default view, the first view is used.
- Using the X and Y coordinates of each field, the order of fields begins at the top left, and moves left to right, then down (in a zigzag-like pattern).
- Fields excluded from the form's default view are randomly included at the bottom of the list in a notification.
- You can only include data fields in an email. Other field types such as panel fields are ignored.

To guarantee the order of fields in a notification, create a special view named ARNotification. Exclude any panel fields from this view; otherwise, notifications default to the default view.

4 To view additional email-related fields, click Show Unused Fields. Use these fields to override default parameters relating to the email system. For information about these parameters, see the *BMC Remedy Email Engine Guide*.

5 Save the Notify filter or escalation action.

Using a dynamic delivery method

To determine the notification method at runtime, select Cross-Reference in the Mechanism field. (See "To define the delivery mechanism, recipients, and contents" on page 96.) With this option, users are notified based on the numerical value in an integer or radio (selection) field in the current request.

When you select the Cross-Reference delivery mechanism, another field appears in the filter or escalation named Reference Field. Enter the field variable for the referenced field in this field, for example, $DeliveryType$. Only integer fields and radio fields appear in the Field Selector dialog box.

To populate this field at runtime, you must also define workflow that inserts a numeric value between 0 and 3 in this field, or sets the equivalent radio field value, at runtime. In a radio field, the first option has a value of 0, the second a value of 1, and so on. These values represent the following delivery mechanisms:

- **0**—Do not notify
- **1**—Alert
- **2**—Email
- **3**—User default

Using an external delivery mechanism

To use a delivery method that does not use Alert or the BMC Remedy Email Engine, select Other in the Mechanism field. (See "To define the delivery mechanism, recipients, and contents" on page 96.)

When you select the Other delivery mechanism, the Code field appears in the filter or escalation. Select a value from 4 through 98 in the Code field to indicate the delivery mechanism to use.

With this option, the notification is written to a file in the following path:

- **Windows**—`*arInstallDir*\arserver\db\notification*NN*.arn`, where *NN* is the number in the Code field.
- **UNIX**—`*arInstallDir*/db/notification*NN*.arn`, where *NN* is the number in the Code field.

OLE Automation action

Use the OLE Automation action in:

- Active links

Use the OLE Automation action to share functionality between applications that support OLE. For more information, see the *Integration Guide*, "OLE automation," page 269.

Open Window action

Use the Open Window action in:

- Active links

Use the Open Window action to open an AR System form in one of six window types. The window type determines the form open mode and general window behavior. The fields in the active link editor change depending on the window type selected to allow you to configure the appropriate options for the Open Window action.

The Open Window action includes the following window types:

- **Dialog**—Open a form that acts as a dialog box from a parent form. The Open Window action defines what data is transferred from the parent form to the dialog box when the dialog box opens, and what data is transferred from the dialog box back to the parent form when a Commit Changes action occurs.
  - Data transfer from the parent form to the dialog box is based on the field mapping you create in the On Dialog Open Action table.
  - Data transfer from the dialog box back to the parent form is *not* automatic. You must create a Commit Changes active link action (see "Commit Changes action" on page 82) that executes before or when the dialog box closes to cause data transfer. The data transferred is determined by the field mapping you create for the On Dialog Close Action table.

  See "To define the Open Window action for dialog boxes" on page 103 and the *Form and Application Objects Guide*, "Using a display-only form as a dialog box," page 158.
- **Search**—Open a form in Search mode, using the specified server and form. You can either set default field values in the opened form or map values from the parent form to the search form. See "To define the Open Window action for Search or Submit window types" on page 106.
- **Submit** —Open a form in New mode, using the specified server and form. You can either set default field values in the opened form or map values from the parent form to the search form. See "To define the Open Window action for Search or Submit window types" on page 106.
- **Modify**—Open a form in Modify mode, using the specified server and form along with a qualification to select the entries to be displayed when the window opens. See "To define the Open Window active link action for Modify or Display windows" on page 108.
- **Display**—Open a form in Display mode, using the specified server and form along with a qualification to select the entries to be displayed when the window opens. See "To define the Open Window active link action for Modify or Display windows" on page 108.

- **Report**—Open a report form to display the results of a search, using the specified server and form. You also specify the report form to use and the search criteria to select the records to include in the report. See "To define the Open Window active link action for Report windows" on page 112.
- **Modify Directly**—Opens a form in Modify mode and skips workflow events except for the Window Open and Display events. See "To define the Open Window active link action for Modify or Display windows" on page 108.
- **Display Directly**—Opens a form in Display mode and skips workflow events except the Window Open and Window Display events. See "To define the Open Window active link action for Modify or Display windows" on page 108.

Popup —Opens a form like a pop-up on the browser. It behaves like an existing AR System pop-up and cannot be decoupled from the parent window. See "To define the Open Window action for Popup" on page 117.

NOTE

For all types of Open Window actions, you can hard code the source data server and form in the active link, or you can use the SAMPLE DATA data source to supply a source data server and form at runtime. For information about using a dynamic data source, see "Using a dynamic data source or destination in workflow" on page 71.

Creating Open Window actions for dialog boxes

Use a dialog window with a display-only form to prompt for user input and transfer the data entered to the parent form.

▶ To define the Open Window action for dialog boxes

1 Right-click the If Action or the Else Action panel header.

2 Choose Add Action > Open Window.

3 From the Window Type list, select Dialog.

The fields required to define the Open Window action for dialog boxes appear.

4 From the Data Source list, select SERVER or SAMPLE DATA:

- To define a specific server and form, select SERVER, and go to step a.
- To allow the server and form to be defined at runtime, select SAMPLE DATA, and go to step b.

a (*Data Source is SERVER*) Select the Server Name, Form Name, and Form View Name to be used as the dialog box.

Servers that appear in the Server Name list are those to which you are currently logged in.

Selecting a form view name is optional. If you do not enter anything in the Form View Name field, the form's default view is used. For more information, see *Form and Application Objects Guide*, "Creating and managing form views," page 393.

b (*Data Source is SAMPLE DATA*) Define the sample and runtime server, form, and view names or values:

- Use the Sample Server Name and Sample Form Name fields to select the server and form name to use while defining the Open Window action.
- Use the Runtime Server Value, Runtime Form Value, and Runtime Form View Value fields to identify the fields on the parent form or the AR System keywords that will define the source server, form, and view name for the dialog box at runtime.

For more information about using the SAMPLE DATA data source, see "Using a dynamic data source or destination in workflow" on page 71

5 Select or clear the Show Close Button check box. If the check box is:

- **Selected**—A close box appears in the upper-right corner of the dialog box in BMC Remedy User. This option has no effect in the web client, because browsers always contain an X (close) button in the top right corner.

NOTE

Selecting this option does *not* add an AR System button field to the form. Instead of using a generic window close box, consider creating buttons that predictably exit the dialog box and accomplish the actions that you want to occur. See "Using Commit Changes with a dialog box" on page 83.

- **Cleared**—A close box does not appear. For BMC Remedy User clients, you must create a Close Dialog action to close the dialog box and return the focus to the parent form.

6 In the On Dialog Open Action table, map the values for fields to populate when the dialog box opens:

a In the Field column, select fields to be populated in the dialog box, and then click OK.

The list of available fields comes from the dialog box form selected in the Form Name or Sample Form Name field.

b In the Value column, enter an expression to select field values from the parent form, as described in "Mapping fields to values" on page 68.

You can use field names, AR System keywords, the results of a function, or static values to set values in the dialog box fields, and you can build an expression to combine these value types.

7 In the On Dialog Close Action table, map the fields and values to be passed back to the parent form when the dialog box closes:

a In the Field column, select fields in the parent form to receive values from the dialog box when it closes.

The list of available fields comes from the primary form for the active link.

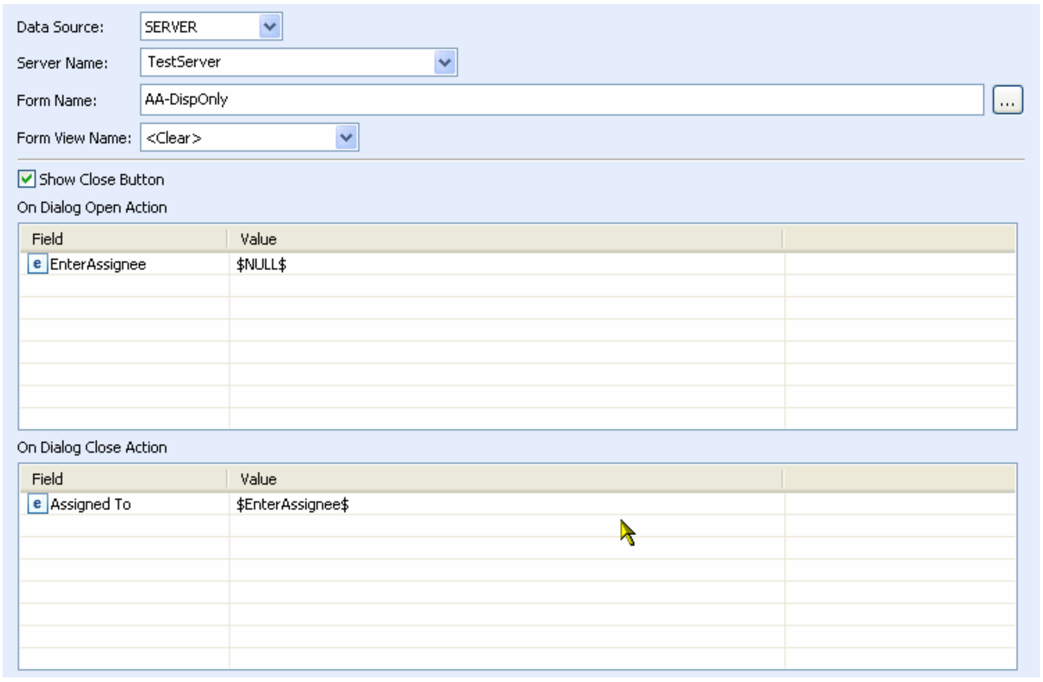
b In the Value column, enter an expression to select field values from the parent form, as described in "Mapping fields to values" on page 68.

NOTE

Mapping fields and values in the On Dialog Close Action table identifies the fields and values to populate on the parent form, but the mapping does not cause the values to transfer. You must use a Commit Changes action to transfer the defined field values to the parent form. See "Commit Changes action" on page 82.

Figure 4-13 shows an example Open Window active link action for the Dialog window type. In this example, the Open Window action will open a display-only form named AA-DispOnly as a dialog box, obtaining the form from the server TestServer. The field EnterAssignee is a field on the display-only form, and its initial value is set to `$NULL$`. When the dialog box is closed, a Commit Changes action will transfer the value in the EnterAssignee field to the Assigned To field on the parent form.

Figure 4-13: Open Window active link action, Dialog window type



Data Source: SERVER
Server Name: TestServer
Form Name: AA-DispOnly
Form View Name: <Clear>
✓ Show Close Button

On Dialog Open Action

Field	Value
EnterAssignee	$NULL$

On Dialog Close Action

Field	Value
Assigned To	$EnterAssignee$

Creating Open Window actions for Search or Submit windows

The search and submit window types open a form in Search mode or New mode, respectively. You can open a blank form, or you can populate fields using field defaults, AR System keywords, values from the current request, static data, or the results of a formula.

▶ To define the Open Window action for Search or Submit window types

1. Right-click the If Action or the Else Action panel header.
2. Choose Add Action > Open Window.
3. From the Window Type list, select Search or Submit.
   - **Search**—Opens the specified form in Search mode.
   - **Submit**—Opens the specified form in New mode.

   The fields required to define the Open Window action for search and submit window types appear.
4. To control where the window opens in the web client, select one of the following Target Locations:
   - **New**—Opens the form in a new browser window.
   - **New Without Toolbar**—opens the form in a new browser window with the browser toolbar, menu bar, and location bar hidden.
   - **Current**—Opens the form in the current browser window.
   - **<String>**—Allows you to specify a frame name, which opens the form inside the frame. In the case of a view field frame, the name is VF<FieldId>; for example, VF536870913.
   - **<Field>/<Keyword>**—Allows you to select a field or keyword that will determine the window open location at runtime.

   NOTE
   In BMC Remedy User, the Open Window action always opens a new window within the current instance of BMC Remedy User, and the Target Location field is ignored.

5. If you have selected **<String>** or **<Field>/<Keyword>** in step 4, perform the following:
   a. Select or clear the Inline Form check box. If the check box is:
      - Selected—the view field provides inline behavior for the rendering form. See *Form and Application Objects Guide*, "Inline Forms" on page 150.
      - Cleared—the view field provides the frame based behavior for the rendering form.

NOTE

View fields maintain the existing behavior unless the Inline Form option is checked.

b Perform the following in the field next to the Target Locations:

- For **<String>**, enter a view field id with the prefix `VF`. For example, for a view field with ID `1234`, enter `VF1234`.
- For **<Field>/<Keyword>**, click the ellipsis and select a character field that contains the `VF` expression at runtime.

NOTE

In BMC Remedy User, the Open Window action always opens a new window within the current instance of BMC Remedy User, and the Target Location field is ignored.

6 From the Data Source list, select SERVER or SAMPLE DATA:

- To define a specific server and form, select SERVER, and go to step a.
- To enable the source data server and form to be determined at runtime, select SAMPLE DATA, and go to step b.

a (*Data Source is SERVER*) Select the Server Name, Form Name, and Form View Name to be used as the dialog box.

Servers that appear in the Server Name list are those to which you are currently logged in.

Selecting a form view name is optional. If you do not enter anything in the Form View Name field, the form's default view is used. For more information, see *Form and Application Objects Guide*, "Creating and managing form views," page 393.

b (*Data Source is SAMPLE DATA*) Define the sample and runtime server, form, and view names or values:

- Use the Sample Server Name and Sample Form Name fields to select the server and form name to use while defining the Open Window action.
- Use the Runtime Server Value, Runtime Form Value, and Runtime Form View Value fields to identify the fields on the parent form or the AR System keywords that will define the source server, form, and view name for the dialog box at runtime.

For more information about using the SAMPLE DATA data source, see "Using a dynamic data source or destination in workflow" on page 71.

7 (*Optional*) To set field values in the form when it is opened, use the field mapping area:

- To set all fields to their default values, if any, select the Set Fields to Default Values check box.
- To map values to specific fields, select fields and define values in the field mapping table. See "Mapping fields to values" on page 68.

If you do not define any field values, a blank form opens in Search or New mode. When you save the active link without field values, BMC Remedy Developer Studio warns that no field values have been mapped, but the active link is saved.

The Class Search active link in the AR System Sample application contains an example Open Window action that uses a search window.

Creating Open Window actions for Modify or Display windows

The Modify and Display window types allow you to search for data in the database and display the results to the user.

▶ To define the Open Window active link action for Modify or Display windows

1. Right-click the If Action or the Else Action panel header.
2. Choose Add Action > Open Window.
3. From the Window Type list, select Modify or Display.
   - **Modify**—A user with the correct permissions can open and edit records in the results list.
   - **Display**—Users can open but not edit records in the results list.

   The fields required to define the Open Window action for a Modify or Display window appear.
4. From the Display Type list, choose a display type:
   - **List Only**—In BMC Remedy User, displays a list of results in the results pane. The user can open the record by clicking an item in the list.

     In the web client, the results are shown in a split display, including the results pane and the details pane.
   - **Detail Only**—In BMC Remedy User, the first matching record appears in the details pane, with Next and Previous buttons.

     In the web client, only the first matching entry is displayed. No toolbar buttons are available to move between entries, but you can add Next and Previous buttons and support them with workflow.
   - **Split Window**—The results are displayed in a split window, with the results list pane in the top half, and the first entry displayed in the details pane in the bottom half.
   - **Clear**—In BMC Remedy User, the results are displayed according to the display type specified by the Show Result List Only user preference.

     In the web client, the results are shown in a split display, including the results list pane and the details pane.
5. To control where the window opens in the web client, select one of the following Target Locations:
   - **New**—Opens the form in a new browser window.
   - **Current**—Opens the form in the current browser window.

- **<String>**—Allows you to specify a frame name, which opens the form inside the frame. In the case of a view field frame, the name is `VF<`*`FieldId`*`>`; for example, `FV536870913`.
- **<Field>/<Keyword>**—Supply a field or keyword reference that will contain a valid value for the target at runtime. Valid values for the Target identifier are:
  - `to-screen:`
  - `to-print:`
  - `to-file:`

  Include a space after the colon. For example, to create a screen target based on a field reference, the value in the field you select contains `to-screen: `.

NOTE

In BMC Remedy User, the Current and <String> options are ignored.

6 If you have selected **<String>** or **<Field>/<Keyword>** in step 4, perform the following:

a Select or clear the Inline Form check box. If the check box is:

- Selected—the view field provides inline behavior for the rendering form. See *Form and Application Objects Guide*, “Inline Forms” on page 150.
- Cleared—the view field provides the frame based behavior for the rendering form.

NOTE

View fields maintain the existing behavior unless the Inline Form option is checked.

7 From the Data Source list, select SERVER or SAMPLE DATA:

- To define a specific server and form, select SERVER, and go to step a.
- To allow the source data server and form to be determined at runtime, select SAMPLE DATA, and go to step b.

a (*Data Source is SERVER*) Select the Server Name, Form Name, and Form View Name to be used as the dialog box.

Servers that appear in the Server Name list are those to which you are currently logged in.

Selecting a form view name is optional. If you do not enter anything in the Form View Name field, the form’s default view is used. For more information, see *Form and Application Objects Guide*, “Creating and managing form views,” page 393.

b (*Data Source is SAMPLE DATA*) Define the sample and runtime server, form, and view names or values:

- Use the Sample Server Name and Sample Form Name fields to select the server and form name to use while defining the Open Window action.

- Use the Runtime Server Value, Runtime Form Value, and Runtime Form View Value fields to identify the fields on the parent form or the AR System keywords that will define the source server, form, and view name for the dialog box at runtime.

For more information about using the SAMPLE DATA data source, see "Using a dynamic data source or destination in workflow" on page 71.

8 In the Qualification field, complete the following steps:

a Define a qualification that defines the records to return when the window opens.

Click the ellipsis to use the Expression Editor dialog box, or type the qualification, using content assist to select fields and keywords. For information about creating qualifications, see Chapter 3, "Building qualifications and expressions."

b (*Optional*) Select Suppress Empty List to specify that an empty list does not appear if no results are returned from the search.

c From the If No Requests Match list, select a handling option to control how BMC Remedy User or the web client responds when no matches are found in the selected form:

- **Do Not Show Any Message**—No message appears regardless of results returned.
- **Show Default Message**—Standard system-generated message appears if no requests match.
- **Show Message**—A message that you define appears.

  The fields for defining a message appear in the workflow editor.

9 If you selected Show Message:

a From the Message Type list, select the appropriate message type.

b In the Message Number field, enter a message number.

c In the Message Text field, enter the text of the message.

To build an expression using fields or keywords, click the ellipsis button.

For more information about these parameters, see "Message action" on page 88.

10 (*Optional*) To specify a sort order for the results, or a polling interval, click Show Advanced.

11 To sort the requests of the search, complete the Sort Order table:

a Click Add, and then select fields to sort on in the Field Selector dialog box.

b Click OK to add the fields to the Sort Order table.

c To move fields up or down in the Sort Order table, use the Up and Down buttons.

d To select ascending or descending order, click the field in the Sort Order column, and then select Ascending or Descending.

e To remove a field from the Sort Order table, select the field, and then click Remove.

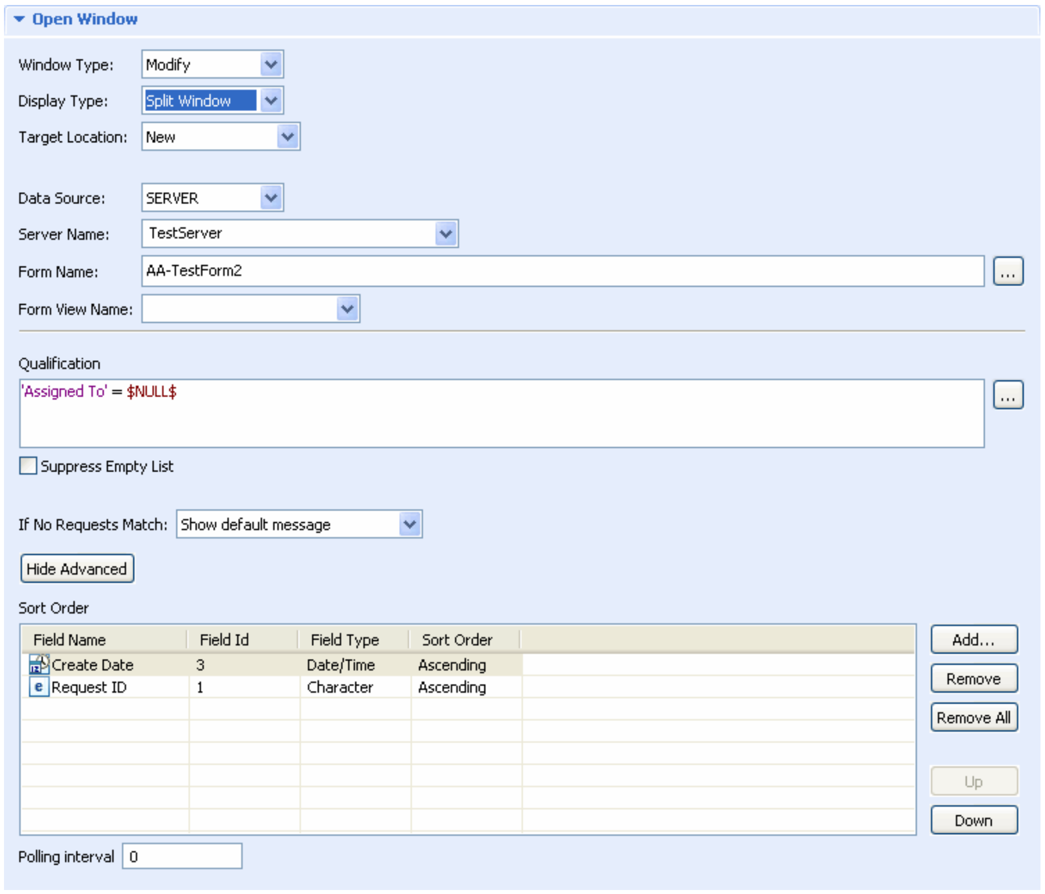
f To clear the table, click Remove All.

12 (*Optional*) To define a polling interval for BMC Remedy User clients, enter the interval in minutes in the Polling Interval field.

This option causes BMC Remedy User to reissue the search at the specified interval while the window is open.

Figure 4-14 shows an Open Window action for a Modify window type. In this example, the Open Window action will search for requests in the form AA-TestForm2, in which the Assigned To field is empty. If the search finds matching requests, the Modify window display the results of the search. A results list will appear in the top half of the window and the first matching record will appear in the detail pane in the bottom half of the window. The results list will be sorted by Create Date and then by Request ID. If no requests match the search qualification, a default message will appear.

Figure 4-14: Open Window active link action, Modify window type



Open Window

Window Type: Modify
Display Type: Split Window
Target Location: New

Data Source: SERVER
Server Name: TestServer
Form Name: AA-TestForm2
Form View Name:

Qualification
'Assigned To' = $NULL$

Suppress Empty List

If No Requests Match: Show default message

Sort Order

Field Name	Field Id	Field Type	Sort Order
Create Date	3	Date/Time	Ascending
Request ID	1	Character	Ascending

Polling interval 0

Creating Open Window actions for reports

Use the Report window type to search for records matching a qualification and display the results in a report. You can use any compatible report format, including AR System reports and Crystal reports. For information about creating and using reports in AR System, see:

- The BMC Remedy User help
- The *BMC Remedy Mid Tier Guide*, "Using Crystal reports with AR System," page 111
- The *BMC Remedy Mid Tier Guide*, "For your end users: Creating reports in a browser," page 155

▶ To define the Open Window active link action for Report windows

1 Right-click the If Action or the Else Action panel header.

2 Choose Add Action > Open Window.

3 From the Window Type list, select Report.

The fields required to define the Open Window action for a Report window appear.

4 To control where the report displays, select one of the following Target Locations:

- **New**—Opens the form in a new browser window.
- **Current**—Opens the form in the current browser window.
- **<String>**—Allows you to specify a frame name, which opens the form inside the frame. In the case of a view field frame, the name is `VF<FieldId>`; for example, `FV536870913`.
- **<Field>/<Keyword>**—Supply a field or keyword reference that will contain a valid value for the target at runtime. Valid values for the Target identifier are:
  - `to-screen:`
  - `to-print:`
  - `to-file:`

  Include a space after the colon. For example, to create a screen target based on a field reference, the value in the field you select contains `to-screen: `.

NOTE

In BMC Remedy User, the Current and <String> options are ignored.

5 From the Data Source list, select SERVER or SAMPLE DATA:

- To define a specific server and form, select SERVER, and go to step a.
- To allow the source data server and form to be determined at runtime, select SAMPLE DATA, and go to step b.

a (*Data Source is SERVER*) Select the Server Name, Form Name, and Form View Name to be used as the dialog box.

Servers that appear in the Server Name list are those to which you are currently logged in.

Selecting a form view name is optional. If you do not enter anything in the Form View Name field, the form's default view is used. For more information *Form and Application Objects Guide*, "Creating and managing form views," page 393.

b (*Data Source is SAMPLE DATA*) Define the sample and runtime server, form, and view names or values:

- Use the Sample Server Name and Sample Form Name fields to select the server and form name to use while defining the Open Window action.
- Use the Runtime Server Value, Runtime Form Value, and Runtime Form View Value fields to specify the fields on the parent form or the AR System keywords that will define the source server, form, and view name for the dialog box at runtime.

For more information about using the SAMPLE DATA data source, see "Using a dynamic data source or destination in workflow" on page 71.

6 From the Report Type list, select a report type:

- **AR System**—Use an AR System report.
- **Crystal**—Use a Crystal Report, if installed.
- **<Field>/<Keyword>**—Supply a field or keyword reference that will contain a valid value for the report type at runtime. Valid values for this field are:
  - The Report Type identifier (21=).
  - The corresponding numeric report type (1 for AR System and 2 for Crystal).

  For example, to create an AR System report based on a field reference, the value in the field must contain 21=1 at runtime.

Other report options might appear, depending on which report engine software you have installed on your system.

7 From the Report Location list, select where the report resides:

- **Embedded**—Report is stored as part of the active link.

NOTE

If you make changes to an embedded report after creating the active link, save the active link to use the updated report.

- **Local**—Report resides on the client's local disk.
- **Report Form**—Report resides on the reporting server in the Report form.

  For web clients, the reporting server must be defined in the AR System User Preference form. For BMC Remedy User, the reporting server must be defined in the local user preferences or in the AR System User Preference form. If the reporting server is not defined, the report does not run.

  The Report form has a unique index, which guarantees that the report name is unique.

8 To save an embedded report to another area on the network, click Save to Disk.

   An embedded report is saved with the definition of the active link, on the server where the active link is created. Use Save to Disk if you need to access to the report on an AR System server where the embedded report does not reside locally.

9 In the Report Name field, enter the full path name for the report.

   If the report is located on a network, enter a path with a drive mapping or the full server and directory path (for example, `V:\shared\report2.rpt` or `\\`*`serverName`*`\`*`path`*`\`*`fileName`*).

   - For Embedded reports, the path name must be for a location on or accessible to, the BMC Remedy Developer Studio client machine. This allows AR System to store the report with the Open Window action when you save the active link.
   - For Local reports, select <String> or <Field / Keyword>:
     - **<String>**—Enter the full path to the report. To navigate to the report location, click Browse.

       The path must be to a location on or accessible to the user's machine. This allows AR System to run the report from the client machine when the Open Window action executes.
     - **<Field>/<Keyword>**—Select a field or keyword that will define the full path to the report at runtime.
   - For Report Form reports, select <String> or <Field / Keyword>:
     - **<String>**—Enter the name of the report as entered in the Name field of the appropriate entry in the Report form.
     - **<Field>/<Keyword>**—Select a field or keyword that will contain the name of the report at runtime.

10 From the Report Destination list, select where you want your report to appear:

   - **Screen**—The report appears in a window as determined by the option selected in the Target Location field.

     For Crystal Reports, if a report is run from a regular form window, the report is displayed in a window in BMC Remedy User. If the report is run from a form opened as a dialog box (modal window), the report is displayed in a new window and in front of the dialog box. The Crystal Report is scrollable, resizable, and printable.
   - **Printer**—The report is sent to your default printer as defined in your Windows environment (BMC Remedy User only).
   - **File**—The report is saved to a file (BMC Remedy User only).

     For Crystal Reports, a dialog box allows the user to select the file format and destination. In the user tool a Crystal Dialog is shown that allows the user to select the Format and the Destination. File formats include PDF, HTML, RPT, Microsoft Excel, Microsoft Word, and others. Possible destinations include a file, an application, a Microsoft Exchange folder, others.

For the web client, you must choose Screen. The user must print the report from the browser.

- **<Field>/<Keyword>**—The report destination is determined at runtime.

  Create an expression to identify a field reference in the Report Destination field. The value of the field at runtime dynamically defines the destination of the report. Valid values for the destination identifier are:

  - `to-screen:`
  - `to-print:`
  - `to-file:`

  In each case, the colon must be followed by a space character in the field value. For example, to specify a screen destination using a field reference, the value in the field must be `to-screen: `.

11 In the Qualification field, complete the following steps:

a Define a qualification that defines the records to return when the window opens.

Click the ellipsis to use the Expression Editor dialog box, or type the qualification, using content assist to select fields and keywords. For information about creating qualifications, see Chapter 3, "Building qualifications and expressions."

b From the If No Requests Match list, select a handling option to control how BMC Remedy User or the web client responds when no matches are found in the selected form:

- **Do Not Show Any Message**—No message appears regardless of results returned.
- **Show Default Message**—Standard system-generated message appears if no requests match.
- **Show Message**—A message that you define appears.

  The fields for defining a message appear in the workflow editor.

12 If you selected Show Message:

a From the Message Type list, select the appropriate message type.

b In the Message Number field, enter a message number.

c In the Message Text field, enter the text of the message.

To build an expression using fields or keywords, click the ellipsis button.

For more information about these parameters, see "Message action" on page 88.

13 (*Optional*) To specify a sort order for the results, or to enter other advanced options, click Show Advanced.

14 To sort the data in the report, complete the Sort Order table as described in "Creating Open Window actions for Modify or Display windows" on page 108.

15 (*Web clients only*) To use entry IDs rather than the qualification to select entries for the report, select the appropriate option from the Entry IDs list:

- **<String>**—Enter the Entry ID
- **<Field>/<Keyword>**—Select a field or keyword that will contain the Entry ID at runtime.

Select a field reference in the Entry IDs field list that passes the ID numbers of the entries selected for the report in the Query List View. If there is more than one entry ID, separate them with commas. These IDs take precedence over any search entered in the Qualification tab. However, if the Entry IDs field is blank, the qualification criteria are used to create entries for the report.

16 For web clients, if your third-party report engine supports query override, enter Yes or No in the Query Override field to indicate whether you want the query in the results list or table field to override the query stored in the report.

Make sure that the Query Override field in the Report form is consistent with the Query Override in this Open Window active link so that the report runs correctly.

17 From the Report Operation list, select the appropriate option:

- **Edit**—Only for Web clients using AR System reporting on a *pre-6.3* mid tier, where the window type is Report.
- **Run**—Only for AR System reports on web clients. Select this option to run a report by using the Start command on the ReportType entry for the selected report type.
- **Create**—Only for Web clients using AR System reporting on a *pre-6.3* mid tier, where the window type is Report.
- **<Field>/<Keyword>**—Enter a field or keyword that will contain the Report Operation type at runtime.

NOTE

In the pre-6.3 mid tier, you used *Create* and *Edit* operations for Open Window active link actions with a Window Type of Report to create and modify AR System reports. In the 6.3 and later versions of the mid tier, Create and Edit operations are not supported. To create and edit reports from an Open Window active link, use the Submit Window Type, and select the ReportCreator form.

18 From the Character Encoding list, select the appropriate option:

- **Use server**—Use the character encoding defined for the AR System server in the report. This is the default.
- **Alternate character sets**—Select an alternate character set for the report. For example, to use Korean characters, select Korean.
- **<Field>/<Keyword>**—Select a field or keyword that will contain the character encoding at runtime.

Creating Open Window actions for Popup window

The Popup Open Window action prompts for user input and transfers the data entered to the parent form.

The Popup type provides an advanced look to the pop-ups in BMC Remedy Mid Tier. It does not open in a new window or on a new tab. Instead, it appears as a child window at the center of the parent window. It closes automatically when you close the parent window. You cannot move a pop-ups outside the parent window.

You can handle the close option on the pop-ups in BMC Remedy Mid Tier using the `Show Close Button` option. For more details, see "To define the Open Window action for Popup" on page 117.

- When you enable the option, the pop-up window opens with a close option on the title bar.
- When you disable the option, the pop-up window does not contain the title bar and the close option.

▶ To define the Open Window action for Popup

1 Right-click the If Action or the Else Action panel header.

2 Choose Add Action > Open Window.

3 From the Window Type list, select Popup.

The fields required to define the Open Window action for Popup appear.

4 From the Data Source list, select SERVER or SAMPLE DATA:

- To define a specific server and form, select SERVER, and go to step a.
- To allow the server and form to be defined at runtime, select SAMPLE DATA, and go to step b.

a (*Data Source is SERVER*) Select the Server Name, Form Name, and Form View Name to be used as the pop-up.

Servers that appear in the Server Name list are those to which you are currently logged in.

Selecting a form view name is optional. If you do not enter anything in the Form View Name field, the form's default view is used. For more information about default view, see *Form and Application Objects Guide*, "Creating and managing form views."

b (*Data Source is SAMPLE DATA*) Define the sample and runtime server, form, and view names or values:

- Use the Sample Server Name and Sample Form Name fields to select the server and form name to use while defining the Open Window action.
- Use the Runtime Server Value, Runtime Form Value, and Runtime Form View Value fields to identify the fields on the parent form or the AR System keywords that will define the source server, form, and view name for the pop-up at runtime.

For more information about using the SAMPLE DATA data source, see "Using a dynamic data source or destination in workflow" on page 71

5 Select or clear the Show Close Button check box. If the check box is:

- **Selected**—A close box appears in the upper-right corner of the pop-up window on the browser.
- **Cleared**—A close box does not appear. You must create a Close Window action to close the pop-up and return the focus to the parent form. Otherwise, you cannot close the pop-up window.

6 In the On Dialog Open Action table, map the values for fields to populate when the pop-up opens:

a In the Field column, select fields to be populated in the pop-up, and then click OK.

The list of available fields comes from the pop-up form selected in the Form Name or Sample Form Name field.

b In the Value column, enter an expression to select field values from the parent form, as described in "Mapping fields to values" on page 68.

You can use field names, AR System keywords, the results of a function, or static values to set values in the pop-up fields, and you can build an expression to combine these value types.

7 In the On Dialog Close Action table, map the fields and values to be passed back to the parent form when the pop-up closes:

a In the Field column, select fields in the parent form to receive values from the pop-up when it closes.

The list of available fields comes from the primary form for the active link.

b In the Value column, enter an expression to select field values from the parent form, as described in "Mapping fields to values" on page 68.

NOTE

Mapping fields and values in the On Dialog Close Action table identifies the fields and values to populate on the parent form, but the mapping does not cause the values to transfer. You must use a Commit Changes action to transfer the defined field values to the parent form. See "Commit Changes action" on page 82.

Using a dynamic data source in Open Window actions

To create an Open Windows action that uses runtime information to identify the source server and form, specify SAMPLE DATA instead of SERVER for the data source. See "Using a dynamic data source or destination in workflow" on page 71.

To use the SAMPLE DATA data source in an Open Window action, you must enter values into the Server Name and Form Name fields. The Form View setting is optional.

Allowing users to select a view based on preferences

You can create an alternate view of an existing view that can dynamically open at runtime for specific users. The user controls the view selection by setting the Open Window View Extension field in the user preferences. The alternate view must start with the same prefix as an existing view.

For example, if an existing view has the label `Inventory`, an alternate view can have the label `Inventory_Large Font`. Select `Inventory` in the Form View field of the active link, and instruct users to specify `_Large Font` in the Open Window View Extension field in user preferences.

Push Fields action

Use the Push Fields action in:

- Active links
- Filters
- Escalations

The Push Fields action enables you to automate updates to the database. You can transfer values from selected fields in the current request to another request in a different form or in the same form. The Push Fields action can update or create a request.

You can push values from all field types, including tables, hidden fields, fields not in the active view, or fields in no views at all. However, you cannot push values *to* non-data fields, such as table fields, column fields, and panel holders.

Each Push Fields action can push data from the current form to one other form, although you can update multiple requests in that form. To push data from the current form to more than one form, you must create a separate Push Fields action for each destination form.

▶ To define a Push Fields active link, filter, or escalation action

1 Right-click the If Action or the Else Action panel header.

2 Choose Add Action > Push Fields.

3 From the Data Destination list, select SERVER or SAMPLE DATA:

- To specify a specific server and form, select SERVER, and go to step a.
- To determine the server and form at runtime, select SAMPLE DATA, and go to step b.

a. (*Data Destination is SERVER*) Select the Server Name (active links only) and Form Name to push data to.

   Servers that appear in the Server Name list are those to which you are currently logged in. In the Form Name field, the forms that appear in the Form Selector dialog box are from the server specified in the Server Name field.

   Filters and escalations can push data only to forms on the server where the workflow is running.

b. (*Data Destination is SAMPLE DATA*) Define the sample and runtime server and form names or values:

   - In the Sample Server Name (active links only) and Sample Form Name fields, select the sample server and destination form name to use to design the Open Window action.
   - In the Runtime Server Value and Runtime Form Value fields, identify the fields on the current form or the AR System keywords that will identify the destination server and form at runtime.

   For more information, see "Using a dynamic data source or destination in workflow" on page 71.

4. In the Qualification field, enter a qualification, if any, to specify the request to which the Push Fields action will push the data.

   Click the ellipsis button to use the Expression Editor dialog box, or type the expression directly in the Qualification field. For more information about creating qualifications, see Chapter 3, "Building qualifications and expressions."

   - To select fields in the Expression Editor dialog box, first select either the current form or the destination form in the Form list. The list of Available Fields changes to display fields from the selected form.
   - As you build the expression, fields from the current form are delimited with dollar signs ($). Fields from the destination form are delimited with single quotation marks (').
   - To use content assist to build the expression, press Ctrl+Space to get a list of all possible choices. Alternatively, type a dollar sign ($) to get a list of fields from the current form and keywords. Type a single quotation mark (') to get a list of fields from the destination form.

   If you associated additional forms and want to use a field from a form other than the primary form, enter the appropriate field ID in the appropriate format, for example, '`$fieldID$`' or '`fieldID`'. The field must exist on the form.

   NOTE
   Using a qualification causes the Push Fields action to run a query to obtain the list of matching requests, so make sure to optimize the Push Fields qualification for best system performance. See "Creating efficient qualifications" on page 56.

- To create a Push Fields action that does not search for existing records but instead always creates a request, do not enter a qualification. Instead, select the following values:
  - In the If No Requests Match field, select Create a New Request.
  - In the If Any Requests Match field, select Take No Action.

5 From the If No Requests Match list, select an option to control how the system responds when no matches are found in the destination form:

- **Display 'No Match' Error**—Returns an error message and stops processing.
- **Take No Action**—Skips this action and proceeds to the next action.
- **Create a New Request**—Uses the data specified in the Push Fields action to create a request.

6 From the If Any Requests Match list, select an option to control how the system responds when multiple matches are found in the selected form:

- **Display 'Any Match' Error**—Returns an error message and stops processing.
- **Modify First Matching Request**—Pushes data to the first request that meets the qualification.
- **Modify All Matching Requests**—Pushes data to every request that meets the qualification.
- **Take No Action**—Skips this action and proceeds to the next action.

7 Use the field mapping table to map the fields and values to push from the current form to the destination form as follows:

- To use matching field IDs to map the values of all matching fields from the current form to the destination form automatically, select the Matching IDs option. The field mapping table is disabled, and you cannot select specific field mappings.

  In this case, all matching field IDs (except for table columns, panel holders, and core fields such as Modified Date and Request ID) are automatically set in the destination form from the values in the current form.

TIP

When you use Matching IDs, the values are dynamic. This means that only the setting is stored as part of the Push Fields action, and not the actual values. Therefore, you can add data fields to these forms at a later stage and the action uses them when executed.

- To use the Auto Map dialog box, click Auto Map.

  The Auto Map dialog box opens with all matching fields entered in the list. To delete a field you do not want to update, select the field and click Remove. After making selections in the Auto Map dialog box, click OK.

TIP

You can use Auto Map to map fields according to the field name or field ID. When you use Auto Map with field names, the mappings are dynamic. However, when you use Auto Map with field IDs, the mappings are stored with the form. In this case, if you later add fields to the forms, they will not be part of the action and must be added manually.

- To map specific fields and to use an expression to define the value, follow the steps described in "Mapping fields to values" on page 68.

  In the Field column, select the fields in the destination form to which the Push Fields action will push data. In the Value column, use an expression to define the values that the Push Fields action will push to the fields in the destination form.

  To define the value, you can use field values from the current form, keywords, static values, or the result of a function. In active links, you can also use the result of a DDE operation. For information about using the result of a function, see "Assigning values using function results" on page 251. For more information, see *Integration Guide*, "Using active links with DDE," page 290.

8 Save the active link, filter, or escalation.

Using a dynamic data destination

To create a Push Fields action that uses runtime information to identify the destination server and form, specify SAMPLE DATA instead of SERVER for the data destination. At runtime, the field values are pushed to a server and form defined by workflow or selected by the user. See "Using a dynamic data source or destination in workflow" on page 71.

NOTE

Dynamic Push Fields actions cause browsers to perform extra HTTP fetches and cause the AR System server to perform extra queries to determine the data types for remote fields. To enable the mid tier to fetch these data types ahead of time and avoid performance degradation, avoid using field references to store the server or form names.

Run Macro action

Use the Run Macro action in:

- Active links

The Run Macro action runs a macro created in BMC Remedy User. This workflow action only runs in BMC Remedy User, and is not supported in the web client. Most operations that can be performed from a macro can be performed from other actions. Use the Run Macro action for backward-compatibility with clients prior to the 5.0 release *only*.

The macro can perform any operation or series of operations. If the macro contains parameters, you can specify values for those parameters, including a value from a field in the current request. For example, a macro could use the value in the Customer Name field of the current request to query the database for information about the customer, or you it could generates a report when the user clicks a button in the current request.

To use a macro in an active link, you must first create the macro by using BMC Remedy User. For information about creating a macro, see BMC Remedy User Help)

The macro that you specify is copied into the active link. If you later change the macro, you must modify the active link and specify the macro again.

NOTE

You can use the `$VERSION$` keyword to include or exclude older Windows user tool clients in workflow. For more information, see "`$VERSION$`" on page 228.

You cannot convert a Run Macro action for use in an active link in these cases:

- When you use the query bar inside the macro and:
  - Field names and field labels on the current form are different.
  - You use field IDs not on the current form as a parameter inside the query.
- For any action involving logging in.

▶ To define the Run Macro active link action

1. Right-click the If Action or the Else Action panel header.
2. Choose Add Action > Run Macro.
3. In the Macro Name field, click the ellipsis button and navigate to select the appropriate macro.

   If the macro contains parameters, a list of parameters appears.

4 If parameters are listed, enter any of the following as parameter values:

- A static value
- A keyword
- A reference to any field in the current form

  Field references and keywords must be enclosed within dollar signs, for example, `$Customer Name$`. You can enter a field reference or keyword by clicking in the Value column to open the Expression Editor dialog box, or by using the content assist feature.

NOTE

Macros do not prompt for values when executed from active links, so you must specify parameter values in the Run Macro action. To prompt for values, use an Open Window active link action to capture the values. See "Creating Open Window actions for dialog boxes" on page 103.

5 To save the macro to another area on the network, click Save To Disk.

6 Save the Run Macro active link action.

Run Process action

Use the Run Process action in:

- Active links
- Filters
- Escalations

Use the Run Process action to run an independent process on a client computer or an AR System server. The system executes the command specified in the Command Line field. You can run an AR System application command or workflow process command, or specify an external program to carry out actions such as sending a fax or making a log entry in a specific format.

