Hexis
CYBER SOLUTIONS
a KEYW company

Reporting Guide

HawkEye AP 5.0.1

February 5, 2014

Hexis Cyber Solutions, Inc.
2800 Campus Drive, Suite 150
San Mateo, CA 94403
+1 (650) 830-0484
fax: +1 (650) 631-2810
www.hexiscyber.com

Copyright

Trademarks

The Hexis Cyber Solutions® logo and HawkEye AP are registered trademarks of Hexis Cyber Solutions, Inc., in the United States. Other company, product names or logos mentioned or reproduced in this document may be trademarks or registered trademarks of their respective companies and are hereby acknowledged.

Patents

The software described in this manual contains technology protected by U.S. Patent 7,024,414.

Permitted Use

This document is provided by Hexis Cyber Solutions, Inc., and/or its subsidiaries to entities using the services described herein pursuant to a separate services and/or software license agreement. This document may be used by such entities only for the limited purpose of using the services described herein as authorized by such services agreement. Any other use of this document is strictly prohibited, unless expressly authorized by Hexis Cyber Solutions, Inc., in writing.

Disclaimer and Limitation of Liability

Except as otherwise expressly provided in the applicable services and/or software license agreement, Hexis Cyber Solutions, Inc., and its subsidiaries make no representations or warranties of any kind, whether express or implied, regarding the accuracy of this document and disclaim the implied warranties of merchantability, fitness for a particular purpose, title and non-infringement of third party rights, to the maximum extent permitted by law. Hexis Cyber Solutions, Inc., and its subsidiaries assume no responsibility for any damage or loss resulting from the use of this document. This document is subject to revision from time to time.

Confidential Information

The information contained in this document is confidential and proprietary to Hexis Cyber Solutions, Inc., and its subsidiaries and may not be used or disclosed except as expressly authorized in writing by Hexis Cyber Solutions, Inc. This document is released to the possessor on a strictly secure basis. No authority, express or implied, exists to permit discussion with any third party. By continuing to possess this document, the bearer accepts all direct and indirect responsibility for keeping these materials in confidence. The bearer acknowledges the implicit material commercial harm to Hexis Cyber Solutions, Inc., of unauthorized disclosure of the contents herein, including an acknowledgment of responsibility for such damages caused to Hexis Cyber Solutions, Inc., if any.

Destination Control Statement

ECCN (Export Control Classification Number):5D002 ENC

These commodities, technology, or software were exported from the United States in accordance with the Export Administration Regulations. Diversion contrary to United States law is prohibited.

Table of Contents

PREFACE

This book, the *Reporting Guide*, describes the 5.0.1 version of HawkEye AP software.

HawkEye AP *reporting* includes:

- **reports**—provide information on event data collected from all sources in your Enterprise and stored in the HawkEye AP Event Data Warehouse (EDW); display results in tabular or graphic form.

  The HawkEye AP distribution includes *Analytics Reports*, which provide a comprehensive solution for reporting on event data stored in the EDW.

- **alerts**—provide immediate information about enterprise and system assets and for exception reports; the HawkEye AP Real-Time system collects enterprise alerts and displays them before storing them in the EDW.

HawkEye AP Console is the primary interface for creating and viewing reports and viewing and investigating alerts. Business analysts can use HawkEye AP Console to run reports provided by HawkEye AP, create new reports using a wizard, view and investigate alerts, modify and run saved reports, schedule reports, and create dashboards that customize display of event and alert data. Advanced users can also use the Sensage SQL query language and Perl functions to create reports for special situations. This manual documents the HawkEye AP Console interface, HawkEye AP Analytics, and the Sensage SQL query language and its related Perl functions.

NOTE: HawkEye AP Console also provides an Administration Mode interface that enables you to create users, roles, and assets, and to enable rules. For information on these and other administration tasks, see the *Administration Guide*.

This Preface contains the following sections:

- "Audience for this Book", next
- "Reporting Guide Organization", on page 19
- "Road Map to HawkEye AP Documentation", on page 21
- "Conventions Used in HawkEye AP Documentation", on page 23
- "Contacting Technical Support", on page 24

AUDIENCE FOR THIS BOOK

This book is directed to:

- **business analysts**—who use dashboards and reports to analyze event data

- **developers**—who create SQL queries, views, and reports, and who develop log adapter PTL files

REPORTING GUIDE ORGANIZATION

This book contains the following chapters:

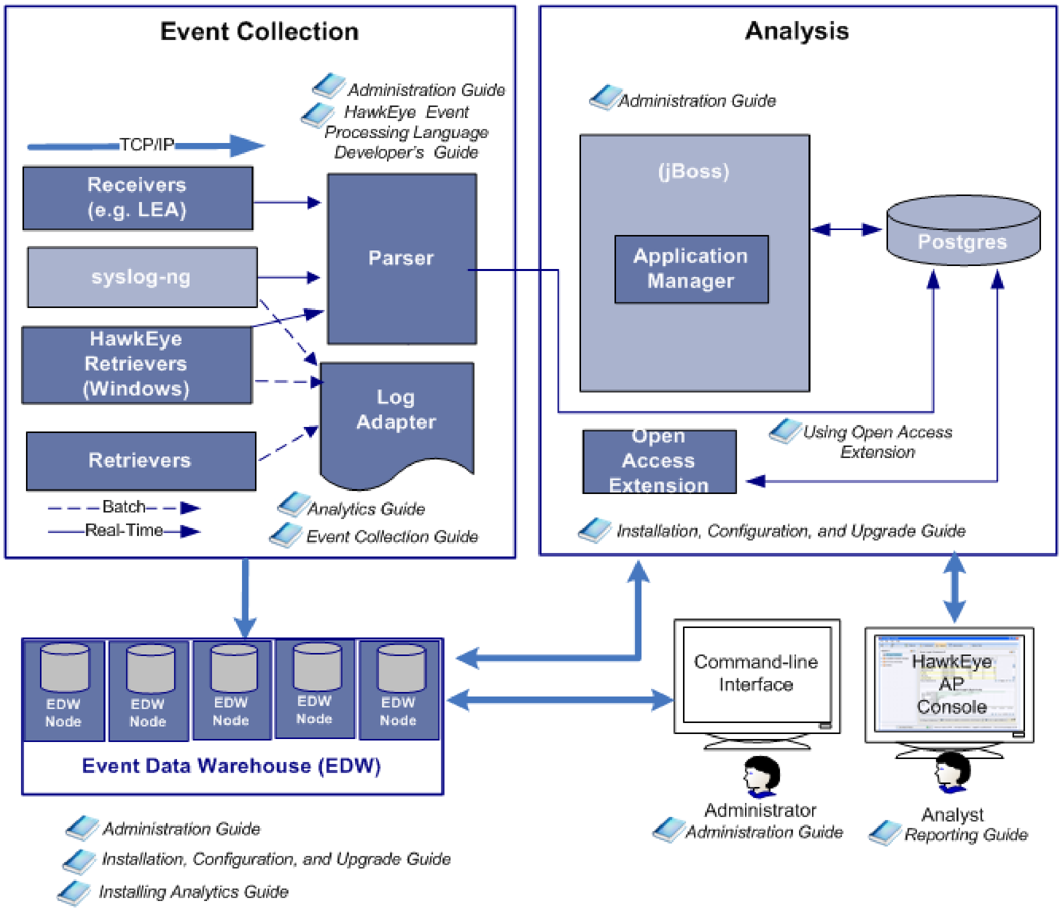
- Chapter 1: Getting Started—Describes high-level architecture, documents how to log in and out of HawkEye AP Console, introduces the HawkEye AP Console interface and its components, documents how to set user preferences, introduces HawkEye AP Analytics reports, and introduces HawkEye AP SQL.
- Chapter 2: Using Dashboards—Describes how to view reports and alerts in a dashboard
- Chapter 3: Running, Viewing, and Managing Reports—Describes how to manage HawkEye AP reports
- Chapter 4: Creating and Editing Wizard Reports—Describes how to create and edit Wizard reports
- Chapter 5: Creating and Editing SQL Reports—Describes how to create and edit SQL reports
- Chapter 6: Creating and Managing Dashboards—Describes how to create and manage dashboards
- Chapter 7: Creating and Editing Schedules—Describes how to create and manage schedules
- Chapter 8: Creating Alerting Rules from Templates—Describes how to create alerting rules
- Chapter 9: Report Libraries Reference—Provides a reference on libraries you can use to create reports.
- Chapter 10: Sensage SQL—Provides HawkEye AP SQL reference