For more information about the AR System application and workflow commands, see Appendix C, "Using Run Process and $PROCESS$ commands." For information about using the Run Process action or the `$PROCESS$` keyword to integrate AR System with an external application, see the *Integration Guide*, "Running external processes (Run Process)," page 259.

NOTE

The Run Process action only executes an independent process; it does *not* return a value. To insert the result of a process into a field, use the Set Fields action with the `$PROCESS$` keyword to set a field by using the value returned from a process (see "Assigning values from process results" on page 253).

Run Process action and access control

If the process executed with the Run Process action runs on the *client*, it uses the permissions of the user who started BMC Remedy User. If the process executed with the Run Process command runs on the *server*, it uses the permissions of the user who started the AR System server. Make sure to configure AR System and network access permissions to correctly enforce access control for processes.

IMPORTANT

For active links that run processes on the server, AR System provides a security feature that allows you to limit active link processes to only execute in a specified directory. For more information, see the permission and configuration information in the *Configuration Guide*.

Using the Run Process command in a browser

To avoid degrading performance on the web, always specify the action in the Run Process command instead of indirectly through a field. For example, the command `PERFORM-ACTION-ADD-ATTACHMENT 536868912` is acceptable. However, the command `$536868914$ 536868912` can degrade performance on the web.

Creating a Run Process action

Use the procedures in this section to define a Run Process action.

▶ To define the Run Process active link, filter, or escalation action

1 Right-click the If Action or the Else Action panel header.

2 Choose Add Action > Run Process.

3 In the Command Line field, enter the command to execute.

You can type the command or you can build the command by using the Expression Editor dialog box. You can include the values from fields from the primary form and keywords in the command, and you can select a process from the list of AR System processes.

- If an expanded value contains spaces, you must insert double quotation marks around the parameter so that the operating system interprets it as a single value.
- The fully expanded command line cannot exceed 4096 bytes.

4 Save the active link, filter, or escalation.

Running AR System application processes from an active link

AR System provides a set of workflow commands and application commands that you can call with the Run Process action. These are described in Appendix C, "Using Run Process and $PROCESS$ commands."

The process commands named `GET-CHANGE-FLAG`, `SET-CHANGE-FLAG`, `SET-RO-TYPE`, and `PERFORM-ACTION-`*`actionName`* are workflow-specific commands that carry out various workflow actions. These commands are executed on the client when run in active links, and on server when run in a filter or escalation.

The process commands named `Application-`*`commandName`* can only be executed by the server. To use an Application command in an active link, you use a special syntax to identify the server where the command will run:

- To run the command on the current AR System server, enter

  `@@:`*`processCommand`*

- To run the command on a difference AR System server, enter

  `@`*`serverName`*`:`*`processCommand`*

Running external processes on a client computer

For active links, when you run a process on a client computer, consider whether you need to verify that the process can run on the client computer. If the process is not available on all hardware platforms and operating systems, you can build a qualification for the active link by using the `$HARDWARE$` and `$OS$` keywords to verify that the client is running on an appropriate platform or operating system at the time the active link executes. See Chapter 3, "Building qualifications and expressions," for more information.

Command line syntax guidelines

BMC recommends the following guidelines for command line syntax:

- For external processes, adjust the command syntax appropriately for the platform on which the server is running, and include the explicit path to the command; for example, `/home/jim/bin/`*`command`*.

  In the Windows environment, specify the drive; for example, `d:\home\jim\bin\`*`command.bat`*.

- In the UNIX environment, the process runs under a Bourne shell.
- On a Windows server, you can only run a process that runs in a console (such as a `.bat` script or an executable like `runmacro.exe`).
- Use double quotation marks around substituted fields when the values might contain spaces or other special characters; for example, `/bin/cmd "$`*`field`*`$"`.

- Substituted field values that contain hard returns or other special characters can have unexpected results.
- The AR System server does not interpret environment variables for the Run Process action. To use environment variables, the action must run in the context of a command window or shell on the client or server computer. Therefore, include the path to the command-line executable in the command-line statement, as in the following Windows example (`cmd.exe`):

  ```
  “C:\Program Files\I386\cmd.exe %windir%\system32\mplay32.exe
  %windir%\Media\chimes.wav".
  ```

  For UNIX, you must include a reference to `/bin/sh` in the path.

For examples using the Run Process action to integrate AR System with other applications, see the *Integration Guide*, “Running external processes (Run Process),” page 259.

Service action

Use the Service action in:

- Active links
- Filters
- Escalations

The Service action triggers filters that have an Execute On condition of Service. You can now use the Service action in all three types of workflow:

- Active links and filters—The Service action uses the input field mapping along with an optional database request ID to provide output data to the current transaction for use in further processing. In this case, no changes are required to the database.
- Escalations—The Service action used in an escalation does set values in the database, if output data is provided.

The Service action uses fields and values in the input field mapping to determine values for the output field mapping. The Service action can perform its service on the current form or on any other form in the database. The Service action can work with an AR System web service to obtain external services or with a Set Fields filter action to consume an internal AR System service.

The filter called by the Service action accesses a request by using the input field values in the transaction, or by retrieving values from the database, and returns output values to the calling request.

NOTE

If the filter started by a Service action runs a Push Fields action, the output field value list is not affected. If the filter runs a Set Fields action, the output field value list uses any modified values.

WARNING

If a user is running a pre-7.1 version of BMC Remedy User and activates a Service active link action, the user receives the following error message:
`AR Warning 96 (Capability not supported by this AR System client)`

▶ To define the Service active link, filter or escalation action

1 Right-click the If Action or the Else Action panel header.

2 Choose Add Action > Service.

3 From the Data Source list, select SERVER or SAMPLE DATA:

- To define a specific server and form, select SERVER, and go to step a.
- To allow the server and form to be defined at runtime, select SAMPLE DATA, and go to step b.

a (*Data Source is SERVER*) Select the Server Name and Form Name for the source data.

Servers that appear in the Server Name list are those to which you are currently logged in.

b (*Data Source is SAMPLE DATA*) Define the sample and runtime server and the sample and runtime form:

- Use the Sample Server Name and Sample Form Name fields to select the server and form name to use while defining the Open Window action.
- Use the Runtime Server Value and Runtime Form Value fields to identify the fields on the current form or the AR System keywords that will define the source server and form for the Service action.

For more information about using the SAMPLE DATA data source, see “Using a dynamic data source or destination in workflow” on page 71.

4 In the Request ID field, select the field that will contain the Request ID of the source form entry to use in case the Service filter needs to retrieve data from the database.

This creates a Request ID mapping. When the transaction data is not sufficient to complete the Service action’s output mapping, the Service filter uses the Request ID mapping, if valid, to retrieve values from the database for the fields that are not in the input field list or in the transaction.

If the Service filter is able to fulfill the output mapping by using the input mapping, the Request ID mapping is not used. In other words, the Service action gives preference to transaction values. If Request ID is NULL, the output value will also be set to NULL.

The Request ID value for the action is resolved before firing the Service, based on the Request ID mapping. If the Request ID mapping does not use the associated form’s Request ID field, then the mapped field value is treated as the Request ID for the target or service form.

The Request ID mapping is required when creating the Service action, but its value can be NULL or invalid at runtime.

5 In the Input Mapping table, map the input fields and values or value expressions to provide input values to the service filter.

6 In the Output Mapping table, map the fields and values to receive the output data from the service filter.

7 Save the Service active link action.

Set Fields action

Use the Set Fields action in:

- Active links
- Filters
- Escalations

The Set Fields action sets field values *in the current request*. You can use the Set Fields action to load specific values into selected fields each time certain conditions are met. This enables you to automate field updates for a request.

For example, a filter could automatically set the Status field to Assigned when a name is entered in the Assigned To field, or an active link could change the field menus on the Type and Item fields depending on the user's selection from the Category menu.

NOTE

To move data *from* the current request *to* another request, use a Push Fields action. See "Push Fields action" on page 119.

Set Fields data sources

The value set in a field can include static values and AR System keyword values, as well as a value retrieved from another data source. You can retrieve a value from any of the following general types of data sources:

- A field in the same form.
- A field in another form on the same server or a different server.
- A result from an SQL command. See "Assigning values through SQL statements" on page 244.
- A result from a filter API plug-in service (filters and escalations only). See "Assigning values by issuing requests to a Filter API Plug-In service" on page 249.
- The result of a function. See "Assigning values using function results" on page 251.
- A result from an independent system process. See "Assigning values from process results" on page 253.

- A result from an arithmetic operation. See "Assigning values by using arithmetic operations" on page 255.
- A result from accessing a Web service (filters and escalations only). See the *Integration Guide*, "Creating web service clients," page 66.
- A result from a DDE request. See the *Integration Guide*, "Using active links with DDE," page 290.

The procedures in this section describe how to set field values by using AR System keywords, static values, and data from the current request or from any form on an AR System server. For information about using the other data sources to set field values, see the referenced sections and guides.

NOTE

Because filters and escalations execute with administrator permissions, field values set through a filter or escalation are updated even if the user does not have permission to change the field. However, active links execute with the permissions of the user, so field values set through an active link are updated in the database only if the user has permission to change the field. For information about changing the filter permissions, see "Restricting filter data retrieval by user permissions" on page 301.

Defining a Set Fields action

To define a Set Fields action, you first identify the data source option for the field value or values to be set. The procedures in this section describe how to create a Set Fields action that uses the following data source options:

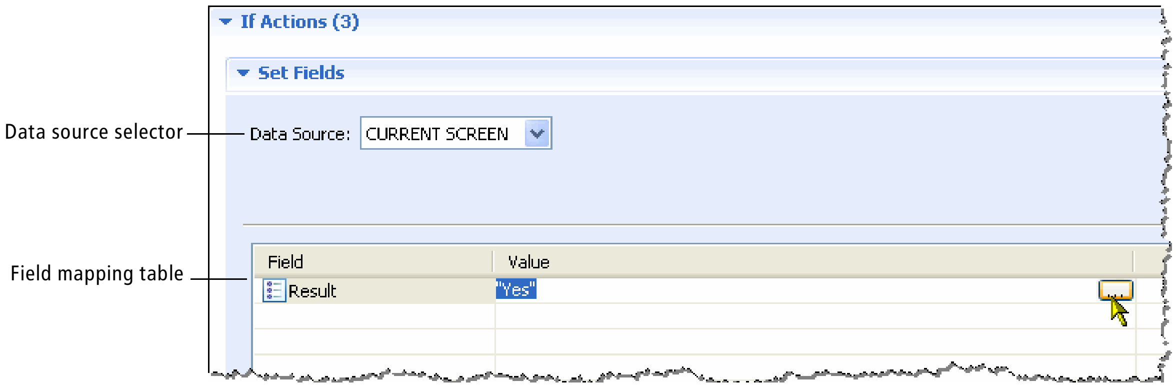
- CURRENT SCREEN (active links) or CURRENT TRANSACTION (filters and escalations)
- SERVER (active links, filters, and escalations)
- SAMPLE DATA (active links, filters, and escalations)

For information about using the data source options SQL, FILTER API, and WEB SERVICES, see the sections referenced in "Set Fields data sources" on page 129.

Depending on the option selected, the Set Fields panel changes to allow you to define the specific data to retrieve. For all data source options, the Set Fields panel includes a table containing the columns Field and Value, where you define the fields to be changed with the Set Fields action, and the value to enter in each field.

For example, Figure 4-15 shows a Set Fields action in the Sample: DialogYes active link from the AR System sample application.

Figure 4-15: Set fields panel for active link using CURRENT SCREEN data source



Data Source: CURRENT SCREEN

Field	Value
Result	"Yes"

Using the CURRENT SCREEN and CURRENT TRANSACTION data sources

When you define the data source for a Set Fields action as CURRENT SCREEN (active links) or CURRENT TRANSACTION (filters and escalations), the field is set with a value that is not from the database. You can set the field value by getting a value from another field in the current request (active links or filters), by defining a static value, by using an AR System keyword, or by using the result of a function, process, or arithmetic operation.

▶ To use the CURRENT SCREEN or CURRENT TRANSACTION data source

1 Right-click the If Actions or Else Actions panel header.

2 Choose Add Action > Set Fields.

3 From the Data Source list, select CURRENT SCREEN (active links) or CURRENT TRANSACTION (filters and escalations.) This is the default setting.

4 In the field mapping table, click the first blank row in the Field column, and then click the ellipsis button to open the Field Selector dialog box.

5 In the Field Selector dialog box, enter the field to which you want to assign a value in the Field column.

6 In the Value column for this field, build an expression to define the value to set, as described in "Mapping fields to values" on page 68.

You can set the field value using another field in the current request, a keyword, a static value, or other data sources, such as the result from a process, function, filter API. For information about using external data sources, see the sections and guides referenced in "Set Fields data sources" on page 129.

7 When the expression is complete, click OK to close the Expression Editor dialog box.

NOTE

If you enter a server name and form different than the CURRENT SCREEN, but you do not reference a field from another form, there is no need to retrieve a value from the database, and the server name and form revert to CURRENT SCREEN when the workflow object is reopened. Also, If you enter a form different than the CURRENT TRANSACTION but you set the field by using a static value or a keyword or if you do not set the field from another form, there is no need to retrieve a value from the database, and the form reverts to CURRENT TRANSACTION when the workflow object is reopened.

8. To change additional fields using the same Set Fields action, continue selecting fields in the Field column, and then define the appropriate expression in the Value column.

NOTE

If you set multiple fields in one action, the order of the set fields might execute differently on BMC Remedy User and web clients and might give you different results. If you have dependent fields, you should always break up the set fields into multiple Set Field actions so that field dependencies are executed correctly. This guarantees that BMC Remedy User and web clients work the same.

Using the SERVER data source

When you define the data source for a Set Fields action as SERVER, you can set the field value by getting a value from another request or form, as follows:

- For *active links*, from another request in this form or from another form on this server or another server.
- For *filters and escalations*, from another request in this form or from another form on this server.

You can also use static values, AR System keywords, and the results of a function, process, or arithmetic operation with the SERVER data source.

To use data from a server that is determined dynamically at runtime, use the data source SAMPLE DATA instead of SERVER. See “Using the SAMPLE DATA data source” on page 135.

▶ To use the SERVER data source

1. Right-click the If Actions or Else Actions panel header.
2. Choose Add Action > Set Fields.
3. From the Data Source list, select SERVER.

4 (*Active links only*) From the Server Name list, select the server that contains the data to be retrieved.

To cause another server to appear in the Server Name list, you must be logged in to that server. You can also enter the server name manually, but you must be able to log in to the specified server with the current user name and password. Otherwise, the Login dialog box does not appear for the other server.

5 In the Form Name field, select the form from which the Set Fields action should retrieve data:

- If the source form is the primary form for the active link, filter, or escalation, no change is required. The primary form appears in the Form Name field by default.
- To select a different form as the source form, click the ellipsis button and then select the appropriate form from the Available Forms list.

  To locate a form quickly in the list, use the Filtering Options or the Locate field in the Form Selector dialog box. See the *Introduction to Application Development with BMC Remedy Developer Studio* guide, "To filter the contents in an object list," page 38.

6 Use the Qualification field to build an expression to find the request that contains the values you want to retrieve. Click the ellipsis button to use the Expression Editor.

You can also type the expression directly in the Qualification field. See Chapter 3, "Building qualifications and expressions."

In the Expression Editor, the primary form for the active link, filter, or escalation is the "current form." If you selected another form in the Form Name field (step 5), you can use fields from that form in the expression as well.

WARNING

When you use the Qualification field to retrieve data for the Set Field action, the action runs a query to retrieve a list of the requests to get data values from. Therefore, optimize the Set Field If qualification for best system performance.

7 From the If No Requests Match list, select a handling option to control how the system responds when no matches are found in the selected form.

- **Display 'No Match' Error**—Returns an error message and stops processing.
- **Set Fields to $NULL$**—Sets field values obtained from a qualification expression to `NULL` unless the values are static or are based on keywords.

  For example, suppose a field value is based on this expression:

  ```
  $fieldName$ + " on " + $SERVER$
  ```

  If no matches are found, `$fieldName$` is set to `NULL`, and only the static value and the value based on the keyword appear in the field:

  ```
  on serverName
  ```

8 From the If Multiple Requests Match list, select a handling option to control how the system responds when multiple matches are found in the selected form.

- **Display 'Multiple Match' Error**—Returns an error message and stops processing.
- **Use First Matching Request**—Inserts the value of the first request that meets the qualification.
- **Set Fields to $NULL$**—Sets field values obtained from a qualification expression to `NULL` unless the values are static or are based on keywords.

  For example, suppose a field value is based on this expression:

  ```
  $fieldName$ + " on " + $SERVER$
  ```

  If multiple matches are found, `$fieldName$` is set to `NULL`, and only the static value and the value based on the keyword appear in the field:

  ```
  on serverName
  ```

- **Display a List (active links only)**—Displays a selection list so the user can select the appropriate request. The selection list uses the form's result list specification.
- **Use First Matching Request Based On Locale**—Returns localized information to determine the match.

  The contents of the Locale field (special field ID 160) are used along with the current user's locale to select the best match. If there are several matches with the same locale, then the first match is returned.

  If there is no localized server or no Locale field in the form, this option becomes the same as Use First Matching Request option. For more information about using the Locale field with localizing search menus, see the *Form and Application Objects Guide*, "Localizing search menus," page 586.

9 Use the field mapping table to map the fields and values from the source form to the current request as follows:

- To use matching field IDs to automatically map the values of all matching fields from the source form to the current form, select the Matching IDs check box. The field mapping table is disabled, and you cannot select specific field mappings.

  In this case, all matching field IDs (except for table columns, panel holders, and core fields such as Modified Date and Request ID) from the source form are automatically set in the current form.

TIP

When you use Matching IDs, the values are dynamic. This means that only the setting is stored as part of the Set Fields action, and not the actual values. Therefore, you can add data fields to these forms at a later stage and the action will use them when executed.

- To use the Auto Map dialog box, click Auto Map.

  The Auto Map dialog box opens with all matching fields entered in the list. To delete a field you do not want to update, select the field and click Remove. After making selections in the Auto Map dialog box, click OK.

TIP

You can use Auto Map to map fields according to the field name or field ID. When you use Auto Map with field names, the mappings are dynamic. However, when you use Auto Map with field IDs, the mappings are stored with the form. In this case, if you later add fields to the forms, they will not be part of the action and must be added manually.

- To map specific fields and to use an expression to define the value, follow the steps described in "Mapping fields to values" on page 68.

  In the Field column, select the fields to which you want to assign a value. In the Value column, use the expression editor to define the value to set from the source form.

  To set the field value using other data sources, such as the result from a process, function, filter API, and so on, see the sections and guides referenced in "Set Fields data sources" on page 129.

10 Save to save the active link, filter, or escalation.

Using the SAMPLE DATA data source

To create a Set Fields action that uses runtime information to identify the source server and form, use the SAMPLE DATA data source. For general information about using SAMPLE DATA, see "Using a dynamic data source or destination in workflow" on page 71.

To use a dynamic data source in a Set Fields action, your application must include the following objects and functionality:

- If the server and form name for the source data are taken from fields on the primary form, the workflow or user interaction must be designed to populate these fields before executing the Set Fields action.
- The sample form must contain fields with the same field IDs as the potential Set Fields source form on the runtime servers. You use the "sample" fields on this form to complete the field mapping section of the Set Fields action. The "sample" form *must* contain field IDs that exist on any dynamic source forms that are used at runtime.
- All potential Set Fields source forms must contain fields with the correct field IDs.
- If the server and form name for the source data are not taken from keywords, the primary form must contain fields to hold the runtime values of server name (for active links) and form name (for all workflow types). These fields can be populated by workflow or the values can be entered by a user (if active link).

Special considerations for Set Fields actions

For *active links*, Set Fields actions that execute on After Submit or After Modify update the specified fields on the screen. However, they do not write the values to the database, nor do they update the change flag (Save button).

You can use the Set Fields action to load values into hidden fields, into fields not in the active view, or into fields in no views at all. However, you cannot use the Sets Fields action for panel holders.

When you use a set fields action with attachments, the order of the attachments might change.

Values loaded into diary fields with the Set Fields action are handled as follows:

- Filters and escalations add the new text to the old text.
- Active links replace the new text.

This difference of functionality is an important consideration whether you might use an active link in one situation, but a filter in another.

Wait action

Use the Wait action in:

- Active links

The Wait action suspends a guide in BMC Remedy User so that the user can interact with a field. After making a response, the user can continue the guide by pressing Tab or by clicking a button. The user can also terminate the guide during a Wait action.

For example, to create a training guide, you can use a series of workflow actions to walk a user through a form, and use the Wait action to pause the guide while the user completes each field. For a detailed example see "Creating interactive guides" on page 155.

NOTE

The Wait has no effect in the web client or outside of a guide. Also, the Wait action does not work for the Search execution condition.

▶ To define a Wait active link action

1. Right-click the If Action or the Else Action panel header.
2. Choose Add Action > Wait.
3. In the Label for Continue Button field, enter the text you want to appear on the Continue button in the Prompt Bar in BMC Remedy User, for example, "Continue" (the default), "Next Step," or "Finish."
4. Save the active link or filter.

Chapter

5 Defining guides and guide actions

This section describes how to use BMC Remedy Developer Studio to create and modify active link guides and filter guides.

The following topics are provided:

Process for creating guides

A guide is a workflow program consisting of a series of active links or filters that can be used for a variety of tasks. When creating guides, it is helpful to plan them out first. Use the following steps before you begin creating active link guide objects.

Step 1 Plan the guide.

- What will the guide be used for? To lead users through a form? To invoke a dialog box? To create a computational subroutine? To specify an entry point guide in the Application List field used on a home page form?
- What class of user will it guide? Advanced users? Or will you have to guide novice users step-by-step through a form?
- What will the guide form look like? Will you have to create a simple form for novice users?
- Will the guide be shared among multiple forms?
- Will your guide function as an access point for applications?

Step 2 Design the guide.

- How many steps or dialog boxes will the guide require?
- What will be the order of workflow in the guide and are there points where the guide should take different actions, depending on conditions?
- If you are using dialog boxes, plan the layout of forms, fields, and buttons.
- Will you require data validation?
- What will your prompts say if it is a navigational or training guide?

Step 3 Create the active links or filters that you want to use within the guide.

- Use only one active link or filter for each step.
- Identify what guide actions you will need to use to control the order of workflow. Actions specific to guides are Call Guide, Exit Guide, Goto Guide Label, and Wait. For information about these actions, see "Specifying workflow actions" on page 63.
- Identify where you will need to enter Guide labels and keep track of any Goto Guide Label targets used.

NOTE

If you reuse or copy active links or filters for your guide that were designed for other applications, make sure they do not have any Execute On conditions and that the actions, associated forms, and other specifications are appropriate for use in the guide.

Step 4 Create the guide.

a Set the appropriate specifications for the guide as described in "Creating guides" on page 140.

b Add the appropriate active links or filters to the guide as described in "Adding workflow to the guide" on page 142.

c If you are using the Go to Guide Label active link action, add the labels needed for Go To targets.

d Specify the appropriate guide permissions.

e Specify the appropriate guide change history.

f Create or modify the help text for a guide.

For more information about permissions, building and using change history, and creating help text, see the *Introduction to Application Development with BMC Remedy Developer Studio* guide, "Working with existing objects," page 48.

Step 5 Debug your guide before sending it to your users.

- Use active link logging to generate a list of active links that are executed and the order in which they are executed.

  You can create a log file of active link activity that includes all of the active links in a guide. This option logs information about active link activity for each operation, including which active links executed and whether active link execution was successful. Active link logging is enabled in BMC Remedy User. For information about enabled active link logging, see BMC Remedy User help.

- Use filter logging to generate a list of filters that are executed and the order in which they are executed.
- Provide visual cues as the guide executes, so users knows what actions to expect.
- Make sure the entry point guide appears on the Application List field on your home page form and that the link starts the guide.
- Create Message active link actions as temporary placeholders to understand how your guide works.

Step 6 (BMC Remedy User only) To send the completed guide to users through email, choose File > Send > Form > Mail Recipient.

Your MAPI-compliant email tool is launched with the guide as an attachment so that you can send the guide to the appropriate users. When users receive the guide attachment, they can drag the icon to their desktops and use it as a shortcut to start the guide in BMC Remedy User.

When modifying a guide, you do not need to send the modified guide to your users. The shortcut that you sent to your users is a pointer to the guide definition that resides on the AR System server.

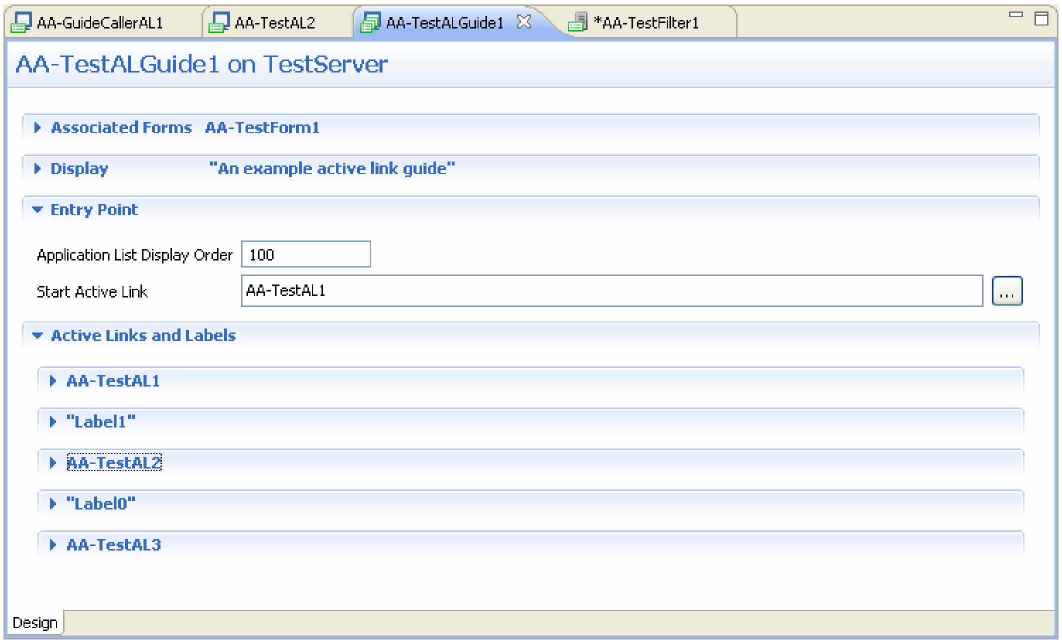
After you have deleted a guide, tell your users to delete the guide shortcuts from their desktops. If users try to start a guide after it has been deleted from the server, they receive an error message.

Creating guides

The procedures in this section describe how to create an active link or filter guide. For both types of guides, you define the basic guide settings, and then add the existing active links, filters, and guide labels that will make up the workflow of the guide.

Figure 5-1 shows an example active link guide in the editor area.

Figure 5-1: Creating an active link guide



AA-TestALGuide1 on TestServer

Associated Forms AA-TestForm1

Display "An example active link guide"

Entry Point

Application List Display Order 100

Start Active Link AA-TestAL1

Active Links and Labels

AA-TestAL1

"Label1"

AA-TestAL2

"Label0"

AA-TestAL3

Setting up a guide

This procedure describes how to create and name the guide, associate it to a form, enter descriptive information about the guide, and how to make an active link guide act as an entry point.

▶ To define basic settings for active link guide or filter guide

1. From the All Objects list, open an existing active link guide or filter guide, or create a new one.
2. In the Associated Forms panel, click Add.
3. In the Form Selector dialog box, select the appropriate form, and then click OK.

   To locate a form quickly in the list, use the Filtering Options or the Locate field in the Form Selector dialog box. See the *Introduction to Application Development with BMC Remedy Developer Studio* guide, "To filter the contents in an object list," page 38.

   The form or forms that you associate with the guide determine which active links or filters you can include in the guide.

   You can associate a guide with more than one form. If so, the guide is considered shared, and special considerations apply. See "Shared workflow" on page 32.
4. If you associated the guide with more than one form, select the correct primary form from the Primary Form list.

   The primary form is the reference form for the guide.
5. (Optional) In the Display panel, enter the Application List Label for the guide.

   Make the Application List Label descriptive and indicative of the guide's function. Application List Labels can be as many as 255 bytes.

   - For an active link guide that acts as an entry point, the Application List Label appears in the Application List on the home page form, and as the name of the entry point in the server object list in BMC Remedy User and the web client.

     If you do not supply an Application List Label, the guide name is used as the guide label.
   - For filter guides and active link guides that are not entry points, the Application List Label appears only in BMC Remedy Developer Studio, for example, in the Guide Selector dialog box when you create a Call Guide action.
6. (Optional) In the Description field, enter a description of the active link guide suitable for users. You can enter a maximum of 2000 bytes.

   - For an active link guide that acts as an entry point, this description appears in the message area of the Object List dialog box in BMC Remedy User. (It does not appear when the object list is viewed in the web client.)
   - For filter guides and active link guides that are not entry points, the contents of the Description field appear only in BMC Remedy Developer Studio, as with the Application List Label field.

7 (Optional, active links only) For active link guides that will be an entry point, open the Entry Point panel and enter values to define the entry point:

- In the Application List Display Order field, enter a numerical value. This value determines the location of the guide in the Application List on the home page form.
- In the Start Active Link field, select the starting active link for the guide. This active link can use different actions, but it should at least use the Open Window action.

  Entering values in these two fields in the Entry Point panel has the effect of making an active link guide an entry point. Entering a value in one of these fields makes the other field required.

For more information about using entry points in an application and about the home page, see the *Form and Application Objects Guide*, "Defining entry points and home pages," page 433.

8 Save the active link guide or filter guide, supplying a name for the guide.

The guide name appears in the following locations:

- In the server Object List in BMC Remedy Developer Studio.
- For active links that are entry points with no Application List Label defined, the guide name appears in the Application List on the home page and in the object list.

Guide names must be unique on each AR System server. There is no enforced convention for specifying guide names, but it is helpful to make the name descriptive and indicative of the guide's function. Guide names can be up to 80 characters, including spaces. Names can include double-byte characters, but avoid using numbers at the beginning of the name.

Adding workflow to the guide

The procedure in this section describes how to add existing active links or filters to a guide. At runtime, the guide executes the selected active links or filters in the order in which they appear in the list. You can add the same active link or filter to a guide more than once.

This procedure also describes how to add guide labels to the guide. Guide labels act as a marker in the list of active links or filters, and work with the "Go to Guide Label" action to control the flow of workflow. See "The Go to Guide Label active link or filter action" on page 123.

After adding items to a guide, you can take the following actions:

- Change the order of active links, filters, and labels in the guide.
- Open an active link or filter directly from the guide to edit it.
- Delete items from the list.

▶ To add active links, filters, and labels to a guide

1 Open the guide with which you want to work.

2 To add active links to an active link guide:

   a Right-click Active Links and Labels, and then select Add Active Link(s).

   b In the Active Link Selector dialog box, select the active links to add from the Available Active Links list, and then click OK.

   The available active links are those associated with the form or forms listed in the guide's Associated Forms panel.

3 To add filters or labels to a filter guide:

   a Right-click Filters and Labels, and then select Add Filter(s).

   b In the Filter Selector dialog box, select the filters to add from the Available Filters list, and then click OK.

   The available filters are those associated with the form or forms listed in the guide's Associated Forms panel.

4 To add a guide label to either type of guide:

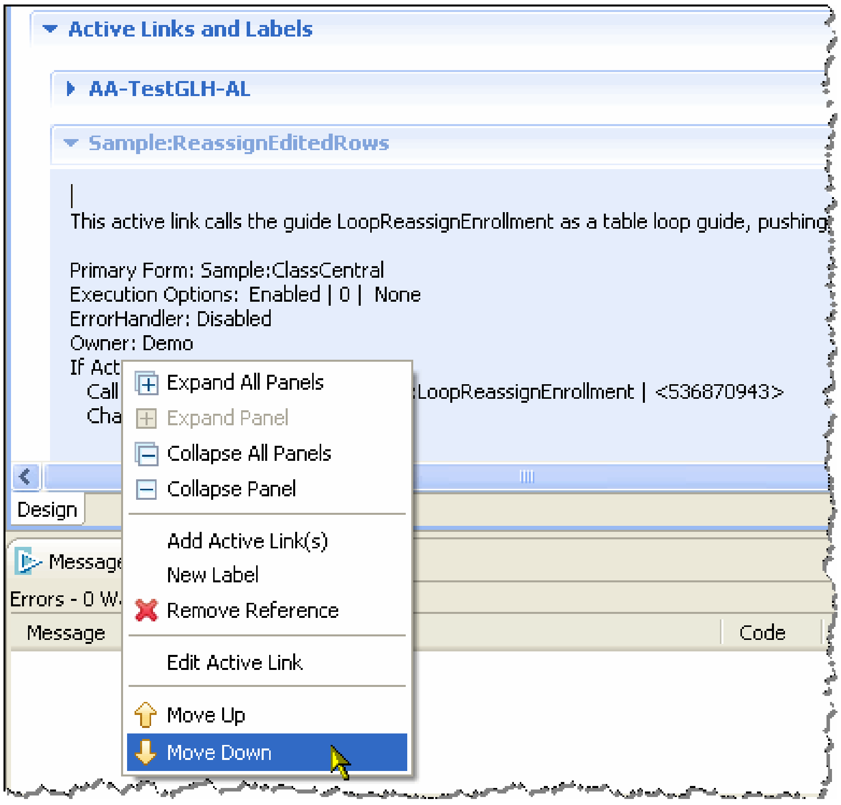
   a Right click Active Links and Labels (for active link guides) or Filters and Labels (for filter guides).

   b Select New Label, and then type the name of the label in the field provided.

   You use this label name when you define a Go to Guide Label active link or filter action. See "Go to Guide Label action" on page 86.

5 To change the order of items in the guide, right click any item in the list and then select Move Up or Move Down, as shown in Figure 5-2.

Figure 5-2: Changing the order of workflow objects in a guide



This active link calls the guide LoopReassignEnrollment as a table loop guide, pushing

Primary Form: Sample:ClassCentral
Execution Options: Enabled | 0 | None
ErrorHandler: Disabled
Owner: Demo
If Act
Call LoopReassignEnrollment | <536870943>
Cha

6 To add an active link, filter, or label before an item already in the list:

a Right click the item after which you want to add the new active link, filter, or label.

b To add an active link or filter, select Add Active Link(s) or Add Filters(s)

c To add a new label, select New Label.

When you add the new item, it appears in the list before the existing item.

7 To open an active link or filter directly from the guide list, right-click the item in the list and then select Edit Active Link or Edit Filter.

The active link or filter opens in an editor window. This allows you to make quick changes to existing active links or filters while adding them to a guide.

8 To delete an item from the list, right-click the name of the active link, filter, or label to be removed, and then select Remove Reference.

9 Save the active link guide or filter guide.

Setting active link guide or filter guide properties

This section describes how to set properties for an active link guide or filter guide, including assigning permissions (active link guides), creating help text, and editing change history.

For active link guides, you must assign Visible permission to the access control groups, roles, or users that are allowed to view and execute the active link guide. The same groups, roles, or users must also have permission to execute the active links contained in the active link guide. (Filter guides run filters, which execute on the server with administrator-level permissions, so you do not assign permissions to filter guides.)

As with other object types, use the Help Text property to provide information about the guide to users, application developers, and administrators.

AR System automatically tracks the change history of all objects in the Change History property. However, while you are developing guides or making changes to them, you can supply additional change history information.

For more information, see:

- *Introduction to Application Development with BMC Remedy Developer Studio* guide, "Updating change history," page 53
- The *Introduction to Application Development with BMC Remedy Developer Studio* guide, "Providing help text," page 54
- The *Form and Application Objects Guide*, "Assigning permissions," page 58

▶ To set permissions for an active link guide

1. In the Properties tab, expand Permissions, and then click in the Value column.
2. Click the ellipsis button.
3. In the Permissions dialog box, move groups with permission to use the guide to the Permissions list. Move groups without permission to the guide to the No Permissions list.

TIP

Make sure the same groups or users have permissions to the active links in the guide. If a user is running the guide but does not have permission to an included active link, the guide skips that active link.

▶ To add help text to a guide

1. In the Properties tab, expand Help Text, and then click in the Value column.
2. Click the ellipsis button, and then type the help text in the Help Text dialog box.

▶ To add information to the Change History property

1. In the Properties tab, expand Change History if necessary.
2. Click in the Value column for the New Description property, and then click the ellipsis.
3. Type the change description in the New Description dialog box.

   AR System automatically records the owner of a guide, the administrator who last modified the guide, and the date of the modification.

Shared guides

Guides can be shared by multiple forms. You define a shared guide like you define a guide for an individual form, except that you attach the guide to multiple forms. If you do not want the guide to be shared, select only one form. Also, share any active links in the shared guide that you want to execute on multiple forms. If a guide contains active links that do not belong to or are not shared with the current form, those active links are skipped when the guide is executed.

NOTE

Changes you make to shared active links affect all guides and forms that use them.

For more information about creating shared workflow, see "Shared workflow" on page 32.

The sequence of active links in the guide takes precedence over any execution condition previously defined for the active links. You can redirect the active links by using the Go to Guide Label action. If you are creating active links that are used *only* in a guide, do not include an Execute On condition in them. Both the condition and its execution order are ignored when the guide's active links are executed.

Guides use the following procedures when determining which form to run on:

Step 1 Search for the current form.

Step 2 If the current form belongs to the guide, run the guide against that form.

Step 3 If the current form does not belong to the guide, open a new window with a reference form and run the guide.

Because a guide can be shared by one or more forms, it also can contain multiple active links from those forms. All active links associated with the selected forms appear in the list when you add an active link to the guide. (See "Exit Guide action" on page 86 for more information.) As a result, you can access those active links *only* from the forms listed in the Form Name field. Active links that are not associated with a form that is associated with the guide do not execute.

If you select multiple forms, the guide is attached to all of them. The first form you select becomes the primary form. You can change the primary form by using the drop-down list.

Shared guide examples

The following examples illustrate shared guide behavior. Assume you have created guide X and shared it with forms A, B, and C.

▶ To create a shared guide—example 1

1 Create a Call Guide active link action for forms A, B, and C.

2 Open form A and execute the Call Guide action.

Guide X is executed on form A.

▶ To create a shared guide—example 2

1 Create a Call Guide active link action for forms A, B, and C.

2 Open form C and execute the Call Guide action.

Guide X is executed on form C.

▶ To create a shared guide—example 3

1 Designate Form B as the reference form.

2 Create a Call Guide active link action for forms A, B, C, and D.

3 Open form D and execute the Call Guide action.

A new window opens with the reference form B.

Guide X is executed on form B in a new window.

▶ To create a shared guide—example 4

1 Form A is designated as the reference form.

2 Open guide X from the Open dialog box.

Guide X is executed on form A, the reference form.

Active link guides

Here are some possible uses of active link guides:

Table 5-1: Uses for active link guides

Type of guide	Possible uses
Interactive guides	Walk users through filling out a form or direct users through each step of a procedure, much like a wizard. (These are also known as "navigational guides.") Interactive guides that use the Wait action can be implemented in BMC Remedy User, but not in the web client.
Computational guides	Used for manipulating data as a background process without any user interaction. (These are also known as "procedural guides.") A good example of a procedural guide is one that uses workflow to step through the rows that appear in a table field. For more information, see "Using active link guides in client-side table fields" on page 151.
Entry point guides	Allows users access to sophisticated tasks through the Application List field on your home page form. For example, an entry point guide allows users to access reports or Modify or Display windows. For more information, see the *Form and Application Objects Guide*, "Defining entry points and home pages," page 433.

An Active Link guide can be executed through the Open dialog box in BMC Remedy User (File > Open > Classic), through workflow, or through an email attachment that users can then drag to their Desktop as a shortcut.

The following actions control the execution of guides:

- **Call Guide active link action**—Executes or invokes a guide. For example, you can create a button on a form that uses the Call Guide active link to invoke a guide.

  You can use the Call Guide active link action to execute a guide from any client-side workflow, even from inside another guide. If a guide calls another guide on the same form, the result is a stack of guides. Control is not returned to the calling guide until the called guide exits or the nested guide executes a wait action. For more information about the Call Guide action, see "Call Guide action" on page 76.

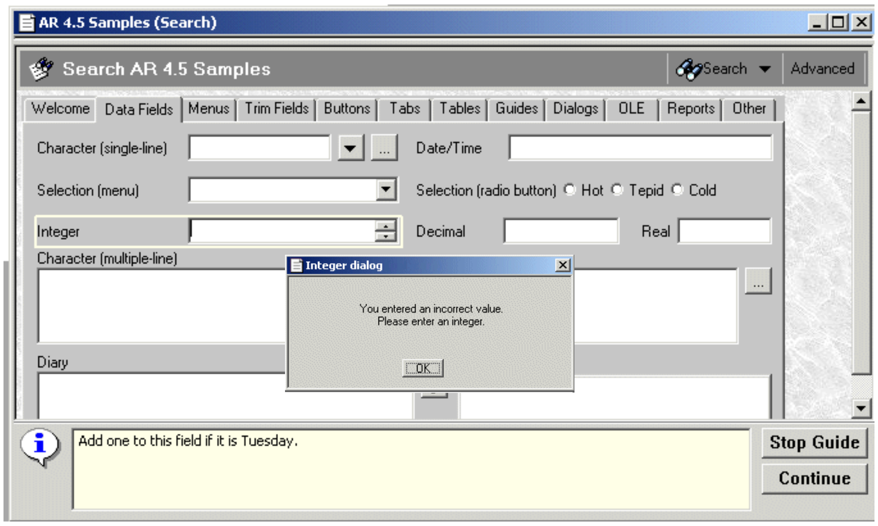
- **Go to Guide Label active link action**—Breaks a guide's normal sequence. When the Go to Guide Label active link is executed inside a guide, the action that follows the label is executed next.

  You can use this action to create data validation in a guide to make sure that users entered the correct information in a form. For example, if a user enters a number that is not recognized in the underlying form, the guide opens a warning dialog box for the user. When the user clicks OK in the dialog box, the guide continues for the user to enter the correct integer. When the user finishes working through all the steps in the guide, the guide closes.

You can also create a guide that invokes other guides if the user has entered wrong information. When the user has completed all the necessary information, the second guide closes and enters the correct information back into the first guide.

The following figure shows an example of an interactive guide using the Go to Guide Label.

Figure 5-3: Guide using Go to Guide Label active link action



You entered an incorrect value.
Please enter an integer.

Add one to this field if it is Tuesday.

For more information about the Go to Guide Label action, see "Go to Guide Label action" on page 86.

- **Exit Guide active link action**—Exits a guide. The Exit Guide action is helpful if users enter incorrect information into a guide or if conditions for the guide to run are not met. Under these conditions, you might decide to exit the guide or all guides.

  Typically, you do not need to use the Exit Guide action to quit the guide. A guide that has completed its actions successfully exits without using the Exit Guide action.

  For more information about the Exit Guide action, see "Exit Guide action" on page 86.

How active links interact with guides

Most of the active link Execute On conditions do not interact with a running guide. If the Execute On condition occurs while a guide is running, active links are executed normally, independent of the guide. For example, if a guide executes a Run Macro action that sends a request, the Execute On Submit condition is processed before the next guide action. However, active links that are available to the form and independent of the guide can interact with active links contained in a guide. See the following table for a summary of how active links interact in guides.

You can use the `$GUIDE$` keyword in qualifications to determine whether an active link is executing as part of normal workflow or in the context of a guide. When `$GUIDE$ = $NULL$`, the guide is not running. Otherwise, the name of the current guide is the value of the `$GUIDE$` keyword.

NOTE

If an action generates an error during the processing of an active link, no further actions occur.

For more information about qualifications, keywords, and active links, see Appendix A, "Operators, wildcards, keywords, and NULL values."

Table 5-2: How active links interact with guides

Execute On Condition	Interaction
Submit After Submit Search Modify After Modify Display	These conditions can be triggered only when a guide is executing a Wait action or as the result of a Run Macro action that the guide executes. The conditions are executed in sequence.
Window Open	Active links with this condition are executed during guide execution. Thus, if a guide action causes a window to open, any active links tied to the Window Open condition are executed before the next guide action is executed. If a guide is executed by an active link tied to a Window Open condition, then the guide is initialized but not started, because the user interface is not yet visible. Next, any other active links with the Window Open condition are executed, which is followed by any Set Default processing (described in the next row). When the user interface is set up, the guide executes.
Set Default	This is the most subtle of all conditions. There are several cases to consider: **Case 1:** Active links tied to a Set Default condition are executed when a guide is waiting and the user chooses the Set to Defaults menu item. **Case 2:** A guide is executed by an active link tied to a Window Open condition. In this case, the guide is initialized but not started, because the user interface is not yet visible. After the guide has initialized, the active links tied to the Set Default condition execute. After the field list and display symbols are set up, the guide is executed. **Case 3:** A guide is executed from the Open dialog box in BMC Remedy User. In this case, the guide is initialized first, and then the guide opens the form. The Window Open condition occurs and is followed by the Set Default condition. Lastly, the guide executes. **Case 4:** When the "on new" or "on search" behaviors are set to "set fields to default values," any active links tied to the Set Default condition are executed when a guide is waiting or as the result of a Run Macro action executed by the guide (after a send or a search).

Table 5-2: How active links interact with guides

Execute On Condition	Interaction
Un-Display Window Close	If you close the form on which a guide is running, the guide is terminated. Active links tied to the Window Close condition are executed before the guide terminates. Usually this occurs while a guide is waiting, but it can happen when a guide executes a Run Macro action.
Button / Menu Item	Active links that have this condition execute when a guide waits. BMC Remedy User treats form buttons as fields. If you click a button while a guide is in a wait, the guide displays the "user is off the guide" dialog and executes the active links tied to the button.
Row Double Click or Return Menu / Row Choice	These conditions can be triggered only when a guide is executing a wait action. If the field focus is changed by an action associated with the On Return trigger, then the Wait condition on the guide ends.
Gain Focus	When a guide executes a Change Field Set Focus action, any active links associated with the field's Gain Focus condition execute before the next guide action is executed. When a guide is in a wait condition, the user cannot change field focus unless the running guide is terminated.
Lose Focus	When a guide executes Change Field Set Focus action, any active links associated with the current field's Lose Focus condition execute immediately. The newly focused field's Gain Focus condition is then processed. After all Lose Focus and Gain Focus active link processing is completed, the next guide action is executed. When a guide is in a wait condition, the user cannot change field focus unless the running guide is first terminated.
Copy To New	No interaction.
Interval	If an Interval condition executes during an active link guide, the guide waits until all the workflow related to the interval condition is complete before the remainder of the guide finished.
Event	Active links with this condition are executed during execution of an event. Thus, if a guide action sends an event, any active links tied to the Event condition are executed on the target window before the next guide action is executed.

Using active link guides in client-side table fields

Active link guides are useful for creating workflow that walks you through the rows in a *client-side* table field. (For *server-side* table fields, see "Using a filter guide to loop through a table field" on page 160.) The workflow goes through every row in the table and then perform selected workflow actions on particular rows or sets of rows. For example, you can create workflow to find a specific row in a table field, then perform actions based on specific criteria. (For more information about optimization, see "Enhancing the performance of table loops" on page 154.)

You can also build workflow to:

- Skip rows, based on mathematical calculations on field values.
- Find a row that has changed values and perform an action on it, using the `$ROWCHANGED$` keyword.
- Find a row that has been selected and perform an action on it, using the `$ROWSELECTED$` keyword.

For more information about keywords, see Appendix A, "Operators, wildcards, keywords, and NULL values."

The following example procedure creates simple loop that searches through all the rows of a table field until it finds a ticket with a specific user's name on it. The workflow then sets a field with values from a column in the table field.

NOTE
This procedure assumes you already know how to create forms, fields, workflow, and guides.

▶ To create an active link guide that loops through a table field

1. Create a form (for example, **Loop Test**):
   - Add a button field (**Button**).
   - Add a table field that includes at least the following fields as columns. Use the Tree / Table property in the Properties tab to add:
     - Submitter (**Column**)
     - Request ID (**Column2**)
   - Save the form.
2. Create an active link (**Loop Test SF Active Link**):
   - Associate it to the Loop Test form.
   - Enter a Run If Qualification: **'Column' = $USER$**

     This condition executes the active link only if the value of the Submitter field is set to the person who is logged in to BMC Remedy User.
   - Add a Set Fields action and in the field mapping table, set:
     - Name: **Short Description**
     - Value: **$Column2$**

     This action puts the Column2 value, which is the value of the Request ID field, into the Short Description field.
   - Save the active link.

3 Create an active link guide (**Loop Test Guide**):
   - Associate it to the form.
   - Add the **Loop Test SF Active Link** to the guide.
   - Save the guide.

4 Create an active link (**Loop Test Active Link Button**):
   - Associate it to the form.
   - Set the Execution Options to **Button/Menu Field** and select the **Button** field.
   - Add a **Call Guide** action:
     - In the Guide Name field, select **Loop Test Guide**
     - In the Guide Field field, select the table field added to the form in Step 1.
   - Set Table Loop to "All rows."

     This action activates the guide and causes it to loop through the rows in the table field.
   - Save the active link.

5 Log in to BMC Remedy User as user **Demo** and open the **Loop Test** form in New mode.

6 As a test, create several tickets, but only one with **Demo** as the value of the Submitter field.

7 Open the **Loop Test** form in Search mode.

8 Click the table field to refresh it. The tickets you just created appear as rows.

9 Click the button field on the form.

   The active link guide is triggered and loops through all the rows in the table field until it finds a row with **Demo** as the value of the Submitter field. The workflow then fills in the Short Description field with the value of the Request ID field.

Example from the AR System sample application

In the AR System sample application, the button "Save Enrollee Edits" triggers the following actions:

- Clicking the button calls the `Sample:ReassignEditedRows` active link.
- `Sample:ReassignEditedRows` contains a Call Guide action, which calls the `Sample:LoopReassignEnrollment` active link guide.
- `Sample:LoopReassignEnrollment` is a table loop guide, which calls the active link `Sample:LoopPushEnrollee`.
- `Sample:LoopPushEnrollee` uses a qualification and a Push Fields action to locate the correct row or rows in the table and update them.
- When the guide `Sample:LoopReassignEnrollment` has completed looping through the table, it returns control to the `Sample:ReassignEditedRows` active link (the active link that called the guide).

- `Sample:ReassignEditedRows` executes its next action, which is a Change Field action that refreshes the table.
- `Sample:ReassignEditedRows` exits, and the form is now updated.

Enhancing the performance of table loops

With a large number of records, you might see slow performance in the table loop, especially as more records are selected. The following list suggests some tips for improving performance:

- Use data chunking in the table field. For example, in the Advanced Display tab of the Field Properties window, set the size of the chunk to 50. This means that the AR System processes only 50 records at a time.
- If you are not using data chunking, you can specify a maximum number of rows to be displayed in the table field, for example, 50. To set this property, enter a number of maximum rows in the Table Property tab of the Field Properties window.

  For more information, see the *Form and Application Objects Guide*, "Working with tables," page 233.
- In BMC Remedy User, set the Limit Number of Items Returned to some specified number, for example, 500 (by choosing Tools > Options > Behaviors).
- Make sure you define a Run If condition in your active link. Otherwise, the design of the table loop actually goes through every single record in the table and checks against the active links that are associated with that table.
- Limit the active links in the table loop to run only against rows that are selected. To do so, append `` `$ `` (backquote character followed by a dollar sign) to the end of the names of the active links contained within the table loop guide. (You do not need to change the name of the active link that calls the guide, the names of the table loop actions, or the name of the active link guide referenced by the call guide.)

Using interactive active link guides in BMC Remedy User

One use of a guide is to help a user navigate through a series of interactive steps. To the user, a guide looks like any other application that you open in BMC Remedy User. When using BMC Remedy User with the Prompt Bar, a guide includes the Stop Guide and Continue buttons.

When the guide prompts a user for information, the user enters the information, and then clicks Continue (or presses the Tab key). Users can quit the guide at any time by clicking Stop Guide.

A guide that walks users through a form depends especially on the Wait active link action and the prompt bar in BMC Remedy User. To create the typical steps in a navigation-style guide, use the following active links in this sequence:

- **Change Field (Set Focus)**—Sets focus to a field and highlights the field.
- **Message (Prompt)**—Provides the user with instructions and information in the prompt bar.
- **Wait**—Temporarily suspends the guide while waiting for user input.

  These first three active links should be set for every field in the form.
- **Commit Changes**—After the user enters all the information on the form, saves the information and then creates a request.

Guides can also be used to set up an application environment. For example, you can create a guide that presets a form with tabs, specific colors, default field values, and so on.

Finally, you can create a guide that uses dialog boxes, much like a wizard. This is especially important in a web environment, because the Wait active link action does not work in the web client. Instead, you must create dialog boxes for user interaction within the guide. For more information, see the *Form and Application Objects Guide*, "Using a display-only form as a dialog box," page 158.

Creating interactive guides

The following section describes how you can create an interactive (or navigational) guide. This example walks you through filling out three fields in a form. When you click the Help button, the interactive guide is activated. A halo appears around the first field you must enter information into, and instructions appear in the prompt bar. After following the instructions, you click the Continue button for more instructions.

This example uses the Wait and Call Guide actions, but you can easily include more advanced actions (for example, the Go to Guide Label action). This procedure assumes you already know how to create forms, fields, and active link workflow.

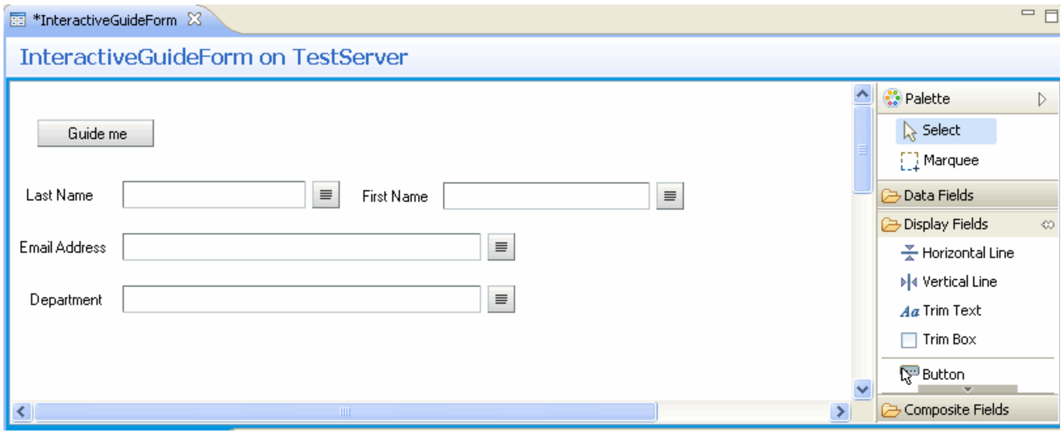
NOTE

Interactive guides are not supported in the web client.

▶ To create an interactive guide

1 Create a sample form (for example, the **Interactive Guide Form**).

Figure 5-4: Example form for interactive active link guide



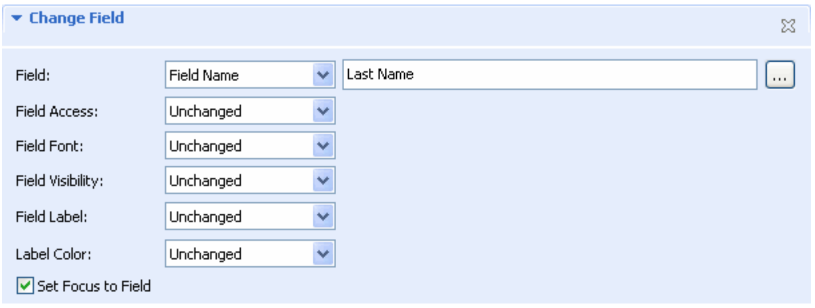
Users can complete the fields on this form by clicking Guide Me, which activates the interactive guide.

2 Create and save four active links, one for each of the fields on the form, for example **ALG: Last Name**, **ALG: First Name**, and so on.

For each active link:

- Associate the active link to the **Interactive Guide Form**.
- Do not select any Execution Options and leave the Run If Qualification blank.
- Add a Change Field action:
  - Select the appropriate field for the active link. For example, in **ALG: LastName**, select the **Last Name** field.
  - Select Set Focus to Field.

Figure 5-5: Example Change Field action



Change Field

Field:	Field Name	Last Name
Field Access:	Unchanged	
Field Font:	Unchanged	
Field Visibility:	Unchanged	
Field Label:	Unchanged	
Label Color:	Unchanged	

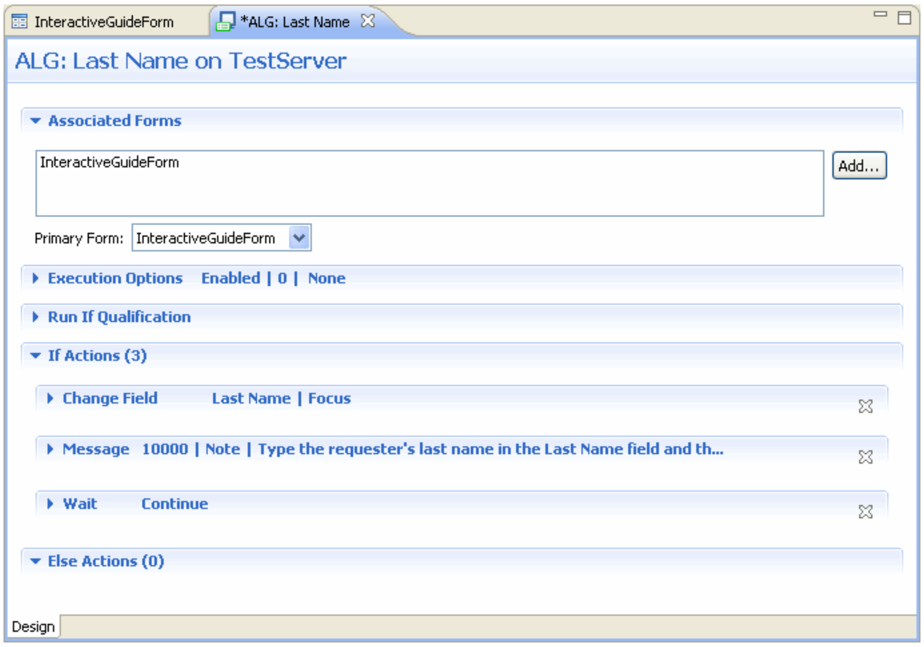
[✓] Set Focus to Field

- Add a Message action:
  - Enter an appropriate instruction for the field. For example, "Type the requester's last name in the Last Name field, and then click Continue."
  - Select Show Message in Prompt Bar.
- Add a Wait action:
  - For all active links except the one that will be last in the guide, leave "Continue" as the label for the Continue button.
  - For the last active link, enter "Finish" as the label for the Continue button.
- Assign the Public group permissions for the active link.
- Save the active link.

TIP

After creating the first active link, use Save As to create the others, then edit the actions and active link name appropriately.

Figure 5-6: Example active link for the interactive guide



InteractiveGuideForm | *ALG: Last Name

ALG: Last Name on TestServer

Associated Forms

InteractiveGuideForm

Add...

Primary Form: InteractiveGuideForm

Execution Options Enabled | 0 | None

Run If Qualification

If Actions (3)

Change Field Last Name | Focus

Message 10000 | Note | Type the requester's last name in the Last Name field and th...

Wait Continue

Else Actions (0)

Design

3 Create an active link guide (for example, **Incident Guide**):

- Associate it to the **Interactive Guide Form**.

TIP

You can create an entry point guide by clicking the Entry Point check and choosing an active link that opens a window. For more information, see the *Form and Application Objects Guide*, "Creating form entry points," page 439.

- In the Active Links tab of the active link guide, add the four active links you just created, in the following order:
  - **ALG:Last Name**
  - **ALG: First Name**
  - **ALG:Email Address**
  - **ALG:Department**
- Grant the Public group Hidden access permissions to the active link guide.
- Save the active link guide.

4 Create another active link that calls the active link guide (for example, **Call Incidents Guide**):

- Associate it to the **Interactive Guide Form**.
- Select the Button / Menu Field Execution Condition, and specify the **Guide Me** button.
- Add a Call Guide action, and specify the **Incident Guide**.
- Grant public permissions to the active link.
- Save the active link.

5 Test the active link guide:

a In BMC Remedy User, open the **Interactive Guide Form** form in New mode.

b Click the **Guide Me** button.

The focus should move to the first field and the message should appear prompting you to complete the field.

Filter guides

Filter guides are used to create reusable components of filter workflow by adding computational subroutines within filter processing. This provides for more sophisticated workflow solutions and allows easier reuse of functionality between forms.

A filter guide is a list of filters that perform a task on a particular form. You create the filters *before* you insert them into a guide. Filters that execute in the context of guides are not triggered by the same Execute On conditions that trigger a filter during normal workflow. Filters are evaluated in their order within the guide, subject to redirection by the way of the Go to Guide Label action. If a filter used in a guide does have an execution condition, its execution condition is ignored.

You use three filter actions to implement filter guides:

- **Call Guide**—Starts the filter guide. Like active link guides, the Call Guide action can be nested, calling yet another filter guide. The number of nested Call Guides cannot be deeper than the maximum depth of the filter action stack. For more information about the Call Guide action, see "Call Guide action" on page 76.
- **Exit Guide**—Terminates the filter guide. This action is ignored if it is not running inside a filter guide. For more information about the Exit Guide, see "Exit Guide action" on page 86.
- **Go to Guide Label**—Redirects the flow of execution within a guide to a specific filter. This action is ignored if it is not running inside a guide. If the guide label is not found, this action is ignored as well. For more information about the Go to Guide Label action, see "Go to Guide Label action" on page 86.

Filter guides allow a developer to group a set of filters into a single unit of work (that is, a subroutine) and call only the filters referenced in sequence inside the guide without caring about the execution order of other filters. They also let developers call the filter guide under many different circumstances tied to multiple different forms. In essence, developers can create a piece of functionality that can be called as a unit of work, and not care about the execution order across all filters. In that way, developers can focus on what the filter guide as a unit of work accomplishes. For example, take the following filters with their execution orders:

Filter Name	Execution order
A	100
B	102
C	104
D	102
E	110
F	99

By design, each filter executes in its own execution order, based on which events trigger the filter action.

By contrast, a filter guide could call these filters, in the following defined sequence:

Filter Name
A
C
E

This same filter guide can be triggered in different circumstances, at different times, from different forms.

NOTE

Filter guides do not affect the "phases" of filter processing. A Set Fields action still executes in Phase 1, a Push Fields action in Phase 2, and so on. For more information about filter phases in AR System, see "Filter processing in the AR System server" on page 177.

You use essentially the same procedures to create filter guides as you do for active link guides. For more information, see "Creating guides" on page 140.

Using a filter guide to loop through a table field

Filter guides are useful for creating filter workflow that steps through the rows in a server-side table field. A server-side table field is a field in a supporting form that allows you to manipulate data on the server and then return the result to the client, rather working with the data directly in the client. This approach improves performance by reducing the amount of network traffic between the client and the server.

The filter guide workflow goes through every row in the table and then performs selected filter actions on particular rows or sets of qualifying rows. These qualifications can be built with reference to table column values.

Because filters run on the server, you cannot use server-side table fields to highlight rows for display. You cannot select rows on server-side table fields, like you can on the client side. The user whose search initiates the server-side table loop must have at least hidden permissions to the table field.

TIP

For best performance, combine all operations on a server-side table field into a single filter guide. This avoids retrieving the data from the server more than once.

This table looping functionality works only inside the filter guide. Filters function on only one AR System server. If a server-side table field points to a form on a remote server, the filter guide does not work.

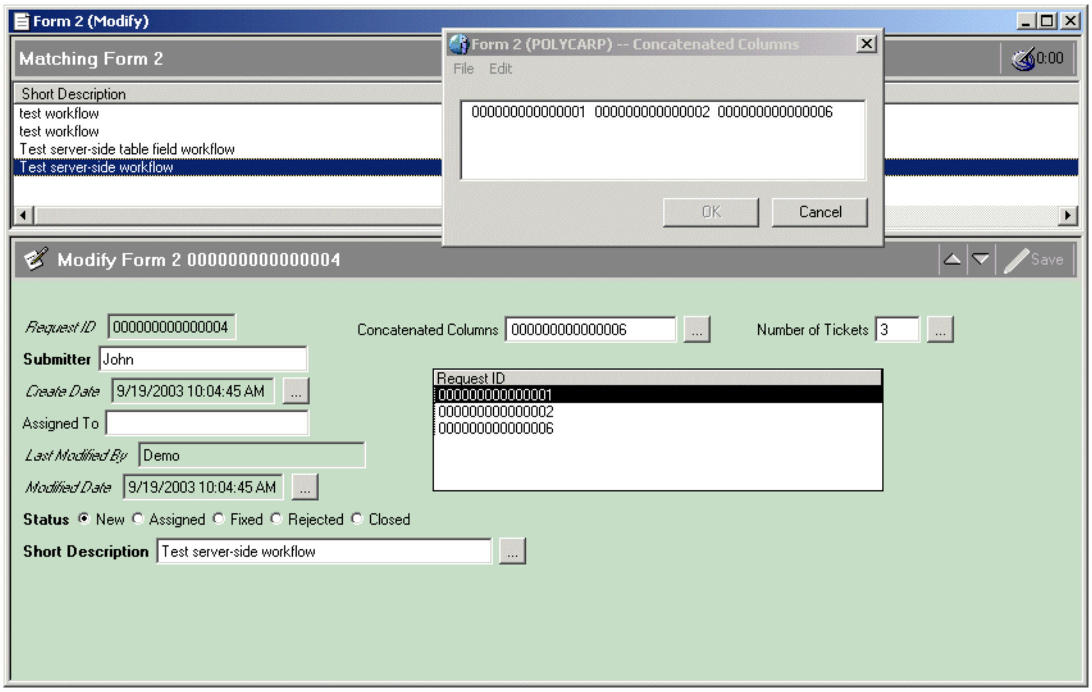
You can also perform calculations on the columns of the server-side table field, using the `COLAVG`, `COLCOUNT`, `COLMAX`, `COLMIN`, and `COLSUM` functions inside a filter guide. Use these functions just as you do with active link guides.

For more information about table fields and functions, see the *Form and Application Objects Guide*.

Server-side table field example

The following procedure constructs a simple loop that searches through all the rows of a server-side table field on a helper form (`Form1`) until it finds all the tickets with a specific user's name on them. The workflow then uses values from a column in the table field to set fields on a second form (`Form2`). The first Set Fields action concatenates all the returned values of the table field column into a character field; the second action uses the `COLCOUNT` function to display the number of columns returned.

Figure 5-7: Example of server-side table workflow


Form 2 (Modify)
Matching Form 2
Form 2 (POLYCARP) -- Concatenated Columns
File Edit
OK
Cancel
Modify Form 2 000000000000004
Save

Short Description
test workflow
test workflow
Test server-side table field workflow
Test server-side workflow

000000000000001 000000000000002 000000000000006

Request ID 000000000000004
Submitter John
Create Date 9/19/2003 10:04:45 AM
Assigned To
Last Modified By Demo
Modified Date 9/19/2003 10:04:45 AM
Status [✓ New] Assigned Fixed Rejected Closed
Short Description Test server-side workflow
Concatenated Columns 000000000000006
Number of Tickets 3

Request ID
000000000000001
000000000000002
000000000000006

You can enhance this example by including additional in-line workflow, for example, notifying a user about the number of rows returned.

NOTE

This procedure assumes you already know how to create forms, fields, filter workflow, and filter guides.

▶ To create a filter guide that loops through a server-side table field

1. Create a form (for example, **Form1**).
2. Create another form (for example, **Form2**):
   - Add an integer field to display the number of tickets (for example, **Tickets**).
   - Add a character field to display all the entry IDs of the columns returned (for example, **Concatenated Columns**). Align these fields side-by-side so that the results show all the tickets for a specific user.
   - Add a table field:
     - Select Form1 as the source form.
     - Enter the qualification **'Submitter' = $Submitter$**

       Using this qualification in the table field allows the database to select the entries that match the $Submitter$ field. This can improve performance because neither the AR System server nor the client needs to process the other entries in the form.
     - Add the Request ID field (becomes **Column**)
     - Add the Submitter field (becomes **Column2**)
   - (Optional) Hide the table field.

     Because you use server-side table fields exclusively for computation, you should hide them on forms that users interact with.
3. Create a filter (**Form2 Set Field**):
   - Associate the filter with **Form2.**
   - Add a Set Fields action:
   - In the mapping table, map the field **Concatenated Columns** to the value **($Concatenated Columns$ + " ") + $Column$**

     The workflow concatenates all the returned columns into the Concatenated Columns field and separate them with a space. $Column$ references the table column that contains the request IDs.
   - Add another Set Fields action, and map **Tickets** to **COLCOUNT($Column$)**

     The action uses the COLCOUNT function to fill the **Tickets** field with the value of how many tickets this user has created.
   - Save the filter
4. Create a filter guide (**Form2 Filter Guide**):
   - Associate the filter guide with **Form2.**
   - In the Filters and Labels panel, add **Form2 Set Field** to the guide.
   - Save the filter guide.

5 Create a filter (**Form2 Call Guide**):

- Associate the filter with **Form2.**
- Select the Execution Option **Submit**.

  This filter executes when you create tickets in Form2 and then display them.
- Add a **Call Guide** action and enter **Form2 Filter Guide** as the guide name.
- Set Table Loop to All Rows, and then select the table field on **Form2**.

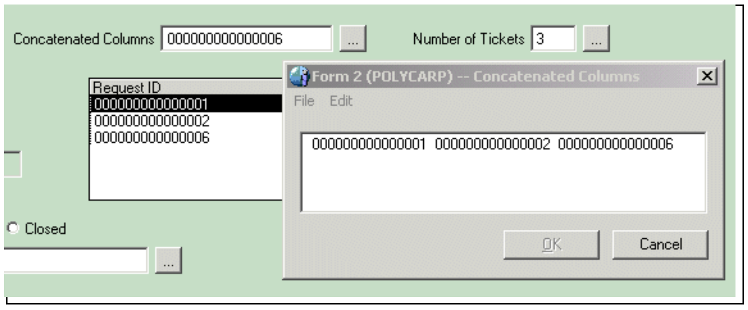
  This action activates the guide and causes it to loop through the rows in the table field.

6 To test the guide:

- Log in to BMC Remedy User and open **Form1** in New mode.
- Create several tickets, including two or three with **John** as the value of the Submitter field.
- Open **Form2** in New mode and create a ticket for John.
- Perform a search for John's tickets, then click the request in the Results List.

The filter guide is triggered and loops through all the rows in the table field until it finds the rows with **John** as the value of the Submitter field. The workflow then completes the other fields in the form with the number of John's tickets created in Form1. In addition, the concatenated entry IDs from the returned entries appear in the Concatenated Columns field.

Figure 5-8: Tickets returned and columns displayed



Concatenated Columns: 000000000000006
Number of Tickets: 3

Request ID
000000000000001
000000000000002
000000000000006

Form 2 (POLYCARP) -- Concatenated Columns

000000000000001 000000000000002 000000000000006

You can use similar functionality to loop through hundreds or even thousands of requests in the database, bypassing both the server and the client. By contrast, you *could* use client-side processing to calculate this information, but doing so increases network traffic between the client and the server and can impact performance.

Server-side table field implementation issues

Keep these server-side table field limitations in mind:

- When creating workflow for server-side tables, filters with the Call Guide action can be defined to loop over table columns. The values retrieved during the looping always comes from the *last* row of the table. To retrieve different values, postfix two special characters (`!) on the filters within the filter guide. Adding these special characters causes your filter to execute synchronously, instead of deferring into a later filter processing pass. For example, if your filter within the Filter guide is named FormA: Push New Entry to Form B, postfix those special characters to look like this:

  ```
  FormA: Push New Entry to Form B`!
  ```

- Data conversion between values in a column of one type should be able to be set or pushed to fields of a different type for most data types. But unlike client-side table fields, there are some limitations in server-side tables for the following conversions:

  CHARACTER to DECIMAL

  CHARACTER to REAL

  For example, `123.45 six` converted to decimal became `0.00`, and converted to real became `0.000000`.

  On the other hand, `123.45` does convert properly to `123.45` and `123.450000`.

- Setting values from columns of integer, real, decimal, or currency type (and sometimes character type) to date or time fields does not yield NULL for negative values such as -3 or -3.01. Instead, the lowest values of the date and time type are set.

  For example, the lowest value of date fields is `1/1/4713 BC`, while the lowest value of time fields is `12:00:00 AM`.

Chapter

6 Using buttons and menu bar items to execute active links

You can set an active link to execute when a user clicks a button in a form or chooses a menu item from a top-level menu. (The active link can be linked to a button or a menu item, but not both.) If an active link is executed from a menu item, you can also execute the same active link through a corresponding toolbar button.

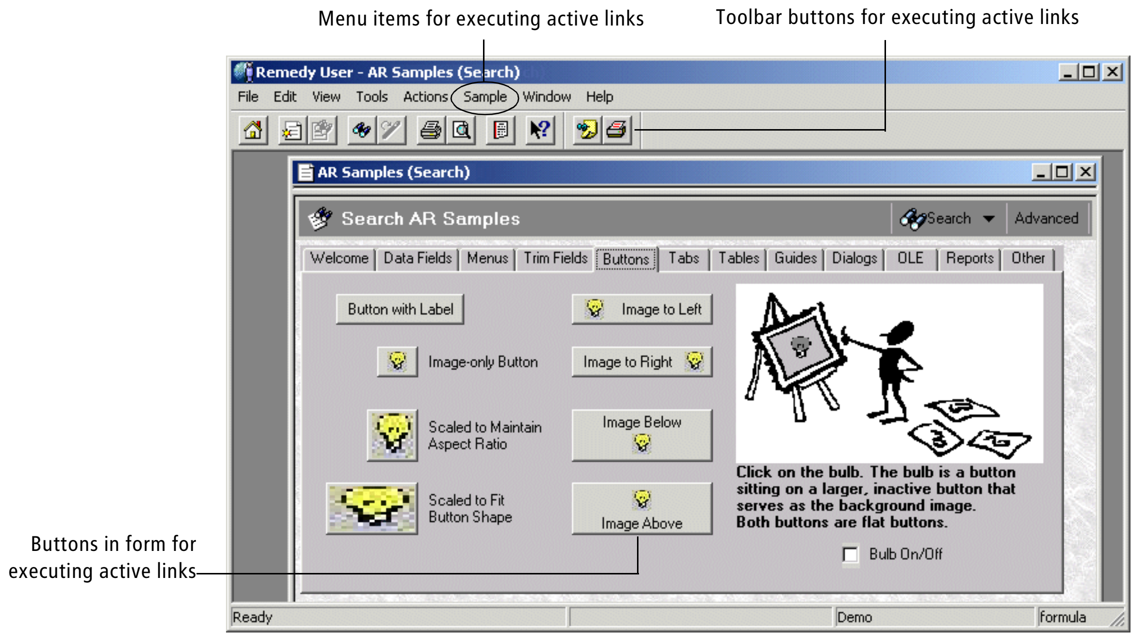
This section discusses how you can use buttons and menu bar items to execute active links.

The following topics are provided:

Understanding how buttons and menu items work with active links

Buttons and menu items are fields in AR System; you can use them in workflow as in any other field. For example, an active link can show, hide, enable, or disable buttons and menu items. Because these fields do not have data behind them and they are used to control workflow, they are called *control fields*.

Figure 6-1: Form with active link buttons and menu bar items

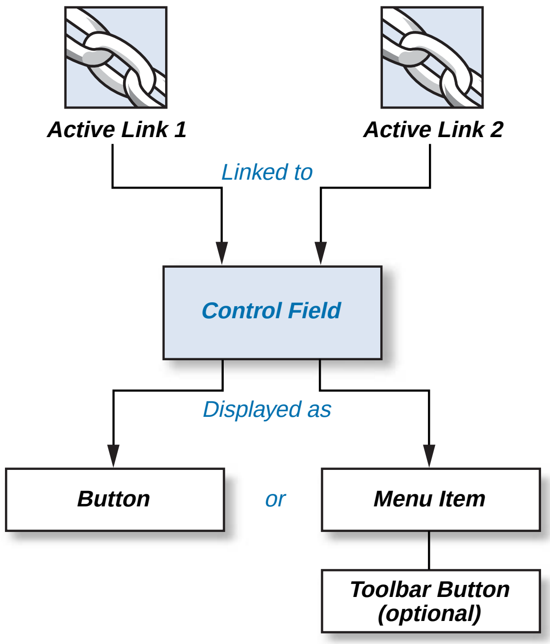


Click on the bulb. The bulb is a button sitting on a larger, inactive button that serves as the background image. Both buttons are flat buttons.

You can associate more than one active link with a button or menu item. A good use of this capability is to define active links that execute based on current conditions, such as the platform on which the tool is running.

For example, you can define an active link to execute only if the tool is running on a PC. You can then associate another active link with the same field—this time defining the active link so that it executes only if the web client is running. Users on either platform can then perform the same action to execute the appropriate active link for their platform.

Figure 6-2: Active links as buttons or menu items



To add a button to a form, see the *Form and Application Objects Guide*, "Creating button fields," page 210.

Creating and modifying menus and toolbar items

For each form, you can add menus and menu items to the BMC Remedy User menu bar. Each menu can have menu items, submenus, and menu separators. Only *menu items* can be connected to active links.

From menu items, you can also create toolbar buttons.

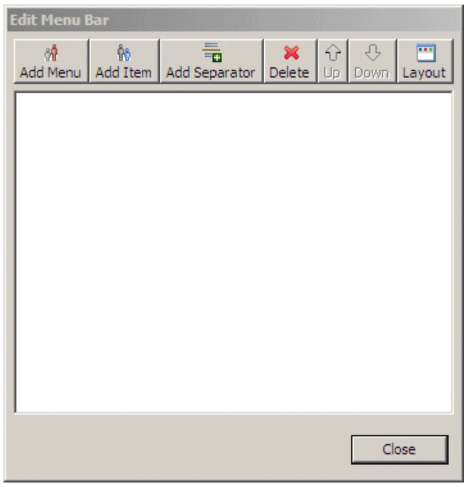
▶ To add menus, menu items, and separators

1 Open the form with which you want to work.

2 Choose Form > Edit Menu Bar.

The Edit Menu Bar dialog box appears.

Figure 6-3: Edit Menu Bar dialog box



3 Add menus, menu items, and separators by using the Add Menu, Add Item, and Add Separator buttons.

Avoid creating a menu item at the top level; otherwise, the menu item will appear in the menu bar.

4 To rename a menu or menu item:

a Select the menu or menu item.

b In the Properties tab, edit the values for Name and Label.

The maximum length for each value is 80 characters.

5 To add help text for a menu item:

a Select the menu item.

b In the Properties tab, enter the help text for Menu Item Help.

The text appears on the status bar in BMC Remedy User.

6 To specify the initial state menu or menu item:

a Select the menu or menu item.

b In the Properties tab, set Field Access to Enabled (the default) or Disabled.

c Set Visible to True (the default) or False.

Workflow can change either of these settings.

The values for all properties defined in the *Form and Application Objects Guide*, “Field properties,” page 501.

7 To delete a menu, menu item, or separator, select it, and click Delete.

Child menus and items are deleted when you delete the parent menu.

8 Save the form.

Creating accelerator keys for your menu items

You can designate one character of a menu or menu item as an accelerator key. For example, if you create a menu item called “Print Report,” you can designate the letter R, and the R is underlined when the user presses the Alt key.

When creating accelerator keys, do not choose a letter that other menus on the menu bar use. Do not duplicate the choices for menu items in your menu.

▶ To designate a character as an accelerator key

1 Select the menu or menu item in the Edit Menu Bar dialog box.

2 In the Properties tab, click the Label property, and enter an ampersand (&) before the character.

Creating toolbar buttons

You can create a toolbar button from a menu item that appears with a form in BMC Remedy User. If a user clicks the toolbar button, it behaves in the same fashion as if the user had selected the menu item.

NOTE

If you create a toolbar button from a menu separator, it is plat form-dependent, but it is usually a line or an extra space.

▶ To create a toolbar button

1 Select a menu item in the Edit Menu Bar dialog box.

2 In the Properties tab, set Has Toolbar Item to True.

3 Click the Image property, and click the ellipsis (...) button that appears in the Value column.

4 From the Image dialog box, select the image for the toolbar button.

You can select either of the following options:

- **Embedded Image**—Select this option, and click Browse. The image is embedded in the form (using more disk space).
- **Image Reference**—Select this option, and click Select to select an image from the images stored in AR System. For more information, see the *Form and Application Objects Guide*, “Working with images,” page 359.

  The only image size supported for toolbar buttons is 16 pixels wide by 16 pixels high.

5 To save the image to an area on your computer or the network, click Save to File.

6 Click OK to close the Image dialog box.

7 To add a tooltip for the button, in the Properties tab, enter text for Tooltip.

This text appears in BMC Remedy User when the user holds the cursor over the toolbar button.

8 To rearrange the order of the toolbar buttons in BMC Remedy User:
   a Click Layout in the Edit Menu Bar dialog box.
   b In the dialog box that appears, use the Up and Down buttons to rearrange the order.
   c Click OK.

   If you do not perform this step, the toolbar buttons appear in no particular order.
9 Save the form.

Associating active links with buttons and menu items

The Active Links property of a menu item enables you to choose which active links are executed with a button or menu item (and any corresponding toolbar button). For these active links, the user must belong to a group with permission for the active link *and* the button or menu item:

- If a user has access to all active links associated with a button or menu item, but not to the button or menu item itself, the button or menu item is not displayed in the form.
- If the user has access to the button or menu item, but not to any of its active links, clicking the button or menu item does not execute the active link.

For more information about setting button permissions, see the *Form and Application Objects Guide*, "Assigning permissions," page 58.

▶ To associate active links with buttons and menu items

1 Open the form with which you want to work.
2 Complete one of the following actions:
   - For a button, select the button so that its properties appear in the Properties tab.
   - For a menu item, choose Form > Edit Menu Bar, and then select the menu item.
3 In the Properties tab, click Active Links, and click the ellipsis (...) button that appears in the Value column.
4 In the Active Links dialog box, select which active links you want to view:
   - **All Active Links**—Displays all of the active links in the specified form.
   - **Available Active Links**—Displays only those active links in the specified form that are not assigned to a button or menu item.
5 To the Selected Active Links list, move the active links you want associated with the button or menu item.
6 Click OK, and save the form.

Deleting buttons and menu items

Buttons and menu items that are deleted from a form are deleted from the database.

▶ To delete a button

1 Open the form.

2 Right-click on the button and choose Delete.

▶ To delete a menu item

1 Open the form.

2 Choose Form > Edit Menu Bar.

3 Select the item, and click Delete.

▶ To delete a toolbar button

1 Open the form.

2 Choose Form > Edit Menu Bar.

3 Select the menu item from which you want to delete a toolbar button.

4 In the Properties tab, set Has Toolbar Item to False.

Modifier keywords for use in workflows

The following modifier keywords can help verify whether a user has pressed the Shift, Alt, or Control keyboard modifier keys:

- `$SHIFT_KEY$`
- `$ALT_KEY$`
- `$CTRL_KEY$`

Appendix A, "Keywords" provides a description about the keywords. You can define a workflow in BMC Remedy Developer Studio using the modifier keywords on a Control field such as push button, url-style button, image button. This can also be executed on a table field.

Using the modifier keywords

You can use the modifier keywords in any workflow. The following example describes how to use the `$SHIFT_KEY$` keyword. The workflow opens an AR System form on a new window when the user performs Shift+Click on the Control field or on the table field in BMC Remedy Mid Tier.

NOTE

`$SHIFT_KEY$` is not limited to the workflow that is defined to open a form in a new window. Also, you can use other modifier keywords in the workflow instead of `$SHIFT_KEY$`.

▶ To open an AR System form on a new window using the $SHIFT_KEY$

1 Add a workflow object on a form or use an existing workflow object, such as a button or a table.

NOTE

If you are creating a workflow to a Control field, skip step 2 through step 3.

2 Under the Properties window, add columns to the table.

3 Set the Drilldown Property to False.

4 Create an active link and associate the workflow object with a form. See "To associate a workflow object with a form" on page 36.

5 Define the execution options for active links. See "To define the execution options for filters" on page 43.

6 Under Execution Options, click the Field ellipsis button and select Button or Table Field, based on step 1.

7 Under the Run If Qualification section, click the ellipsis button to open the Expression Editor.

8 In the Expression Editor dialog box, click Keywords.

9 Select the `$SHIFT_KEY$` keyword and assign the value of `1` to that keyword. For example, `$SHIFT_KEY$ = 1`.

NOTE

You can also use `$CTRL_KEY$` or `$Alt_KEY$` and assign the value of `1` to that keyword.

10 Click OK to close the Expression Editor dialog box.

11 Right-click the If Action panel header.

12 Choose Add Action > Open Window and select a window type.

13 For the Target Location field, select New.

14 Right-click the Else Action panel header.

15 Choose Add Action > Open Window and select a window type.

16 For the Target Location field, select Current.

Chapter

7 Workflow processing

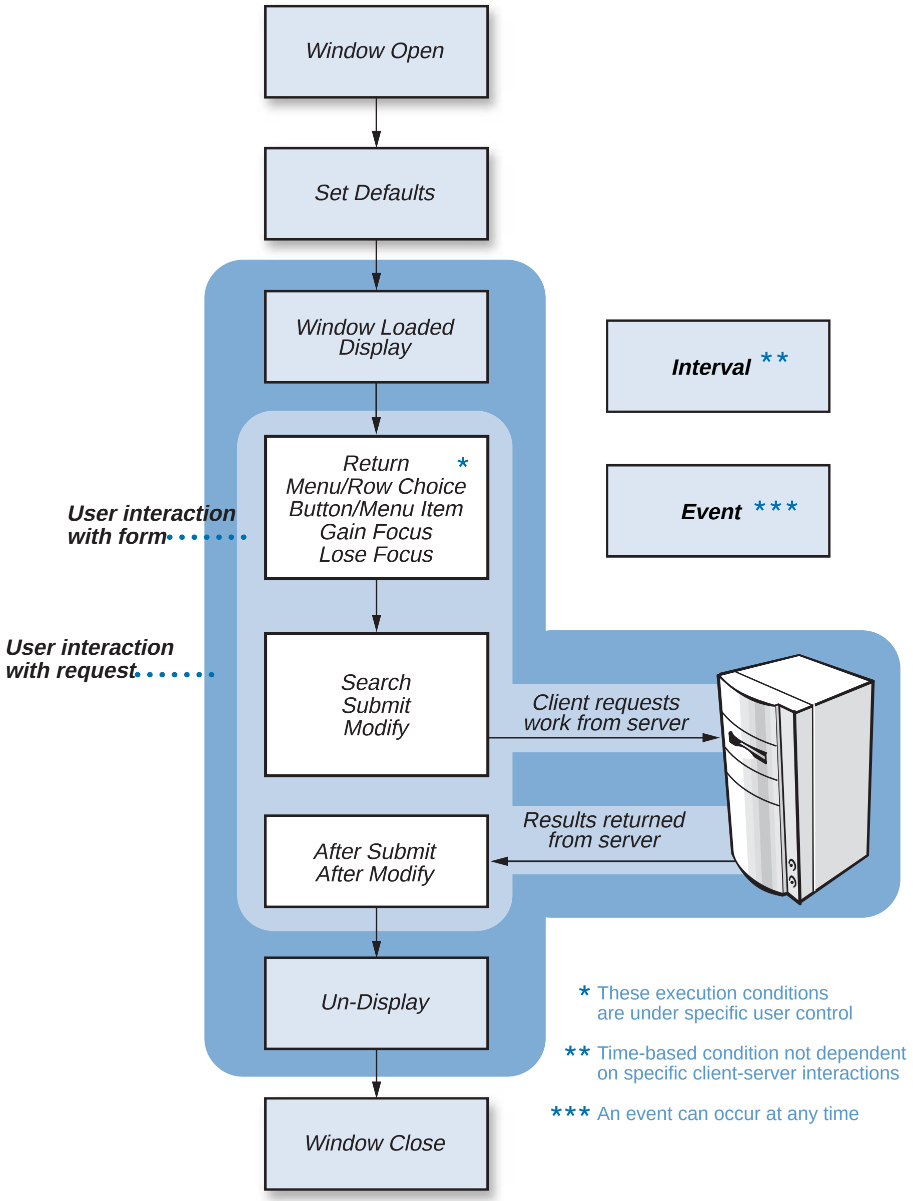
This section outlines how workflow is processed on the client side (active links) and on the server side (filters).