  NOTE: HawkEye AP SQL enables data retrieval, but does not support data update or delete. These functions are unnecessary to log analysis and prohibited by compliance
- Chapter 11: SQL Functions—Provides reference for the rich library of functions that HawkEye AP SQL supports. Supported tasks include string matching, time handling, and mathematical functions, lookups, and conditional evaluations.
- Chapter 12: Perl Subroutines—Provides reference on how to create, debug, and use Perl routines from within HawkEye AP SQL. These functions allow you to perform complex transforms on data.

Road Map to HawkEye AP Documentation

This document, the *Reporting Guide*, is part of the larger documentation set of your HawkEye AP system. Figure P-1 illustrates HawkEye AP components and modules in the context of their function within the HawkEye AP system.

Figure P-1: Road Map to HawkEye AP Documentation



The table below describes all the manuals in the HawkEye AP documentation set and the user roles to which they are directed.

Role	Tasks	Documentation
Business Analyst	• Use and create dashboards • View and create reports • Monitor enterprise and exception alerts • Schedule reports & dashboards • Create Alerting Rules from Templates	*Reporting Guide*

Role	Tasks	Documentation
Business Analyst	• Learn about Analytics • Use IntelliSchema views • Learn about the Foundation and Compliance Analytics packages • Learn about additional Analytics packages	*Analytics Guide*
Report Developer or Security Analyst	• Use Sensage SQL, Sensage SQL functions, and libraries to create reports or query the EDW • Create and use Perl code in Sensage SQL statements • Use the DBD Driver to query HawkEye AP from other locations	*Reporting Guide*
Security System Administrator	• Configure retrievers, receivers, and collectors • Configure HawkEye Retriever • Create log adapter PTL files	*Event Collection Guide*
Security Analyst	• Create parsing rules, alerting rules, and configurable alerting rule templates • Manage rules	*HawkEye Event Processing Language Developers Guide*
System Administrator	• Install HawkEye AP • Configure HawkEye AP and its components	*Installation, Configuration, and Upgrade Guide*
System Administrator	• Install Analytics	*Installing Analytics*
System Administrator	• Manage the HawkEye AP Event Data Warehouse (EDW) • Manage the Collector • Manage users, groups, and permissions • Archive to nearline storage • Manage assets & monitor security alerts • Monitor log source health • Monitor system health • Troubleshoot • Error Messages	*Administration Guide*
System Administrator	• Install and configure log adapters	*Analytics Guide*
Developer	• Access EDW data using open standards as ANSI SQL, ODBC, and JDBC	*Using Open Access Extension*
Legal	Monitor third-party licenses	*Third-Party Open Source Licensing*

TIP: You can access the manuals listed above from:

- HawkEye AP Console online help

  Click **Help** > **Help Contents**.

- HawkEye AP Welcome page

Click the **Documentation** hyperlink.

For more information, see “Logging into HawkEye AP Console” in the *Administration Guide*.

Conventions Used in HawkEye AP Documentation

This convention...	Indicates...	Example
bold text	Names of user interface items, such as field names, buttons, menu choices, and keystrokes	Click **Clear Filter**.
italic text	Indicates a variable name or a new term the first time it appears	`http://<host>:<port>/index.mhtml` Use the *whammerjammer* to adjust the whamming frequency.
`Courier text`	Indicates a literal value, such as a command name, file name, information typed by the user, or information displayed by the system	`atquery localhost:8072 myquery.sql`
SMALL CAPS	Indicates a key on the computer keyboard	Press ENTER.
`{ }`	In a syntax line, curly braces surround a set of options from which you must choose one and only one. **NOTE**: Syntax specifications for SELECT statements include curly braces as part of the `{INCLUDE_BAD_LOADS]` keyword.	`{ start | stop | restart }`
`[ ]`	In a syntax line, square brackets surround an optional parameter	`atquery [options] <host>:<port> -`

`|`	In a syntax line, a pipe within square brackets or curly braces separates a choice between mutually exclusive parameters
	NOTE: Syntax for defining a Nearline Storage Address (NSA) includes a pipe.
		`{ start | stop | restart }`
		`[g|m]`

`...`	In a syntax line, ellipses indicate a repetition of the previous parameter	The following example indicates you can enter multiple, comma-separated options: `<option>[, <option>[…]]`

This convention...	Indicates...	Example

backslash (\)	A backslash in command-line syntax or in a command example behaves as the escape character on Unix. It removes any special meaning from the character immediately following it. In HawkEye AP documentation, a backslash nullifies the special meaning of the newline character as a command terminator. Without the backslash, pressing ENTER at the end of the line causes the Unix system to execute the text preceding the ENTER. Without the backslash, you must allow long commands to wrap over multiple lines as a single line.	`atquery --user=administrator \`
		`--pass=pass:p@ss localhost:8072\`
		`-e='SELECT * FROM system.users;'`

CONTACTING TECHNICAL SUPPORT

For additional help, email support@hexiscyber.com or call +1 650 830-0484, Option 2. Also see the Hexis Cyber Solutions Technical Support web page at http://www.hexiscyber.com/content/ for HawkEye AP documentation, product downloads, and additional information on contacting support to escalate help on issues that impact your production environment.

CHAPTER 1

Getting Started

This chapter describes the following sections:

- "Overview", next
- "Accessing HawkEye AP Console", on page 25
- "Navigating HawkEye AP Console", on page 27
- "How does your data get to HawkEye AP Console?", on page 37
- "Reporting and HawkEye AP Analytics", on page 38
- "Querying and Sensage SQL", on page 38

OVERVIEW

HawkEye AP reporting enables you to make use of the event data collected by your HawkEye AP deployment. HawkEye AP collects the event data from information systems in your organization and transforms the data into reports and alerts that help identify threats and areas of concern in your organization. The primary interface for viewing and manipulating these reports and alerts is HawkEye AP Console.

HawkEye AP Console is a graphical application that runs on a Windows workstation. HawkEye AP Console remotely accesses the HawkEye AP Event Data Warehouse (EDW) to retrieve stored event data for display on your local workstation. HawkEye AP Console also receives alerts, which represent streaming event data that matches pre-defined criteria.

Not only does HawkEye AP Console enable you to view event data, it also enables users with appropriate permissions to:

- consolidate reports, alerts, and other elements into *dashboards* for quick access to important information
- create folders that organize reports and dashboards for viewing
- create, edit, run, and schedule reports and dashboards
- create users and grant them permissions
- create and manage libraries
- manage assets and rules

This chapter provides an overview of HawkEye AP Console functionality. The following chapters in this book provide details on using HawkEye AP Console and HawkEye AP reporting.

ACCESSING HAWKEYE AP CONSOLE

This section describes the following topics:

- "Logging into HawkEye AP Console", next
- "Logging out of the HawkEye AP Console", on page 27

Logging into HawkEye AP Console

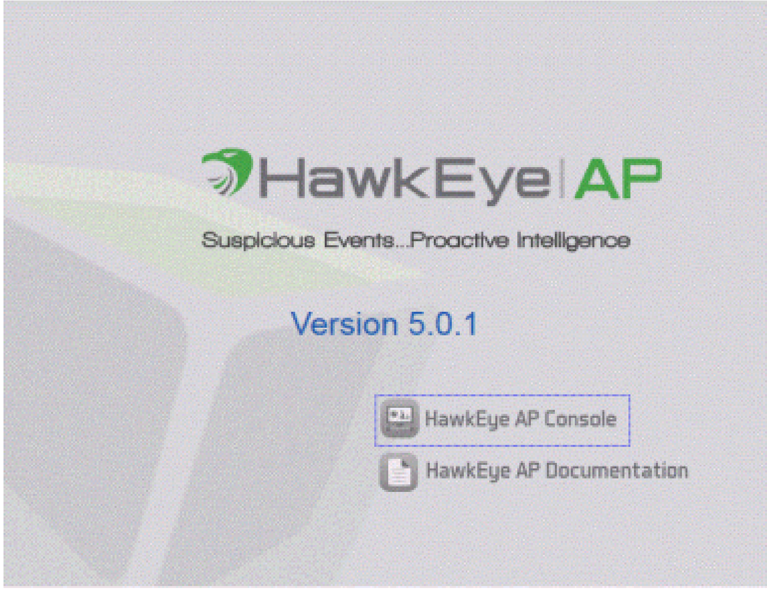
To access HawkEye AP Console, first open the HawkEye AP Welcome page in your Web browser. For the Web address of the welcome page as well as your user name and password, contact your HawkEye AP administrator.

To log into the HawkEye AP Console

1 Enter the HawkEye AP Web address in your Web browser.

The HawkEye AP Welcome page displays, as shown in Figure 1-1.

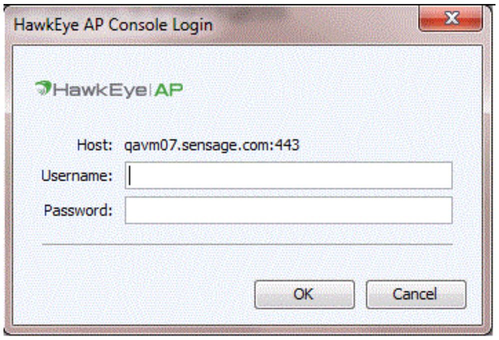
Figure 1-1: HawkEye AP Welcome Page



2 Click **HawkEye AP Console**.

3 Java Web Start launches and the HawkEye AP Console login window displays, as shown in Figure 1-2.

Figure 1-2: HawkEye AP Console Login Window



Host: qavm07.sensage.com:443

Username:

Password:

4 Enter your **User Name** and **Password**.

5 Click **OK**.

The HawkEye AP Console window displays, as shown in Figure 1-3.

Logging out of the HawkEye AP Console

To log out of HawkEye AP Console

From the menu, select **File** > **Exit**.

NAVIGATING HAWKEYE AP CONSOLE

- "Introduction to the Interface", next
- "Setting Preferences", on page 35
- "Changing Your Password", on page 36
- "Accessing Online Help", on page 36

Introduction to the Interface

HawkEye AP Console provides graphical access to event data collected by a HawkEye AP deployment, system data, and tools for viewing and manipulating the data. HawkEye AP Console has three different modes:

- Dashboards

  Dashboards Mode enables you to organize and view HawkEye AP reports and alerts on one or more pages that group the data meaningfully and facilitate interpretation of the data. Dashboards can display images and text that enhance the displayed data.

  This mode enables creation and display of multiple dashboards, each with multiple pages. Typically dashboards display information for a particular audience or purpose and are organized within folders to further group their data meaningfully.

  You can schedule, run, print, export, and email dashboard pages and their reports. You can perform some of those same operations on reports that display on a dashboard. The dashboard as an entity can be locked, which hides the title bar and container outline of each displayed widget and prevents repositioning of the widgets. User permissions determine access to dashboards and dashboard items.