The following topics are provided:

Active link processing

Active link conditions have an order and relationship to one another as well as to the client and server.

Figure 7-1: Active link processing


Window Open
Set Defaults
Window Loaded Display
Interval **
Event ***
Return *
Menu/Row Choice
Button/Menu Item
Gain Focus
Lose Focus
User interaction with form
User interaction with request
Search
Submit
Modify
Client requests work from server
Results returned from server
After Submit
After Modify
Un-Display
Window Close
* These execution conditions are under specific user control
** Time-based condition not dependent on specific client-server interactions
*** An event can occur at any time

NOTE

The Interval and Execute active link conditions are not included in Figure 7-1. Interval conditions are time-based and thus not dependent on specific client-server interactions; “events” can be sent during any change of application state.

How data is loaded into a form and available to be acted on often depends on the window mode and how the user interacts with fields on the form. The order in which active links are processed can help you determine which one to specify.

Following is an example of an active link process using Window Open:

1 Use the Window Open condition to set field values when the values are required for workflow processing *before* the request is actually displayed. These values are used *only* as temporary data for validation purposes in workflow. But when the window opens, the actual values displayed in these fields are not this temporary data, but the values specified by the user preference settings.

2 Use the Display condition for any values you want the user to see when the request is displayed.

3 Use the Window Open condition to prevent the window from opening and generating an error message when the user does not have appropriate permissions.

The way your active link executes can be affected by how users have set their preferences for the web client or their BMC Remedy User Behavior options. Use caution when using Set Fields or Push Fields actions triggered by Window Open that rely on specific initial field values. For example, if your active link relies on default field values appearing, it does not work if the users sets the option to clear all fields on Window Open.

More than one active link (or filter) can execute on the same execution condition, and the output of one can affect another.

Filter processing in the AR System server

The AR System server processes filters in an execution order you specify. As the AR System server considers actions in each filter, it categorizes those actions into one of three phases. (See "Filter phases" on page 178.) In general, actions are carried out such that database operations are deferred until after all actions that modify the database have been performed. At that point, a single database transaction is opened and all the database operations are performed. This keeps transaction time and table and row locks in the database to a minimum.

Some complex applications, however, require database operations to be performed immediately. For example, an application might be used as a subsystem of another application, where the results of the subsystem's work are needed immediately by the calling application. For more information, see "Overriding filter processing phasing" on page 184.

TIP

You can combine the Filter, SQL, and API logs to "see" how AR System interacts with the database and how the different phases interact with each other, especially if you are creating complicated workflow. For more information, see the *Optimizing and Troubleshooting Guide*, "Combined server and client logging," page 70.

Filter execution order

Execution order determines the order in which filters are executed. When you create a filter, you assign it an execution order value. If one filter has a lower execution order than another, the actions for the first filter are executed or queued before the actions for the second one.

Filter phases

Actions within filters are carried out in one of three phases. All Phase 1 actions are carried out immediately. Phase 2 and Phase 3 actions are queued to be carried out later. See “Filter processing example” on page 180.

Using a phased approach to filter processing helps to make sure the following actions occur:

- Notifications are sent and that processes are run only *after* the database operations are successful. If any database operation fails, all subsequent actions are suppressed, and database changes are rolled back. So, the system defers to a final phase the operations that should not run until database transactions have been committed and there is no chance that a rollback occurs.

  For example, you can create a filter to notify Shipping that a purchase order is ready to be fulfilled. If, during the filter processing, an error occurs, the chain of events are rolled back, and the transaction is not committed to the database. Without phases in filter processing, Shipping would be notified and the equipment would be sent, but no record of the task would be committed to the database.
- All data values are complete and available to the notification that is sent as well as to Push Fields operations that create related records.
- Obtaining a write lock (which is an exclusive lock) for a Push Fields operation is delayed until the point at which a write is performed to the main data table. This shortens the duration of the exclusive portion of the transaction, and improves throughput.

Phase 1 actions

The following Phase 1 actions are always performed as soon as they can be:

- Call Guide
- Exit Guide
- GoTo
- Go To Guide Label
- Log to File
- Message
- Set Fields

Phase 2 actions

Phase 2 actions are queued when they are encountered and performed after all Phase 1 actions from all filters execute. If a Phase 2 action triggers more Phase 1 actions, then those Phase 1 actions are performed before the next Phase 2 action. If a Phase 2 action triggers more Phase 2 actions, then they are added to the end of the Phase 2 queue.

Each request with a filter executing against it has its own Phase 2 queue. So if a Push Fields action triggers the execution of a nested filter against the target request, the Phase 2 action in that filter that apply to that request are queued separately in the Phase 2 queue for that request. Those Phase 2 actions are run after the Phase 1 action in the nested filter are complete. The remaining Phase 2 actions in the outer filter are run after the nested filter Phase 2 actions.

Phase 2 actions include:

- Push Fields
- Direct SQL

Phase 3 actions

Phase 3 actions do not include any database interaction. All Phase 3 actions encountered are queued and performed after all Phase 1 and Phase 2 actions.

Unlike the separate Phase 2 queues, there is one Phase 3 queue for the request against which filter execution is originally triggered and any other requests which have filters

Phase 3 actions include:

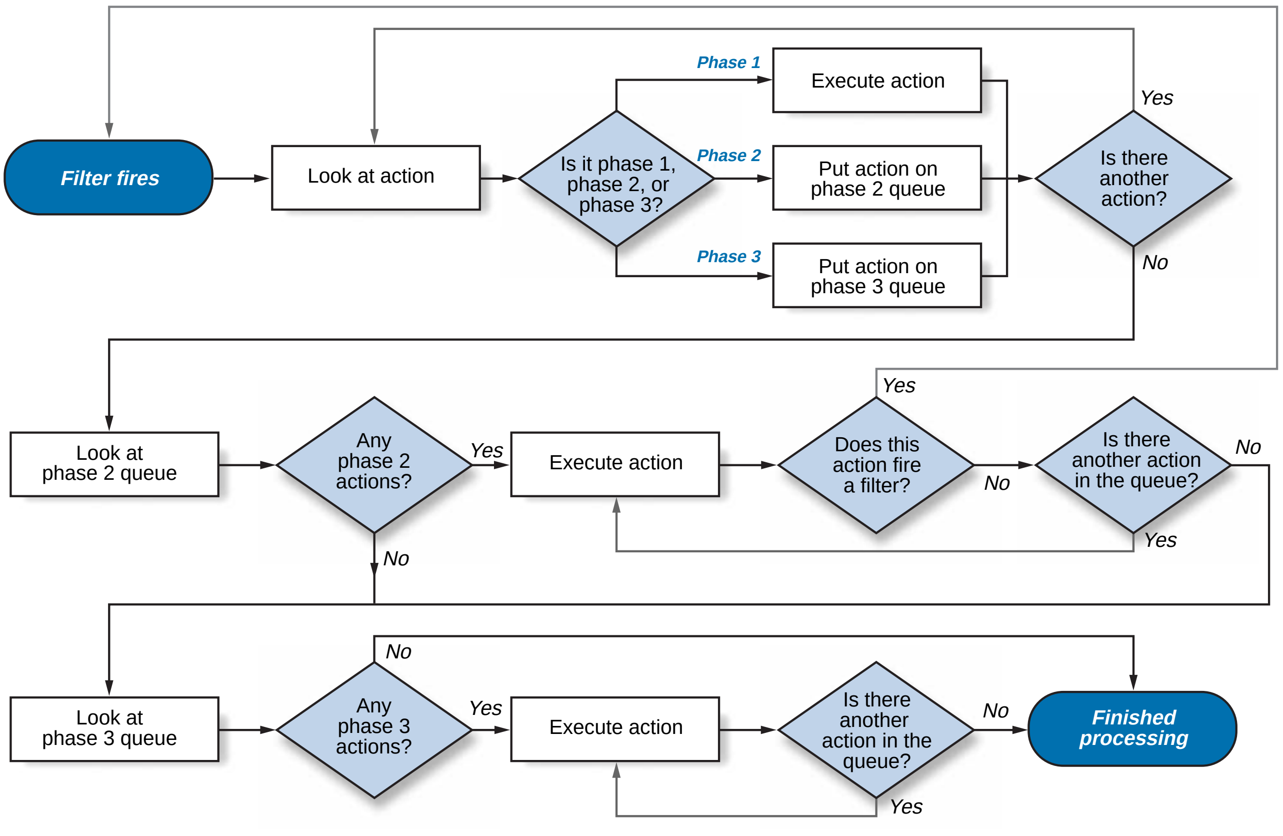
- Notify
- Run Process
- DSO

Phase 3 actions are presumed successful, but can fail and not affect the operation's success.

How AR System processes filters

Figure 7-2 illustrates how AR System processes filters.

Figure 7-2: Filter processing



Filter processing example

Suppose we have form A, form B, and form C.

Form A has two filters, `F1` and `F2`, that execute when form A is modified.

Filter `F1` on form A has the following attributes:

- Execution order 1
- Set Fields action (A1)
- Push Fields action (A2) to form B
- Notify action (A3)
- Another Set Fields action (A4)
- Another Push Fields action (A5) to form C

Filter `F2` on form A has the following attributes:

- Execution order 2
- Set Fields action (A6)
- Notify action (A7)

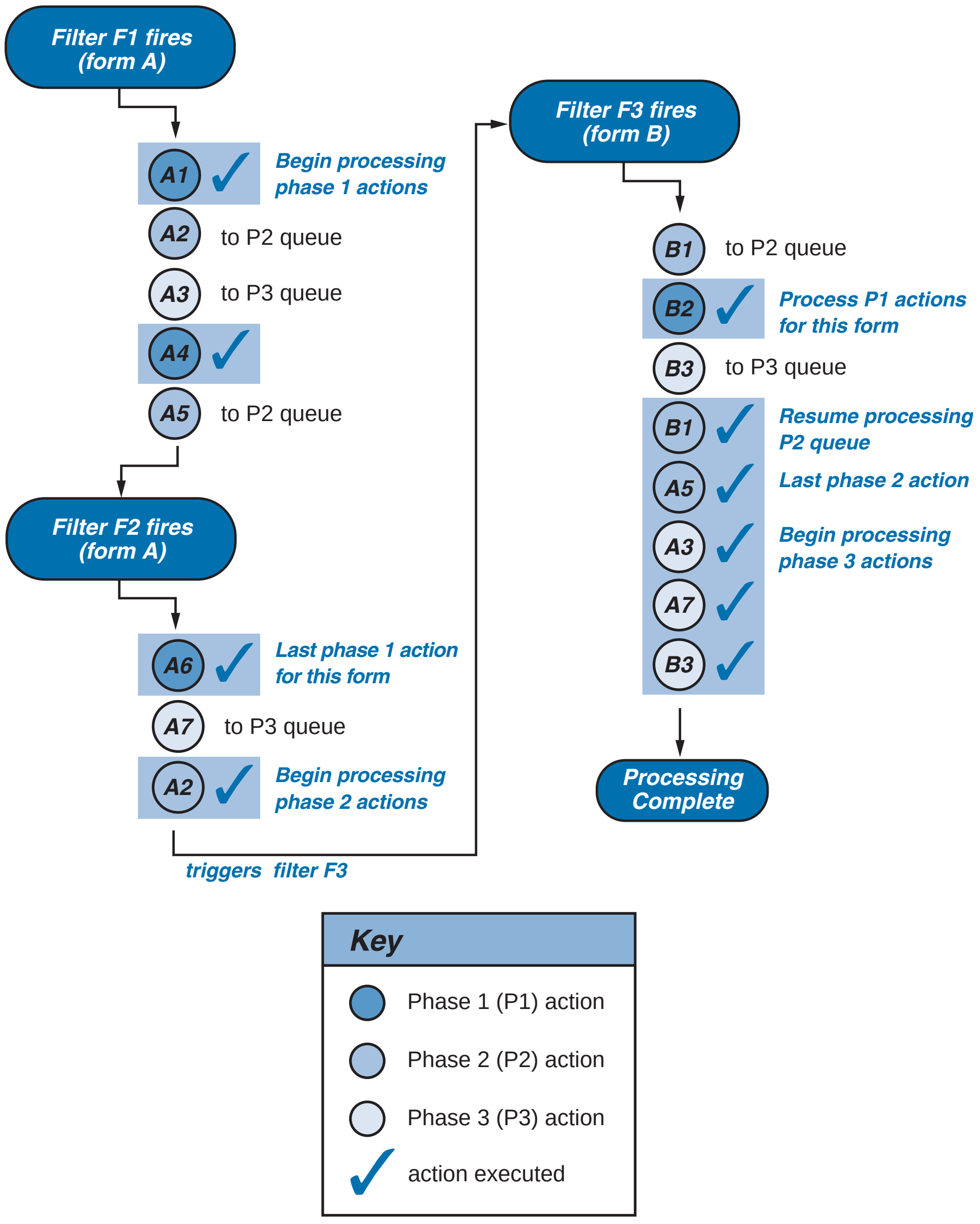
Form B has a filter, `F3`, that fires when form B is modified. Filter `F3` has the following attributes:

- Push Fields action (B1) to form C
- Set Fields action (B2)
- Notify action (B3)

Form C has no filters.

The following figure and steps illustrate the filter processing in the example.

Figure 7-3: Filter processing example



Step 1 The server acts on the filters by execution order, and for each filter it examines all of the actions to be performed. For filter `F1`, the server performs action A1 immediately because it is a Phase 1 action. It puts action A2 on the Phase 2 queue for the form A request. It puts action A3 on the Phase 3 queue. It then performs action A4 because it is a Phase 1 action. Finally, it puts action A5 on the Phase 2 queue for the form A request.

Step 2 The server then moves on to the second filter, `F2`. It performs A6 immediately because it is a Phase 1 action. It puts action A7 at the end of the Phase 3 queue.

Step 3 Having performed all available Phase 1 actions (A1, A4, and A6) the server performs the first action on the current Phase 2 queue, A2.

Step 4 A2 modifies an request on form B. Form B's filter, `F3`, fires when the form is modified . The server examines all of the actions to be performed by filter `F3`. The server puts action B1 on the Phase 2 queue for the form B request. It performs action B2 immediately because it is a Phase 1 action. It puts action B3 at the end of the Phase 3 queue.

Step 5 After performing B2, the server has no more Phase 1 actions, so it performs the action on the current Phase 2 queue, B1. The Phase 2 queue for the form B request is empty, so the server returns to the form A request and performs A5, the action on the form A request Phase 2 queue. Those actions do not cause any filters to fire.

Step 6 After performing all actions on the Phase 2 queues the database transaction is committed. At this point, the operation succeeds.

The database transaction means the operation fails if any Phase 1 or Phase 2 action fails, because the DB transaction is rolled back. It also means that if all Phase 1 and Phase 2 actions succeed, the operation succeeds even if a Phase 3 action fails. So, in the example, if one of the notifications cannot be sent for some reason, the modify operation still succeeds.

Step 7 The server then performs the actions on the Phase 3 queue—A3, A7, and B3.

The following table summarizes the phase to which each action belongs and in what order each action was carried out.

Phase	Actions (in order of execution)
Phase 1	A1 A4 A6
Phase 2	A2 (fires filter F3)
Phase 1	B2 (from filter F3)
Phase 2	B1 A5
Phase 3	A3 A7 B3

If an error occurs anytime during Phase 1 or Phase 2 processing, all processing stops, the database changes are rolled back, and the server handles the error as described in "Error processing" on page 187.

Filter phasing exceptions

There are exceptions to the phasing process:

- For *get* actions (filters with Execute On set to Get Entry) and *delete* actions, Phase 1 and Phase 3 actions occur together.
- For *get* operations, there is generally no database change, so the actions need not be phased. However, there might be database changes as the result of Push Fields actions that the get operation triggers.
- For *delete* operations, you cannot defer actions because the delete action removes the current record, so the record would not be available for the actions later. The subsequent actions need to run when the data is still present.

The following tables further describe when certain operations are processed.

Filter phasing for Create and Merge operations	
1 Filters run (including Phase 1 actions).	
2 Create operation. (Entry ID is created.)	*Database transaction occurs.*
3 Phase 2 actions occur.	
4 Phase 3 actions occur.	

Filter phasing for Set operations	
1 Filters run (including Phase 1 actions).	
2 Phase 2 actions occur.	*Database transaction occurs.*
3 Entry is modified in the database.	
4 Phase 3 actions occur.	

Filter phasing for Get operations	
1 Data is retrieved from the database.	
2 Filters run (including all Phase 1 and 3 actions).	
3 Phase 2 actions occur.	*Database transaction occurs.*

Filter phasing for Delete operations	
1 Filters run (including all Phase 1 and 3 actions).	
2 Phase 2 actions occur.	*Database transaction occurs.*
3 Entry is deleted in the database.	

Overriding filter processing phasing

Sometimes, processing filter actions in phases is limiting or inefficient. For example, an application might be used as a subsystem of another application, where the results of the subsystem's work are needed immediately by the calling application.

You can override phasing for Direct SQL, Notify, Run Process, and Distributed Server Option actions. If you override Push Fields actions, the actions take the intermediate values, but the database transaction is deferred.

WARNING

Be *very* careful about how you use overriding filter processing phasing. Improper use can produce inconsistent, confusing, and potentially inaccurate results. It is recommended that you avoid using this capability unless it is critical to your functionality.

Remember, if you override the phases for a filter, Request IDs and create dates are not available during a Create operation. Also, a modified date is not available during a Create or Modify operation (`$TIMESTAMP$` might be a suitable workaround in these situations). Furthermore, if there is a failure in the server, users might receive notifications for a request that does not really exist.

Finally, the data values used during a given operation are the data values at the point at which the action is performed. This data value might not be the final value for some of the fields. The operation can be performed with intermediate values instead of the final values that you might expect.

As an alternative, display-only fields are available, and they retain their values throughout the transaction. If your workflow needs an intermediate value at a later stage, storing it in a display-only field for later access is often the best answer. You get all the advantages offered by filter-action phases with respect to the transaction in progress and still retain the intermediate values.

The following sections discuss two methods for overriding filter processing phasing:

- Using a special override naming convention
- Releasing pending operations

Using a special override naming convention

To override the phases for a specific filter, for example, one of a sequence of filters executed in a filter guide, you can force the filter run actions in-line (sequentially) by using a special convention for the filter name. When you create or modify the filter, its name must end in a back quote character followed by an exclamation point (`` `! ``). For example, a filter named `SendNotification` that has a Notify action performs the action in Phase 3. However, a filter named `` SendNotification`! `` that has a Notify action runs the notification during Phase 1.

Put the Phase 3 actions to run in Phase 1 in a filter with the override naming convention in a filter guide. Put actions that require normal phasing in filters without the special name elsewhere in the guide.

Releasing pending operations

Use the `Application-Release-Pending` Run Process command in a filter or escalation to open a database transaction to perform operations as various filter actions are performed instead of deferring the operations.

When an `Application-Release-Pending` command is run, any Phase 2 actions queued by child filters are executed. This makes the changes due to these action available to the remaining action in the parent filter.

For example, assume form A has workflow that includes a Push Fields actions that pushes values to form B and that form B has a filter with a Modify condition that is triggered by the Push Fields action. The form A workflow is suspended while the form B filter is executed and the form B filter is said to be a child of the form A workflow. If the form B filter runs Phase 2 actions, they remain queued until the form A workflow completes all its Phase 1 actions. This means that the form A workflow cannot use the modifications made by the form B filter Phase 2 actions.

If the form A workflow includes a Run Process action with the `Application-Release-Pending` command after the Push Fields action and the form B filter execution, the Phase 2 actions queued by the for B filter are run at that point so that the rest of the form A workflow actions can use the modifications they make.

NOTE

In release 7.6.04, `Application-Release-Pending` was modified to automatically run in filter Phase 1. Therefore, you no longer need to use the special filter naming convention to override filter phasing for this command.

Delaying entry creation

To delay the creation of individual form entries so that they can be created in bulk, use the `Application-Set-Filter-Phasing` Run Process command. This command determines whether form entries are created when the workflow operation to create them occurs or whether they are created in bulk during a later filter phase.

For more information, see "Process commands" on page 261.

Error processing

The server reacts to error conditions in various ways, depending on what operation you are running. In general, errors are handled according to where you are within filter processing with regard to the database transaction:

- If an error is encountered during filter processing and the filter has no error handler, all processing immediately stops, no further filters or actions run, and an error is returned back to the caller.
- If the error occurs during a Phase 1 or 2 action but *before* the database transaction is completed, the operation is cancelled, the database transaction is rolled back, and no change is made to the database.
- If an error occurs during Phase 3 actions, processing continues, and changes are not rolled back. However, if you override filter processing so that your Phase 3 actions are processed *before* the database transaction, errors are handled according to the "before-transaction-commit" rules. The error handling itself does not change, but using the override causes actions that are normally deferred to take place before the database transaction; therefore, any errors encountered are handled as an error *before* the transaction is complete and terminate the transaction and roll back.

To summarize, overriding phasing for a filter moves Phase 3 actions into Phase 1 actions. Such actions take on the behavior of other Phase 1 actions (that is, if an error occurs while processing the action, all subsequent filter processing is halted, and any database changes are rolled back).

Error handling filters

You can specify an error handling filter that the server executes before it performs the default error handling described in the preceding section. Using an error handler, workflow can recover from an error or change the handling of Message Actions of type Error produced by third-party AR System applications when called from your custom applications.

The procedure for creating an error handling filter is much the same as creating any filter. To specify and enable an error handler filter, use the Error Handler in the filter editor. See "Creating filters" on page 43. For an example of an error handling filter, see the Sample application distributed with AR System. The `Sample:GetMoreInfo` filter has an error handler, `Sample:GetMoreInfoError`.

The error handler filter and the filter whose errors it handles must be associated to the same form in the Associated Forms list for the filter. This ensures the error handler has access to the fields the other filter references. If no error handler is specified or if the error handler is not enabled, AR System handles errors as described in "Error processing" on page 187.

Error handler execution

When a filter is executed as an error handler, the execution differs from the usual filter execution in the following ways:

- The Execution Order value and Execute On selections on the Basic tab of the Filter properties window are ignored. The Enable check box is not ignored. The filter must be enabled to be executed.
- Three keywords that give information about the error that caused the error handler to be called are defined:
  - `$ERRNO$`—The number of the first error on the stack.
  - `$ERRMSG$`—The error message for the first error on the error stack.
  - `$ERRAPPENDMSG$`—The appended message for the first error on the error stack, if any.

  Some error conditions place more than one error number, message, and appended message on the error stack. The error handler has access to the first one only.
- Filter processing phasing is overridden. Phase 3 actions are not deferred; they are run immediately. The error handling filter executes in-line like a filter whose name ends with a back quote and an exclamation point (`!). BMC recommends that you use this naming convention for error handling filters to indicate that they execute in-line. However, this is not required.
- When all the If actions of the error handling filter complete successfully, the error is considered handled, the error information and keyword is cleared, and the filter in which the error occurred continues execution with the next action. If another error occurs during the execution of the filter, the error handler is executed again.
- If the Else actions of the error handler filter are executed or if the If actions terminate in error as described in the next section, the error information is not cleared, the remaining actions in the filter where the error occurred are not run, and the server continues to process the error. If the filter was called from another filter, the server might call another error handler as described in "Error handling for nested filter calls" on page 189. Otherwise, the server processes the error as described at the beginning of "Error processing" on page 187.

Errors in error handling filters

An error can occur while an error handler is running its If actions. If the error handler has its own error handler, that error handler is called (unless calling that inner error handler would exceed the Maximum Stack of Filters value). If that error handler handles the error, the If actions of the outer error handler continue to run.

If the outer error handler does not have its own error handler or if its error handler does not handle the error, the remaining If actions of the outer error handler are not run, the error information for the error in the If actions is added to the error stack below the original error, the outer error handler does not handle the original error, and the error is treated as described in the next section on nested filter calls.

An error can occur while an error handler is running its Else actions. If the error handler has its own error handler, that error handler is called (unless calling that inner error handler would exceed the Maximum Stack of Filters value). If that error handler handles the error, the Else actions of the first error handler continue to run.

If the outer error handler does not have it own error handler or if its error handler does not handle the error, the remaining Else actions of the outer error handler are not run and the error information for the error in the Else actions is added to the error stack below the original error.

Because the outer error handler was running its Else actions when the error occurred whether or not the error is handled, the outer error handler does not handle the original error and it is treated as describe in the next section on nested filter calls.

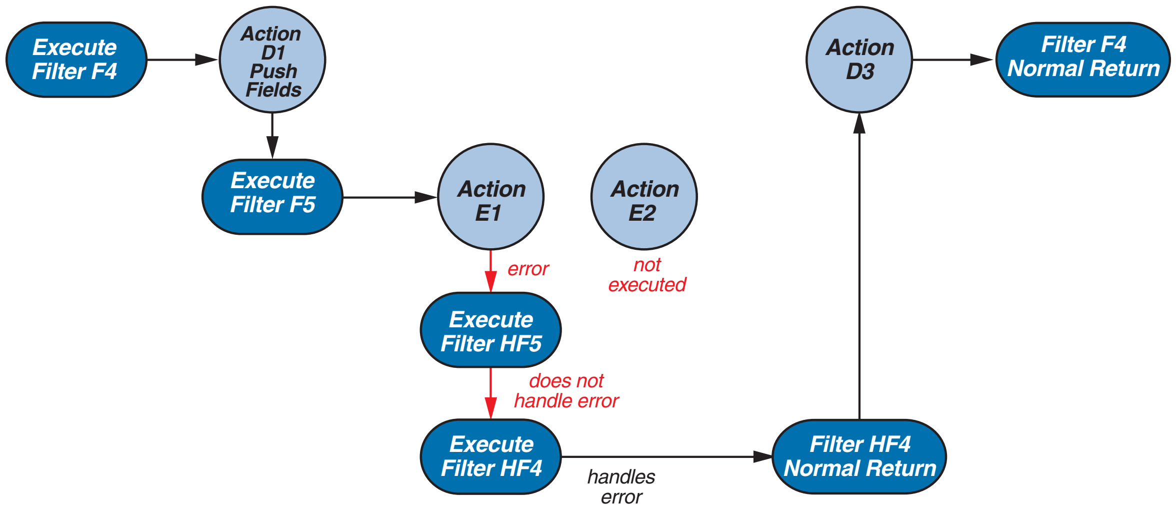
Error handling for nested filter calls

A filter can be executed from another filter, so a filter can execute some number of levels down in a stack of executing filters. If an error occurs in such a filter and the error is not handled, either because the filter has no error handler or because the error handler does not handle the error, the remaining actions in the filter where the error occurred are not run and the error is passed one level up the stack.

The AR System server treats the error as it had occurred in the filter that executed the filter where the error actually occurred. That filter can, in turn, handle the error or not. Thus, the error can pass up though the stack of running filters, terminating each until an error handler handles it. If no error handler handles the error, the AR System server performs default error handling.

If an outer error handler handles an error that occurs in a nested filter call, the remaining actions in the nested call, if any, are not run. In this case, the server does not report an error, but some actions are skipped. Figure 7-4 illustrates how an action can be skipped. Filter F4 has error handler HF4. When F4 runs a Push Fields action to another form, it executes filter F5. Filter F5 has error handler HF5. An error occurs in F5, but HF5 does not handle it. Filter HF4 is called and handles the error in F5. But the actions in F5 that were pending when the error occurred are not run because HF4, the error handler for F4, handled the error.

Figure 7-4: Error in Nested Filter Execution, Error Handler for Outer Filter Handles Error



Using filter guides with error handling filters

To write a filter that handles several errors, write the qualification to select exactly the errors the filter must handle, store the values of the filter error keywords in fields in the form, and call a filter guide. The qualifications of the filters in the guide select the filter to process each error. If the last filter in the guide has no qualification, it must process the error or errors the other guide filters do not process. The guide filters are not error handlers. They get the error keywords values from the fields in the form, not from the keywords. The actual error handling filter that called the guide succeeds because its If actions are run and do not fail. The guide filters must process all the errors that pass the qualification of the error handler.

To handle errors from separate actions differently, put the actions in separate filters with different error handlers and put those filters in a filter guide.

Tracing active link, filter, or escalation activity

You can create a log file of active link, filter, or escalation activity. This option logs information about active link, filter, or escalation activity for each operation, including what executed and whether execution was successful.

Active links are executed on the client, so logging is activated in BMC Remedy User. For information about activating active link logging, see BMC Remedy User Help.

Filters and escalations are executed on the server. For more information, see the *Optimizing and Troubleshooting Guide*, "Using log files," page 43.

NOTE

Errors encountered during the execution of escalations are recorded in the `arerror.log` on the AR System server. If an escalation terminates in error consistently, a message is written to the log *every time* the escalation is executed and fails. If the escalation qualification does not select any requests and the alternate actions, if any, are run, a message is written to the escalation log on the server (by default, `arescl.log`).

Chapter

8 Using Analyzer to find problems in your applications

The BMC Remedy Developer Studio Analyzer option can help you optimize and troubleshoot your AR System application. When you run the Analyzer, it finds objects that are not consistent with the Analyzer rules that you select. The rules check for potential performance problems, such as inefficient database queries, and potential errors, such as workflow that cannot be run.

The following topics are provided:

Analyzer rules and messages

Analyzer finds potential performance problems and errors by checking the server objects that you select against the rules that you specify. Table 8-1 lists the rules that you can use to analyze your objects.

Table 8-1: Analyzer rules (Sheet 1 of 2)

Category	Rule	Description
Structural	Inefficient Queries	Identifies inefficient queries, such as queries that do not use an index
	Unused Workflow	Identifies workflow objects that have no Execute On options enabled and are not included in any guide
	Server Referenced by Name	Identifies workflow object that include references to servers by name and might not be portable
	Unreachable Else Actions	Identifies workflow objects with Else actions that have no Run If qualification
	Process Validity	Identifies workflow objects that contain Run Process actions or Set Fields actions with a `$PROCESS$` value that have invalid server references
	Invalid Access Point	Identifies forms or call guides that are referenced as part of another application and that are not declared as an access point in the current application
	Duplicate Message Number in Message Action	Identifies duplicate numbers used in more than one Message action
	Run Macro Action Used in Active Link	Identifies active links where the Run Macro action is used
	Set Fields Order Is Not Predictable	Identifies workflow objects where the Set Fields action source and target values are interdependent
	Field References in SQL Command	Identifies SQL commands that use field references
	Unused menu	Identifies menus that are not attached to any fields or are not used in any workflow
	Invalid Declaration of Global Fields	Identifies global fields with either of the following characteristics: ■ The field is a Character field whose input length is greater than 255. ■ The field is a Diary or Attachments field.
	Form Is Not a Part of Any Deployable Application	Identifies forms that are not part of a deployable application, which means that its associated workflow are also not included
	Irrelevant Field Indexing	Identifies fields that are indexed but do not exist or should not be indexed
	Trim Fields Are Not Visible in Any View	Identifies trim fields that are defined on the form but are not present in any view

Table 8-1: Analyzer rules (Sheet 2 of 2)

Category	Rule	Description
Structural (continued)	Number of Aliases for Selection Fields Is Inconsistent on Views	Identifies selection fields that have an inconsistent number of aliases of different views (An example of this might occur in localized views.)
	Unused Guide	Identifies guides that are not used in any workflow
	Guide's Workflow Forms Belong to Different Application	Identifies guides that have multiple workflows, and the forms associated to the workflow belong to different deployable applications
	Unreferenced Views Not Set as Default View	Identified unreferenced views that are not used in any workflow or are not set as the default view
	Bounding Box Is Too Small	Identifies fields that are cut off because they do not fit in their bounding box
	Display-Only Fields Not Used In Any Workflow	Identifies display-only fields that are not used in any workflow
	Unused Control Fields	Identifies unused control fields (buttons)
	Referenced Fields Do Not Exist	Identifies fields that are referenced but do not exist
Performance	Active Link Execute On	Identifies workflow objects that have two or more of the following Execute On conditions enabled: ■ Window Open ■ Window Loaded ■ Display
	Large image	Identifies large background images and large images on buttons
	QBE Match	Identifies character fields where the query-by-example (QBE) property is set to Anywhere
	$PROCESS$ as Assignment Value	Identifies workflow where `$PROCESS$` is used as an assignment value in a Set Fields action (This excludes special Run Process commands.)
	Field Length Greater Than 255 Characters	Identifies fields with a length that is greater than 255 characters
Security	Run Process	Identifies when a process command uses a field variable as a part of a process argument or as the entire command; also identifies whether the server setting to use a directory for Run Process active link actions is enabled Additionally, for UNIX servers, this rule identifies whether the following server settings are enabled: ■ Using shell control for Run Process actions ■ Using back quote characters (‘) in Run Process actions
	System Password	Identifies system passwords that have not been set (such as the DSO User password)
	User Password	Identifies user passwords that have not been set; notifies you if the server setting for saved logins is enabled for all users

In the Analyzer View tab, Analyzer reports all of the inconsistencies it finds. Most messages in the Analyzer View tab have a severity of informational, warning, or error, as described in Table 8-2.

Table 8-2: Analyzer messages (Sheet 1 of 10)

Message	Severity	Description	Solution
Filter or escalation cannot run this process command	Error	This Run Process command is not valid in a filter or an escalation.	Remove the invalid Run Process command.
Filter or escalation cannot run this process command	Error	This `$PROCESS$` command is not valid in a filter or an escalation.	Remove the invalid `$PROCESS$` command.
Invalid $PROCESS$ command	Error	This `$PROCESS$` command must be run on a server, so a server reference is required.	Add a @@ or `@serverName` reference to the `$PROCESS$` command.
Invalid Run Process command	Error	This Run Process command must be run on a server, so a server reference is required.	Add a @@ or `@serverName` reference to the Run Process command.
Invalid server reference	Error	The reference to a server is not valid in an active link or escalation Run Process or `$PROCESS$` command.	Remove the @@ or `@serverName` reference from the Run Process or `$PROCESS$` command.
This $PROCESS$ command must be run on the client.	Error	This `$PROCESS$` command cannot be run on a server, so the server reference is invalid.	Remove the @@ or `@serverName` reference from the `$PROCESS$` command.
This Run Process command must be run on the client	Error	This Run Process command cannot be run on a server, so the server reference is invalid.	Remove the @@ or `@serverName` reference from the Run Process command.
Field references found in direct SQL commands.	Error Informational	This is a level 1 result if the entire SQL command is created from one field such as `$Short Description$`. Any command can be run from the field value, and this can be exploited. This is a level 3 result if a field reference is used as part of a command, for example: `Select $Short Description$ From $Character Field$` The SQL statement might be malformed. The AR System server does not validate SQL commands used in Direct SQL actions.	Validate the SQL command manually.

Table 8-2: Analyzer messages (Sheet 2 of 10)

Message	Severity	Description	Solution
No password set for a <license type> licensed user: <user name>	Error	For user `Demo`, this is a level 1 result. For all other users, this is a level 2 result. Applies only if External Authentication is *not* used. For more information on external authentication, see the *Configuration Guide.*	For security reasons, you should enforce passwords.
	Warning	Indicates that the user named in the message has an empty password.	
$PROCESS$ is used as an assignment value in a Set Fields action	Informational	When `$PROCESS$` is used in Set Fields actions, the system must wait for the process to complete and return a value. Depending on the process, this could impair performance.	Evaluate the need for using `$PROCESS$` results to assign field values.
A control field is not used in any workflow	Informational	A form contains a control field (such as a button) that is not connected to any workflow, so it is not being used.	Delete the unused control field, or attach it to the appropriate workflow. Choose Form > Add / Remove Fields In View.
A display-only field is not used in any workflow	Informational	Display-only fields do not store data. If they are not referenced by workflow, they usually do not serve a purpose. This might occur with fields that were created for a purpose at one time, but the application changed, and they are no longer useful.	Verify whether the display-only field is used. If it is not used, delete it.
A form is not part of any deployable application	Informational	The form is not a part any deployable application, so its associated workflow is also not included in an application. If the application is deployed into a production environment, the workflow will not serve any useful purpose.	Add the form to a deployable application.

Table 8-2: Analyzer messages (Sheet 3 of 10)

Message	Severity	Description	Solution
A global field has a length greater than 255	Informational	Global fields are display-only fields of the following types: ■ Regular global fields share data across multiple windows or forms. The field ID range for these global fields is 1000000 through 1999999. ■ Window-scoped global fields share the data across multiple records in the same window. The field ID range for these global fields is 3000000 through 3999999. Global fields cannot be any of the following field types: ■ Character whose input length is greater than 255 ■ Diary ■ Attachment	Change the global field to a type *other than* the following types: ■ Character whose input length is greater than 255 ■ Diary ■ Attachment
A guide is not used in any workflow	Informational	A guide was found that is not connected to any workflow, so it is not being used.	Delete the unused guide, or attach it to the appropriate workflow.
A menu is not being used in any fields or workflow	Informational	A menu was found that is not attached to any fields or is not being used in any workflow (such as a Change Field action).	Delete the unused menu, or attach it to the appropriate workflow.
A trim field is not used in any view	Informational	A trim field (such as a line) is defined on the form but is not present in any view. Note: BMC Remedy Developer Studio does not usually allow this, but a legacy form from a definition file could contain this issue.	Delete the unused field. Choose Form > Add / Remove Fields In View.
A view is not used in any workflow or is not the default view	Informational	A view is not being used in any workflow (such as an Open Window action), or a view is not set as the default view.	Make sure that the view is being used efficiently.
Back quotes allowed in Run Process command	Informational	Applies only to UNIX servers. Back quotes should not be permitted in a Run Process command.	Set the following option in the `ar.conf` file: `Allow-Backquote-In-Process-String:F`
Duplicate message numbers in Message action	Informational	If the same message number is used in more than one Message action, it is difficult to determine which piece of workflow triggered the message. Different messages might be associated with the same number.	Evaluate all of the Message actions, and change the message numbers as needed.

Table 8-2: Analyzer messages (Sheet 4 of 10)

Message	Severity	Description	Solution
EXTERNAL used in the query	Informational	The field contents are not explicit. EXTERNAL keywords can be used in queries in AR System to dynamically retrieve qualifications from a field value. These are indeterminate, and the result indicates that such a qualification is used.	It is possible that the query is malformed and incorrect. Try to avoid such queries, unless the field value is being well calculated.
Field has QBE match property set to "Anywhere"	Informational	If the QBE Match property on a character field is set to Anywhere, the database cannot optimize its search and will perform a table scan.	Evaluate the requirement of this setting for the field. Avoid using this setting if possible.
Server setting for saved login enabled for all users	Informational	This is a security concern if user desktops are shared.	Set Users Prompted For Login option to Always in the Configuration tab of the AR System Administration: Server Information form.
Set fields order is not predictable	Informational	Because the order for the setting of fields in active links at run time. cannot be predicted, target fields should not be used as source fields in same Set Fields action.	Split the Set Fields action into two or more Set Fields actions.
The number of aliases for a field is not the same on each form view	Informational	A selection field appears on several views of a form, but the number of aliases for the field is not the same on each form view. For example, consider a form with two views: one in English and one in Spanish. A selection field on these views has four options. A fifth is added to the field. An alias is added to the English view but not to the Spanish view, and the number of aliases is inconsistent. The English view now contains five options and five aliases. The Spanish view contains five options and four aliases. Consequently, the field on the Spanish view will display four Spanish words and one English word.	Make sure that each view has the correct number of aliases for the selection field.
Workflow for a form connected to a guide belongs to a different application	Informational	A guide can have multiple workflows, but the forms associated with the workflow can belong to different deployable applications. This can cause inconsistent results.	Make sure that the guide is working properly.

Table 8-2: Analyzer messages (Sheet 5 of 10)

Message	Severity	Description	Solution
- <name> Password has not been set	Warning	An AR System Service Password was not set, and AR System uses the default password. The possibility of abuse is greater if these passwords are left as defaults. The following passwords are analyzed: ■ Application service ■ DSO user (if DSO is licensed) ■ DSO target (if DSO is licensed) ■ Mid tier (if the mid tier is licensed) ■ Plug-in server ■ Plug-in target	Set these passwords using options on the Connection Settings tab of the AR System Administration: Server Information form.
!= operator	Warning	When a qualification uses the != operator (for example, `'Status' != "Closed"`), the database is likely to perform a full table scan.	Rewrite the qualification without the != operator. When possible, use the less than operator (<) or greater than operator (>) instead.
Active link will fire multiple times for same operation	Warning	Active link is executed on more than one of the following: ■ Window Open ■ Window Loaded ■ Display ■ After Submit The same active link might execute multiple times for the same operation. This can impair performance.	Evaluate all such active links and see if different active links can be created for each Execute On condition.
Bad field references	Warning	A display-only field has been used in a query. Display-only fields do not have any associated columns in the corresponding database table. Their data is not persistent. Use of these fields in queries will cause a run-time error.	Avoid using display-only fields in queries to avoid failure during run time. Even without an error, using a display-only field in a query is not recommended because there is no corresponding persistent data for these fields.
Else actions cannot be run	Warning	The workflow never runs these Else actions because it does not have a Run If qualification.	Specify a Run If condition, or remove the Else actions.

Table 8-2: Analyzer messages (Sheet 6 of 10)

Message	Severity	Description	Solution
Empty qualification	Warning	No qualification is specified for an operation that queries the database. Empty queries typically cause a scan of the entire table. The exception is when an empty qualification is used in a Push Fields action and the If No Records Match option is set to Create New Request. In this situation, the server does not look up the table.	Avoid using empty qualifications. Try to use a query that would always evaluate to true.
Explicit server reference	Warning	An explicit reference to a server by its name might fail when the workflow is moved to another server.	If the server referenced is the one running the workflow, use the form of the workflow that references it implicitly to make your application portable. If a different server is referenced, update the name if you move the workflow.
Field is not appropriate for indexing	Warning	Fields defined in the index list do not exist or those fields are no longer to appropriate for indexing. For example, a character field's size is 50, and the field is indexed. Then, the field's size is changed to unlimited. Unlimited fields cannot be indexed	Remove the field from the index list.
Field used in arithmetic expression	Warning	For queries where the field is used in an arithmetic expression (for example, if A is an integer field, and the following query is used: `'A' * 3 < 20`), indexes or queries cannot be optimized effectively because the expression must be evaluated for each row in the table and then evaluated against the other part of the query.	Avoid queries that use fields in arithmetic expressions. To improve the query (used: `'A' * 3 < 20`), the expression can be changed to `'A' < (20/3)`. Then, an index and optimizations can be used because `20/3` is evaluated separately and based on its value. The correct rows then can be fetched from the database table.
Field(s) not indexed	Warning	Fields are not indexed. For example, you have a qualification such as `'A' = "John"`, and field A is not indexed.	Review the index list and the fields in the query list to determine the number of times the field is used in queries. Determine if it makes sense to index field A.
Invalid access point references	Warning	The form or call guide action is part of an application, and that form or action is not declared as an access point in that application.	Declare the form or call guide action as an access point in the application to which the form belongs.

Table 8-2: Analyzer messages (Sheet 7 of 10)

Message	Severity	Description	Solution
Large background image used for form	Warning	Using large background images can significantly affect the performance of the application.	Avoid using large images as background images for forms. Use smaller background images.
Large image used for field	Warning	Using large images for field can significantly affect performance by slowing down the downloading process.	Use smaller images for fields.
`LIKE` match string starts with a wildcard	Warning	When the match string of a `LIKE` operator in a qualification starts with a wildcard character (for example, `'A' LIKE "%John"`), the database cannot use any index or any optimizations and will perform a full table scan.	Rewrite the qualification without the wildcard character at the beginning of the search string.
LIKE operator is followed by a wildcard character	Warning	If you use the `LIKE` operator and a wildcard at the beginning of the string, the database will not use an index, and the database will not be able to use optimizations. Additionally, the database might perform a complete table scan. Examples include `'A' LIKE "%John"` and `'A' LIKE "_John"`.	Avoid such queries where possible.
Long character field used in query	Warning	Indexes cannot be created on long character fields (fields greater than 255 characters). For long character fields, AR System uses a different table and maintains a link with the actual table for the form. Using these long fields can lead to considerable performance problems.	Avoid using long character fields in queries.
No shell control specified for active link Run Process actions	Warning	Applies to UNIX servers only. You can specify shell control for running active link processes	Set this option in the Advanced tab on the AR System Administration: Server Information form.
No specific directory specified for active link Run Process actions	Warning	A potential security problem exists if the directory from which processes can be run from active links is not secure.	Specify a value for Active Link Run Process Directory on the Advanced tab of AR System Administration: Server Information form.