  For more information, see:

  - Chapter 2: Using Dashboards
  - Chapter 6: Creating and Managing Dashboards

- Reports

  Reports Mode enables you to create, modify, and run reports, and to view report output. This mode enables creation of two types of report (Wizard and SQL), display of report data in tabular or chart format, and linking of reports. Whereas Dashboards mode enables you to view and run existing reports, Reports mode allows you to view, create, edit, and run reports.

Reports mode also facilitates management of existing reports, which includes organizing report definitions in folders and viewing information about each definition, such as the disk space it consumes and the number of times it has been run.

Permissions determine what reports and data a user can view and manage. For example, users with permission to access data in only certain reports or certain columns can see only their permitted subset of reports or data. Users are further restricted as to whether they can edit reports or only view or run them.

For more information, see:

- Chapter 3: Running, Viewing, and Managing Reports
- Chapter 4: Creating and Editing Wizard Reports
- Chapter 5: Creating and Editing SQL Reports

- Administration

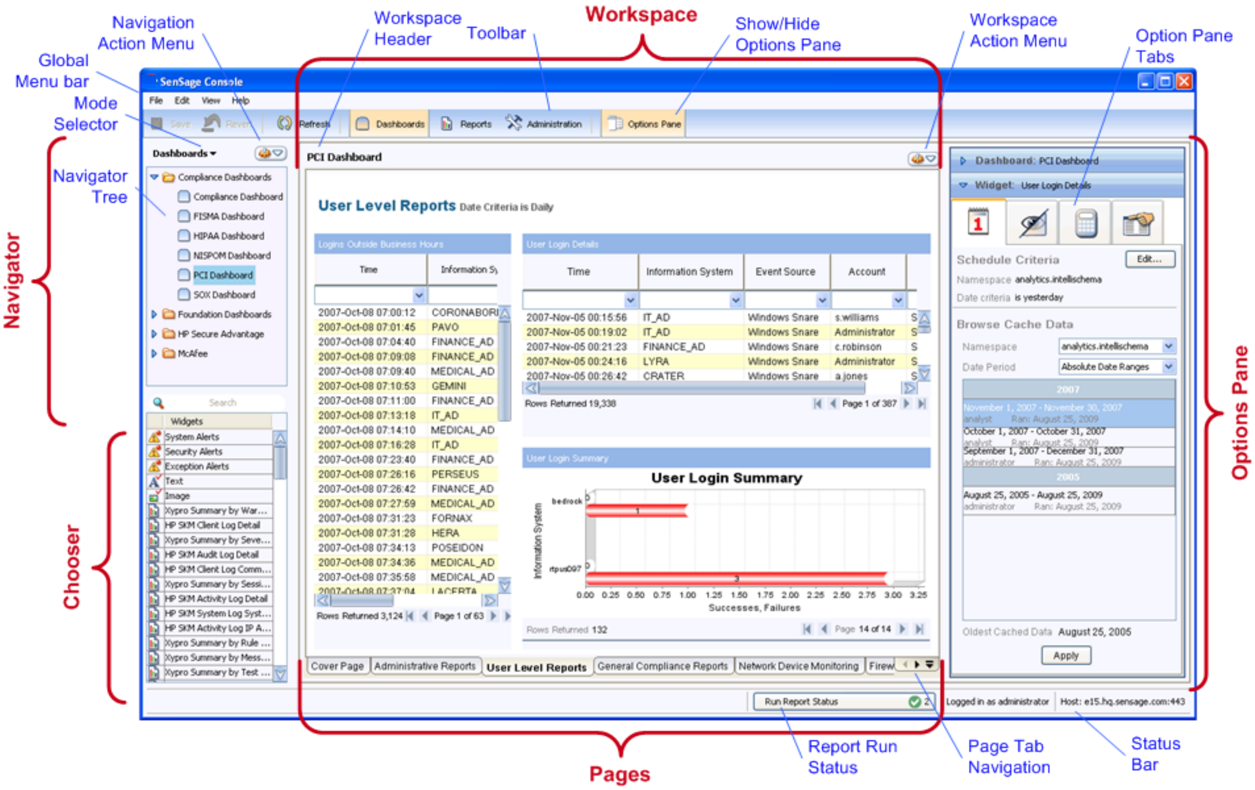
  Administration Mode enables administrators and those with the necessary permissions to create and modify schedules for reports and dashboards, create and manage users and roles, system and security assets, libraries, HawkEye AP rules, Distribution filters, and associate reports to security alerts.

  For more information, see:

  - Chapter 7: Creating and Editing Schedules
  - “Associating a Report to a Security Alert”, on page 164
  - “Creating Alerting Rules from Templates”, on page 247
  - Users, Roles, and Permissions in Chapter 8, “Administering Users and Authentication” in the *Administration Guide*
  - Creating Security Assets in Chapter 10, “Administering Assets and Monitoring Alerts” in the *Administration Guide*
  - Command Line: Creating and Managing Users, Roles, and Permissions in Chapter 8, “Administering Users and Authentication” in the *Administration Guide*.

Figure 1-3 illustrates the layout of HawkEye AP Console. Although the figure illustrates Dashboards mode, most of these functions are available in all three modes.

Figure 1-3.: Common Interface Components



Time	Information Sy
2007-Oct-08 07:00:12	CORONABORI
2007-Oct-08 07:01:45	PAVO
2007-Oct-08 07:04:40	FINANCE_AD
2007-Oct-08 07:09:08	FINANCE_AD
2007-Oct-08 07:09:40	MEDICAL_AD
2007-Oct-08 07:10:53	GEMINI
2007-Oct-08 07:11:00	FINANCE_AD
2007-Oct-08 07:13:18	IT_AD
2007-Oct-08 07:14:10	MEDICAL_AD
2007-Oct-08 07:16:28	IT_AD
2007-Oct-08 07:23:40	FINANCE_AD
2007-Oct-08 07:26:16	PERSEUS
2007-Oct-08 07:26:42	FINANCE_AD
2007-Oct-08 07:27:59	MEDICAL_AD
2007-Oct-08 07:31:23	FORNAX
2007-Oct-08 07:31:28	HERA
2007-Oct-08 07:34:13	POSEIDON
2007-Oct-08 07:34:36	MEDICAL_AD
2007-Oct-08 07:35:58	MEDICAL_AD
2007-Oct-08 07:37:04	LACERTA

Rows Returned 3,124 Page 1 of 63

Time	Information System	Event Source	Account	
2007-Nov-05 00:15:56	IT_AD	Windows Snare	s.williams	S
2007-Nov-05 00:19:02	IT_AD	Windows Snare	Administrator	S
2007-Nov-05 00:21:23	FINANCE_AD	Windows Snare	c.robinson	S
2007-Nov-05 00:24:16	LYRA	Windows Snare	Administrator	S
2007-Nov-05 00:26:42	CRATER	Windows Snare	a.jones	S

Rows Returned 19,338 Page 1 of 387

Rows Returned 132 Page 14 of 14

Schedule Criteria

Namespace analytics.intellischema

Date criteria is yesterday

Browse Cache Data

Namespace analytics.intellischema

Date Period Absolute Date Ranges

2007

November 1, 2007 - November 30, 2007
analyst Ran: August 25, 2009

October 1, 2007 - October 31, 2007
analyst Ran: August 25, 2009

September 1, 2007 - December 31, 2007
administrator Ran: August 25, 2009

2005

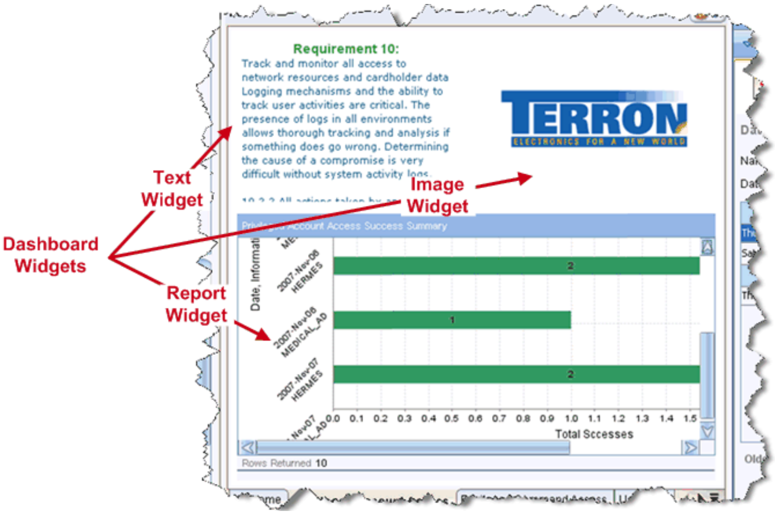
August 25, 2005 - August 25, 2009
administrator Ran: August 25, 2009

Oldest Cached Data August 25, 2005

Run Report Status 2

Figure 1-4 illustrates a section of another workspace. This workspace displays three widgets of different types: a report formatted as a chart, an image that displays the example company logo, and a text widget that displays relevant compliance text.

Figure 1-4: Three Types of Dashboard Widgets



Requirement 10:
Track and monitor all access to network resources and cardholder data Logging mechanisms and the ability to track user activities are critical. The presence of logs in all environments allows thorough tracking and analysis if something does go wrong. Determining the cause of a compromise is very difficult without system activity logs.

Rows Returned 10

The following sections describe the common interface components:

- “Global Menu Bar”, next
- “Global Tool Bar”, on page 30
- “Navigator”, on page 31
- “Workspace”, on page 31
- “Chooser”, on page 33
- “Options Pane”, on page 35
- “Status Bar”, on page 35

Global Menu Bar

HawkEye AP Console provides the following menus regardless of mode:

- **File**
  - Save
  - Revert to Saved
  - Exit from HawkEye AP Console
- **Edit**
  - Change Password
  - Set preferences

    For more information, see “Setting Preferences”, on page 35.
- **View**
  - Change mode to Dashboard, Reports, or Administration
  - Hide or show the Options Pane, Tool Bar, and Status Bar
  - Refresh the current display
- **Help**
  - Open HawkEye AP product documentation
  - Obtain build information about HawkEye AP Console; this information is useful for when talking to Hexis Cyber Solutions Technical Support

Global Tool Bar

HawkEye AP Console provides icons for performing the following operations:

- Save
- Revert
- Refresh
- Change mode to Dashboards, Reports, or Administration
- Hide or show the Options Pane

Navigator

The Navigator facilitates movement among the three modes and the items specific to each mode. It comprises:

- **Mode Selector**—facilitates switching among the three modes, which display different items in the Navigator Tree.
- **Navigator Tree**—displays all items available in the current mode. For Dashboards and Reports modes, this tree contains a hierarchy of folders that organize the dashboards and reports. For Administration mode, this tree organizes such items as schedules, libraries, and assets.
- **Navigator Action Menu**—provides operations specific to Dashboards and Reports mode. For example:
  - Dashboards mode—provides options to create a new dashboard or folder.
  - Reports mode—provides options to create a new report definition or folder.

  NOTE: Administration mode does not have a Navigator Action Menu because there are no actions that would apply to the Navigator. In Dashboard and Reports mode, the Action Menu creates items that are accessed from the Navigator. For example, in these modes you can create and remove folders. In the Administration mode, the items listed in the Navigator are fixed. You cannot modify or perform actions on these items.

Workspace

The Workspace provides the primary area in which you view and manage data. The Workspace comprises:

- **Header**

  The workspace header identifies your current location in the HawkEye AP Console. Header values vary according to your current mode. Figure 1-3 illustrates a dashboard workspace whose header identifies “PCI” as the name of the current dashboard.

- **Workspace Action Menu**

  The Workspace Action Menu provides operations specific to the current mode and selected item.

- **Workspace Pane**

The primary workspace component is the pane that displays one or more pages, each of which contains items to be viewed or manipulated and the tools for creating and managing these items. Workspace items and actions vary according to your current mode.

- **Pages**

  In Dashboards mode, you can create multiple pages to organize reports, alerts, and other items by function, usage, or user access. In Reports and Administration mode, pages are created automatically as you open different report, schedule, or library definitions for editing. All modes display at least one page by default.

  The bottom of the workspace displays tabs with the name of each page. If there are more tabs than can fit within the display, a page-tab navigation tool displays that facilitates moving among the pages. You can also change the relative position of a page. Additionally, in Dashboards mode, you can add and rename pages; for more information, see "Creating a Page", on page 213.

Page Navigation

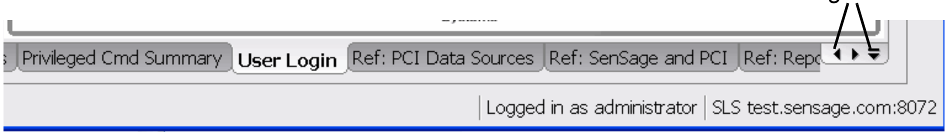
All three modes can display multiple pages. This section describes how to find a page, change focus to a page, and change the relative position of a page.