Table 8-2: Analyzer messages (Sheet 8 of 10)

Message	Severity	Description	Solution
No subquery of an AND qualification uses an index	Warning	If the query involves an `AND` operator, at least one subquery must include an indexed field to make the query efficient. In the following example, neither subquery uses an index: `'A' = "John" AND 'B' LIKE "%Smith"` Field A is not indexed, and field B is indexed, but the wildcard character with the `LIKE` operator forces a table scan.	Make sure that subqueries favor index usage.
`NOT` operator	Warning	When a qualification uses the `NOT` operator (for example, `NOT('Status' = "Closed")`), the database is likely to perform a full table scan.	Rewrite the qualification without the `NOT` operator.
One of the subqueries of an OR qualification does not use any index	Warning	If the query includes an OR operator, all parts of the subqueries must have a field that is indexed to make the query efficient. For example: `'A' = "John" OR 'B' LIKE "%Smith"` Field A and field B are indexed. `'A' = "John"` can use the index, but `'B' LIKE "%Smith"` cannot use an index because of the wildcard character with the `LIKE` operator.	Make sure that subqueries favor index usage.
Referenced fields do not exist	Warning	Referenced fields do not exist.	Remove the cross-reference.
Run Macro action used in active link	Warning	Run Macro actions work only in BMC Remedy User and are not supported in browsers. When deployed in a browser, these actions do not function properly.	Use an active link action that is compatible with browsers.

Table 8-2: Analyzer messages (Sheet 9 of 10)

Message	Severity	Description	Solution
The bounding box is too small to fit the data box and label box. OR The [height | width] of the [data | label] box is too small to fit the field's [data | label].	Warning	The bounding box does not allow the entire field to appear. For example, a selection field contains two radio buttons. If five radio buttons are added but the bounding box is not made larger, the bounding box is too small to display all of the radio buttons. Note: BMC Remedy Developer Studio does not allow this, so the problem occurs only if a field is changed programmatically.	Resize the field so that the entire field appears.
Unused active link	Warning	The active link cannot be run.	Add an execution option to the active link, or include it in a guide.
Unused filter	Warning	The filter cannot be run.	Add an execution option to the filter, or include it in a guide.
Usage of != operator	Warning	The != operator causes the database to perform a complete table scan. For example, `'Status' != "Closed"`.	Avoid using != operator. Use > or < where possible. For example, replace the previous example with `'Status' < "Closed"`.
Usage of NOT operator.	Warning	Using the `NOT` operator usually causes the database to perform a complete table scan. For example, `NOT ('Status' = "Closed")`.	Avoid using the `NOT` operator.
No field found in subquery	Warning (for Oracle databases) Informational (for all other databases)	For example, you have a qualification such as `1 = 0`. Depending on the database used, this could lead to a table scan. For Oracle databases, this is a level 2 result. The Oracle database does not evaluate queries that do not involve fields in memory, but performs a table scan instead. Apply the solution for an Empty Qualification result, if possible. For all other databases, this is a level 3 result for any other database. Other databases evaluate such expressions in memory instead of doing a table scan.	Evaluate your query and see if a field can be used in the query and produce the same result.

Table 8-2: Analyzer messages (Sheet 10 of 10)

Message	Severity	Description	Solution
Length of field is greater than 255	Warning for all databases (except Oracle) Informational for Oracle database	For all databases, except Oracle, this is a level 2 result. The AR System server handles fields with lengths greater than 255 differently, and this can impair performance. For Oracle databases (where the character field length limit is 4000), this is a level 3 result for character fields with lengths between 255 and 4000.	Unless absolutely necessary, avoid using long character fields in queries.
$fieldID$ is used in a $PROCESS$ or Run Process command	Warning Information	A `$PROCESS$` or Run Process command uses `$`*`fieldID`*`$` as the entire command (for example, `$`*`fieldID`*`$`) A `$PROCESS$` or Run Process command uses `$`*`fieldID`*`$` as an argument (for example, `ABCprocess.exe $`*`fieldID`*`$`)	Make sure that the `$PROCESS$` or Run Process command is valid.

Analyzing server objects

Run the Analyzer wizard to analyze server objects. In the wizard, you select the objects to analyze and the rules to apply.

The Analyzer uses object relationships that the AR System server records. For the Analyzer to be available, you must enable the relationships feature.

▶ To enable the relationships feature so that the Analyzer is available

1. In a browser or BMC Remedy User, open the AR System Administration Console, and click System > General > Server Information.

   The AR System Administration: Server Information form appears.
2. Click the Configuration tab.
3. Select the Record Object Relationship check box.
4. Click Apply.
5. Restart the server.

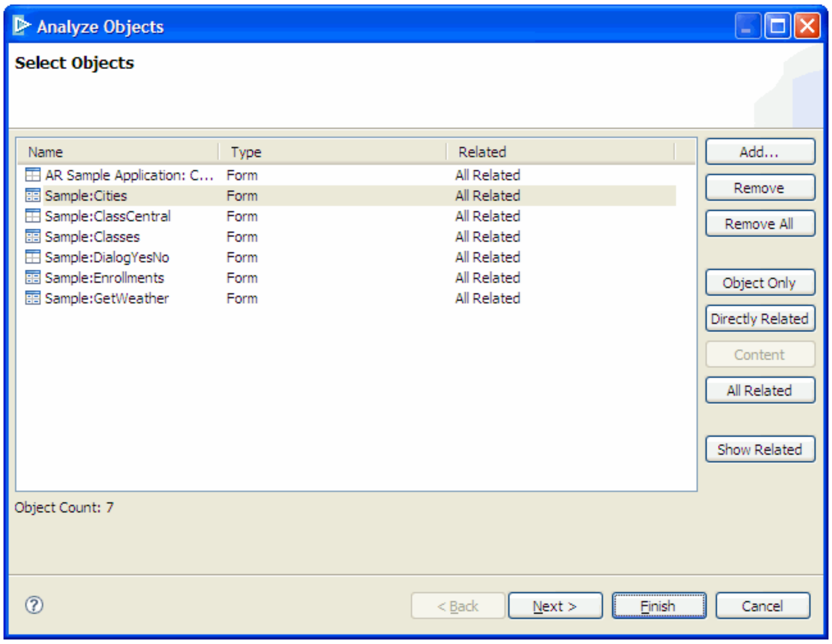
   For more information, see the *Configuration Guide*, "Server Information—Configuration tab," page 128.

▶ To analyze server objects

1. Start the wizard:
   - To analyze the object in an application or working list, right-click its item in the AR System Navigator and choose Analyze.
   - To analyze a collection of objects, select them in an object list, right-click, and choose Analyze.
   - To open the Analyze Objects wizard with no objects selected, click the Analyze Objects button in the BMC Remedy Developer Studio toolbar. In this case, you choose the server on which to analyze objects in the first page of the wizard.

   The Analyze Objects wizard displays the Select Objects page, shown in Figure 8-1.

Figure 8-1: Analyze Objects wizard—Select Objects



Name	Type	Related
AR Sample Application: C...	Form	All Related
Sample:Cities	Form	All Related
Sample:ClassCentral	Form	All Related
Sample:Classes	Form	All Related
Sample:DialogYesNo	Form	All Related
Sample:Enrollments	Form	All Related
Sample:GetWeather	Form	All Related

Object Count: 7

2 To add objects, click the Add button and select them in the Add Items dialog box.

3 To remove objects, select them and click Remove, or click Remove All.

4 To determine which related objects are included in the analysis, select objects and click one of the following buttons:

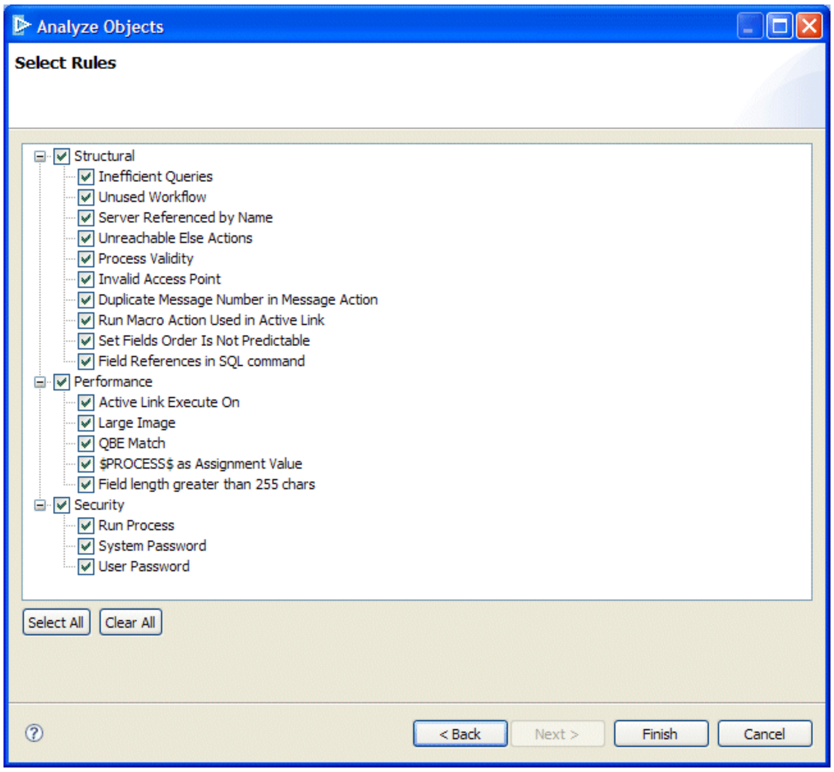
- Object Only
- Directly Related
- Content
- All Related

You can also click the Related column in the list and choose a setting.

For descriptions of the options, see the *Form and Application Objects Guide*, "Exporting object definitions, views, and applications," page 560.

5 Click Next to display the Select Rules page, shown in Figure 8-2.

Figure 8-2: Analyze Objects wizard—Select Rules



- ☑ Structural
  - ☑ Inefficient Queries
  - ☑ Unused Workflow
  - ☑ Server Referenced by Name
  - ☑ Unreachable Else Actions
  - ☑ Process Validity
  - ☑ Invalid Access Point
  - ☑ Duplicate Message Number in Message Action
  - ☑ Run Macro Action Used in Active Link
  - ☑ Set Fields Order Is Not Predictable
  - ☑ Field References in SQL command
- ☑ Performance
  - ☑ Active Link Execute On
  - ☑ Large Image
  - ☑ QBE Match
  - ☑ $PROCESS$ as Assignment Value
  - ☑ Field length greater than 255 chars
- ☑ Security
  - ☑ Run Process
  - ☑ System Password
  - ☑ User Password

6 Select the rules to apply to the selected objects.

7 Click Finish.

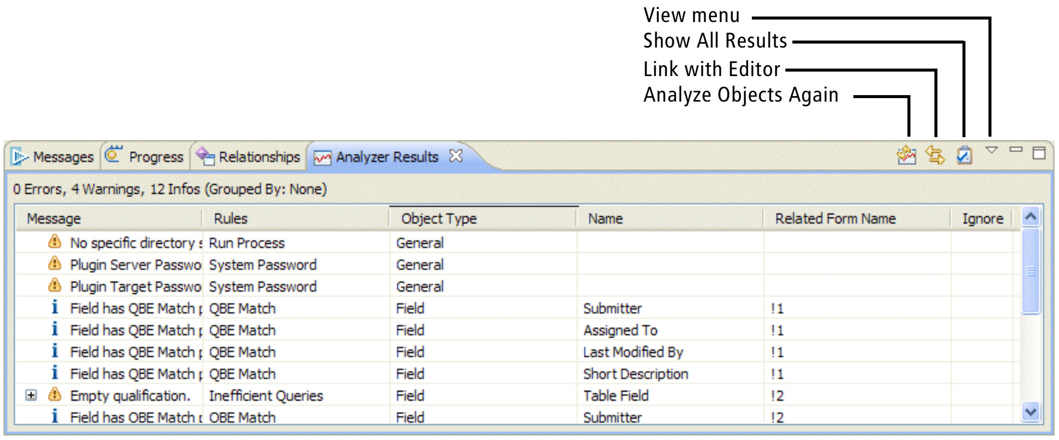
As BMC Remedy Developer Studio analyzes the objects, it displays the state of analysis in the status bar and in the Progress tab.

When the analysis is complete, BMC Remedy Developer Studio displays the results in the Analyzer View tab.

Working with the Analyzer View tab

The Analyzer View tab reports the results of the analysis, as shown in Figure 8-3.

Figure 8-3: Analyzer View tab



0 Errors, 4 Warnings, 12 Infos (Grouped By: None)

Message	Rules	Object Type	Name	Related Form Name	Ignore
No specific directory s	Run Process	General			
Plugin Server Passwo	System Password	General			
Plugin Target Passwo	System Password	General			
Field has QBE Match p	QBE Match	Field	Submitter	!1	
Field has QBE Match p	QBE Match	Field	Assigned To	!1	
Field has QBE Match p	QBE Match	Field	Last Modified By	!1	
Field has QBE Match p	QBE Match	Field	Short Description	!1	
Empty qualification.	Inefficient Queries	Field	Table Field	!2	
Field has QBE Match p	QBE Match	Field	Submitter	!2	

The Analyzer View tab includes the following columns:

- **Message**—The text and details of the messages. The text is listed in Table 8-2 on page 196.

  The messages can be listed individually or grouped by message severity or any of the columns. By default, the messages are grouped by related form name. In Figure 8-3, some of the groups are expanded to show the individual messages. Some of the messages can be expanded further to show details of the potential problem.

- **Rules**—The rules that defines the potential problems, as listed in Table 8-1 on page 194.
- **Object type**—The types of the objects.
- **Name**—The names of the objects where the Analyzer found the potential problems.
- **Related Form Name**—The forms associated with the objects where the Analyzer found the potential problems.
- **Ignore**—The results that are ignored. When you select results to ignore, the settings are stored on the AR System Ignored Analyzer Results form. For information about how to ignore results, see "To ignore results" on page 211.

▶ To group the results

- From the View menu ( ) on the Analyzer View tab, choose an option from the Group By menu:
  - Message
  - Object Type
  - Name
  - Related Form
  - Severity
  - Result Summary
  - None
  - Field In Query

NOTE

The Field in Query option appears only if you selected Inefficient Queries in the Analyzer wizard. If you choose Field in Query, the Message column lists the objects that contain fields with inefficient queries. Click the plus sign next to the object to drill down and learn how the fields are used inefficiently in queries.

▶ To display a description of the potential problem

- Right-click the message line and choose Description.

The Description dialog box displays the message description and solution as listed in Table 8-2 on page 196. Use the up and down arrow keys to view the descriptions of other messages.

▶ To display details about the location of the potential problem

- Click the plus sign before the message in the Messages column to display the values of attributes of the message.

▶ To edit the form or workflow object

- Double-click the message line in the Analyzer View tab.

The object editor opens. If a field, workflow action, or Run If qualification is the subject of the message, it is selected in the editor.

▶ To analyze the same objects with the same rules

- Click the Analyze Objects Again button ( ) on the Analyzer View tab.

▶ To analyze an object automatically when you save it

- Click the Link With Editor button ( ) on the Analyzer View tab.

When Link With Editor is selected, each time you save an object in an editor, the Analyzer analyzes that object only using all the rules and displays the resulting messages.

▶ To ignore results

1 Select one or more rows that contain the results or object types that you want to ignore.

2 Right-click and choose Ignore.

3 In the Confirm dialog box that appears, select the appropriate option:

   - Skip Selected Result Only
   - Skip All Results of Selected Types

4 Click OK.

   The settings are stored on the AR System Ignored Analyzer Results form.

   For information about how to restore results that are ignored, see “To restore results that are ignored.”

▶ To restore results that are ignored

1 Click the Show All Results button ( ) on the Analyzer View tab.

   Results that are ignored appear, and a check mark appears in the Ignore column to indicate that the results are to be ignored.

2 Select the results that you no longer want to ignore.

3 Right-click and choose Ignore.

4 In the Confirm dialog box, click Yes.

Launching the Analyzer from a command line

The installation directory for BMC Remedy Developer Studio contains an `analyzer.bat` file, which allows you to run the Analyzer from a command line. When the `analyzer.bat` file is run, you receive a report in CSV format.

▶ To run the Analyzer from a command line

1 Create a backup copy of the `Analyzer.bat` file in the *`ARSystemInstallDir`*`/DeveloperStudio` folder.

2 Add the appropriate options to the command line in `Analyzer.bat`, for example:

```
analyzer -x "computer1" -u "Demo" -f "Sample%%"
```

   For a list of available options, see “Options for Analyzer from a command line.”

3 At the command line, run the batch file.

   For examples, see “Examples of the Analyzer command” on page 213.

Options for Analyzer from a command line

Table 8-3: Options for Analyzer from a command line

Option	Parameter	Description
`-u`	*`user`*	Required login parameter that identifies the user account.
`-p`	*`password`*	Optional login parameter that identifies the user account. If the user account has no password, omit the `-p` option.
`-x`	*`serverName`*	Required login parameter that specifies the server to log in to.
`-w`	*`authenticator`*	The name of an external authentication string or Windows domain. This is related to the Login window's Authentication field, which is discussed in the *Configuration Guide*, "Setting up an authentication alias," page 76.
`-portnum`	*`TCPPortNumber`*	Used to log in when the portmapper is turned off.
`-o`	*`CSVFilePath`*	CSV file path for analyzer results.
`-excludeIgnoredResults`		Results marked as Ignored should not be included in the report.
`-a`	*`activeLinkName`*	Analyze the specified active link.
`-A`		Analyze all active links.
`-f`	*`formName`*	Analyze the specified form.
`-F`		Analyze all forms.
`-n`	*`applicationName`*	Analyze the specified application.
`-N`		Analyze all applications.
`-q`	*`escalationName`*	Analyze the specified escalation.
`-Q`		Analyze all escalations.
`-t`	*`filterName`*	Analyze the specified filter.
`-T`		Analyze all filters.
`-r`	*`relatedType`*	Include related object for analysis. Valid options are: ■ `none` (default) ■ `directly` ■ `all` ■ `content`
`-modifiedAfter`	*`date`*	Analyze objects that are modified after *`date`*. Use the *`M/d/yy h:mm`* format. Time details are optional.
`-modifiedBefore`	*`date`*	Analyze objects that are modified before *`date`*. When you use with `-modifiedAfter` with `-modifiedBefore`, you can analyze modified objects between two dates. Use the *`M/d/yy h:mm`* format. Time details are optional.

You can use the following percent (%) wildcard when specifying names for the -a, -f, -n, -q, and -t options.

- *%string* analyzes objects with a name ending with *string*.
- *string%* analyzes objects with a name beginning with *string*.
- *%string%* matches *string* anywhere in object name.

For example, TMS% analyzes all active links with a name beginning with TMS.

Examples of the Analyzer command

- In the following example, the Demo user analyzes all of the filters on the vmw23prem95 server, and the results are entered in the filter.csv file.

  ```
  C:\Program Files\BMC Software\ARSystem\DeveloperStudio>analyzer
  -x "vmw23prem95" -portnum 2044 -u "Demo" -p "" -T
  -o "D:\csv\filter.csv"
  INFO: Analyzer started: Tue Jul 07 13:13:22 GMT+05:30 2009)
  Jul 7, 2009 2:13:27 PM
  com.bmc.arsys.studio.analyzer.core.AnalyzerApplication
  printResultCount
  INFO: Analyzed  544 objects
  1 Errors, 148 Warnings, 594 Infos
  Jul 7, 2009 2:13:27 PM
  com.bmc.arsys.studio.analyzer.core.AnalyzerApplication start
  INFO: Analyzer completed: Tue Jul 07 18:13:27 GMT+05:30 2009)
  ```

- In the following example, the Demo user analyzes the R_Like form with the directly related option, and the results are entered in the like.csv file.

  ```
  C:\Program Files\BMC Software\ARSystem\DeveloperStudio>analyzer
  -x "premq617" -u "Demo" -p "" -f "R_Like" -r "Directly"
  -o "D:\csv\like.csv"
  INFO: Analyzer started: Tue Jul 07 14:08:11 GMT+05:30 2009)
  Jul 7, 2009 2:08:14 PM
  com.bmc.arsys.studio.analyzer.core.AnalyzerApplication
  printResultCount
  INFO: Analyzed  4 objects
  0 Errors, 14 Warnings, 6 Infos
  Jul 7, 2009 2:08:14 PM
  com.bmc.arsys.studio.analyzer.core.AnalyzerApplication start
  INFO: Analyzer completed: Tue Jul 07 14:08:14 GMT+05:30 2009)
  ```

- In the following example, the `Demo` user analyzes the forms that start with R, are modified after June 1, 2009 but before July 1, 2009. The results are entered in the `wildcard_date.csv` file.

  ```
  C:\Program Files\BMC Software\ARSystem\DeveloperStudio>analyzer
  -x "premq617" -u "Demo" -p "" -f "R%"  -modifiedAfter "6/1/09"
  -modifiedBefore "7/1/09" -o "D:\csv\wildcard_date.csv"
  INFO: Analyzer started: Tue Jul 07 14:13:22 GMT+05:30 2009)
  Jul 7, 2009 2:13:27 PM
  com.bmc.arsys.studio.analyzer.core.AnalyzerApplication
  printResultCount
  INFO: Analyzed  20 objects
  0 Errors, 14 Warnings, 94 Infos
  Jul 7, 2009 2:13:27 PM
  com.bmc.arsys.studio.analyzer.core.AnalyzerApplication start
  INFO: Analyzer completed: Tue Jul 07 14:13:27 GMT+05:30 2009)
  ```

- In the following example, the `Demo` user ignores the results that were specified to be ignored in the Analyzer in BMC Remedy Developer Studio. (See "To ignore results" on page 211.) The results are entered in the `a3.csv` file.

  ```
  C:\Program Files\BMC Software\ARSystem\DeveloperStudio>analyzer
  -x "premq617" -u "Demo" -p "" -f "a%"  -o "D:\csv\a3.csv"
  -excludeIgnoredResults
  INFO: Analyzer started: Tue Jul 07 15:50:24 GMT+05:30 2009)
  Jul 7, 2009 3:50:26 PM
  com.bmc.arsys.studio.analyzer.core.AnalyzerApplication
  printResultCount
  INFO: Analyzed  11 objects
  0 Errors, 4 Warnings, 2 Infos
  Jul 7, 2009 3:50:26 PM
  com.bmc.arsys.studio.analyzer.core.AnalyzerApplication start
  INFO: Analyzer completed: Tue Jul 07 15:50:26 GMT+05:30 2009)
  ```

Appendix

A Operators, wildcards, keywords, and NULL values

This section provides descriptions of the operators, wildcards, and keywords that you can use when defining qualification criteria or building field values. It also discusses `NULL` values in fields and how they are used with AR System.

The following topics are provided:

Operators

This section describes how each operator is evaluated when you use multiple operators in a qualification. You can use operators to build logical expressions. For additional information, see Chapter 3, "Building qualifications and expressions."

Operator types

The following table lists the operators that you can use when building a qualification.

Table A-1: Operators (Sheet 1 of 4)

Operator	Action
`AND (&&)`	Logical `AND` of the result of two conditions. (The result is true if both conditions are true.) For example, `'Status'="New" AND 'Assigned-to'="Andy"` finds all new requests assigned to Andy. You can use the symbol `&&` instead of the word `AND`.
`OR (||)`	Logical `OR` of the result of two conditions. (The result is true if either condition is true.) For example, `'Status'="New" OR 'Assigned To'="Andy"` finds all new requests and all requests assigned to Andy (no matter what their status). You can use the symbol `||` (two vertical lines) instead of the word `OR`.
`NOT (!)`	Negates the condition that follows. (If the condition is false, the result is true.) For example, `NOT 'Status'="New"` finds all requests that are not new. You can use the symbol `!` instead of the word `NOT`.
`LIKE`	Performs a pattern search using wildcard characters, including `%`, `-`, `[ ]`, `[-]`, and `[^]`. For example, `'Submitter' LIKE "Bob%ton"` finds all requests with a submitter name that begins with the letters "Bob" and ends with the letters "ton"—such as Bob Compton and Bobby Fenton. See "Wildcards" on page 183. The `LIKE` operator applies to character and diary fields only. `LIKE` with comma separators can be used as the accrue operator in full text searches. For more information about full text search, see the *Configuration Guide*, "Using full text search," page 295. Note: Some databases do not support all the possible wildcard matching characters. To determine whether there are restrictions on the wildcard characters that you can use in SQL queries, check your database documentation.

Table A-1: Operators (Sheet 2 of 4)

Operator	Action

`EXTERNAL()`	Enables all or part of a qualification to come from a field on the form. Used in qualifications for active link or filter Run If conditions, Push Fields or Set Fields actions, table field properties, and Search type menu definitions. `EXTERNAL()` is *not* used in escalations and flashboard variable qualifications.
	
	Enter a character field name or field ID into the `EXTERNAL()` operator, for example, `EXTERNAL($field_name$)` or `EXTERNAL('50')`.
	
	`EXTERNAL()` evaluates to True when:
	■ The field specified contains an expression that evaluates to True.
	■ The field is empty.
	■ The field does not exist or is not a character field.
	
	You can also concatenate expressions. For example:
	`EXTERNAL($field_name$) AND 'Submitter' != $NULL$`
	
	If the qualification in a field called by `EXTERNAL()` includes keywords, you must add an escape character (`\`) to the keywords. This prevents the keywords from expanding before being used in the qualification.
	
	For example, if you have a table field with a qualification of
	`EXTERNAL($Qualify Field$)`
	and Qualify Field is a character field with a value of
	`('Create Date' < $TIMESTAMP$) AND ('Login Name' = $USER$)`
	the table field does not produce the expected results when refreshed. Instead, the keywords expand, producing a qualification such as
	`('Create Date' < 05/22/02 11:00:34 AM) AND ('Login Name’ = Demo)`
	This is an invalid query because the date/time and character values are not enclosed in quotation marks. To prevent the keywords from expanding, write the qualification like this:
	`('Create Date' < $\TIMESTAMP$) AND ('Login Name' = $\USER$)`
	
	In forms viewed in the web client, if `EXTERNAL()` references a field that contains a qualification such as `$Date 1$ <= 'Date 2'`, you must add double quotation marks around `$Date 1$`:
	`"$Date 1$" <= 'Date 2'`
	
	If `EXTERNAL()` references a field whose value is set by a Set Fields or Push Fields action, use four double quotation marks to create a single double quotation mark. For example, to set the field’s value to
	`'Create Date'>="10\31\05" AND 'OtherField'="Hello"`
	set it as follows in the Set Fields or Push Fields action:
	`"'Create Date'>=" + """" + "10\31\05" + """"+ " AND 'OtherField'=" + """"+ "Hello" + """"`
	
	Note: In BMC Remedy User, the `EXTERNAL` operator reads date and time string from the server's time. In the web client, the `EXTERNAL` operator reads the date and time string from client's time zone.
	
	For more information, see “Double quotation marks” on page 55.

Table A-1: Operators (Sheet 3 of 4)

Operator	Action
`+`	Adds two integer, real, or decimal values. Adds an integer, real, or decimal interval to a time value. (Real and decimal values are truncated to an integer value.) Concatenates two character strings. For example, `'Create Date'>$DATE$+28800` finds all requests created after 8:00 a.m. today (where 28800 is the number of seconds in 8 hours).
`-`	Performs any of the following calculations: ■ Subtracts two integer, real, or decimal values. ■ Subtracts two time values (resulting in an integer). ■ Subtracts an integer, real, or decimal interval from a time value. (Real and decimal values are truncated to an integer value.) For example, `'Create Date'>$DATE$-604800` finds all requests created within the last seven days (where 604800 is the number of seconds in one week).
`*`	Multiplies two integer, real, or decimal values. For example, `'Quantity'*'Price'>50` finds all requests where the contents of the Quantity field multiplied by the contents of the Price field is more than 50.
`/`	Divides two integer, real, or decimal values. For example, `'Yearly Expenses'/12>8000` finds all requests where the average monthly expense is greater than $8,000.
`%`	Supplies the modulo of two integer values (the remainder of a division of the values). For example, `'Group ID'%2=1` finds all requests with an odd number in their Group ID field. Note: Use the modulo operator only with integer fields. If you use this operator with fields that have other data types, such as Date/Time, an error occurs.
`<`	Matches contents that are less than the value. For example, `'Create Date'<($TIMESTAMP$-86400)` finds all requests created more than 24 hours ago (where 86400 is the number of seconds in 24 hours).
`>`	Matches contents that are *greater than* the value. For example, `'Create Date'>"10/31/05 00:00:00"` finds all requests created after midnight on October 31, 2005.
`!=`	Matches contents that are *not equal to* the value. For example, `'Status'!="Closed"` finds all requests that are not closed. Note: If possible, avoid using the `!=` operator when building qualifications. For more information, see "Creating efficient qualifications" on page 56. On Sybase databases, you cannot use the != operator against unlimited character fields where the database input length equals 0.

Table A-1: Operators (Sheet 4 of 4)

Operator	Action
`<=`	Matches contents that are *less than or equal to* the value. For example, `'Salary'<=10000` finds all requests with contents of the Salary field less than or equal to 10000.
`>=`	Matches contents that are *greater than or equal to* the value. For example, `'Create Date'>="10/31/05"` finds all requests created on or after October 31, 2005.
`=`	Matches contents that are *exactly equal to* the value. For example, `'Status'=0` finds all requests with a status value equal to the first selection value.

Operator precedence

When you use multiple operators to construct qualification criteria, they are evaluated in the following order:

1. `( )`
2. `NOT (!) -` (unary minus; an operator that takes only one operand; for example, -2.5)
3. `* / %`
4. `+ -`
5. `< <= > >= = != LIKE`
6. `AND (&&)`
7. `OR (||)`

Operators of the same precedence are performed left to right.

You can use parentheses in an expression to override operator precedence. AR System evaluates expressions inside parentheses *first* before evaluating those outside.

Wildcards

The following table lists the wildcards that you can use with the `LIKE` operator in qualifications.

Table A-2: Wildcards

Wildcard	Description	Supported database
`%`	Matches any string of 0 or more characters. Example: `J%son` matches Jackson, Johnson, Jason, and Json. To include leading and trailing characters, you *must* use the % symbol. For example, to match Jill Bobbington, Bobby Fenton, Bob Compton, and Bob Stone in the Submitter field, enter: `'Submitter' LIKE "%Bob%ton%"`	■ Informix® ■ Oracle ■ Microsoft SQL Server
`_`	(Underscore character) Matches any single character. Example: `B_b` matches Bab, Bob, and Bub.	■ Microsoft SQL Server ■ Sybase
`[ ]`	Matches any single character within a specified set. Example: `[abcf]` matches the characters a, b, c, and f. Note: The close bracket (]) functions as a wildcard only when it is accompanied by an open bracket ([).	■ Microsoft SQL Server
`[-]`	Matches any single character within a specified range. Example: `[a-f]` matches the characters a, b, c, d, e, and f. Note: The hyphen functions as a wildcard character only when preceded by an open bracket ([or [^).	■ Microsoft SQL Server
`[^]`	Matches any single character *not* within the specified set or range. Example: `[^abcf]` matches all characters *except* a, b, c, and f. `[^a-f]` matches all characters *except* the characters a, b, c, d, e, and f.	■ Microsoft SQL Server

Wildcard symbols are interpreted as wildcards only when used with the `LIKE` operator; otherwise, they are interpreted literally.

To use percent (%), underscore (_), or open bracket ([) characters as literal text in a LIKE operation:

- **Microsoft SQL Server and Sybase**—Enclose the character in brackets. For example, [_] matches only the _ character, not any single character.
- **Informix**—Insert a backslash before the character. For example, \% matches the % character, not any string of 0 or more characters.

NOTE

On Oracle, you cannot use the `LIKE` operator to search for literal percent (%) or underscore (_) characters.

Validating wildcard pattern matching

When checking input fields for valid characters, the best practices is to check the field for anything other than what is acceptable. For example, the following is the list of acceptable input characters, so if any other character is entered an error message should be sent to the user.

- A to Z in UPPER or lower case
- Numbers from 0 to 9
- Special characters _ (underscore) and - (dash)

The recommended way to set this up is to create an Active Link to generate an error message that informs the user than an invalid character was detected in the string for the field.

The Active Link would fire on Submit, Modify, and Lose Focus actions for the character field being used. For this example the character field is called ‘Input’. The Run If qualification would be:

```
'Input' LIKE "%[^a-zA-Z0-9_-]%"
```

The If Action would contain a Message action (Type: Error).

NOTE

If you need to use the bracket characters (‘[‘ or ‘]’), you must follow the standard regular expression rules and use and escape character (‘\’) because brackets are used to define the syntax. For example, you would use ‘\]’ in order to use the close bracket character.

Keywords

The following table lists the keywords that you can use when building a qualification.

Table A-3: Keywords (Sheet 1 of 8)

Keyword	Value
`$ALT_KEY$`	To verify if the Alt key is pressed. The value is set to 1 when the Alt key is pressed while starting the workflow from a control field, a double-click or return on a table field. Any other value represents that the Alt key is not pressed.
`$APPLICATION$`	The name (not label) of the currently running application. On the web client, this keyword is set to `NULL` when the form is opened from a URL that does not include the application’s name. In BMC Remedy User, this keyword is set to `NULL` when the form is opened outside of an application. This keyword is `NULL` on the server.

Table A-3: Keywords (Sheet 2 of 8)

Keyword	Value
`$BROWSER$`	The browser (Internet Explorer or other) being used in the current session. If the browser is anything other than Internet Explorer, Netscape is returned. For BMC Remedy User, an empty string ("") is returned. This keyword is `NULL` on the server.
`$CLIENT_TYPE$`	The client type of the API program. When used in workflow, this keyword resolves to a number that corresponds to BMC Remedy User, BMC Remedy Developer Studio, DSO, and so on. The number representations of the different client types are in the `ar.h` file, which is located in *ARServerInstallDir*`\ARServer\api\include`. For example, if you use `$CLIENT_TYPE$` in a Run If qualification for an active link, you can cause the active link to execute only for a mid tier client (for example, `$CLIENT_TYPE$ = 9`).
`$CTRL_KEY$`	To verify if the Ctrl key is pressed. The value is set to 1 when the Ctrl key is pressed while starting the workflow from a control field, a double-click or return on a table field. Any other value represents that the Ctrl key is not pressed.
`$CURRENTWINID$`	Window ID that uniquely identifies the current window in the client environment. This is the same ID used to send events to this window. The format and length of window IDs are not constrained. The value returned from `$CURRENTWINID$` should be saved into a display-only character field with a database length of zero (unlimited length). `$CURRENTWINID$` is not the window with focus, but the window ID of the window in which the active link is running. This keyword is `NULL` on the server.
`$DATABASE$`	The type of database used on the current server. This keyword is especially useful for workflow that is dependent on the database environment.

Table A-3: Keywords (Sheet 3 of 8)

Keyword	Value
`$DATE$`	For date/time fields, this keyword evaluates to the current date, and the time defaults to midnight. Anything stored in a date/time field is stored as the number of seconds since UNIX epoch time, which includes the date and time. Note: When using `$DATE$` in a search on a Date/Time field, you can add or subtract date/time by using a value that is based on seconds. For example: *`dateTimeField`* `> $DATE$ - 86400` (86400 is the number of seconds in 24 hours.) For Date fields, this keyword evaluates to the current date. For Time fields, this keyword evaluates to 12:00:00 AM. Note: The `$DATE$` keyword is not expanded when default values are set. This allows the value to be set to the date the form is submitted rather than the date the form is opened. Consequently, the literal keyword is displayed in the field before the request is submitted.
`$DEFAULT$`	The default value for the field, if any.
`$DRAGSRCFIELDID$`	An integer representing the ID of the field from which a drag operation started. Use this keyword in workflow to reference the source field immediately after a drag operation has started. This keyword is useful with the `Application-Copy-Field-Value` command. For more information, see "Process commands" on page 261.
`$DROPTGTFIELDID$`	An integer representing the ID of the field on which a drop operation occurred.
`$ERRNO$`	The number of the first error on the stack. Defined only during filter error handler execution.
`$ERRMSG$`	The error message for the first error on the stack. Defined only during filter error handler execution.
`$ERRAPPENDMSG$`	The appended message for the first error on the error stack, if any. Defined only during filter error handler execution.
`$EVENTDATA$`	The value that identifies the data of the event. This keyword is `NULL` on the server.
`$EVENTSRCWINID$`	Window ID that uniquely identifies the event source window in the client environment. The value returned from `$EVENTSRCWINID$` should be saved into a display-only character field with a database length of zero (unlimited length). This keyword is `NULL` on the server.

Table A-3: Keywords (Sheet 4 of 8)

Keyword	Value
`$EVENTTYPE$`	The value that identifies the type of the event. The value returned from `$EVENTTYPE$` should be saved into a display-only character field with a database length of zero (unlimited length). This keyword is `NULL` on the server.
`$FIELDHELP$`	The help text in the Help Text tab of the Field Properties window. Web applications do not support the `$FIELDHELP$` keyword; it returns `NULL`. This keyword is `NULL` on the server.
`$FIELDID$`	The ID of the field that currently has focus in the client. Returns `NULL` if the `FIELDHELP` keyword is not implemented.
`$FIELDLABEL$`	The label of the field that currently has focus in the client.
`$FIELDNAME$`	The name of the field that currently has focus in the client.
`$FILTER_ERRNO$`	An error-handling keyword that was replaced by `$ERRNO$`, but remains for backward compatibility purposes.
`$FILTER_ERRMSG$`	An error-handling keyword that was replaced by `$ERRMSG$`, but remains for backward compatibility purposes.
`$FILTER_ERRAPPENDMSG$`	An error-handling keyword that was replaced by `$ERRAPPENDMSG$`, but remains for backward compatibility purposes.
`$GROUPIDS$`	Returns the list of the group IDs of which the current user is a member. If there are no groups, the value is `NULL`. If there are groups, the value is a string displayed in the following format: `;`*`groupID`*`;`*`groupID`*`;`*`groupID`*`;`
`$GROUPS$`	Returns the list of the groups of which the current user is a member.
`$GUIDE$`	The name of the currently running guide. (The name is `$NULL$` if no guide is running.) This keyword is useful for determining if an active link or filter is executing as a part of normal workflow or in a guide.
`$GUIDETEXT$`	The text that is entered under the Help tab of a guide definition. Web applications do not support the `$GUIDETEXT$` keyword; it returns `NULL`. This keyword is `NULL` on the server.

Table A-3: Keywords (Sheet 5 of 8)

Keyword	Value
`$HARDWARE$`	The hardware on which the process is running. This is the name each hardware vendor has given their hardware: ■ On UNIX platforms, the name returned by the `uname -m` command (for example, sun4c). ■ On PC platforms, the processor type (for example, PC i486). You can use this keyword to build filters, escalations, and active links that execute only if the process is running on an appropriate platform. Web applications do not support the `$HARDWARE$` keyword; it returns `NULL`.
`$HOMEURL$`	The mid tier translates `$HOMEURL$` to the base URL of the current page. BMC Remedy User translates `$HOMEURL$` to the default web path specified on the AR System Administration: Server Information form. The `$HOMEURL$` keyword enables you to use the relative path to the application resources when creating workflow. For example, in the Set Fields action, you can enter the following text for the Set Value for the View Field: `$HOMEURL$/Resources/test.html` The URL is resolved at the runtime and loads a different file without changing the Set Fields action.
`$INBULKTRANSACTION$`	Indicates if the current execution is within a bulk transaction. The values returned are: ■ 0 = False ■ 1 = True The keyword always expands to a value of 0 on AR System clients.
`$LASTCOUNT$`	The number of requests returned from the most recent search. You can use this keyword with *any* search, including one run from the search window, a Set Fields operation, a macro, a table refresh, and so on. Note: The `$LASTCOUNT$` keyword is not expanded when default values are set. This allows the value to be set when the form is submitted rather than when the form is opened. Consequently, the literal keyword is displayed in the field before the request is submitted.
`$LASTID$`	Upon a successful submit, the keyword contains the Request ID of the most recently created request during the user's login environment. Note: The `$LASTID$` keyword is not expanded when default values are set. This allows the value to be set when the form is submitted rather than when the form is opened. Consequently, the literal keyword is displayed in the field before the request is submitted.

Table A-3: Keywords (Sheet 6 of 8)

Keyword	Value
`$LASTOPENEDWINID$`	Sent Event keyword that resolves to the ID of the window that was last opened. Every window that is opened, including children, is saved as the last opened window. The value of this keyword is in effect until another window is opened. The format and length of window IDs are not constrained. The value returned from `$LASTOPENEDWINID$` should be saved into a character field with a database length of zero (unlimited length). This keyword is `NULL` on the server.
`$LOCALE$`	The language and country code for the specified locale, in the format *`language_countryCode`*. To view a list of the language and county codes, open a form in BMC Remedy Developer Studio. In the Properties tab, click in the Locale property. Locales appear in the Value list for the property.
`$NULL$`	A null value (no value).
`$OPERATION$`	The current operation.
`$OS$`	The operating system of the computer on which a process is running. You can use this keyword to build workflow that executes conditionally, based on the current operating system. Web applications do not support the `$OS$` keyword; it returns `NULL`.
`$ROLES$`	For a deployable application, returns the list of roles that map to groups to which the current user belongs.
`$ROWCHANGED$`	Evaluates the state of a row in a table field: ■ 0 = Loaded / unchanged. ■ 1 = Modified. ■ 2 = New. ■ 3 = Deleted. The row is hidden from users but visible to workflow. Values 2 and 3 are returned only by AR System 7.6.02 or later. This keyword is valid only in the context of a table loop guide.
`$ROWSELECTED$`	Evaluates if a row in a table field is selected: ■ 0 = Not selected ■ 1 = Secondary selection ■ 2 = Primary selection This keyword is valid only in the context of a table loop guide. Note: The `$ROWSELECTED$` keyword is not expanded when default values are set. This allows the value to be set when the form is submitted rather than when the form is opened. Consequently, the literal keyword is displayed in the field before the request is submitted.
`$SCHEMA$`	The name of the current form.

Table A-3: Keywords (Sheet 7 of 8)

Keyword	Value
`$SCHEMA_ALIAS$`	The singular alias used for a form. This is the form's active alias, which comes from one of the request aliases of the active view (VUI).
`$SERVER$`	The name of the current AR System server.
`$SERVERTIMESTAMP$`	For Date / Time fields, the current date and time on the server. For Time fields, the current time on the server. For Date fields, the current date on the server. Note: The `$SERVERTIMESTAMP$` keyword is not expanded when default values are set. This allows the value to be set to the date and time (on the server) the form is submitted rather than the date and time the form is opened. Consequently, the literal keyword is displayed in the field before the request is submitted.
`$SHIFT_KEY$`	To verify if the Shift key is pressed. The value is set to 1 when the Shift key is pressed while starting the workflow from a control field, a double-click or return on a table field. Any other value represents that the Shift key is not pressed.
`$TCPPORT$`	The TCP / IP port of the local AR System server. This keyword is an integer data type. Web applications do not support the `$TCPPORT$` keyword; it returns `NULL`.
`$TIME$`	For Date / Time fields, the current time (date defaults to current day). Anything stored in a date / time field is stored as the number of seconds since UNIX epoch time, which includes the date *and* time. For Time fields, the current time. For Date fields, the number of seconds since 12:00:00 a.m., which is converted to a date value. Note: The `$TIME$` keyword is not expanded when default values are set. This allows the value to be set to the time the form is submitted rather than the time the form is opened. Consequently, the literal keyword is displayed in the field before the request is submitted.
`$TIMESTAMP$`	For Date / Time fields, the current date and time. For Time fields, the current time. For Date fields, the current date. Note: The `$TIMESTAMP$` keyword is not expanded when default values are set. This allows the value to be set when the form is submitted rather than when the form is opened. Consequently, the literal keyword is displayed in the field before the request is submitted.
`$USER$`	The current user login.