Changing Focus to a Page

To open a page whose name is hidden among a set of page tabs, you must first display its page tab. To do so, you can either use the Page-Tab Scroll icons to move page-by-page to the right or left or use the Tab List icon to move to a specific page from a list of all page names. Figure 1-5 illustrates the Page-Tab Scroll icons and Figure 1-6 illustrates the Tab List icon.

Figure 1-5: Scrolling to a Page

Click to scroll right or left.



To scroll page-by-page

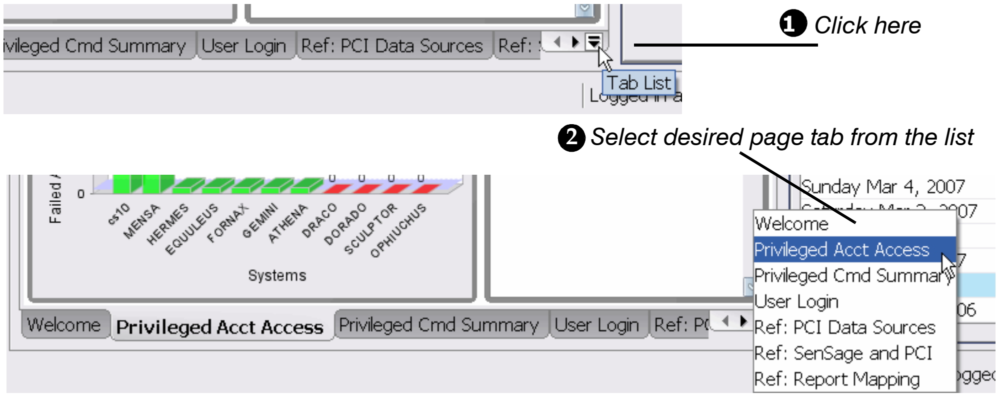
Click the right or left scroll icon.

To see available pages and move to a specific page by name

1 Click the **Tab List** icon.

2 Select the desired page name from the popup, as illustrated in Figure 1-6.

Figure 1-6: Moving to a Page by Name


❶ *Click here*

❷ *Select desired page tab from the list*

Changing the Relative Position of a Page

To change the position of a page

Select the page tab, drag it to the desired position and release the mouse.

Chooser

The Chooser provides a list of items and the ability to sort them and search them in the list. The types of items change as you change modes. For example, in Dashboards mode, the Chooser lists all widgets that you can drag and drop into the Dashboard.

The types of Chooser items determine what you can view and manipulate in the workspace.

- **Dashboards mode**—The items are system, exception report, and security alert widgets, report widgets, and text and image widgets.
- **Reports**—The items include report definitions as well as tables and views.
- **Administration**—The items depend on the activity you are performing. For example, when you create or edit a schedule, the items include report definitions, dashboards, and folders of reports and dashboards.

The Chooser comprises a search field and a scroll list. You can use the search field to limit the items that display in the scroll list. When you create a Wizard report or a schedule, the search field also includes a scope dropdown that limits item to a specific type. In addition to limiting display to only the desired items, you can use the Chooser sort option to change the order of display.

Limiting Display by Scope

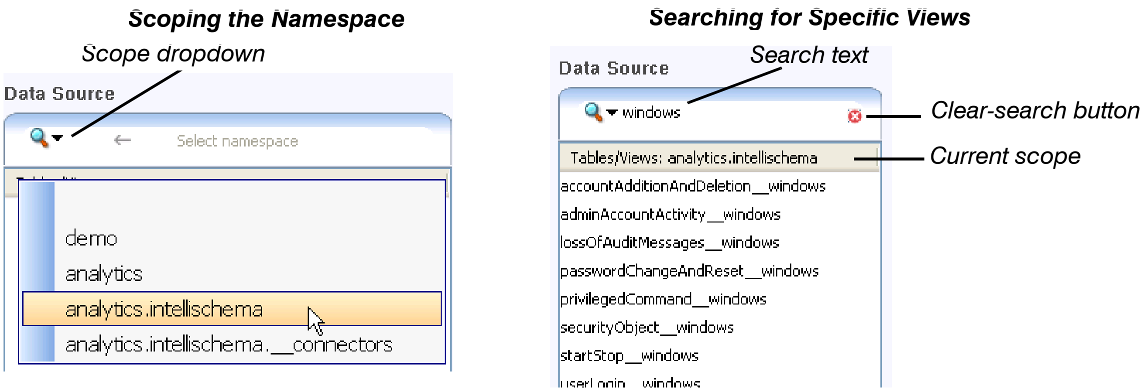
Scope enables you to limit items by group if more than one group is available. The Chooser provides scope during Wizard Report creation, when it enables you to scope the report namespace. HawkEye AP uses *namespaces* to organize EDW data objects (tables and views). For more information, see “Namespaces: Using a Single Report or Dashboard to Access Different Data”, on page 43. Figure 1-7 illustrates the Wizard Report Chooser.

Limiting Display by Text Search

Text search enables you to limit items by the characters in their names. Figure 1-7 illustrates the Chooser during Wizard Report creation. If you have access to more than one namespace, you must first select the namespace whose table or view will provide the data for your report.

In the graphic below, the user first selects the `analytics.intellischema` namespace. Next the user limits Chooser display to only views that contain the text "windows" in the name.

Figure 1-7: Example of Scope and Search Text in Wizard Report Creation


Scoping the Namespace
Scope dropdown
Data Source
Select namespace
demo
analytics
analytics.intellischema
analytics.intellischema.__connectors
Searching for Specific Views
Search text
Data Source
windows
Clear-search button
Tables/Views: analytics.intellischema
Current scope
accountAdditionAndDeletion__windows
adminAccountActivity__windows
lossOfAuditMessages__windows
passwordChangeAndReset__windows
privilegedCommand__windows
securityObject__windows
startStop__windows
userLogin__windows

NOTE: The Clear-search button displays only after you enter search text.

By default, all items within the selected scope display in the scroll list. As you enter values in the search field, the set of displayed items reduces to only those whose names contain the current search text.

NOTE: Names of Chooser items match even if they do not begin with the search text. The search text can appear anywhere in the name. For example, assume the Chooser displays the following reports:

- SOX
- Southern Central Firewall
- OSO
- Ontario

If the user enters "`o`" as the search text, none of the reports are removed from display because all names contain an "o". However, if the user enters "`so`" as the search text, the "Ontario" report no longer displays. Moreover, if the user enters “`sox`”, only the "SOX" report displays.