Table A-3: Keywords (Sheet 8 of 8)

Keyword	Value
`$VERSION$`	The software version running on the client (active links) or server (filters and escalations). This keyword includes any applicable patch number.
`$VUI$`	The label of the form view displayed in the active window. This keyword is `NULL` on the server.
`$VUI_TYPE$`	The view's platform (such as web or Windows). When used in workflow, this keyword is the platform value of the active form view. For filters and escalations, this keyword resolves to an empty string. This keyword is similar to `$VUI$` except that its value is the VUI type of the active view. The values for this keyword are `NONE`, `WINDOWS`, `WEB(RELATIVE)`, `WEB(FIXED)`, `WIRELESS`. The number representations of the different view platforms are in the `ar.h` file.
`$WEEKDAY$`	The current day of the week as a character string. Note: The `$WEEKDAY$` keyword is not expanded when default values are set. This allows the value to be set to the day of the week the form is submitted rather than the day of the week the form is opened.

You can use the `$OPERATION$` keyword in qualifications to determine whether the filter or active link should execute on the current operation. For example, the qualification `$OPERATION$="SET"` causes the filter or active link to execute when a request is modified but not when a request appears in display mode. You can also use this keyword to capture audit information. For example, setting a field to the value:

```
"Filter X executed on operation "+$OPERATION$
```

tells you whether a filter executed during a submit operation (`CREATE` returned) or a modify operation (`SET` returned).

Table A-4: $OPERATION$ values (Sheet 1 of 2)

$OPERATION$ value	Description	Filter execution conditions	Form mode
`CREATE`	Set when creating a request.	Submit	New
`DELETE`	Set when deleting a request.	Delete	
`DIALOG`	Set when opening a form as a dialog box.		Dialog
`GET`	Set when retrieving a request for display.	Get	Display
`MERGE`	Set when merging requests.	Merge	
`QUERY`	Set when searching for requests.		Search

Table A-4: $OPERATION$ values (Sheet 2 of 2)

$OPERATION$ value	Description	Filter execution conditions	Form mode
`SET`	Set when modifying a request.	Modify	Modify
`SET ALL`	Set when a Push Fields action that is configured to modify all matching requests modifies a request.	Modify	Modify All

Functions

Table A-5 describes functions that you can use to specify field values in the following workflow actions:

- Open Window
- Push Fields
- Service
- Set Fields

All functions listed in Table A-5 work with active links, filters, and escalations, with the following exceptions:

- The Hover, Listget, Listsize, Mapget, and Template functions work only in active links.
- The Encrypt and Decrypt functions work only in filters and escalations.

Table A-5: Functions (Sheet 1 of 12)

Function	Arguments	Return	Description
`AR_FUNCTION_ DROPPEDROWINDEX`	(field ID)	int	For a row on which a drop event occurs on a list view, tree view, or cell-based table field, returns the 1-based row index of the row for the most recent drop operation.
`AR_FUNCTION_ DROPPEDCOLUMNINDEX`	(field ID)	int	For a column on a drop event occurs on a list view, tree view, or cell-based table field, returns the 1-based column index of the column for the most recent drop operation.

Table A-5: Functions (Sheet 2 of 12)

Function	Arguments	Return	Description
`COLAVG`	(column)	any (matches column)	For a given column on a list view, tree view, or results list table field, returns the average of all non-`NULL` row values as a real type. Exceptions: ■ For char columns, converts the column values to numeric and averages the numeric values. ■ If the column type is currency, the average of the currency value is calculated, and the currency type is ignored. ■ If the column type is enum (radio button or selection), the average of the enum ID is calculated. ■ `COLAVG` ignores date/time, date, and time columns; and returns a `NULL` value.
`COLCOUNT`	(column) or (table)	int	For a list view, tree view, or results list table field, returns the total number of rows. For a column field, returns the total number of non-`NULL` rows in a given column.
`COLMAX`	(column)	any (matches column)	For a given column on a list view, tree view, or results list table field, returns the maximum value of the row values.
`COLMIN`	(column)	any (matches column)	For a given column on a list view, tree view, or results list table field, returns the minimum value of the row values.
`COLSUM`	(column)	any (matches column)	For a given column on a list view, tree view, or results list table field, returns the sum of all values as a real type. Exceptions: ■ For char columns, converts the column values to numeric and sums up the numeric values. ■ If the column type is currency, the sum of the currency value is calculated, and the currency type is ignored. ■ If the column type is enum (radio button or selection), the sum of the enum ID is calculated. ■ `COLSUM` ignores date/time, date, and time columns; and returns a `NULL` value.

Table A-5: Functions (Sheet 3 of 12)

Function	Arguments	Return	Description

`CURRCONVERT`	(currency, type, timestamp)		For a given currency, converts currency to a different type according to new date, and recalculates the functional currency values.
			Use syntax such as:
			`CURRCONVERT($`*`currencyField`*`$, $`*`currencyField2`*`$.TYPE$, $TIMESTAMP$)`
			Use other values for the type and timestamp, such as a character field and a date field. For example:
			`CURRCONVERT($`*`currencyField`*`$, $`*`characterField`*`$, $`*`dateField`*`$)`
			For information about data conversion rules for currency fields, see the *Form and Application Objects Guide*, "Workflow considerations," page 374.

`CURRSETDATE`	(currency, timestamp)		For a given currency, sets a new date of currency and recalculates the functional currency values.
			Use syntax such as:
			`CURRSETDATE($`*`currencyField`*`$, $TIMESTAMP$)`
			Use other values for the timestamp, such as a date field or integer. For example:
			`CURRSETDATE($`*`currencyField`*`$, 1026779689)`
			For information about data conversion rules for currency fields, see the *Form and Application Objects Guide*, "Workflow considerations," page 374.

`CURRSETTYPE`	(currency, type)		For a given currency, sets new type of currency and recalculates the functional currency values.
			Use syntax such as:
			`CURRSETTYPE($`*`currencyField`*`$, $`*`currencyField2`*`$.TYPE$)`
			Use other values for the type, such as a character field. For example:
			`CURRSETTYPE($`*`currencyField`*`$, $`*`characterField`*`$)`
			For information about data conversion rules for currency fields, see the *Form and Application Objects Guide*, "Workflow considerations," page 374.

Table A-5: Functions (Sheet 4 of 12)

Function	Arguments	Return	Description

`CURRSETVALUE`	(currency, value)		For a given currency, sets new value of currency and recalculates the functional currency values.
			Use syntax such as:
			`CURRSETVALUE($`*`currencyField`*`$, $`*`currencyField2`*`$.VALUE$)`
			Use another field to set the currency value, such as a decimal field. For example:
			`CURRSETVALUE($`*`currencyField`*`$, $`*`decimalField`*`$)`
			For information about data conversion rules for currency fields, see the *Form and Application Objects Guide*, "Workflow considerations," page 374.

`DATE`	(timestamp)	char	For a given date, returns the date portion of the time stamp.
`DATEADD`	(datepart, number, date)	date	Adds a specified number of days, weeks, months, or years to the date and returns the new date. Specify `datepart` using one of the following quoted values: ■ **Year**—"`year`", "`yy`", or "`yyyy`" ■ **Month**—"`month`", "`mm`", or "`m`" ■ **Day**—"`day`", "`dd`", or "`md`" ■ **Week**—"`week`", "`wk`", or "`ww`" The `date` parameter is the date value to add to. For example, to add 10 weeks to the 05/20/02 date, enter: `DATEADD("ww", 10, "05/20/08")`
`DATEDIFF`	(datepart, startdate, enddate)	int	Depending on the value of `datepart` (day or week), returns the number of days or weeks between the start date and the end date. Specify `datepart` using one of the following quoted values: ■ **Day**—"`day`", "`dd`", or "`md`" ■ **Week**—"`week`", "`wk`", or "`ww`" For example, to find the number of days between date1 and date2, enter: `DATEDIFF("dd", $date1$, $date2$)`

Table A-5: Functions (Sheet 5 of 12)

Function	Arguments	Return	Description
`DATENAME`	(datepart, date)	char	Depending on the value of `datepart` (month or weekday), returns the name of the month or the name of the day of the week corresponding to `date`. Specify `datepart` using one of the following quoted values: ▪ **Month**—"`month`", "`mm`", or "`m`" ▪ **Weekday**—"`weekday`" or "`wd`" For example, to find the weekday for December 31, 2007, enter: `DATENAME("wd", "12/31/07")`
`DATENUM`	(datepart, date)	int	Depending on the value of `datepart` (year, month, week, day, or weekday), returns the numeric value of the year (4 digits), month (1 to 12), week (1 to 52), day (1 to 31), or weekday (1=Sunday, 2=Monday, and so on). Specify `datepart` using one of the following quoted values: ▪ **Year**—"`year`", "`yy`", or "`yyyy`" ▪ **Month**—"`month`", "`mm`", or "`m`" ▪ **Day**—"`day`", "`dd`", or "`md`" ▪ **Week**—"`week`", "`wk`", or "`ww`" ▪ **Weekday**—"`weekday`" or "`wd`" For example, `DATENUM("mm", "12/31/07")`returns 12.
`DAY`	(timestamp)	int	Returns the day of the time stamp (1 to 31).

`DECRYPT`	(cyphertext, key)		Used in Set Fields filter actions only. Returns the unencrypted text value of the encrypted text (cyphertext), using the encryption key (key). The return value is smaller than the size of the encrypted string. When decrypting, you must use the same key that you used when encrypting.
			For example, to decrypt the string in `Field1` using the key `my_key`, enter:
			`DECRYPT($Field1$, "my_key")`
			To decrypt a string using a key in `KeyField`, enter:
			`DECRYPT("`*`stringToDecrypt`*`", $KeyField$)`
			Note: Use this function with character fields only. See information about the Encrypted String field in Appendix B, "Reserved fields," of the *Form and Application Objects Guide.*

Table A-5: Functions (Sheet 6 of 12)

Function	Arguments	Return	Description

`ENCRYPT`	(plaintext, key)		Used in Set Fields filter actions only. Returns the encrypted value of a text string (plaintext), using the encryption key (key). The return value is larger than the size of the unencrypted string.
			For example, to encrypt the string in `Field1` using the key `my_key`, enter:
			`ENCRYPT($Field1$, "my_key")`
			To encrypt a string using a key in `KeyField`, enter:
			`ENCRYPT("`*`wordToEncrypt`*`", $KeyField$)`
			Note: Use this function with character fields only. See information about the Encrypted String field in Appendix B, "Reserved fields," of the *Form and Application Objects Guide*.

`HOUR`	(timestamp)	int	Returns the hour of the time stamp (0 to 23).
`HOVER`	(column)	char	For a column in a list view, tree view, a results list table field, or a cell-based table, the string value of the column in the row being hovered over is returned. For a table itself, the row number of the hovered over row is returned. For a given attachment pool, a string value containing the file name (if any) of the hovered attachment is returned. For example, to get the `HOVER` value for a table column with ID 536880913, enter: `HOVER(536880913)` Note: When the field being referenced for the HOVER function is not being hovered over, an empty string is returned. This function should be used in conjunction with an On Hover active link execution option.
`LEFT`	(char,int)	char	For single-byte languages. Returns the left-most *bytes* of the first parameter (char) up to the number of *bytes* indicated by the second parameter (int). For example, to set the value of a field to the first ten bytes of the Submitter name, enter: `LEFT($Submitter$,10)`
`LEFTC`	(char,int)	char	For single-byte or multi-byte languages. Returns the left-most *characters* of the first parameter (char) up to the number of *characters* indicated by the second parameter (int). For example, to set the value of a field to the first ten characters of the Submitter name, enter: `LEFTC($Submitter$,10)`

Table A-5: Functions (Sheet 7 of 12)

Function	Arguments	Return	Description
`LENGTH`	(char)	int	For single-byte languages. Returns the number of bytes in the string (char) in BMC Remedy User, or returns the number of characters in the string (char) in browsers. Note: When a length (int) is located in the middle of double-byte characters, the length is treated as (length -1).
`LENGTHC`	(char)	int	For single-byte or multi-byte languages. Returns the number of *characters* in the string (char).
`LISTGET`	(fieldid, index)	string	Returns the value from the list that is maintained by a character field, at the indicated position. If the range of the index is not between `1` and `LISTSIZE()`, this function returns `NULL`. For example, if character field `1234` contains the value `abc;def;ghi;jkl`, calling `LISTGET(1234,2)` returns the value `def`.
`LISTSIZE`	(fieldid)	int	Returns the number of entries in a list or map that is maintained by a character field. If the value is `NULL`, the function returns `0`. For example, if character field `1234` contains the value `abc;def;ghi;jkl`, calling `LISTSIZE(1234)` returns the value `4`.
`LOWER`	(char)	char	Returns all characters in the string (char) as lowercase characters. Note: No conversion is performed to double-byte characters.
`LPAD`	(char,int,char)	char	For single-byte languages. Returns the value that results from padding the first parameter (char) to the left with the value of the third parameter (char) so that the resulting value is the length (in *bytes*) of the second parameter (int) or the length of the original string, whichever is longer. For example, if you want the results of a Set Fields operation to be a 15-byte value with the prefix `LEAD`, followed by zeros, and ending in the contents of the integer field `Call #`, enter: `LPAD($Call #$,15,"LEAD00000000000").` If the `Call #` field contains the number 947, the result of the Set Fields action is `LEAD00000000947.`

Table A-5: Functions (Sheet 8 of 12)

Function	Arguments	Return	Description
`LPADC`	(char,int,char)	char	For single-byte or multi-byte languages. Returns the value that results from padding the first parameter (char) to the left with the value of the third parameter (char) so that the resulting value is the length (in *characters*) of the second parameter (int) or the length of the original string, whichever is longer. For example, if you want the results of a Set Fields operation to be a 15-character value with the prefix `LEAD`, followed by zeros, and ending in the contents of the integer field `Call #`, enter: `LPADC($Call #$,15,"LEAD00000000000").` If the `Call #` field contains the number 947, the result of the Set Fields action is `LEAD00000000947.`
`LTRIM`	(char)	char	Returns the value of (char) after deleting any blank spaces and tabs to the left.
`MAPGET`	(fieldid, key)	int, string	Returns a value from a map that is present within a character field. If the key does not have a value, this function returns `NULL`. For example, if character field `1234` contains the value `type=4;Name=""John Smith"";operation=""="";status=Open`, calling `MAPGET(1234,"Name")` returns the value `John Smith`. Note: You can use double-quotes to escape the value containing spaces or delimiters.
`MAX`	(any,any[,any]...)	any (matches input)	Returns the maximum value of the set specified. The data type of all values must match for the result to be meaningful. For example, to check the current time and the escalation time, and return the greater (latest) value of the two, enter: `MAX ($Escalate Date$, $TIMESTAMP$).` `MAX` returns `$NULL$` only if *all* arguments in the function evaluate to `$NULL$`.

Table A-5: Functions (Sheet 9 of 12)

Function	Arguments	Return	Description
`MIN`	(any,any[,any]...)	any (matches input)	Returns the minimum value of the specified set. The data type of all values must match for the result to be meaningful. For example, to check the current time and the escalation time, and return the lower (earliest) value of the two, enter: `MIN ($Escalate Date$, $TIMESTAMP$).` If any argument evaluates to `$NULL$`, the argument is not considered in the evaluation of the function. For example, if `MyField` is empty (has a value of `$NULL$`), but the field `HasValue` has a value of 10, then `MIN($MyField$, $HasValue$)` returns 10. `MIN` returns `$NULL$` only if *all* arguments in the function evaluate to `$NULL$`.
`MINUTE`	(timestamp)	int	Returns the minute of the time stamp (0 to 59).
`MONTH`	(timestamp)	int	Returns the month of the time stamp (1 to 12).

`REPLACE`	(char,char,char)	char	Returns the value that results from replacing any occurrences of the second parameter (char) found in the first parameter (char) with the contents of the third parameter (char).
			For example, to replace the name Bob with the name Robert, enter:
			`REPLACE ($Submitter$, "Bob", "Robert").`
			To replace all occurrences of the double-quote character (") within a character string with another character, for example, replacing any instance of a double quote with the name Robert, enter:
			`REPLACE ($Submitter$, """", "Robert").`
			As a result, *anywhere* a double quote is found in the Submitter field, the double quote is replaced with the name Robert.

`RIGHT`	(char,int)	char	For single-byte languages. Returns the right-most *bytes* of the first parameter (char) up to the number of *bytes* indicated by the second parameter (int). For example, to set the value of a field to the last four bytes of an account code, enter: `RIGHT($Account#$,4).`

Table A-5: Functions (Sheet 10 of 12)

Function	Arguments	Return	Description
`RIGHTC`	(char,int)	char	For single-byte or multi-byte languages. Returns the right-most *characters* of the first parameter (char) up to the number of *characters* indicated by the second parameter (int). For example, to set the value of a field to the last four characters of an account code, enter: `RIGHTC($Account#$,4)`.
`ROUND`	(real), (decimal), or (currency)	int or currency	Returns the rounded value of a real number or a complex data type. For a decimal value, for example, 5.1 to 5.4 are rounded to 5, and 5.5 to 5.9 are rounded to 6. For a currency value, for example, 5.01 USD through 5.49 USD are rounded to 5.00 USD, and 5.50 USD through 5.99 USD are rounded to 6.00 USD.
`RPAD`	(char,int,char)	char	For single-byte languages. Returns the value that results from padding the first parameter (char) on the right with the value of the third parameter (char) so that the resulting value is the length (in *bytes*) of the second parameter (int) or the length of the original string, whichever is longer. For example, enter `RPAD($Submitter$,20," ")` to add blank spaces after a submitter's name to make the entry 20 bytes long.
`RPADC`	(char,int,char)	char	For single-byte or multi-byte languages. Returns the value that results from padding the first parameter (char) on the right with the value of the third parameter (char) so that the resulting value is the length (in *characters*) of the second parameter (int) or the length of the original string, whichever is longer. For example, enter `RPADC($Submitter$,20," ")` to add blank spaces after a submitter's name to make the entry 20 characters long.
`RTRIM`	(char)	char	Returns the value of (char) after deleting any blank spaces and tabs to the right.
`SECOND`	(timestamp)	int	Returns the seconds from the time stamp (0 to 59).
`SELECTEDROWCOUNT`	(tableId)	int	For tree view tables, it returns the number of leaf nodes represented by the selected node. If a node does not contain a leaf node, this function returns `1`. For list view tables, it returns the number of selected rows. For cell-based tables, it returns the number of selected cells.

Table A-5: Functions (Sheet 11 of 12)

Function	Arguments	Return	Description
`STRIPHTML`	(field ID)	char	Removes all formatting and image information from the contents of a rich-text-formatted field. The function returns a plain text string. See the *Form and Application Objects Guide*, "Adding rich-text-formatting capabilities to a character field," page 186.
`STRSTR`	(char,char)	int	For single-byte languages. Returns the position (in *bytes*) of the second parameter (char) if it is found in the first parameter (char). If the second string is not found, returns a -1. This function is zero-indexed (that is, numbering of bytes begins at 0).
`STRSTRC`	(char,char)	int	For single-byte or multi-byte languages. Returns the position (in *characters*) of the second parameter (char) if it is found in the first parameter (char). If the second string is not found, returns a -1. This function is zero-indexed (that is, numbering of characters begins at 0).
`SUBSTR`	(char,int [, int])	char	For single-byte languages. Returns the substring of *bytes* in the first parameter (char) starting at the position (in *bytes*) indicated by the second parameter (int) and continuing to the position indicated by the third parameter (int). The string is zero-indexed (that is, numbering of bytes begins at 0). If the third parameter is not included, the function returns bytes to the end of the string. If the third parameter is less than the second parameter, the function returns a `NULL` value. For example, to set the value of a field to six bytes of the Location field, skipping a three-byte prefix, enter: `SUBSTR($Location$, 3, 8).`

Table A-5: Functions (Sheet 12 of 12)

Function	Arguments	Return	Description
`SUBSTRC`	(char,int [, int])	char	For single-byte or multi-byte languages. Returns the substring of *characters* in the first parameter (char) starting at the position (in *characters*) indicated by the second parameter (int) and continuing to the position indicated by the third parameter (int). The string is zero-indexed (that is, numbering of characters begins at 0). If the third parameter is not included, the function returns characters to the end of the string. If the third parameter is less than the second parameter, the function returns a `NULL` value. For example, to set the value of a field to six characters of the Location field, skipping a three-character prefix, enter: `SUBSTRC($Location$, 3, 8).`
`TEMPLATE`	(char,char,char,...)	char	Returns a string from the specified template formatted with values substituted for template parameters. The first parameter is the template name. The template parameters and their values follow in pairs. For example: `TEMPLATE("Simple Template", "First Name", $Short Description$, "Last Name", $Submitter$, "City", "Sunnyvale")`
`TIME`	(timestamp)	char	Returns the time portion of the time stamp.
`TRUNC`	(real), (decimal), or (currency)	int or currency	Returns the truncated value of a real number or a complex data type. For a decimal value, for example, 5.1 through 5.9 are truncated to 5. For a currency value, for example, 5.01 USD through 5.99 USD are truncated to 5.00 USD.
`UPPER`	(char)	char	Returns all characters in the string (char) as uppercase characters. Note: No conversion is performed to double-byte characters.
`WEEKDAY`	(timestamp)	int	Returns the weekday of the time stamp (1 to 7, where 1=Sunday and 7=Saturday).
`YEAR`	(timestamp)	int	Returns the year portion of the time stamp.

NULL values, relational algebra, and AR System

This section explains how `NULL` values are used and interpreted within relational algebra qualifications, and discusses some special interpretations and overrides used within AR System.

About the NULL value

The `NULL` value is a special value that is different from any valid value for any data type. It is a legal value in any data type, and is used to represent missing or inapplicable information. Another way of thinking about `NULL` is as no value or the lack of information about the value.

Relational algebra and qualifications involving NULL

All searching in the database uses the SQL language, which is based on the concepts of relational algebra. The following example illustrates how these concepts apply to the `NULL` value.

Suppose there is a form with a selection field called Field X, which allows two possible states: Yes or No. The form contains 20 records of which Field X is set to Yes for 7 of them, No for 8 of them, and `NULL` for 5 of them. When you search for the records, you receive the following results:

- Search for all tickets (without any conditions) results in 20 records returned.
- Search for all tickets where `'Field X' = NULL` results in 5 records returned.
- Search for all tickets where `'Field X' = "Yes"` results in 7 records returned.
- Search for all tickets where `'Field X' = "No"` results in 8 records returned.

If you now search for all tickets where `'Field X' != "Yes"`, you might expect to receive 13 records returned (8 records where the field is No and 5 where it is `NULL`) but you receive only 8 records. This is correct behavior according to the rules of relational algebra. `NULL` is `NULL`; that is, it has no value. You must explicitly look for `NULL` values as they are not implicitly included in queries that search for values.

Relational algebra does not follow Boolean logic, where conditions are either true or false. Instead, every condition evaluates as one of `TRUE`, `FALSE`, or `UNKNOWN`. This is called three valued logic. The result of a comparison is `UNKNOWN` if either value being compared is the `NULL` value. Rows satisfy a search condition if the result of the condition is `TRUE`. Rows for which the condition is `UNKNOWN` do not satisfy the search condition.

The only way to find a row with a `NULL` value for a field in a search is to explicitly search for whether the field has a `NULL` value. Testing for `NULL` is the only case that a `NULL` value matches. So, in the example, to find all entries that are not Yes or are `NULL`, the qualification is

```
'Field X' != "Yes" OR 'Field X' = NULL
```

These criteria find the 13 records in the example data set.

Any comparison other than *equal to* and *not equal to* results in a value of UNKNOWN. In AND operations, at least one item is NOT TRUE, so the qualification does not succeed. In OR operations, this clause with NULL is UNKNOWN, so the qualification depends on the result of the other clauses. If all clauses evaluate to UNKNOWN, the qualification fails as a qualification.

If a NULL value is involved in an arithmetic operation, the result of the operation is NULL. In other words, any time a NULL value is involved, the entire operation becomes NULL.

As an example for this functionality, look at the following qualification

```
('Field A' = 5) OR ('Field B' > 'Field C' + 37)
```

If Field C has a NULL value, the second clause evaluates to UNKNOWN. Since the operation is an OR, the result of the qualification depends on whether Field A is set to 5 (success) or not (failure).

An alternative to using NULL is to have a value or state that represents unknown. Then, you always assign the value of Yes, No, or Unknown. Filters can be used to assign Unknown if the field is NULL. Then, the field always has a value and you do not have the issue about working with a NULL value.

NULL values and AR System

In qualifications, a NULL value follows all the characteristics of relational algebra as defined in the previous section. Since search operations are passed directly to the database and they base their searches on relational algebra, this means AR System is based on relational algebra.

However, you are in full control of value assignment and the use of values within workflow. In these cases, you do not follow the strict rules about NULL handling. This allows the system to return a result more in line with what is expected. For example, if you wanted to perform a Set Fields operation where you assigned a Name field by concatenating the First Name, Middle Initial, and Last Name field, you might make an assignment like this:

```
'Name' = $First Name$ + $Middle Initial$ + $Last Name$
```

If processing used strict relational algebra, and the Middle Initial was NULL, the name assigned is be NULL. According to relational algebra, a NULL value within an operation causes the result of the operation to be NULL.

Instead, AR System treats a NULL value within workflow actions (other than queries) as follows:

- **Character, Diary, Timestamp, Selection**—An empty string
- **Integer, Real, Decimal**—A value of 0

In the example, the name is assigned the First and Last name without middle initial.

By treating a NULL value as an empty value rather than as UNKNOWN in a true relational algebra sense, the result is what is expected instead of a NULL value.

Appendix

B Additional ways to use the Set Fields action

In addition to the basic instructions described in "Set Fields action" on page 129, this section discusses various ways you can use the Set Fields active link action to assign a value in a field.

The following topics are provided:

- Assigning values through SQL statements (page 244)
- Assigning values by issuing requests to a Filter API Plug-In service (page 249)
- Assigning values using function results (page 251)
- Assigning values from process results (page 253)
- Assigning values by using arithmetic operations (page 255)

For information about assigning values from a DDE request, see the *Integration Guide*, "Using active links with DDE," page 290.

Assigning values through SQL statements

In addition to setting field values based on AR System data, you can retrieve data from other sources by submitting SQL queries for use within AR System workflow. Much like issuing a search to get data from another form, you can issue an SQL statement to retrieve data from any relational database table. This feature enables you to select data from databases other than AR System databases, which is useful for integrating AR System with external data sources.

NOTE

You can also use vendor forms to access external data, which could reduce the risk of exposing your application to any issues with the other source. For information about accessing external data using vendor forms, see the *Integration Guide*, "Vendor forms," page 183.

The ability to submit SQL statements enables you to issue complex queries to the database. This is useful if you want to use database features specific to a particular database platform.

The SQL statement you use to set a field value in the Set Fields action has a different result than the Direct SQL action (see "Direct SQL action" on page 83). In the Set Fields action, you use SQL to search the database for information and then use the returned values to set fields. With the Direct SQL action, the SQL command is not expected to return a value.

To run more than one SQL command, use stored procedures or functions or any other extension supported by your database. A stored procedure with a Set Fields action executes all its commands but does *not* return a value.

For the most effective use of SQL statements, you must have a general understanding of relational databases and a specific understanding of the relational database underlying your AR System.

WARNING

Because AR System passes SQL commands to the database without checking the syntax, *all* commands are submitted to the database. Make sure all submitted commands achieve the needed result. Your SQL commands should comply with ANSI SQL standards, so that single quotes are reserved for strings, and double quotes are reserved for use with database object names only.

▶ To assign a value by submitting an SQL command

1 From the Server Name list, make one of the following selections:

- For *active links*, select the server that contains the value that you want to retrieve.
- For *filters* and *escalations*, select the current server.

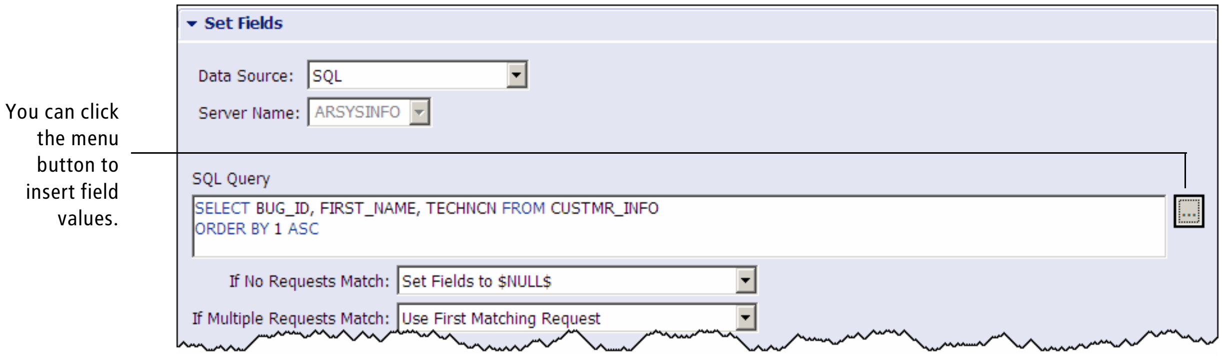
2 From the Data Source list, select SQL.

3 In the SQL Query field, enter the SQL command to issue to the database.

For example, you can enter an SQL statement with a command to display three columns of data from a table and sort the data in ascending order based on the first column:

```
SELECT BUG_ID, FIRST_NAME, TECHNCN FROM CUSTMR_INFO
ORDER BY 1 ASC
```

Figure B-1: SQL statement with SELECT command in SQL Query field

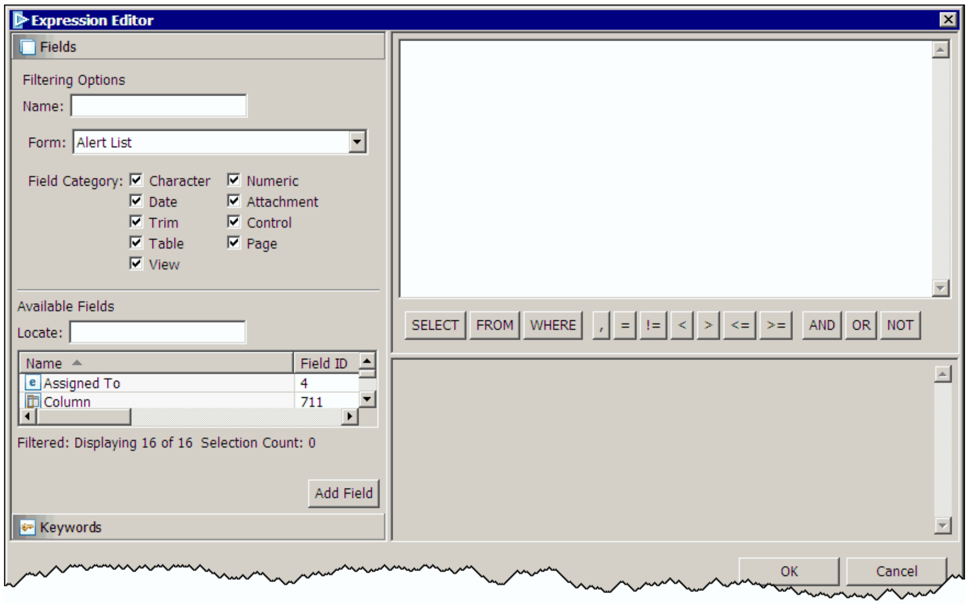


Data Source: SQL
Server Name: ARSYSINFO
SQL Query:
SELECT BUG_ID, FIRST_NAME, TECHNCN FROM CUSTMR_INFO
ORDER BY 1 ASC
If No Requests Match: Set Fields to $NULL$
If Multiple Requests Match: Use First Matching Request

You can enter a command by typing it, or by entering a qualification in the Expression editor. When you type a command, you can use Ctrl+S to display a completion menu for SQL syntax, and $ to display a completion menu for fields and keywords.

If you are not sure which fields to reference, you can build an SQL statement in the Expression Editor. Entering qualifications in the Expression Editor.

Figure B-2: Using the Expression Editor to enter a qualification



Name:
Form: Alert List
Field Category: ☑ Character ☑ Numeric ☑ Date ☑ Attachment ☑ Trim ☑ Control ☑ Table ☑ Page ☑ View
Locate:

Name	Field ID
Assigned To	4
Column	711

Filtered: Displaying 16 of 16 Selection Count: 0

NOTE

Enter only one SQL command for each Set Fields action. Do not end the SQL command with `run` or `go`.

Use the SQL Query field menu button to insert field values or keywords in the SQL statement using the Expression Editor. As shown in the following example, fields and keywords must be enclosed in dollar signs to indicate that the server should expand these values before issuing the command:

```
SELECT BUG_ID, FIRST_NAME, TECHNCN FROM CUSTMR_INFO WHERE
colName = '$field$'
```

You might have to insert single quotation marks manually around the parameter, depending on the content of the expanded value and the context in which you are using it. For example:

- If `colName` is a character field in the `CUSTMR_INFO` table, you must add single quotation marks around `$field$` so that the database interprets the expanded field value as a character string.
- If `colName` is a numeric field, using the single quotation marks results in an SQL syntax error.
- You also get an SQL syntax error if you omit the quotation marks but `field` contains character data.

NOTE
AR System does not verify the validity of your SQL command. The syntax you use must be recognized by the underlying SQL database on which AR System is running.

4 From the If No Requests Match list, select a handling option to control how the system responds when the SQL command returns no matches.

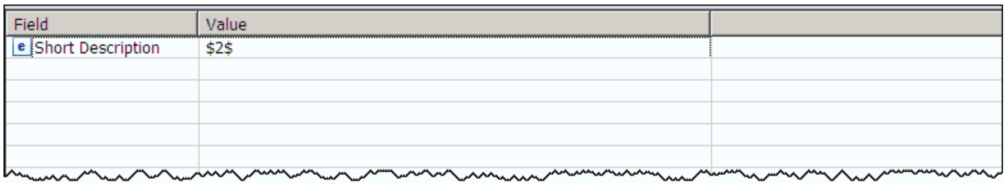
For more information about the options in this step, see "Set Fields action" on page 129.

5 From the If Multiple Requests Match list, select a handling option to control how the system responds when the SQL command returns multiple matches.

For more information about the options in this step, see step 7 in "To use the SERVER data source" on page 132.

6 From the Name list, select the field that you are setting with this action.

Figure B-3: Selecting field being set



Field	Value	
Short Description	2	

7 From the Value list, select SQL Result Column, and then select $*n*$.

The $*n*$ variable represents the number of a column in the SQL result table constructed from the results of the SQL command. When the active link, filter, or escalation executes:

a The SQL command is issued to the database.

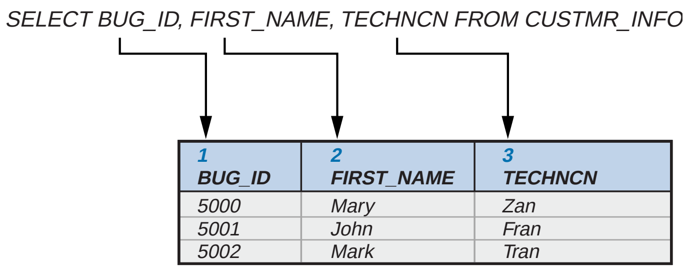
b The results of the SQL command are used to construct an SQL result table.

c The value from column 1 of the SQL result table is loaded into the field that contains a 1, the value from column 2 of the SQL result table is loaded into the field that contains a 2, and so on.

If an SQL command includes three columns, use a $*n*$ variable as high as 3. If you specify a $*n*$ variable that is greater than the number of columns in the SQL command, a NULL value is returned. If you use an asterisk in an SQL command, for example SELECT * FROM CUSTMR_INFO, the menu lets you select an $*n*$ variable as high as 10. However, if you know that 15 values are returned, entering 14 works. Because the first column in the form table is used to set the field that contains 1 (and so on), you must know the order of the form columns to load the correct data into the correct field.

Figure B-1 on page 245 shows an example of values assigned to fields. If you enter the same $*n*$ variables (that is, 1 in the Long Description field, 2 in the Short Description field, and 3 in the Work Around field) and the SQL command shown in Figure B-1, the returned results create an SQL result table that looks like the following figure.

Figure B-4: Results returned from SQL command



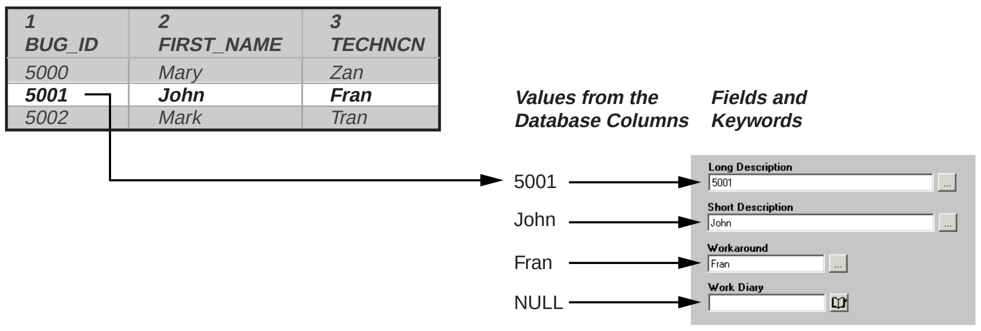
SELECT BUG_ID, FIRST_NAME, TECHNCN FROM CUSTMR_INFO

1 BUG_ID	2 FIRST_NAME	3 TECHNCN
5000	Mary	Zan
5001	John	Fran
5002	Mark	Tran

Because this action also specifies that multiple matches should display a selection list, a selection list of available SQL result table entries appears when the active link executes.

If you select the second selection list entry, the contents of BUG_ID are loaded into the Long Description field (1), the contents of FIRST_NAME are loaded into the Short Description field (2), and so on, as shown in the following figure.

Figure B-5: How database columns are inserted into fields



Entering a `$4$` variable value without an actual fourth column in the SQL command inserts a `NULL` value into the field.

Performing the SQL operation

Observe the following general rules for using SQL commands:

- You need not use every value that is returned from the SQL command. If you do not use any values from the SQL command, the Set Fields action acts like the default selection CURRENT SCREEN (active links) or CURRENT TRANSACTION (filters or escalation), and the system removes your SQL query.
- You can use the same value in more than one field.
- You can issue only one SQL command per action. You cannot enter two commands separated by a semicolon and have both commands run. To run a set of commands, create separate actions, or create a stored procedure and run that. However, stored procedures do not return values.
- Turn on AR System server SQL logging to debug the SQL syntax if it returns unexpected values or results. A good debugging strategy is to start an SQL interpreter (for example, `isql` for Sybase, SQL*Plus for Oracle, Command Center for IBM® DB2®, or Microsoft ISQL / w for Microsoft SQL Server) and to enter the same SQL command directly into the database to verify its validity.
- Because there is no error checking on the SQL statement, run the SQL statement directly against the database (as a test) before you enter it into the SQL Command field. You can then copy and paste the tested SQL command directly into the SQL Command field.
- If the SQL operation fails, an AR System error message and the underlying database error message appear.
- You can affect database performance by how an SQL query is written. If the row has many columns, a `SELECT *` SQL command can have a greater performance impact than if you select specific columns. For more information, see your relational database documentation.

Database security issues

To use SQL commands, familiarize yourself with the features of your underlying database. All SQL commands are issued by the following users:

- For Oracle, Microsoft SQL Server, and Sybase databases, the `ARAdmin` user (or the AR System Database User defined during installation).
- For Informix databases, the user who controls the `arserverd` process.
- For DB2 databases, the user who installs AR System.

Depending on which database you are using, the data must be accessible to the user issuing the command. If you are running AR System as one of these users without permission to access the database, you *cannot* issue the SQL command.

To access databases other than AR System databases, use the database name as part of the SQL command syntax, for example, using MS SQL:

```
DatabaseName.owner.table
getafix.ARAdmin.CUSTMR_INFO
```

Assigning values by issuing requests to a Filter API Plug-In service

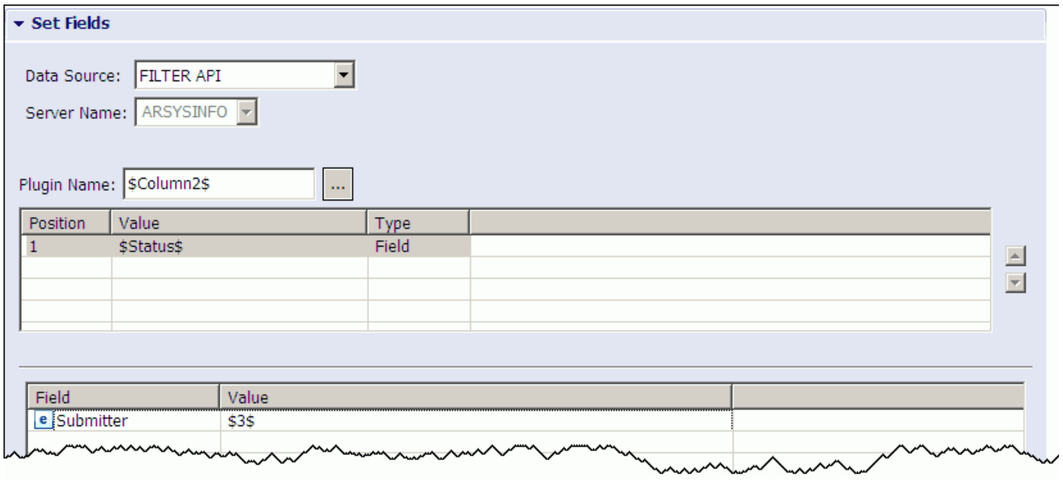
For *filters* and *escalations*, you can pass values from the Set Fields window to a filter API plug-in service. If the service is running, it processes the request issued and return the values to the AR System server. These values are then loaded into the fields.

Before creating a Set Fields action that uses a filter API plug-in, you must create a filter API plug-in and perform certain configuration steps. For more information, see the *C API Reference*, “AR filter API plug-in functions,” page 568.

▶ To assign values by issuing requests to a Filter API Plug-In service

1. In the Create Filter window, choose Set Fields from the Add Action context menu.
2. From the Server Name list, select the current server for filters and escalations.
3. From the Data Source list, select FILTER API.

Figure B-6: Filter API request



Data Source: FILTER API

Server Name: ARSYSINFO

Plugin Name: $Column2$

Position	Value	Type
1	$Status$	Field

Field	Value
Submitter	3

4 In the Plugin Name field, enter the name of a filter API plug-in service.

From the list, you can also select a field that contains a service name. The service name is passed to the plug-in server. If the plug-in server is running and the service exists, the request is processed. Otherwise, the Set Fields action fails and an error is logged.

The Plugin Name list contains all filter API plug-ins that are registered with the plug-in server. For more information, see the *C API Reference*, "AR filter API plug-in functions," page 568.

5 From the Enter Input Values list, enter the values to be passed to the filter API plug-in.

6 After you create a value, click Add to enter it into the Input Value List.

7 Repeat steps 5 and 6 for each value you want to add.

8 In the Input Value List, make any necessary changes:

- Shift the position of a value by clicking the up or down arrows.
- Edit a value by clicking it, changing its value, and then clicking Modify.
- Change the field type by clicking the description and using the drop-down list to select a different type, for example, `Integer` instead of `Char`.
- Remove a row by selecting it and clicking Delete.

These input values are passed to the filter API.

9 From the Name list, select the field that you are setting with the filter API data.

10 From the Value list, select Filter API Values, and then select `$n$`.

The `$n$` variable represents the values that are returned by the filter API plug-in service, where `n` is the index of the value in the returned value list. The menu list provides values up to 20, but you can enter any value. The `$n$` variables work much like the values returned by the SQL command, as described in "To assign a value by submitting an SQL command" on page 244.

You can combine these values using functions and operations, for example, `$4$ + $3$`, that combine the fourth value and the third value of the output value list returned by the filter API.

NOTE

If you do not specify any filter API output values when you save the filter or escalation, it is saved as an ordinary Set Fields action. If you specify an invalid `$n$` variable—for example, the filter API returns only four values but you specify `$5$`—this action is equivalent to assigning a `NULL` value.