SORTING ITEMS

The Chooser enables ascending and descending sort by item name in all modes. However, when you view widgets in the Dashboards Chooser or items to be scheduled in Administration mode, the Chooser displays a second column that allows you to sort by item type.

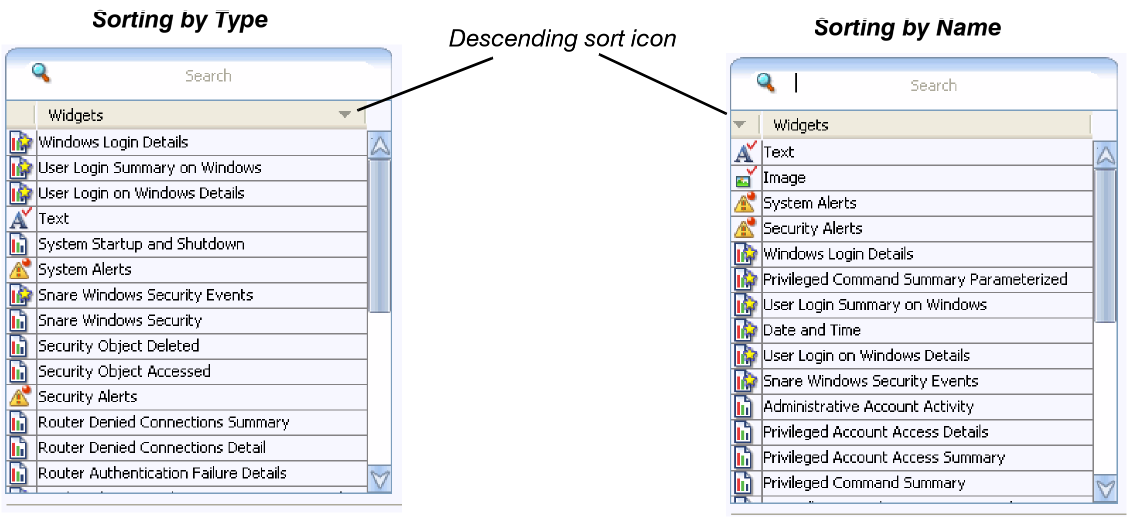
Because Reports mode displays only one type of item at a time, the Chooser scroll list contains only one column that identifies either the report definition or the table/view by name.

In Dashboards mode and when assigning items to a schedule, the Chooser displays the following columns:

- icons that identify the type of widget or item—primary sort defaults to this column
- text that identifies the name of the widget or item

Figure 1-8 illustrates both types of sorts as used in Dashboards mode.

Figure 1-8: Sorting Items in the Dashboard Chooser



Windows Login Details
User Login Summary on Windows
User Login on Windows Details
Text
System Startup and Shutdown
System Alerts
Snare Windows Security Events
Snare Windows Security
Security Object Deleted
Security Object Accessed
Security Alerts
Router Denied Connections Summary
Router Denied Connection Detail
Router Authentication Failure Details

Text
Image
System Alerts
Security Alerts
Windows Login Details
Privileged Command Summary Parameterized
User Login Summary on Windows
Date and Time
User Login on Windows Details
Snare Windows Security Events
Administrative Account Activity
Privileged Account Access Details
Privileged Account Access Summary
Privileged Command Summary

NOTE: For all three modes:

- Sorting is case insensitive.
- The sort icons do not display in the column header until you click in the header.

Although Reports mode enables sorting only by name and not also by widget type, the sorting described below pertains to it as well.

To sort widgets

1 Click in the column header to display the sort icon; if there are two columns, click in the desired column header.

2 Click the sort icon a second time to reverse the order of the sort.

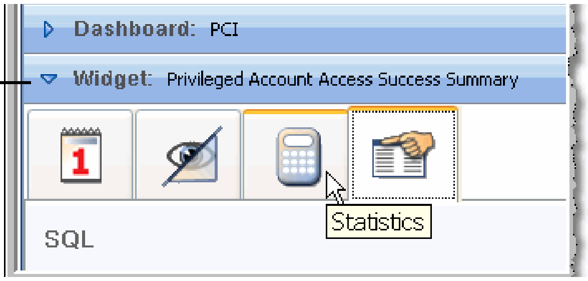
Options Pane

The Options Pane exposes all context-sensitive operations that you can apply against the items in your workspace. The operations differ by mode and by the type of item selected. You can hide and show the Options Pane to maximize your workspace area or to apply options.

To open or close the Options Pane, either click the Options Pane icon Options Pane above the workspace or select **Show Options** from the Workspace Action Menu.

Some item types display multiple icon tabs. For example, in Dashboards mode, the dashboard as well as its report widgets display a set of icon tabs. Figure 1-9 illustrates the tabs for a report widget. Currently open to the SQL tab, this widget also provides tabs for Date Options, Show/Hide columns, and Statistics. In the graphic below, the user is about to click the Statistics tab.

Figure 1-9: Icon Tabs in Options Pane



Dashboard options are closed.

Report widget options are opened.

Status Bar

As illustrated in Figure 1-3, the status bar in Dashboards mode displays the status of running reports, the name of the user currently logged into HawkEye AP Console, and the current EDW instance. In Reports mode, the status bar also displays the total disk space consumed by the full set of cached reports and the total number of report definitions. The status bar in Administration mode displays the name of the user currently logged into HawkEye AP Console and the current EDW instance.

Setting Preferences

From the **Edit** menu, you can set preferences. Click **Preferences …** to display the **Preferences** dialog. There are three tabs in this dialog:

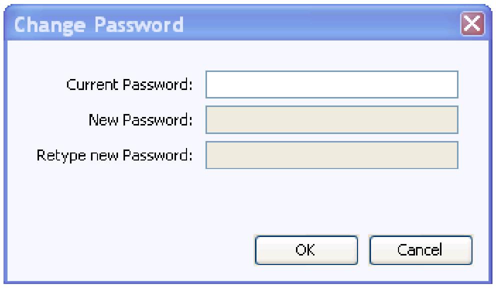
- **General**
  - Reports
    - *Set the maximum number of rows to display on each page for cached reports*
    - *Set whether the Report Wizard in Reports Mode displays both tables and views or only views*
  - Miscellaneous
    - *Specify whether the week begins on* `Sunday` *or* `Monday`
    - *Set smooth fonts to enhance display of text*
    - *Specify whether HawkEye AP Console speeds dashboard display by pre-loading pages in the currently open dashboard*

      To facilitate opening each page in the current dashboard, specify how long HawkEye AP Console should wait for you to press a key or move the mouse in HawkEye AP Console before it begins loading the next page in the open dashboard. Keep this option set to `0` (zero) to disable the feature.
- **Time Zone**
  - Select the time zones you want to display in date and time dropdowns
- **Export**
  - Specify encoding and the separator character for CSV-formatted reports

Changing Your Password

From the **Edit** menu, you can change your password. Click **Change Password …** to display the dialog box below. Note that if you're an administrator, the system ignores what you enter for Current Password - whatever you enter makes no difference.