When the filter or escalation executes:

a The service name and input list value are passed from the AR System server to the plug-in server.

b The plug-in server passes the request to the filter API and waits for a response.

c The filter API processes the request and passes back the values to the plug-in server.

d The plug-in server passes the value to the AR System server.

e The AR System server fills the fields with the output values that were returned by the filter API.

Assigning values using function results

AR System supports a set of functions that you can use in the Set Fields action. The functions enable you to manipulate data so that you can control aspects of values that you are loading into fields. For example, you can use the UPPER function to convert an ASCII string loaded in a field to uppercase.

▶ To use a function in a Set Fields operation

1 Create a Set Fields action as described in “To use the CURRENT SCREEN or CURRENT TRANSACTION data source” on page 131, but enter information in the Name and Value fields as follows:

   a From the Name list, select the field that you are setting with this action.

   b From the Value list, click Functions to display a list of available functions.

   Table A-5 on page 229 describes each function that AR System supports.

2 Select the appropriate function.

   The function appears in the field with a set of parentheses to its right.

3 For the selected function, enter the arguments within the parentheses.

Table A-5 on page 229 describes each function that AR System supports.

4 Repeat steps 1 through 4 for each field that you want to set with a function.

NOTE

If the value of any of the arguments of a function is `NULL`, the result of the function is `NULL` (the field is empty). To avoid this result, use a qualification that verifies the operation and includes a value for all arguments. For *filters* and *escalations*, if you use an empty field as a parameter in a function, it is considered a `NULL` value. The exception to this rule is the *third* parameter for the `REPLACE` function. If the third parameter is `NULL`, it is interpreted as an empty string.

Specifying arguments that use a comma as a decimal separator

Specifying arguments that use a comma as a decimal separator (in a German locale, for example) requires special consideration for the following functions:

- LPAD
- RPAD
- MAX
- MIN
- SUBSTR

▶ To specify an argument that uses a comma as a decimal separator

1 Use a hidden, read-only field, and assign a default value to it.

2 Use the appropriate field keywords as arguments.

For example, instead of using `SUBSTR($Character Field$, 56, 65)` use `SUBSTR($Character Field$, $Integer1$, $Integer2$)` where `Integer1` and `Integer2` are hidden, read-only fields with default values of 56 and 65 assigned to them, respectively.

Table A-5 on page 229 lists the functions you can use in a Set Fields action.

Assigning values from process results

You can set a field to the `stdout` value that results from running a specified process. The process used to set the field can be located on one of the following machines:

- The client system (active links only)
- The server system on which an AR System server has been installed

WARNING

If the process runs on the server, it uses the permissions of the user who started the AR System server. If the process runs on the client, it uses the permissions of the user who started BMC Remedy User. This can have security implications for your system.

The syntax identifies where the process that you want to run is located.

For *active links*, you can run a process in the following ways:

- **On the client computer**—To access a process located on the same machine as the client, use the following syntax:

  `$PROCESS$ processToRun`

- **On the current AR System server**—To run a process on the current AR System server and set a field in a form that resides on that server, use the following syntax:

  `$PROCESS$ @@:processToRun`

- **On any specific AR System server**—To run a process located on specific AR System server and set a field located in a form that resides on the current AR System server, use the following syntax:

  `$PROCESS$ @ARSserver:processToRun`

  where *ARSserver* is the name of a specific AR System server where the process runs.

For *filters* or *escalations*, the syntax for loading the return of a process is as follows:

`$PROCESS$ processToRun`

The `$PROCESS$` tag indicates that all text that follows is a command line. The command line can include substitution parameters from the current screen to enable values to be placed into the command line before it is executed. The command cannot exceed 4096 bytes after the substitution parameters are expanded. The actual maximum length is limited by the operating system in use with AR System. Select substitution parameters (and the `$PROCESS$` string) from the Value list.

For a list of available `$PROCESS$` commands, see “Process commands” on page 261.

When the action is executed:

1 The specified command line is executed.

2 The calling program waits for the process to be completed.

3 The program reads and processes all data returned to `stdout` according to the exit status code that the process returns.

4 If the process returns an exit status code of 0, the returned data is used as the value for the field.

The data is expected in text format and is converted, as needed, to match the data type of the target field. If the process returns a code other than 0, it is assumed that there was an error and the process failed. In this case, the returned value is treated as the text of an error message and is returned to the user.

If the process is located on a server, activity for that server thread is blocked until the process is completed or the process interval is exceeded. If the process has timed out, the server stops its blocking action but does not stop the process. However, the server ignores any process response after the time-out. You can use the Timeouts tab of the AR System Administration: Server Information form to configure the process interval so that the server continues processing other tasks, even if the requested process response has not been received. For more information, see the timeouts and configuration information in the *Configuration Guide*, "Configuring AR System servers," page 122.

For active links, when you design an active link that loads field values from a process that is run on the client, be aware of the hardware platforms and operating systems that your clients might be using. The process that you are specifying might not be available on all platforms and operating systems. If your users run the client tools on more than one type of platform or operating system, you can build a qualification for the active link by using the `$HARDWARE$` and `$OS$` keywords to verify that the client is running on an appropriate platform and operating system at the time the active link executes. See Chapter 3, "Building qualifications and expressions," for more information.

When assigning values from process results, remember the following tips:

- Adjust your command syntax appropriately for the platform on which your server is running and include the explicit path to the command; for example, `/home/jim/bin/`*`command`*. In a Windows environment, you also must specify the drive; for example, `d:\home\jim\bin\`*`command.bat`*.
- On a Windows server, you can only run a process that runs in a console (such as a `.bat` script or `runmacro.exe`).
- In a UNIX environment, the process runs under a Bourne shell.
- Use double quotation marks around substituted fields when the values might contain spaces or other special characters; for example, `/bin/cmd "$`*`field`*`$"`.
- Substituted field values that contain hard returns or other special characters can have unexpected results.

Assigning values by using arithmetic operations

You can use arithmetic operations, such as "+" or "-" to compute a value that you can use in a Set Fields action. The same arithmetic operations allowed for specifying qualifications are used to build a computed value (see "Operators" on page 216). The operation must meet all of the rules for arithmetic operations and produce a result with a type that is compatible with the target field.

NOTE

If you include a process result or DDE result (active links only) in a mathematical operation, the process definition must *not* be contained within parentheses and it must be the last item in the operation because all data after the tag `$PROCESS$` or `$DDE$` is considered to be part of the command line or DDE definition.

In some cases, arithmetic operators can also be used for concatenation. Some examples of valid arithmetic operators are:

```
$TIMESTAMP$ - $CREATE-DATE$
$FIRST NAME$ + " " + $LAST NAME$
"hostname = " + $PROCESS$ hostname
```

Appendix

C Using Run Process and $PROCESS$ commands

Through the Run Process action or by using the `$PROCESS$` keyword in a Set Fields action, you can initiate separate processes, including both AR System commands and outside applications. In general, these actions are used to perform operations in support of your application.

This section describes the AR System commands designed to be run by a Run Process action or a `$PROCESS$` command and the syntax for using them. For more information see the *Integration Guide*, "Running external processes (Run Process)," page 259.

The following topics are provided:

Overview

AR System provides a set of commands that you can use with Run Process actions and the `$PROCESS$` keyword in Set Fields actions. These process commands do not cause the system to run an external operating system process. Instead, the AR System server or client recognizes these commands and performs the operations directly.

"Application" type process commands always run on the AR System server. Other process commands can run on the client or the AR System server as appropriate to the action.

When to use a Run Process action or a Set Fields action with $PROCESS$

Run Process is a workflow action that you can add to active links, filters, or escalations. `$PROCESS$` is a keyword available only in Set Fields actions. To determine whether you need to use a Run Process action or a Set Fields action with `$PROCESS$`, consider these differences:

- To capture a result from the process, you must use Set Fields with `$PROCESS$`. You can then use the result from the process in subsequent workflow.
- Asynchronous versus synchronous operation:
  - The Run Process action starts an asynchronous process. This means that the next workflow action is executed immediately, without waiting for results from the Run Process action. If the Run Process action fails or returns an error, the remaining workflow still runs.
  - A Set Fields action with `$PROCESS$` is synchronous. In other words, the workflow waits for the process to complete before initiating the next action.
- Filter phasing:
  - In a filter, the Run Process action is executed in phase 3.
  - In a filter, a Set Fields action with `$PROCESS$` runs in phase 1.

For information about filter phases, see "Filter processing in the AR System server" on page 177.

NOTE

If necessary, you can use a special filter naming convention to cause a Run Process action to run in filter phase 1. See "Overriding filter processing phasing" on page 184.

For more information about using the results from a Set Fields action with `$PROCESS$`, see "Assigning values from process results" on page 253.

Types of process commands

There are two sets of process commands in AR System. Commands named with "`Application-`" are always executed on the AR System server. These are described in Table C-2 on page 261.

Commands named with "`PERFORM-ACTION-`", as well as `GET-CHANGE-FLAG`, `SET-CHANGE-FLAG`, and `SET-RO-COLOR` are always executed by the workflow engine. If initiated from an active link, these commands run on the client. If initiated by a filter or escalation, they run on the server. Some workflow process commands are specifically designed only for use in active links, some are designed for use in filters and escalations, and some can be used in any type of workflow. These process commands are described in Table C-3 on page 271.

Process command syntax

This section describes special syntax considerations for Application commands, case sensitivity and use of quotation marks, and syntax for certain commands that include a qualification.

Server syntax for Application commands in an active link

Because Application commands always run on the server, you must use the following syntax when executing an Application command from an active link:

- To run the command on the current server:

  `@@:`*`processCommand {parameters`*`]`

- To run the command on a different AR System server:

  `@`*`serverName`*`:`*`processCommmand [parameters`*`]`

To run the process from a Set Fields action with the `$PROCESS$` keyword, enter `$PROCESS$` before the command. This indicates that whatever follows is the process command. For example:

`$PROCESS$ @@:`*`processCommand {parameters`*`]`

For `PERFORM-ACTION` commands and other workflow commands, simply enter the command and its arguments. The workflow engine determines whether the command runs on the client or on the current server.

Case sensitivity and using quotation marks

Run Process and `$PROCESS$` commands are case-sensitive. If incorrect capitalization is used, an error can occur.

When you define a run process command, you must use quotation marks in some cases to indicate that the content inside the quotation marks makes up a single parameter:

- If a parameter includes a space or a special character, put quotation marks around the parameter.
- Surround any keywords with quotation marks in case the substituted value contains spaces or special characters.
- If a value contains one or more quotation marks, double the quotation marks and put quotation marks around the entire parameter.

The following table lists examples of these guidelines.

Table C-1: Using quotations in process commands

Value	Syntax
AR System	`"AR System"`
AR “System” User	`"AR ""System"" User"`
$SCHEMA$	`"$SCHEMA$"`

Syntax exception—Application commands with qualifications

Any arguments *before* the final argument follow the rules described in the previous section.

These rules do not apply for the *final* argument of Application commands where the final argument is a qualification or a substitution string. Commands that take this format include the following commands:

- `Application-Parse-`*`xxx`*
- `Application-Format-`*`xxx`*
- `Application-Query-Delete-Entry`

For the final argument, no formatting is enforced.

For example, to run the `Application-Query-Delete-Entry` command in a filter, using form `ABC DEF` with the qualification `'Field ID' = "Fred"`, use:

```
Application-Query-Delete-Entry “ABC DEF” '536870913' = “Fred”
```

The form name parameter is quoted to keep it as a single argument. However, for the *final* argument (the qualification `'536870913' = "Fred"`), there are no surrounding quotation marks for the full expression.

Process commands

When you create a Run Process action or a Set Fields action with `$PROCESS$`, BMC Remedy Developer Studio enables you to select any of the commands described in this section. You must understand how each command operates to determine whether it is appropriate to use in your workflow.

Table C-2 describes Application commands, and Table C-3 on page 271 describes the `PERFORM-ACTION` and related workflow commands. Along with the syntax and description for each command, the tables include the following information:

Column	Plus sign (+) indicates
AL only	Use the command only with active links.
Filter or Esc only	Use the command only with filters and escalations.
Returns a value	The command returns a value. To capture the retuned value, you must use the command in a Set Fields action with the `$PROCESS$` key word.

Table C-2 describes Application commands, which are always executed on the server. If you use one of these commands in an active link, you must use the `@@:`*`processCommand`* or `@`*`serverName`*`:`*`processCommand`* or syntax.

Table C-2: Application commands (Sheet 1 of 10)

AL only	Filter or Esc only	Command and description	Returns a value

		`Application-Bus-Time-Add "`*`startTime`*`" [ "`*`amount`*`"`
		`[ "`*`amountUnits`*`" [ "`*`holidayScheduleName`*`"`
		`[ "`*`workdayScheduleName`*`" ] ] ] ]`
		Returns a new time that is the requested offset into the future, taking availability and business hours and holidays into account.
		Offset is a value of 0 or greater than 0. The default is 1 hour.
		Offset unit values are:
		■ **1**—Seconds
		■ **2**—Minutes
		■ **3**—Hours
		■ **4**—Days
		For more information, see the *Configuration Guide*, "Using Business Time in the AR System server," page 215.
			+

		`Application-Bus-Time-Diff "`*`startTime`*`" "`*`endTime`*`"`
		`[ "`*`holidayScheduleName`*`" [ "`*`workdayScheduleName`*`" ] ]`
		Returns an integer that represents the number of seconds between the start and stop time, taking business hours into account.
		For more information, see the *Configuration Guide*, "Using Business Time in the AR System server," page 215.
			+

Table C-2: Application commands (Sheet 2 of 10)

AL only	Filter or Esc only	Command and description	Returns a value
		`Application-Bus-Time-Subtract "`*`startTime`*`" [ "`*`amount`*`" [ "`*`amountUnits`*`" [ "`*`holidayScheduleName`*`" [ "`*`workdayScheduleName`*`" ] ] ] ]` Returns a new time that is the requested offset into the past, taking business hours into account. Offset is a value of 0 or greater than 0. The default is 1 hour. Offset unit values are: ■ **1**—Seconds ■ **2**—Minutes ■ **3**—Hours ■ **4**—Days For more information, see the *Configuration Guide*, "Using Business Time in the AR System server," page 215.	+
		`Application-Bus-Time2-Add "`*`startTime`*`" [ "`*`amount`*`" [ "`*`amountUnits`*`" [ "`*`businessTimeSegmentName1`*`" "`*`businessTimeSegmentName2`*`" ... ] ] ]` Performs a business time calculation by starting with the start time and resulting in a new time that adds the requested offset. The command returns a timestamp representing the time calculated. Use this command to recalculate time into the future. For more information, see the *Configuration Guide*, "Using Business Time in the AR System server," page 215.	+
		`Application-Bus-Time2-Assoc-Add "`*`startTime`*`" [ "`*`amount`*`" [ "`*`amountUnits`*`" [ "`*`businessTimeSegmentName1`*`" "`*`businessTimeSegmentName2`*`" ... [ -e "EntityID1" "EntityID2" ... ] ] ] ]` This command contains EntityID parameters, so you do not need to query the Business Segment-Entity Association form. For more information, see the *Configuration Guide*, "Using Business Time in the AR System server," page 215.	+
		`Application-Bus-Time2-Assoc-Diff "`*`startTime`*`" "`*`endTime`*`" [ "`*`businessTimeSegmentName1`*`" "`*`businessTimeSegmentName2`*`" ... [-e "EntityID1" "EntityID2" ... ] ]` This command contains EntityID parameters, so you do not need to query the Business Segment-Entity Association form. For more information, see the *Configuration Guide*, "Using Business Time in the AR System server," page 215.	+
		`Application-Bus-Time2-Assoc-Get-Free-Window "`*`startTimeRange`*`" "`*`endTimeRange`*`" "`*`level`*`" "`*`duration`*`" "`*`earliestStartTime`*`" "`*`latestEndTime`*`" [ "`*`businessTimeSegmentName1`*`" "`*`businessTimeSegmentName2`*`" ... [ -e "EntityID1" "EntityID2" ... ] ]` This command contains EntityID parameters, so you do not need to query the Business Segment-Entity Association form. For more information, see the *Configuration Guide*, "Using Business Time in the AR System server," page 215.	+

Table C-2: Application commands (Sheet 3 of 10)

AL only	Filter or Esc only	Command and description	Returns a value
		`Application-Bus-Time2-Assoc-Get-Next-Window "`*`startTimeRange`*`" "`*`endTimeRange`*`" "`*`duration`*`" "`*`windowFlag`*`" [ "`*`businessTimeSegmentName1`*`" "`*`businessTimeSegmentName2`*`" ... [ -e "EntityID1" "EntityID2" ... ] ]` This command contains EntityID parameters, so you do not need to query the Business Segment-Entity Association form. For more information, see the *Configuration Guide*, "Using Business Time in the AR System server," page 215.	+
		`Application-Bus-Time2-Assoc-Subtract "`*`startTime`*`" [ "`*`amount`*`" [ "`*`amountUnits`*`" [ "`*`businessTimeSegmentName1`*`" "`*`businessTimeSegmentName2`*`" ... [ -e "EntityID1" "EntityID2" ... ] ] ] ]` This command contains EntityID parameters, so you do not need to query the Business Segment-Entity Association form. For more information, see the *Configuration Guide*, "Using Business Time in the AR System server," page 215.	+
		`Application-Bus-Time2-Diff "`*`startTime`*`" "`*`endTime`*`" [ "`*`businessTimeSegmentName1`*`" "`*`businessTimeSegmentName2`*`" ... ]` Performs a business time calculation by computing the difference between the start time and the end time. The return is an integer representing the difference in seconds. Use this command to compare two different times (start time and end time) to get the actual business time. For more information, see the *Configuration Guide*, "Using Business Time in the AR System server," page 215.	+
		`Application-Bus-Time2-Get-Free-Window "`*`startTimeRange`*`" "`*`endTimeRange`*`" [ "`*`level`*`" ] [ "`*`duration`*`" ] [ "`*`earliestStartTime`*`" ] [ "`*`latestEndTime`*`" ] [ "`*`businessTimeSegmentName1`*`" "`*`businessTimeSegmentName2`*`" ... ]` Returns the start of the next available or unavailable free time segment at the same level or a higher level that is *`duration`* seconds long. For more information, see the *Configuration Guide*, "Using Business Time in the AR System server," page 215.	+
		`Application-Bus-Time2-Get-Next-Window "`*`startTimeRange`*`" "`*`endTimeRange`*`" [ "`*`duration`*`" ] [ "`*`windowFlag`*`" ] [ "`*`businessTimeSegmentName1`*`" "`*`businessTimeSegmentName2`*`" ... ]` Returns the start of the next available or unavailable time segment that is *`duration`* seconds long. If *`duration`* is 0 (the default), the command returns either the start of available time segment or the start of the unavailable time segment. For more information, see the *Configuration Guide*, "Using Business Time in the AR System server," page 215.	+

Table C-2: Application commands (Sheet 4 of 10)

AL only	Filter or Esc only	Command and description	Returns a value
		`Application-Bus-Time2-Subtract "`*`startTime`*`" [ "`*`amount`*`" [ "`*`amountUnits`*`" [ "`*`businessTimeSegmentName1`*`" "`*`businessTimeSegmentName2`*`" ... ] ] ]` Performs a business time calculation by starting with the start time and resulting in a new time that subtracts the requested offset. The command returns a timestamp representing the time calculated. Use this command to recalculate time in the past. For more information, see the *Configuration Guide*, "Using Business Time in the AR System server," page 215.	+
		`Application-Command AE-ASSIGN DoAssign -t "`*`processName`*`" -e "`*`RequestID`*`"` Runs a specified Assignment Engine process. *`processName`* is the name of the process to run. It must match the process name in the Processes tab of the Assignment Engine Administration Console. *`RequestID`* is the ID of the request on which the process will run. For active links, the request ID is required. For filters, it is optional. For information about the Assignment Engine, see the *Configuration Guide*, "Using the Assignment Engine," page 265.	
		`Application-Confirm-Group` *`groupID`* Verifies that the current user is a member of the specified group. Returns one of the following integers: ■ **1**—The user is a member of the group. ■ **0**—The user is *not* a member of the group, or the specified group ID does not correspond to a valid group. This command is not context sensitive for a given entry. Validation of group IDs 0, 3, 4, or 7 returns 1.	+
		`Application-Confirm-Password` *`password`* Validates if the password is the password for the current user. For *`password`*, you can use a reference to the field that contains the password, such as field 102 or field 103. This command returns one of the following integers: ■ **1**—The password was confirmed. ■ **0**—The password is not valid. If you used AR System version prior to 6.0 to create workflow involving a Password field (ID 102), the workflow might not function in AR System versions 6.0 and later. Version 6.0 included enhanced encryption and tighter security controls. To work around this issue, use the `Application-Confirm-Password $PROCESS$` command. For more information about the Password field, see the *Form and Application Objects Guide*, "Reserved fields in access control," page 480, and the *Configuration Guide*, "Adding and modifying user information," page 57.	+

Table C-2: Application commands (Sheet 5 of 10)

AL only	Filter or Esc only	Command and description	Returns a value
		`Application-Copy-Field-Value` *`targetFieldID sourceFieldID`* Copies a field on the current form to another field on the current form. Returns one of the following integers: ■ **1**—The assignment failed. ■ **0**—The assignment occurred. To get the return value, use this command in a Set Fields action with `$PROCESS$`. This command cannot be used in a Run Process action.	+

		`Application-Delete-Entry "`*`formName`*`"` *`entryID`*
		Deletes the specified entry.
		For example, to delete the entry on the current form with the entry ID found in the core field 1 (Request ID), enter:
		`Application-Delete-Entry "$SCHEMA$" $1$`
			+

	+	`Application-Event` *`eventNumber eventDetail`* Initiates a server event. These valid values for *`eventNumber`* cause the AR System server to perform the following actions: ■ **1**—Read the configuration file into memory. ■ **2**—Read the group information and definitions from database into memory. ■ **3**—Re-scan the Add or Remove License form. For more information about server events, see the *Configuration Guide*, "Working with the Server Events form," page 197.	
	+	`Application-Format-Qual "`*`form`*`"` *`internalQualifier`* Converts an internal representation of a qualifier into a qualification string. See "Syntax exception—Application commands with qualifications" on page 260.	+
	+	`Application-Format-Qual-Filter "`*`form`*`"` *`internalQualifier`* Converts an internal representation of a qualifier from a filter Run If into a qualification string. See "Syntax exception—Application commands with qualifications" on page 260.	+
	+	`Application-Format-Qual-ID "`*`form`*`"` *`internalQualifier`* Converts an internal representation of a qualifier into a qualification string using ID format. See "Syntax exception—Application commands with qualifications" on page 260.	+
	+	`Application-Format-Qual-L "`*`form`*`" "`*`VUILabel`*`"` *`internalQualifier`* For the indicated VUI, converts an internal representation of a qualifier into a qualification string using labels. See "Syntax exception—Application commands with qualifications" on page 260.	+
	+	`Application-Format-Qual-SField "`*`form1`*`" "`*`form2`*`"` *`internalQualifier`* Converts an internal representation of a qualifier from a Set Fields or Push Fields filter action into a qualification string. See "Syntax exception—Application commands with qualifications" on page 260.	+

Table C-2: Application commands (Sheet 6 of 10)

AL only	Filter or Esc only	Command and description	Returns a value
	+	`Application-Format-Val-SField "`*`form1`*`" `*`fieldID`*` "`*`form2`*`" `*`internalAssignment`* Converts an internal representation of a Set Fields or Push Fields assignment into an assignment statement. See "Syntax exception—Application commands with qualifications" on page 260.	+

	+	`Application-FTS-Reindex-Form "form" [ "scanTime" [ fieldId 1 ] . . . [ fieldId N ]  ]`
		Initiates a full text reindex on the specified form. For example:
		`Application-FTS-Reindex-Form "HPD:Help Desk" "9/28/2010 5:06:52 PM" 10000100 10000200`
		`"scanTime"`
		Indicates the date and time from which the server should scan for updates. For example:
		`"9/28/2010 5:06:52 PM"`
		If the scan time is omitted, the form is completely reindexed.
		The value specified is only used for regular forms. The server treats join, vendor, and view form requests as on-demand scans. These form types record a last scan time value that is used in on-demand scans and scheduled scans. Zero can be used for the scan time to differentiate between a request for an on-demand scan, and a complete form reindex. Because regular forms do not have scheduled scans, the specified value is used.
		Note: The standard format is a character string date, but the server also accepts a numeric timestamp value.
		`[fieldId [1..N]`
		Indicates the fields with data changes by including a range of field IDs. For example:
		`10000100 10000200`
		If no field IDs are specified, the server will not perform analysis to determine if a reindex is needed before proceeding with the reindex. If the scan time is omitted, no field IDs can be specified.
		This advanced option is used when the AR System database has been updated externally, and it is not known if any of the updated columns are associated with full text indexed fields. This option allows the server to determine if any full text indexed fields were affected by the database updates and, if there were none, unnecessary indexing is avoided.	

		`Application-Generate-GUID [ "`*`GUIDPrefix`*`" ]` Generates a globally unique identifier (GUID). The prefix can be a maximum of two characters, which can contain non-alpha characters (although alpha characters are recommended). If you do not include the GUID prefix, it defaults to `ID`.	+

	+	`Application-Get-Approval-Join "`*`form`*`"`
		Retrieves the name of the join form between the application and the AP:Detail form. In the result, the form names are separated by spaces, for example:
		`AP-Sample:Lunch-Detail   AP:Detail`
		This command is used for the Approval Server.
			+

Table C-2: Application commands (Sheet 7 of 10)

AL only	Filter or Esc only	Command and description	Returns a value

	+	`Application-Get-Approval-Join2 "`*`form`*`"`
		Retrieves the name of the join form between the application and the AP:Detail form. In the result, the form names are separated by new lines, for example:
		`AP-Sample:Lunch-Detail`
		`AP:Detail`
		This command is used for the Approval Server.
			+

	+	`Application-Get-DetSig-Join "`*`form`*`"`
		Retrieves the name of the join form between the three-way join form (join between the application form and AP:Detail-Signature) and the names of the AP:Detail-Signature form, AP:Detail form, and AP:Signature form. In the result, the form names are separated by spaces, for example:
		`AP-Sample:Lunch-Detail-Signature   AP:Detail-Signature`
		`AP:Detail   AP:Signature`
		This command is used for the Approval Server.
			+

	+	`Application-Get-DetSig-Join2 "`*`form`*`"`
		Retrieves the name of the join form between the three-way join form (join between the application form and AP:Detail-Signature) and the names of the AP:Detail-Signature form, AP:Detail form, and AP:Signature form. In the result, the form names are separated by new lines, for example:
		`AP-Sample:Lunch-Detail-Signature`
		`AP:Detail-Signature`
		`AP:Detail`
		`AP:Signature`
		This command is used for the Approval Server.
			+

	+	`Application-Get-Form-Alias "`*`form`*`" [ "`*`VUILabel`*`" ]` Retrieves the appropriate form alias for the specified form and VUI. If you do not include a VUI, the default VUI is used.	+
	+	`Application-Get-Form-Name "`*`formAlias`*`" [ "`*`VUILabel`*`" ]` Retrieves the form name for the specified form alias and VUI. If you do not include a VUI, the default VUI is used.	+
	+	`Application-Get-License-Count "`*`licenseName`*`"` Retrieves the number of licenses of the specified type. Use the license name that is used in the License Tool.	+
	+	`Application-Get-Locale-VuiID "`*`form`*`" "`*`VUIType`*`" "`*`localeName`*`"` Retrieves the VUI ID for the locale that you specify. The VUI types are Windows (1) and Web absolute positioning (3). Do not use the `$VUI-TYPE$` keyword.	+
		`Application-Get-Next-Recurrence-Time "`*`formName`*`" "`*`startTime`*`" "`*`recurrenceDefinitionName`*`"` Returns a timestamp representing the recurrence time. For more information, see the *Configuration Guide*, "Using Business Time in the AR System server," page 215.	+

Table C-2: Application commands (Sheet 8 of 10)

AL only	Filter or Esc only	Command and description	Returns a value
		`Application-Invalidate-User` *`userName`* Writes an invalid string for the password value to disable the current user's account. The *`userName`* parameter can be hard-coded, the keyword `$USER$`, or referenced from a field. For syntax information, see "Process command syntax" on page 259.	
	+	`Application-Map-Ids-To-Names "`*`form`*`"` *`qualificationString`* Maps the IDs of the fields or keywords in the string to a name representation.	+
	+	`Application-Map-Ids-To-Names-L "`*`form`*`" "`*`VUILabel`*`"` *`qualificationString`* For the indicated VUI, maps the IDs of the fields or keywords in the string to a name representation using labels. If the field label is blank, the database name is used.	+
	+	`Application-Map-Names-To-Ids "`*`form`*`"` *`qualificationString`* Maps the names of the fields or keywords in the string to an internal ID representation.	+
	+	`Application-Map-Names-To-Ids-L "`*`form`*`" "`*`VUILabel`*`"` *`QualificationString`* For the indicated VUI, maps the labels of the fields or keywords in the string to an internal ID representation. For example, to map the IDs in the given string to names using labels where appropriate, enter: `Application-Map-Ids-To-Names-L "My Form" "" $536870913$` An empty string for the VUI denotes the default VUI for the form.	+
	+	`Application-Parse-Qual "`*`form`*`"` *`qualificationString`* Converts a qualification string into an internal representation. For example, to parse the qualification string into its internal representation, enter: `Application-Parse-Qual "My Form" Integer Field = 99` The qualification string does not need double quotation marks around it because all data after the form name is treated as the qualification string.	+
	+	`Application-Parse-Qual-Filter "`*`form`*`"` *`qualificationString`* Converts a Run If filter qualification string into an internal representation. See "Syntax exception—Application commands with qualifications" on page 260.	+
	+	`Application-Parse-Qual-L "`*`form`*`" "`*`VUILabel`*`"` *`qualificationString`* For the indicated VUI, converts a qualification string with labels into an internal representation. The result of this command is used in the `Application-Format-Qual-L` command. See "Syntax exception—Application commands with qualifications" on page 260.	+
	+	`Application-Parse-Qual-SField "`*`form1`*`" "`*`form2`*`"` *`qualificationString`* Converts a Set Fields or Push Fields filter qualification string into an internal representation. See "Syntax exception—Application commands with qualifications" on page 260.	+

Table C-2: Application commands (Sheet 9 of 10)

AL only	Filter or Esc only	Command and description	Returns a value
	+	`Application-Parse-Qual-SField-L "`*`form1`*`" "`*`form2`*`" "`*`VUILabel1`*`" "`*`VUILabel2`*`"` *`qualificationString`* For the indicated VUI, converts a Set Fields or Push Fields filter qualification string into an internal representation. See "Syntax exception—Application commands with qualifications" on page 260.	+
	+	`Application-Parse-Val-SField "`*`form1`*`"` *`fieldID`* `"`*`form2`*`"` *`assignmentStatement`* Converts a Set Fields or Push Fields filter assignment statement into an internal representation. The result of this command is used in the `Application-Format-Val-SField` command. See "Syntax exception—Application commands with qualifications" on page 260.	+
	+	`Application-Query-Delete-Entry "`*`fieldID`*`"` *`qualificationString`* Deletes all entries matching the specified qualification. See "Syntax exception—Application commands with qualifications" on page 260.	+

Table C-2: Application commands (Sheet 10 of 10)

AL only	Filter or Esc only	Command and description	Returns a value
	+	`Application-Release-Pending` Causes database operations generated by the current workflow to be sent to the database immediately. In a filter, this command changes the usual filter phasing and causes the Run Process action to run in phase 1. Note: Use this advanced feature with caution. The command allows workflow to see the results of previous workflow by causing the previous workflow's actions to be entered into the database. See "Releasing pending operations" on page 185 for a detailed explanation.	
	+	`Application-Set-Filter-Phasing "`*`value`*`"` Determines whether form entries are created when the workflow operation to create them occurs or whether they are created in bulk during a later filter phase. When issued, this command affects all *subsequent* entry create operations for the current API call. Entries already created as a result of the call are not undone. The effect of the command lasts for the duration of the API call or until the command is reissued with a different value. *`value`* can be `1` or `0`: ■ **1**—Entries are created in a later filter phase. ■ **0**—Entries are created immediately. This is the default behavior. Use this parameter only if you previously switched on delayed entry creation for the call and now want to switch it off. Important: If the create phase is delayed, the entries are not immediately added to the database, so their data is unavailable to subsequent workflow actions. This can change the effect of filters that use the data in qualifications or as a source for Set Fields actions. This command does not affect filters whose names end with `\!`. In such filters, all database operations including creates occur in phase 1. When a filter performs an `Application-Release-Pending` Run Process command, all delayed create operations are immediately performed. The new entries are held until the end of filter processing or until they are flushed by another `Application-Release-Pending` Run Process command.	

Table C-3 describes PERFORM-ACTION and other workflow commands, which are executed by the workflow engine and can run on the client or the server as appropriate.

Table C-3: PERFORM-ACTION and other workflow commands (Sheet 1 of 14)

AL only	Filter or Esc only	Command and description	Returns a value
+		`GET-CHANGE-FLAG` Gets the change flag status of the current window. 1 means that changes were made, and 0 means that no changes were made.	+
+		`PERFORM-ACTION-ACTIVE-LINK` Executes all active links associated with the specified Execute On condition (and field ID, as needed). The active links fire as if the Execute On condition indicated occurred. For example, `PERFORM-ACTION-ACTIVE-LINK 8` specifies to run all On Modify active links as if a modify operation was performed. The active links fire, but no modify is actually performed. The options for this command are as follows: ■ **Button**: `PERFORM-ACTION-ACTIVE-LINK 1` *`fieldID`* ■ **Row Double Click or Return**: `PERFORM-ACTION-ACTIVE-LINK 2` *`fieldID`* ■ **Submit**: `PERFORM-ACTION-ACTIVE-LINK 4` ■ **Modify**: `PERFORM-ACTION-ACTIVE-LINK 8` ■ **Display**: `PERFORM-ACTION-ACTIVE-LINK 16` ■ **Menu Choice**: `PERFORM-ACTION-ACTIVE-LINK 128` *`fieldID`* ■ **Lose Focus**: `PERFORM-ACTION-ACTIVE-LINK 256` *`fieldID`* ■ **Set Default**: `PERFORM-ACTION-ACTIVE-LINK 512` ■ **Search**: `PERFORM-ACTION-ACTIVE-LINK 1024` ■ **After Modify**: `PERFORM-ACTION-ACTIVE-LINK 2048` ■ **After Submit**: `PERFORM-ACTION-ACTIVE-LINK 4096` ■ **Gain Focus**: `PERFORM-ACTION-ACTIVE-LINK 8192` *`fieldID`* ■ **Window Open**: `PERFORM-ACTION-ACTIVE-LINK 16384` ■ **Un-Display**: `PERFORM-ACTION-ACTIVE-LINK 65536` ■ **Window Close**: `PERFORM-ACTION-ACTIVE-LINK 32768` ■ **Copy To New**: `PERFORM-ACTION-ACTIVE-LINK 131072` ■ **Window Loaded**: `PERFORM-ACTION-ACTIVE-LINK 262144` ■ **Interval:** `PERFORM-ACTION-ACTIVE-LINK 524288` ■ **Event:** `PERFORM-ACTION-ACTIVE-LINK 1048576` ■ **Table content change:** `PERFORM-ACTION-ACTIVE-LINK 2097152` *`fieldID`* ■ **Hover on Label:** `PERFORM-ACTION-ACTIVE-LINK 4194304` *`fieldID`* ■ **Hover on Data:** `PERFORM-ACTION-ACTIVE-LINK 8388608` *`fieldID`* ■ **Hover on Field:** `PERFORM-ACTION-ACTIVE-LINK 16777216` *`fieldID`* ■ **Expand:** `PERFORM-ACTION-ACTIVE-LINK 33554432` *`fieldID`* ■ **Collapse:** `PERFORM-ACTION-ACTIVE-LINK 67108864` *`fieldID`*	

Table C-3: PERFORM-ACTION and other workflow commands (Sheet 2 of 14)

AL only	Filter or Esc only	Command and description	Returns a value
		`PERFORM-ACTION-ADD-ATTACHMENT` *`fieldID`* `[ "`*`fileName`*`" ]` Adds an attachment to an attachment field, and returns a value of 0 (Successful). If the attachment is not added, the command returns one of the following codes: ■ 1: Canceled ■ 2: Failed For *filters* and *escalations*, the field ID must be an attachment field, and the file name is required. For *active links*: ■ The file name is optional. If omitted, a Browse dialog box is displayed to allow you to select a file name. (If viewed in a web browser, a Browse dialog box is always displayed.) ■ The field ID can be an attachment field or an attachment pool. ■ If the field ID is an attachment field, the attachment is added to the specified field. If the field has a value, the existing value is overwritten. ■ If the field ID is an attachment pool, the attachment is added to the first available field within the pool that has no attachment. If no attachment meets this criteria within the pool, no action is taken. Note: If you use this command in a filter with a Run Process action instead of in a Set Fields action, you must use the filter phase override naming convention *`filterName`*`` `! ``. This causes the action to run in filter phase 1 so that the changes are committed to the database. See "Using a special override naming convention" on page 185.	+
+		`PERFORM-ACTION-APPLY` If the form is open in Search mode, performs the Search operation (clicks the Search button). In Modify or New mode, performs the Apply or Save operation. Note: The only difference between the `PERFORM-ACTION-APPLY` process command and the `Commit Changes` action is that the `Commit Changes` action works differently in conjunction with a dialog box. The `PERFORM-ACTION-APPLY` process command has no effect on a dialog box, it only works with regular forms.	
+		`PERFORM-ACTION-CHANGE-MODE` *`mode`* Changes the mode of the form.	
+		`PERFORM-ACTION-CLEAR-PROMPTBAR` Clears the prompt bar of all messages. This command is useful to run before custom validation occurs (through `PERFORM-ACTION-VALIDATE_NULL-REQUIRED-FIELDS`) if the prompt bar will not be cleared automatically.	

Table C-3: PERFORM-ACTION and other workflow commands (Sheet 3 of 14)

AL only	Filter or Esc only	Command and description	Returns a value
		`PERFORM-ACTION-DELETE-ATTACHMENT` *`fieldID`* Deletes an attachment from an attachment field, and returns a value of 0 (Successful). If the attachment is not deleted, the command returns one of the following codes: ■ 1: Canceled ■ 2: Failed Note: If you use this command in a filter with a Run Process action instead of in a Set Fields action, you must use the filter phase override naming convention *`filterName`*`` ` ``!. This causes the action to run in filter phase 1 so that the changes are committed to the database. See "Using a special override naming convention" on page 185 .	+
+		`PERFORM-ACTION-EXIT-APP` Exits the Windows client or logs out of the web client.	
+		`PERFORM-ACTION-GET-ENTRY` *`"entryID"`* Retrieves the entry based on the entry ID. This command is applicable only for the Display and Modify modes. For information about the related features, see "Ability to modify data on display forms" on page 285.	
+		`PERFORM-ACTION-GET-FIELD-LABEL` *`fieldID`* Returns a field label. Use this command in a Set Fields action with `$PROCESS$` to obtain the return value.	+
+		`PERFORM-ACTION-GET-PREFERENCE` *`fieldID`* Gets the value of the field you specify from the User Preference form. If the field contains a list of values, then it gets the value for *`preferenceName`*. For example, to get the value of the User Locale field, enter the following command: `PERFORM-ACTION-GET-PREFERENCE 20121` where `20121` is the field ID of the User Locale field. To find the field ID: 1 Open the AR System User Preference form in BMC Remedy Developer Studio. 2 Select the field in question, and find the ID property in the Properties tab.	+
+		`PERFORM-ACTION-GO-HOME` Opens the form configured as your home page.	
+		`PERFORM-ACTION-SHOW-TOOLBAR [Value]` Shows or hides the toolbar on a form. If the value to set to 1, the form displays the toolbar. Otherwise, the form hides the toolbar. This command is available for all modes, except Dialog and Popup. For information about the related features, see "Ability to modify toolbar option" on page 285.	

Table C-3: PERFORM-ACTION and other workflow commands (Sheet 4 of 14)

AL only	Filter or Esc only	Command and description	Returns a value
+		`PERFORM-ACTION-HOME-FIELD-REFRESH` Refreshes the Application List field on the home page form. Typically, you use this command to display a subset of entry points based on the values that are dynamically entered into the reserved character field ID 1576.	
+		`PERFORM-ACTION-LIST-APPEND` *`fieldID “value”`* Adds the value to the end of the list maintained by a character field. For example, if character field `1234` contains the value `abc;def;ghi;jkl`, calling `PERFORM-ACTION-LIST-APPEND 1234 "mno"` modifies the value to `abc;def;ghi;jlm;mno`.	
+		`PERFORM-ACTION-LIST-INSERT` *`fieldID position “value”`* Adds the value to the list maintained by a character field, at the specified position. For example, if character field `1234` contains the value `abc;def;ghi;jkl`, calling `PERFORM-ACTION-LIST-APPEND 1234 2 "mno"` modifies the value to `abc;mno;def;ghi;jlm`	
		`PERFORM-ACTION-MAP-GROUPIDS-TO-NAMES` *`fieldID`* Returns a list of group names instead of group IDs. A group list field now always returns a list of group IDs and not group names. To use a group list with a LIKE statement, which requires a text string, first use this command to convert the group IDs to group names. For example, to use a statement such as `$grouplist$ LIKE "%"+'groupname'+"%"`, first add a process command to translate a list of group IDs to a space separated list of group names. For example: `$PROCESS$ PERFORM-ACTION-MAP-GROUPIDS-TO-NAMES $536871123$` Note: This command is only available on AR System 7.6.04 or later servers.	+
+		`PERFORM-ACTION-MAP-PUT` *`fieldID key “value”`* Adds or updates the value in a list or map maintained by a character field. For example, if character field `1234` contains the value `type=4;Name=John`, calling `PERFORM-ACTION-MAP-PUT 1234 Name "George"` modifies the value to `type=4;Name=George`.	
+		`PERFORM-ACTION-NAV-FIELD-SET-SELECTED-ITEM` *`navbarItemID`* Set focus to the specified navigation bar item.	
+		`PERFORM-ACTION-NEXT` Moves to the next request in the Results pane and displays the details in the Details pane.	
+		`PERFORM-ACTION-OPEN-ATTACHMENT` *`fieldID`* Opens an attachment from an attachment field, and returns a value of 0 (Successful). If the attachment is not opened, the command returns one of the following codes: ■ 1: Canceled ■ 2: Failed	+

Table C-3: PERFORM-ACTION and other workflow commands (Sheet 5 of 14)

AL only	Filter or Esc only	Command and description	Returns a value

+		`PERFORM-ACTION-OPEN-URL [ current | new ] "`*`urlString`*`"`
		Opens the specified URL in a browser. For the Windows client, a browser is opened. In a browser, the URL is opened as follows:
		To open the page in a *new* browser window, enter:
		`PERFORM-ACTION-OPEN-URL "`*`urlString`*`"`
		or
		`PERFORM-ACTION-OPEN-URL new "`*`urlString`*`"`
		To open the page in a new browser window with the toolbar, menu bar, and location bar hidden, enter
		`PERFORM-ACTION-OPEN-URL newwithouttoolbar “`*`urlString`*`”`
		To open the page in the *current* browser window, enter:
		`PERFORM-ACTION-OPEN-URL current "`*`urlString`*`"`
		Be sure that the URL is complete and well-formed so that the browser handles it correctly. For example, to open a web page, be sure the URL begins with `http://`. Some partial URLs work correctly when this command is run from BMC Remedy User, but the mid tier always requires a complete and well-formed URL.	

+		`PERFORM-ACTION-PREV` Moves to the previous request in the Results pane and displays the details in the Details pane.	
+		`PERFORM-ACTION-REFRESH-PREFERENCE` *`flag`* Refreshes the preferences for BMC Remedy User. The flag options are: ■ **1**—Saves all modified preferences from BMC Remedy User to the preference server, and then reloads from the preference server. ■ **0**—Discards all modified preferences in BMC Remedy User and reloads from the preference server.	

Table C-3: PERFORM-ACTION and other workflow commands (Sheet 6 of 14)

AL only	Filter or Esc only	Command and description	Returns a value
		`PERFORM-ACTION-SAVE-ATTACHMENT` *`fieldID`* `[ "`*`fileName`*`" ]` Saves an attachment from an attachment field to a file, and returns a value of 0 (Successful). If the attachment is not saved, the command returns one of the following codes: ■ **1**—Canceled ■ **2**—Failed For *filters* and *escalations*, the file is saved on the server machine. The field ID must be an attachment field, and the file name is required. For *active links*: ■ The file is saved on the client machine. ■ The field ID can be an attachment field or an attachment pool. If the field ID is an attachment pool, the first available attachment is saved. ■ The file name is optional. If omitted, BMC Remedy User displays a Save Attachment dialog box using the attachment name for the file name. The web client always displays a dialog box. If the file name is given, the simple file name is displayed in the file name field. Note: If you use this command in a filter with a Run Process action instead of in a Set Fields action, you must use the filter phase override naming convention *`filterName`*`` `! ``. This causes the action to run in filter phase 1. See "Using a special override naming convention" on page 185 .	+
+		`PERFORM-ACTION-SEND-EVENT "`*`target`*`" "`*`eventType`*`" "`*`eventData`*`"` "*`FdataVisualizationModuleFieldID`*" Sends an event to another window. *`target`* is the window to which to send the event. Possible values are: ■ **@**—The "at" sign signifies the parent of the current window. Note: In the web client, an Open Window active link action can open a form in the current window. This window does not have a parent window, so a *`target`* value of **@** is not valid in this case. ■ **#**—The pound sign character signifies all child windows of the current window. ■ *****—The asterisk character signifies all windows managed by the client environment, even the current window. ■ *`eventType`*—The name of the event. This is an arbitrary string defined by the application author (for example, `ChildClosed`). The receiving workflow can access the value of *`eventType`* through the `EVENTTYPE` keyword. ■ *`eventData`*—The data for the event. ■ *`FdataVisualizationModuleFieldID`*—The target for sending events is the module field in the current form. For more information, see "Sending events between windows" on page 295 and "Ability to highlight required fields through workflow" on page 286.	

Table C-3: PERFORM-ACTION and other workflow commands (Sheet 7 of 14)

AL only	Filter or Esc only	Command and description	Returns a value

+		`PERFORM-ACTION-SET-PREFERENCE` *`fieldID`* `"`*`value`*`"`
		Sets the value of the field you specify for the current session only. The preference is not set in the database. For example, to set the value of the User Locale field to Japanese, enter the following command:
		`PERFORM-ACTION-SET-PREFERENCE 20121 "ja_JP"`
		where `20121` is the field ID of the User Locale field and `ja_JP` is the value that represents a Japanese locale.
		
		To find the field ID:
		1 Open the AR System User Preference form in BMC Remedy Developer Studio.
		2 Select the field in question, and find the ID property in the Properties tab.
		
		To find the value or format of a user preference field:
		1 Log in to a preference server using BMC Remedy User.
		2 Choose Tools > Options, and specify a value for a user preference.
		For example, select a Display Locale in the Locale tab.
		3 Open the AR System User Preference form to see what format is used or what value is stored in the corresponding field (for example the User Locale field in the Locale tab).
		This is the format or value you specify for the `value` argument.	

Table C-3: PERFORM-ACTION and other workflow commands (Sheet 8 of 14)

AL only	Filter or Esc only	Command and description	Returns a value
+		`PERFORM-ACTION-TABLE-ADD-ROW` *`tableFieldID rowIndex`* Inserts an empty table row at the specified position on-screen. ("Row" refers to a *row* in list view tables, a *node* in tree view tables, and a *cell* in cell-based tables). This command does not commit the new row to the database. To add data to the empty row, you must use the appropriate active link. This command takes the following arguments: ■ ***tableFieldID***—The ID of the table field to which to add the row. This value is required. ■ ***rowIndex*** (Optional)—The position (row index) at which to insert the row. (Row indexes are based on the version of the table in the client memory data structure.) The specified index determines the actions that occur as follows: ■ **0**—Invalid row index. The current selection and highlight are not changed. ■ **Greater than 0 and less than 100,000,000**—A row is inserted at the specified index, selected, and highlighted. Workflow is then fired on the new row. ■ **Greater than 100,000,000**—A row is inserted at *rowIndex* - 100,000,000. For example, if the specified index is 100,000,005, the row is inserted at index 5. The new row is selected but not highlighted. Workflow is then fired on the new row. ■ **Less than 0 and greater than -100,000,000**—A row is inserted at the specified index multiplied by -1. For example, if the specified index is -500, a row is inserted at index 500. The new row is selected and highlighted. Workflow is then fired on the new row. ■ **Less than -100,000,000**—A row is inserted at *rowIndex* + 100,000,000. For example, if the specified index is -100,000,500, a row is inserted at index -500. The new row is selected, but it is not highlighted. Workflow is not fired. This value can come at runtime from a field on the form (for example, use `PERFORM-ACTION-TABLE-ADD-ROW $Field1$ $Field2$`). If a row already occupies the specified position, this command pushes the existing row and all the rows that follow it down one position. If the specified index is greater than the number of existing rows, the row is inserted at the end of the table. If a row index is *not* specified, a row is inserted at the end of the table, selected, and highlighted. Workflow is then fired on the new row. See the *Form and Application Objects Guide*, "Updating tables on-screen only," page 284.	
+		`PERFORM-ACTION-TABLE-CHANGE-ROW-COL-VISIBILITY` *`tableFieldID`* `[0 | 1]` *`columnFieldID`* Hides (1) or shows (0) the contents of a column field for the current row.	

Table C-3: PERFORM-ACTION and other workflow commands (Sheet 9 of 14)

AL only	Filter or Esc only	Command and description	Returns a value
+		`PERFORM-ACTION-TABLE-CLEAR` *`tableFieldID`* Clears the contents of the table field. For list view table fields, tree view table fields, and alert list fields, returns the table to its initial state. For results list fields, fires workflow and then resets the mode to Query. This is equivalent to pressing the New Search form action button.	
+		`PERFORM-ACTION-TABLE-CLEAR-ROWCHANGED` *`tableFieldID`* Clears the `ROWCHANGED` flag for the current row. Important: This command was removed from AR System 7.6.02 and later releases. In those releases, use the `PERFORM-ACTION-TABLE-SET-ROWSTATE` command instead.	

Table C-3: PERFORM-ACTION and other workflow commands (Sheet 10 of 14)

AL only	Filter or Esc only	Command and description	Returns a value
+		`PERFORM-ACTION-TABLE-DELETE-ROW` *`tableFieldID rowIndex rowToSelect`* On-screen, removes a row from view. In the database, sets the row's state to Deleted. This command does not remove the row from the database. Therefore, table loop guides can still step through the row. ("Row" refers to a *row* in list view tables, a *node* in tree view tables, and a *cell* in cell-based tables.) This command takes the following arguments: ■ ***tableFieldID***—The ID of the table field from which to delete the row. This value is required. ■ ***rowIndex***—The position (row index) of the row to delete. This value is required. (Row indexes are based on the version of the table in the client memory data structure.) This value can come at runtime from a field on the form (for example, use `PERFORM-ACTION-TABLE-ADD-ROW $Field1$ $Field2$`). ■ ***rowToSelect*** (Optional)—The position of the row to select after the row at *rowIndex* is deleted. If this value is specified, a row is selected as follows: ■ **0**—Invalid row index. The current selection is not changed, but the highlight is removed. ■ **NULL**—The current selection and highlight are not changed unless the highlighted row is deleted. In that case, the highlight is removed. ■ **Greater than 0 and less than 100,000,000**—The row at the specified index is selected and highlighted, and workflow is fired. ■ **Greater than 100,000,000**—The row at *rowToSelect* - 100,000,000 is selected but not highlighted, and workflow is fired. For example, if the specified index is 100,000,005, row 5 is selected but not highlighted, and workflow is fired. ■ **Less than 0 and greater than -100,000,000**—The row at the specified index multiplied by -1 is selected and highlighted, and workflow is *not* fired. For example, if the specified index is -5, the row at index 5 is selected and highlighted. ■ **Less than -100,000,000**—The row at *rowIndex* + 100,000,000 is selected but not highlighted, and workflow is *not* fired. For example, if the specified index is -100,000,500, row -500 is selected but not highlighted, and workflow is *not* fired. If the selected row is deleted, it remains selected in memory, but it no longer appears on-screen. This command is valid for list view, tree view, and cell-based tables. See the *Form and Application Objects Guide*, "Updating tables on-screen only," page 284.	
+		`PERFORM-ACTION-TABLE-DESELECTALL` *`tableFieldID`* Deselects all entries in a table field. This command is valid for all types of table fields.	

Table C-3: PERFORM-ACTION and other workflow commands (Sheet 11 of 14)

AL only	Filter or Esc only	Command and description	Returns a value
+		`PERFORM-ACTION-TABLE-GET-CURRENT-CHUNK` *`tableFieldID`* Returns one of the following values: ■ Index of the currently displayed table chunk—Chunk index numbers start at `1`. For example, if the fifth chunk is currently displayed, this command returns `5`. ■ `NULL`—Indicates one of the following conditions: ■ The form does not contain the specified field. ■ The command syntax is incorrect. ■ The table is unrefreshed. ■ `1`—Indicates one of the following conditions: ■ The index of the currently displayed chunk is 1. ■ The table has no chunks. For example, it might be a results list table whose chunk size is not defined, a tree view table, or an alert list table. ■ The table is empty. ■ The specified field is not a table.	+

+		`PERFORM-ACTION-TABLE-GET-SELECTED-COLUMN` *`fieldID`* `[` *`returnType`* `]`
		Returns the field ID or level of the selected node, which starts at 1. If root is selected, the command returns 0. If nothing is selected, the command returns `NULL`. This command works only with tree view table fields.
		The arguments for this command are:
		■ **Field ID**—The field ID of the tree field.
		■ **Return type**—1 returns the selected column's field ID. Anything other than 1 returns the number of the level.
		To return the field ID of a tree with an ID of 536870913, enter:
		`PERFORM-ACTION-TABLE-GET-SELECTED-COLUMN 536870913 1`
		To return the level of a tree with an ID of 536870913, enter:
		`PERFORM-ACTION-TABLE-GET-SELECTED-COLUMN 536870913`
			+

+		`PERFORM-ACTION-TABLE-IS-LEAF-SELECTED` *`tableFieldID`* Returns 1 if selected node is a leaf, and returns 0 if the selected node is not a leaf or if nothing is selected.	+
+		`PERFORM-ACTION-TABLE-NEXT-CHUNK` *`tableFieldID`* Displays the next chunk of data in a table. If the action is for a results list, use reserved field ID 1020.	
+		`PERFORM-ACTION-TABLE-PREV-CHUNK` *`tableFieldID`* Displays the next chunk of data in a table. If the action is for a results list, use reserved field ID 1020. This command is ignored for tree view table fields.	

Table C-3: PERFORM-ACTION and other workflow commands (Sheet 12 of 14)

AL only	Filter or Esc only	Command and description	Returns a value
+		`PERFORM-ACTION-TABLE-REFRESH` *`tableFieldID`* `[` *`startRow`* `]` `[` *`numberToRetrieve`* `]` Refreshes a table. You can optionally specify a start row and a maximum number of rows to retrieve. If the action is for a results list, use reserved field ID 1020. This command is valid for all types of table fields.	
+		`PERFORM-ACTION-TABLE-REPORT` *`tableFieldID`* Runs a report on the selected rows in a table. If no rows are selected, the report is on the entire table. This command is ignored for tree view table fields.	

+		`PERFORM-ACTION-TABLE-SELECT-NODE` *`fieldID rowNumber`* `[` *`columnNumber`* `] [` *`flag`* `]`
		Selects the specified node in a tree view table. The arguments for this command are:
		■ **Field ID**—The field ID of the tree field.
		■ **Row number**—The 1-based row position of the node.
		■ **Column number (optional)**—The column (level) of the node. If you enter an invalid column number, a leaf is selected.
		■ **Flag**—1 expands the selected node. Anything other than 1 is ignored.
		An example use of the command is:
		`PERFORM-ACTION-TABLE-SELECT-NODE 536870913 3 4 1`
		This command selects a node whose tree field ID is 536870913, and the node is in row 3 and column 4. The command also requests that the node be expanded.	

+		`PERFORM-ACTION-TABLE-SELECTALL` *`tableFieldID`* Selects all the entries in a table field. This command is valid for all types of table fields. For a tree view table field, the command selects the root label, which represents all data. If there is no root label, the command selects nothing.	

Table C-3: PERFORM-ACTION and other workflow commands (Sheet 13 of 14)

AL only	Filter or Esc only	Command and description	Returns a value
+		`PERFORM-ACTION-TABLE-SET-ROWSTATE` *`tableFieldID rowIndex rowState`* Sets the state of the row at the specified position. ("Row" refers to a *row* in list view tables, a *node* in tree view tables, and a *cell* in cell-based tables.) This command takes the following arguments: ■ ***tableFieldID***—The ID of the table field from which to delete the row. This value is required. ■ ***rowIndex*** (Optional)—The position of the row to apply the state to. (Row indexes are based on the version of the table in the client memory data structure.) If this value is not specified, one of these actions occurs: ■ If this command is run under a table loop guide, the state is applied to the current row in the guide. ■ Otherwise, the state is applied to all the rows in the table. This value can come at runtime from a field on the form (for example, use `PERFORM-ACTION-TABLE-ADD-ROW $Field1$ $Field2$`). ■ ***rowState*** (Optional)—An integer indicating the state to apply to the specified row: ■ **0 (Loaded or Unchanged)**—(Default) The row was just refreshed. ■ **1 (Modified)**—A Set Fields action was just performed on the row. ■ **2 (New)**—The row was just added. See PERFORM-ACTION-TABLE-ADD-ROW. ■ **3 (Deleted)**—The row was just marked as Deleted and removed from view. It is now hidden from users but visible to workflow. See PERFORM-ACTION-TABLE-DELETE-ROW. If this value is not specified, it defaults to `0` (Loaded or Unchanged). If this value is out of the range, such as `4`, this command does not change the row's state. This command sets the $ROWCHANGED$ keyword to the *rowState* value. This command is valid for list view, tree view, and cell-based tables. See the *Form and Application Objects Guide*, "Updating tables on-screen only," page 284.	

Table C-3: PERFORM-ACTION and other workflow commands (Sheet 14 of 14)

AL only	Filter or Esc only	Command and description	Returns a value
+		`PERFORM-ACTION-VALIDATE-NULL-REQUIRED-FIELDS` *`[display]`* *`[fieldIDsSeparatedByBackslashes]`* Validates missing values or the `$NULL$` value in the data fields. If the Run Process action returns true, it stops the workflow, highlights the field with a colored border (for example, a red border), and displays an error message. The *`display`* parameter has two options to determine where to place the message: `PROMPT` (in the prompt bar) or `POPUP` (in a pop-up box). The *`fieldID`* parameter takes a list of field IDs separated by backslashes to validate specific fields. If this parameter is not specified, the parameter verifies *all* data fields on the form (but automatically excludes non-data fields, such as table fields, column fields, and panel holders). If a field in a specified parameter list is not a data field or not a required field, the Run Process action ignores those fields. Note: If a field listed in the command is a default field and the value was not changed, the field's value is not validated. If that default field is changed and empty, then the field's value is validated. The validation is performed on the browser (client) before passing the value to the server. In the following example, the server validates two fields (8 and 536870913), and the message appears in the prompt bar. `PERFORM-ACTION-VALIDATE-NULL-REQUIRED-FIELDS  PROMPT 8\536870913` For more information, see "Ability to highlight required fields through workflow" on page 286.	+
+		`SET-CHANGE-FLAG [ 0 | 1 ]` Sets the change flag status of the current window to *on* (1) or *off* (0).	
+		`SET-RO-COLOR` *`redCode,greenCode,blueCode`* Sets the background color of read-only fields according to red, green, blue (RGB) color coding. For example, an RGB code for blue is 0,0,255.	

Ability to change the form mode

Use the `PERFORM-ACTION-CHANGE-MODE` (see Table on page 257) run process command in a workflow to change the current mode of displayed form without refreshing the browser window.

For example, assume that by default, a form opens in the New mode to allow a user to create new entries. In some cases, the user might need to search for existing entries. In such cases, the user can define an active link using the `PERFORM-ACTION-CHANGE-MODE` command to change the mode. This way, the user avoids closing the browser windows or going back to select a different option. For information about how to define a run process active link, see "Creating a Run Process action" on page 125.

Ability to modify data on display forms

AR System displays a list of entries on a form when the form is in Modify or Display mode. To reuse a form containing a list of entries, embed the form in a view field on a form that is acting as a container. This allows you to make changes to the entry list without refreshing the browser window. This enables functionality similar to refresh (F5) in the BMC Remedy User Tool and can provide a substantial speed improvement.

You use the `PERFORM-ACTION-GET-ENTRY` command (see Table on page 257) to:

- allow users to edit a view form that is being displayed in a container form.
- hide the entry list or create a custom entry list user interface that can reside on a different form, such as, a cell-based table.

When the user executes the `PERFORM-ACTION-GET-ENTRY` command, depending on the current state of the entry, the following occurs:

- If the user has modified fields and not saved, that is, the change flag is active when the command is called, AR System prompts the user to save the record or cancel the edit.
- If the result list is shown when the command is called, AR System displays the emptied result list to the user with an empty row.

For example, assume that you have a table field on Form A. The table field contains entry IDs describing entries in Form B. Form B is loaded into a view field or a browser window. You can create an active link that sends an event to Form B and includes an entry ID as a parameter of the event. You can then create another active link that acts as an event handler on Form B, which has the `PERFORM-ACTION-GET-ENTRY` run process command. The command enables Form B to load the requested entry only. For information about how to define a run process active link, see "Creating a Run Process action" on page 125.

Ability to modify toolbar option

Use the `PERFORM-ACTION-SHOW-TOOLBAR` command (see Table on page 257) in a workflow to control the toolbar display on forms. With this command, you can do the following:

- display the toolbar on demand, through the initiated workflow
- display the toolbar automatically according to criteria that you provide in the workflow

For example, assume that you want to open Form B on a view field from Form A and you do not want to display the toolbar. Using the `PERFORM-ACTION-SHOW-TOOLBAR` command, you can define a run process action to hide the toolbar on Form B. For information about how to define a run process active link, see "Creating a Run Process action" on page 125.

Ability to highlight required fields through workflow

AR System application developers can use the `PERFORM-ACTION-VALIDATE-NULL-REQUIRED-FIELDS` run process command to create a workflow that checks for missing data in required fields. The validation is performed on the browser client and does not require calls to the server, which improves performance. The command highlights the required fields (for example, with red color borders) that are empty and displays an error message.

NOTE

The workflow highlights required fields in browser clients only. The `PERFORM-ACTION-VALIDATE-NULL-REQUIRED-FIELDS` command is not supported in BMC Remedy User.

In BMC Remedy Mid Tier, the browser validates the Required fields when the following actions are executed:

- Push field action
- Commit Changes action
- `PERFORM-ACTION-VALIDATE-NULL-REQUIRED-FIELDS` Run Process command

▶ To create a workflow to highlight the Required fields

NOTE

Before creating a workflow, set one or more fields on the form to Required. For more information about how to change the entry mode to Required, see the *Form and Application Objects Guide*.

1 Right-click the If Action or the Else Action panel header.

2 Choose Add Action > Run Process.

3 Under the Process tab in the Expression Editor, select `PERFORM-ACTION-VALIDATE-NULL-REQUIRED-FIELDS`.

4 Add parameters to the command, for example:

```
PERFORM-ACTION-VALIDATE-NULL-REQUIRED-FIELDS PROMPT
8\536870913
```

For more information about how to use the process commands, see Table on page 257.

5 Click OK to close the Expression Editor dialog box.

6 Save the active link.

Sending events to the Data Visualization definitions

▶ To send an event to a Data Visualization field

1. Create a form with a Data Visualization field.
2. Choose the module to display in the field.
3. Add the following active link action to send an event type and event data to the module field.

   `PERFORM-ACTION-SEND-EVENT F`*`dataVisualizationFieldID eventType`*
   `[`*`eventData`*`]`

   For example:

   ```
   PERFORM-ACTION-SEND-EVENT F536870914 "DisplayItem" $ItemName$
   ```

   Where:

   - `536870914` is the data visualization field ID.
   - `DisplayItem` is an event type that is recognized by the module.
   - `$ItemName$` is a field on the form that contains the name of the item to be displayed in the module field.

Appendix

D Workflow extras

This section describes additional workflow capabilities you can use in AR System.

The following topics are provided:

Allowing data to be dragged and dropped

Through workflow, you can allow users to drag and drop data among fields. Follow this process to enable this feature:

Step 1 Create fields with Draggable and Droppable properties. (This creates the effect of dragging and dropping.)

Step 2 Create the workflow that copies or moves the data.

You can use several properties, keywords, and functions to create drag-and-drop functionality:

- Field properties
  - Draggable
  - Droppable

  See the *Form and Application Objects Guide*, "Field properties," page 501.
- Execution options for active links
  - Drag
  - Drop

  See "Defining workflow execution options" on page 37.
- Keywords
  - `$DRAGSRCFIELDID$`
  - `$DROPTGTFIELDID$`

  See "Keywords" on page 221.
- Functions
  - `AR_FUNCTION_DROPPEDROWINDEX`
  - `AR_FUNCTION_DROPPEDCOLUMNINDEX`

  See "Functions" on page 229.

For examples of how to enable dragging and dropping, see:

- "Example 1: Dragging and dropping data between character fields" on page 292
- "Example 2: Copying data between tables" on page 293
- "Example 3: Dynamically setting fields that are droppable" on page 294

Which fields can be dragged and dropped

Table D-1 lists which fields you can set as "draggable" (a source field) and which you can set as "droppable" (a target field).

Table D-1: Field types and their drag-and-drop capability

Field	Draggable (the source field)	Droppable (the target field)
Data fields (all)	+	+
Horizontal Line	+	+
Vertical Line	+	+
Trim Text	+	+
Trim Box	+	+
Button	+	+
View		+
Data Visualization		+
Application List		
Horizontal Navigation		
Vertical Navigation		
Table fields (list, tree, and cell-based)	+	+
Alert List	+	+
Results List		
Attachment Pool	+	+
Attachment Field		
Panels and Panel Holders	+	+
Columns		

NOTE

For data fields, users must initiate a drag from the label portion of the field (not from the data portion).

Example 1: Dragging and dropping data between character fields

The following procedure describes how to allow users to copy data between two character fields: Bill To and Ship To.

▶ To allow users to drag and drop data between character fields

1 Create two character fields:

- Bill To
- Ship To

2 Set properties for fields as follows:

- For the Bill To field, set the Draggable property to True.
- For the Ship To field, set the Droppable property to True.

3 Create an active link that copies the data to the drop target when the mouse button is released.

a In the Execution Options panel, enter the Ship To field in the Field field.

b Select the Drop execution option.

c Create a Set Fields active link that sets the Ship To field to contain the contents of the Bill To field.

The Value of the Ship To field should be `$Bill To$`.

Optionally, you can create an active link that responds to a Drop event of a drop target field and performs a Set Fields action:

```
$PROCESS$ Application-Copy-Field-Value $DROPTGTFIELDID$
$DRAGSRCFIELDID$
```

This active link is written once per drop target field and will accommodate any number of drag source data fields. The developer avoids having to write one active link per drag source field to copy data to the drop target field.

For more information, see "Process commands" on page 261.

TIP

To *move* instead of *copy* the data, create another Set Fields action where the Bill To field is set to NULL when the drop action occurs.

Example 2: Copying data between tables

The following procedure describes how to allow users to copy data between two table fields. For example, a table called “Current Inventory” includes a list of inventory items, and a manager needs to remove an item that is broken. With the drag-and-drop function implemented, the manager simply drags the item from the Current Inventory table to a table called “Remove From Inventory.”

During the drag sequence, a drag proxy (3 items selected) appears and displays the number of items being dragged. A label (such as “items selected”) can appear next to the number. To configure the label that appears in the drag proxy, enter text in brackets ([ ]) in the Label property for the table field. You can enter one, two, or no labels, as described in the following procedure.

▶ To allow users to drag and drop data between table fields

1 In the form with the two table fields you want to use in the drag-and-drop sequence, set the Draggable and Droppable field properties for the table fields.

2 Set the Row Label and Row Label Plural properties for the table whose rows will be dragged.

Field property	Description	What to enter

Row Label	Configures the drag proxy’s label when *one* row is selected.
	Enter `{0}` for the count, and include any text before or after `{0}`.
		`labelText{0}labelText`
		For example:
		`{0}item selected`
		Using this example, 1 item selected appears when the user drags over a valid drop target. 1 item selected appears on an invalid drop target.

Row Label Plural	Configures the drag proxy’s label when *more than one row* is selected.
	Enter `{0}` for the count, and include any text before or after `{0}`.
		`labelText{0}labelText`
		For example:
		`{0}items selected`
		Using this example, 3 items selected appears when three rows are selected.

If you do not enter labels, only an icon appears (for example,).

3 Create an active link with the following actions that occur when the Drop execution option occurs on the table:

- **Run Process**—Use the `PERFORM-ACTION-TABLE-ADD-ROW` command to add a row to the table that will be dragged to.

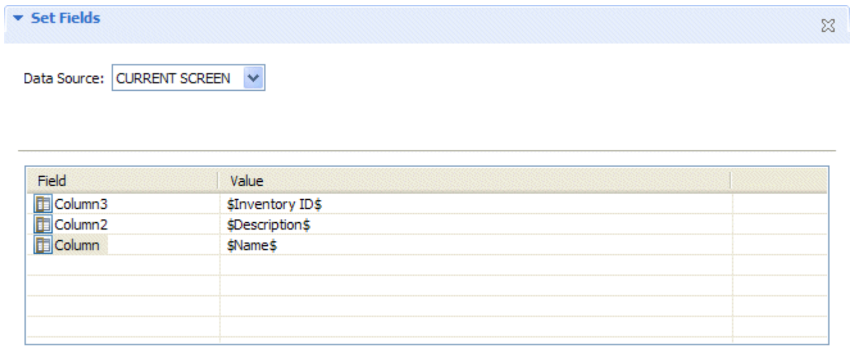
  For example, if the table’s field ID is 536870919, then the command would look like this:

  ```
  PERFORM-ACTION-TABLE-ADD-ROW 536870919
  ```

- **Set Fields**—Set the values of the cells in the new row. This means setting the value for *each column* in the drop target table to the value of each column field in the row being dragged.

Figure D-1 shows an example of the Column fields being set to the value of the Inventory ID, Description, and Name column fields being dragged.

Figure D-1: Setting the values for columns



Data Source: CURRENT SCREEN

Field	Value
Column3	$Inventory ID$
Column2	$Description$
Column	$Name$

Example 3: Dynamically setting fields that are droppable

If you create more than one set of draggable and droppable fields, when a user drags on a field, *all* of the droppable fields are highlighted. Instead, only the target field should be highlighted. To ensure that only the intended droppable field is highlighted, use the Change Field action to dynamically set the intended droppable field and disable to other droppable fields.

For example, the Bill To and Ship To fields are one set of draggable and droppable fields on the Order Entry form. The Item Number Table and the Order List Table comprise another set. The following example explains how to dynamically set the Order List Table so that it is *not* highlighted when the Bill To field is dragged.

▶ To dynamically set a field so that it is not droppable

1. Create a new active link.
2. Select the associated form (in this example, Order Entry).
3. On the Execution Options panel, select the field that will be dragged.

   In this example, the field is Bill To.
4. Select Drag.
5. Create a Change Field action.
   a. Right-click on the If Actions panel, and choose Add Action > Change Field.
   b. Select the field that should be droppable when the draggable field is dragged.

      In this example, the field is Ship To.
   c. From the Field Drop list, select Enable.

6 Create another Change Field action.

   a Right-click on the If Actions panel, and choose Add Action > Change Field.

   b Select the field that should *not* be droppable when the draggable field is dragged.

   In this example, the field is Order List Table should not be droppable.

   c From the Field Drop list, select Disable.

7 Save the active link.

Sending events between windows

You can use active link workflow to send messages to one or more windows. For example, a child window can tell a parent window to execute certain active links. Here, the parent window needs to know when some action has happened on the child window (for example, it is closed) so that workflow on the parent window can refresh related data. The Send Event functionality provides a way for windows to synchronize their data, and a mechanism for a parent window to be notified when a child window has been closed.

WARNING

In some cases, event driven workflow can fail when executed on the mid tier. To avoid this issue, explicitly define the parent-child relationship.

To send events between windows, you coordinate the following mechanisms in BMC Remedy Developer Studio:

- The `PERFORM-ACTION-SEND-EVENT` *`target eventType eventData`* Run Process command, which identifies the window to which to send the event. (For more information, see “Ability to highlight required fields through workflow” on page 286.)
- On Event active link execute condition. (For more information, see “Defining workflow execution options” on page 37.)
- Keywords to identify the event:
  - Current event (`$EVENTTYPE$`).
  - Current event data (`$EVENTDATA$`).
  - Current window (`$CURRENTWINID$`).
  - Window that sent the event (`$EVENTSRCWINID$`).
  - Last opened window (`$LASTOPENEDWINID$`).

  For more information, see “Keywords” on page 221.

NOTE

In a web browser, opening windows from workflow is not synchronous. For example, if an active link contains an Open Window action and the next action is a Send Event Run Process action, the child window does not receive the event. (The active link works correctly in BMC Remedy User.) To solve this issue for the web client, create the workflow so that the parent or child window is loaded, and then the send event action is executed.

Additional considerations for sending events

Using the `$EVENTTYPE$` keyword, the active link running on the Event condition can access the *eventType* specified in the `PERFORM-ACTION-SEND-EVENT` *target eventType eventData* command. (The `$EVENTDATA$` keyword can represent event data to be attached.) Likewise, the recipient using the `$EVENTSRCWINID$` keyword can access the window ID of the window sending the event.

When using the Run Process Send Event action with web browsers, sending a message to a parent window or child windows behaves in an undefined fashion if the contents of the windows are changed by:

- Navigating to a new URL.
- Clicking the Back, Forward, or Reload buttons on the browser.
- Using an Open Window active link action that reloads new data into an existing window, causing the form to be unloaded.

If this event is sent to a data visualization field, the *target* must be F*dataVisualizationFieldID*.

An example of Run Process syntax that sends the `ChildClosed` event to the current window's parent is:

```
PERFORM-ACTION-SEND-EVENT @ ChildClosed
```

The syntax that sends the `ParentClosed` event to the current window's children is:

```
PERFORM-ACTION-SEND-EVENT # ParentClosed
```

NOTE

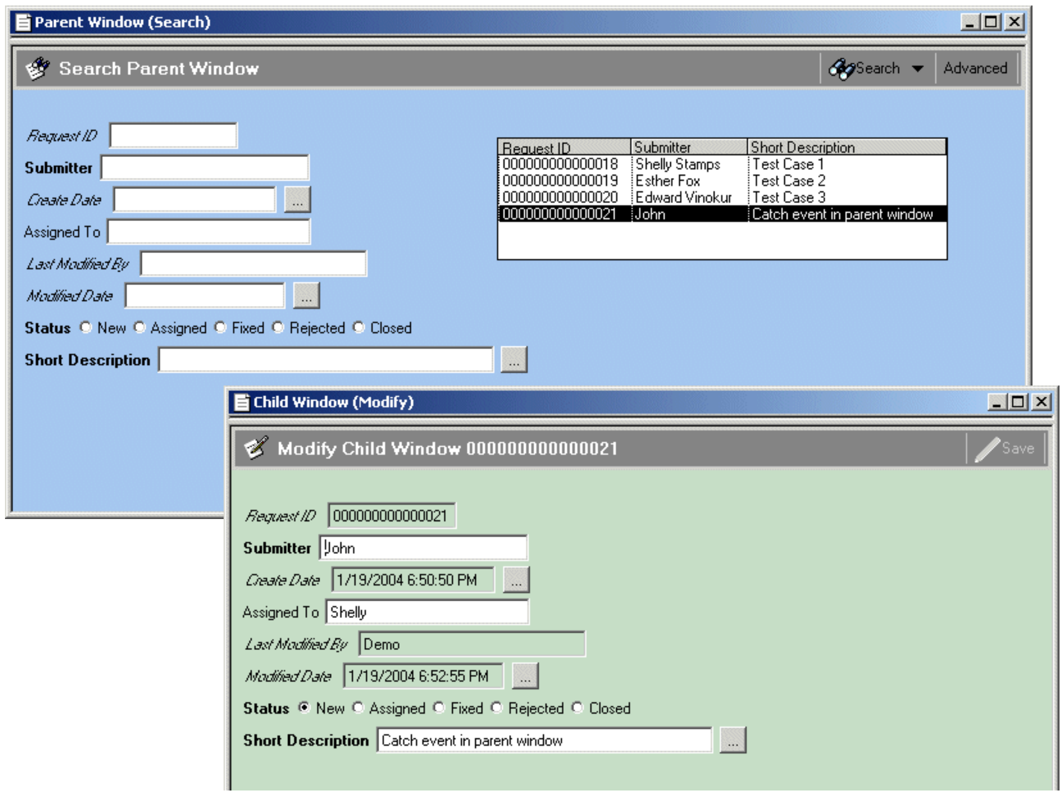
Quotation marks are optional in the Run Process command if the *eventType* does not contain a space. For example, they are optional for:
`PERFORM-ACTION-SEND-EVENT @ EventA`
but you must use them for:
`PERFORM-ACTION-SEND-EVENT @ "This is the event".`

If you are using field references for the *target* and *eventType*, make sure the field is set properly prior to executing the Run Process command.

Example of sending events between windows

The following example describes how you can send an event from a child window to a parent window. When a user modifies a record in the Child Window form, an After Modify condition in the active link automatically sends an event to the Parent Window form and updates a table field.

Figure D-2: Sending event from child window to parent window



Request ID	Submitter	Short Description
000000000000018	Shelly Stamps	Test Case 1
000000000000019	Esther Fox	Test Case 2
000000000000020	Edward Vinokur	Test Case 3
000000000000021	John	Catch event in parent window

Modify Child Window 000000000000021

Request ID	000000000000021
Submitter	John
Create Date	1/19/2004 6:50:50 PM
Assigned To	Shelly
Last Modified By	Demo
Modified Date	1/19/2004 6:52:55 PM
Status	[✓ New] Assigned Fixed Rejected Closed
Short Description	Catch event in parent window

The concepts used in the following procedures are straightforward, so that you can easily reverse sending the event from a parent window to a child window.

NOTE

You cannot have workflow that opens a dialog box and then sends an event to the dialog box. Dialog boxes are modal, and subsequent Send Event actions or active links do not fire until the dialog box closes. In this case, the dialog box no longer exists. Instead, use an On Loaded active link condition in the dialog box to send an event to itself, using the keyword `$LASTOPENEDWINID$` as the target.

To send an event to a parent window

1 Create a form (for example, Parent Window).

2 Create another form (for example, Child Window).

3 Add a table field to the Parent Window form.

4 Select the new table field.

5 Edit the properties of the table:

   a In the Properties tab, select the Tree / Table Property value, and click its ellipsis button.

   b Enter Child Window in the Form Name field.

   The columns of the table field in Parent Form must point back to the Child Window form.

   c Add the following fields to the Table Columns list:

   - Request ID (Column)
   - Submitter (Column2)
   - Short Description (Column3)
   - Assigned To (Column4)

   These columns help you identify which records in the table field to modify.

6 Create an active link (Child Send Event to Parent After Modify) with a Run Process action.

   a Configure the settings in the following panels:

   - **Associated Forms:** Child Window
   - **Execution Options:** After Modify
   - **Run If:** `'Submitter' != 'Assigned To'`

   You want the active link in the child window to fire if the submitter and assignee are not the same user.

   b Right-click on the If Actions tab, and choose Add Action > Run Process.

   c In the Command Line field, enter:

   ```
   PERFORM-ACTION-SEND-EVENT @ “RefreshTable”
   ```

This Run Process command sends the RefreshTable event to the parent window. You will use this same event string later in active link workflow to catch the sent event in the parent window.

TIP

Make sure the target symbols (@, #, *) are used correctly. If you are trying to send an event to a parent window, do not use #.

This Run Process command and its special syntax can send messages to one or more windows. For example, to send an event to all windows, use `PERFORM-ACTION-SEND-EVENT * “All Windows”`. You can also send events to a specific window:

```
PERFORM-ACTION-SEND-EVENT $536870921$ Processing
PERFORM-ACTION-SEND-EVENT $Short Description$ "network is down"
PERFORM-ACTION-SEND-EVENT $My Field$ "process request"
PERFORM-ACTION-SEND-EVENT $LASTOPENEDWINID$ "close now"
```

This command uses an established mechanism in the client environments to support special commands that are not run as executables, but as internal commands within the client. For detailed information about the command syntax, see Appendix C, "Using Run Process and $PROCESS$ commands."

NOTE

Events are asynchronous, and the Run Process command returns immediately. Other windows receive the events and process the workflow at a later time.

7 Create an active link (Parent Catch Event) with a Change Field action to catch the sent event.

   a Configure the settings in the following panels:

   - **Associated Forms:** Parent Window
   - **Execution Options:** Event
   - **Run If:** `$EVENTTYPE$ = "RefreshTable"`

   Because event names are intended to be unique, this active link fires only if it receives the RefreshTable event.

NOTE

You must use quotation marks in the Run If qualification for the event type.

   You can trigger active links by examining the values of the `EVENTSRCWINID`, `CURRENTWINID`, or `EVENTTYPE` keywords in the Run If qualification. Here, the `EVENTTYPE` keyword is used in the Run If qualification to fire the active link if the parent window catches the RefreshTable event sent by the child window.

   b Right-click on the If Actions tab, and choose Add Action > Change Field.

   c Enter the table field's name in the Field field.

   d Select the Refresh Tree / Table check box.

   Here the rows in the table field are automatically refreshed after changes are made to the underlying request.

8 Log in to BMC Remedy User, and open the Child Window form.

9 As a test, create several tickets in the Child Window form, but create one ticket with `John` as the value of the Submitter and Assigned To fields.

10 Open the Parent Window form, and click the table field to refresh it.

11 Double-click the request assigned to John in the Results List to open the Child Window form.

12 Change the assignee to another name and make changes in the Short Description field (for example, enter `Catch event in parent window`), and then save the request.

   When you modify the request, the active link workflow is triggered and sends the child window event to the parent window. The table field in the parent window is refreshed automatically.

If the target window is in the process of executing workflow, that workflow completes *before* the event is received.

TIP

If the active links do not fire in the target window, make sure the event type string in the Run If qualification matches the event type string in the Run Process command.

Controlling row selection in a table field

This section describes how to control which table row is selected in a Set Fields action.

NOTE

Do not use the Set Fields action to set a table row inside a table loop when performing the following procedures. For more information how to create an active link guide that loops through a table field, see "Using active link guides in client-side table fields" on page 151.

▶ To set an integer field with the number of the selected row

1 Create an active link with a Set Fields If action.

2 In Field column of the Set Fields panel, enter the field where you want the row number entered. (This is usually an integer field.)

3 In the Value column, enter the table field name.

   a Click the corresponding cell in the Value column, and then click the ellipsis (...) button.

   b In the Expression Editor, select the table field, and click Add Field.

   c Click OK.

   The field name appears in the Value column in the proper format (for example, `$Help Desk Requests$`).

When the active link is executed, the number of the selected row is entered in the field from step 2. For example, if row 5 is selected, 5 is entered.

▶ To set how a row is selected in a table field

1 Create an active link with a Set Fields If action.

2 In Field column of the Set Fields panel, enter the table field where you want to select a row.

3 In the Value column, enter one of the following:

   - A variable for a field that contains an integer (for example, `$Select Row$`)
   - An integer

If you enter an integer in the Value column, use the following table to learn how to enter values that determine whether the user sees the row highlighted and whether the active link (which is executed through the Row Choice execution option) is executed.

Table D-2: Examples of values that selects a row

Value column contents	Result
n	Selects and highlights row *n* and executes the active link. The user sees the row highlighted.
- *n*	Selects and highlights row *n* and *does not* execute the active link. The user sees the row highlighted.
n + 100000000	Selects row *n* and executes the active link. The user *does not* see the row highlighted. If a row is selected and is already highlighted, the highlighting does not move.
-(*n* + 100000000)	Selects row *n* and *does not* execute the active link. The user *does not* see the row highlighted. If a row is selected and is already highlighted, the highlighting does not move.

Restricting filter data retrieval by user permissions

The Get Data As filter property allows you to specify the permissions that a filter uses for certain operations. Prior to the introduction of this property, all filters in AR System always executed with the permissions of an administrator. In other words, by default, filters can read and write any field in any request, regardless of the permissions on the field or on the request.

When set to User, the Get Data As property supports applications that use row-level security and dynamic groups for access control. This property enables a filter to adopt the permissions of the user that caused the filter to be called when retrieving the data upon which the filter acts. A filter with this property configured for user permissions can still write to any field on any request appropriate to the operation, but can only retrieve entries and fields that are visible to the user that caused the filter to be executed.

The Get Data As filter property has these values:

- **Server**—Run the filter with the permissions of the Administrator. This is the default setting.
- **User**—Run the filter with the permissions of the user that called the filter.

For example, suppose there are two groups with data on a single form that contains salaries for different job levels. Each group should not be aware of the other group's data. In this case, row-level security might be used to identify the records within the form that each group can access.

Without the Get Data As property set to User, a filter cannot avoid accessing all records in the form that match the qualifications in the filter, so it manipulates information for both groups indiscriminately. Operations that should be performed only on records for one group must have additional qualifications to restrict the data, and those additional qualifications must be revisited each time permissions on any element of the form are changed.

With Get Data As set to User, filter operations can be restricted to retrieving only information visible to the groups whose user executed the filter. This is the case regardless of changes made to the form after the filter has been created.

▶ To configure a filter to get data with the permissions of the user

1. In BMC Remedy Developer Studio, create the filter or open an existing filter in the editor.
2. In the Properties tab, expand the Data Access section, and then click the down-arrow in the Value column.
3. Set the value to User.
4. Save the filter.

How Get Data As User works

Filters include a Run If qualifier that accesses data in the record upon which an operation is being performed and determines what actions in the filter should be performed. Qualifiers that are part of actions determine which data the actions affect. The Get Data As user property applies only to qualifiers used in filter actions; it does not apply to Run If qualifiers.

The workflow actions most affected by this property are Set Fields, Push Fields, and Call Guide. Some filter functions are also affected.

Set Fields Action

The Set Fields operation includes a source form and a qualifier that identifies records from which data can be retrieved to be put into the record upon which an operation is being performed. When Get Data As is set to User, the records and fields that match the qualifier and can be set into fields are limited to those visible to the calling user.

All fields within the record on which the operation is being performed can be written, but any data obtained from other records is restricted according to the permissions of the user.

Push Fields Action

The Push Fields operation includes a destination form and a qualifier that is used to identify records into which data can be written from the record upon which an operation is being performed. When Get Data As User is set, the records that the qualifier identifies to be written are limited to those visible to the calling user.

All fields within records to which data is being pushed can be written, but the list of records to which data is pushed is restricted.

Call Guide Action

The Call Guide action causes a filter guide to be run. The guide can run once, or it can run for each entry in a table field. With this property set, the entries and fields that are retrieved for the table field are limited to those visible to the calling user.

The action calling the guide does not affect filter operations in the guide, but the data over which the guide loops is restricted according to the permission of the user.

Workflow Functions

In addition to the Set Fields, Push Fields, and Call Guide actions, setting Get Data As to User affects the workflow functions `COLCOUNT`, `COLAVG`, `COLMIN`, and `COLMAX`. These functions operate over the columns in a table field. When performed in a guide that was called by a filter with the Get Data As User set, the entries and fields used to populate columns in the table field are limited to those visible to the calling user.

When Get Data As User is applied

If a filter has Get Data As set to User, the workflow actions that are subject to this setting are affected whenever the filter fires in the course of an API call. If a filter with Get Data As User set is run in the course of an escalation instead of through an API call, the property is not be applied and the filter actions remain unrestricted.

Filter log output for Get Data As

Filters with Get Data As set to User are identified in the filter log file by the string “`with data retrieval as user` *`userName`*.”

Index