Figure 1-10: Change Password Dialog



Change Password

Current Password:

New Password:

Retype new Password:

Accessing Online Help

HawkEye AP Console online help opens in a separate browser window, which organizes the documentation by users. When you open a document in HTML format, a new window displays with the selected document open for viewing. This window also provides links to each document in the HawkEye AP documentation set and provides an index and search capabilities. When you display the documentation in HTML format, you can search the entire documentation set or only a specified document.

After finding the information you need, you can close online help by clicking the Close Window icon located in the Title Bar.

To access online help

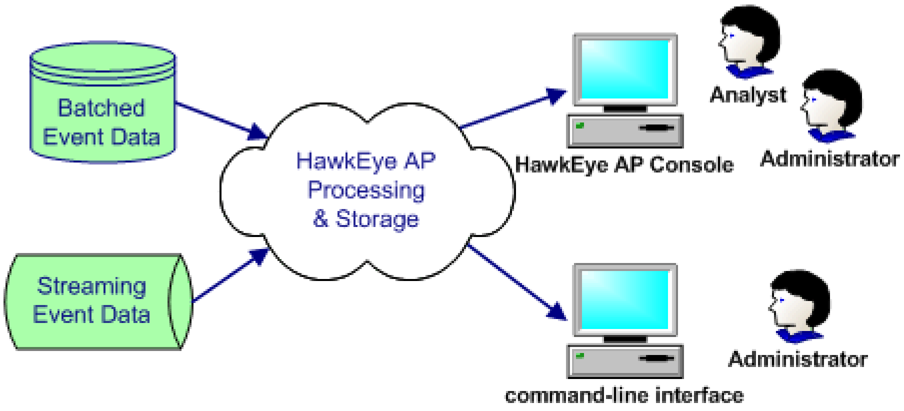
From the menu in any HawkEye AP Console window, select **Help** > **Contents**.

HOW DOES YOUR DATA GET TO HAWKEYE AP CONSOLE?

The HawkEye AP system supports the industry's most comprehensive real-time and historical event management capabilities. Its patented data model and compression technology facilitate online collection and storage of massive volumes of data across an entire network. The HawkEye AP architecture enables rapid querying of and visibility into security threats and facilitates drilldown capability.

Figure 1-11 illustrates data flow at a very high level. The HawkEye AP system receives both streaming and batched event data, processes and stores it, and, when appropriate sends alerts to the HawkEye AP Console.

Figure 1-11: High-Level Event-Data Collection, Processing, and Display



As illustrated in Figure 1-11, events enter the Sensage system from external systems, such as network devices and software applications. Events enter in one of two ways:

- HawkEye AP**Batched**—events are collected at predefined intervals from log files and other event repositories maintained by network devices, software applications, and operating systems. HawkEye AP components poll a data source or repository to retrieve event data and load the data into the Event Data Warehouse (EDW). The EDW makes the event data available to the Application Manager for report and alert management.

- **Streaming**—events flow into the HawkEye AP system as a real-time stream from network devices and software applications that publish the events. HawkEye AP components process the events and raise real-time alerts when the incoming data matches pre-defined criteria.

Analysts can access security alerts and reports through HawkEye AP Console, which provides a powerful graphic interface. Administrators can access EDW data either by using command-line interface (CLI) utilities on a Linux system or by viewing reports and alerts through HawkEye AP Console.

For more detailed information about HawkEye AP architecture and the specific HawkEye AP components that receive or collect data and process it, see Chapter 1: Introduction in the *Administration Guide*.

REPORTING AND HAWKEYE AP ANALYTICS

HawkEye AP Analytics provides a comprehensive solution for reporting on event data collected from all sources in your enterprise and stored in the HawkEye AP Event Data Warehouse (EDW). Without understanding all their information systems and devices, business analysts can use the Analytics reports provided with the HawkEye AP distribution to quickly retrieve and correlate data from multiple log sources. In addition, without understanding the underlying raw data format, analysts can use the IntelliSchema views provided with the HawkEye AP distribution to easily create reports against multiple log sources.

For more information:

- **Viewing reports**—Chapter 2: Using Dashboards
- **Creating reports**—Chapter 4: Creating and Editing Wizard Reports
- **Analytics Guide—**Overview of HawkEye AP Analytics in Chapter 1, "Overview of HawkEye AP Analytics" in the *Analytics Guide*

QUERYING AND SENSAGE SQL

The Event Data Warehouse (EDW) is a database built for and dedicated to loading, storing, and analyzing event data. The HawkEye EDW provides Sensage SQL (SSQL), its own version of SQL, SQL functions, and Perl functions to query event-log entries. A query is a Sensage SQL statement that extracts event-log data from the EDW data store. The EDW also provides utilities to load event-log entries and manage the operation of the EDW.

Typically, analysts use the Report Wizard to create reports against HawkEye AP data. Wizard reports often meet most of your reporting needs. However there are situations that demand more complexity than the queries generated by the Report Wizard allow. For example, you may a need a report to perform the following operations:

- include data from external files
- manipulate lists of data
- concatenate and truncate string data
- nest queries so that the results of one query are used as the input to the next
- use libraries and Perl subroutines

To create a report that takes advantage of HawkEye AP extensions to Sensage SQL, an advanced user must create a SQL report. To create a SQL report requires entering a SQL query directly into the report definition window. The user who creates such a report must understand the SQL query language and the HawkEye AP extensions to the language.

Because this guide documents report creation, it documents those features of Sensage SQL required to write a query against stored data. For more information, see the following chapters in this guide:

- Chapter 10: Sensage SQL
- Chapter 11: SQL Functions
- Chapter 12: Perl Subroutines

For more information on those features of Sensage SQL required to load and store data, see the following chapters in the *Administration Guide*:

- Chapter 3: Loading, Querying, and Managing the EDW
- Chapter 2: Configuring and Managing HawkEye AP

CHAPTER 2

Using Dashboards

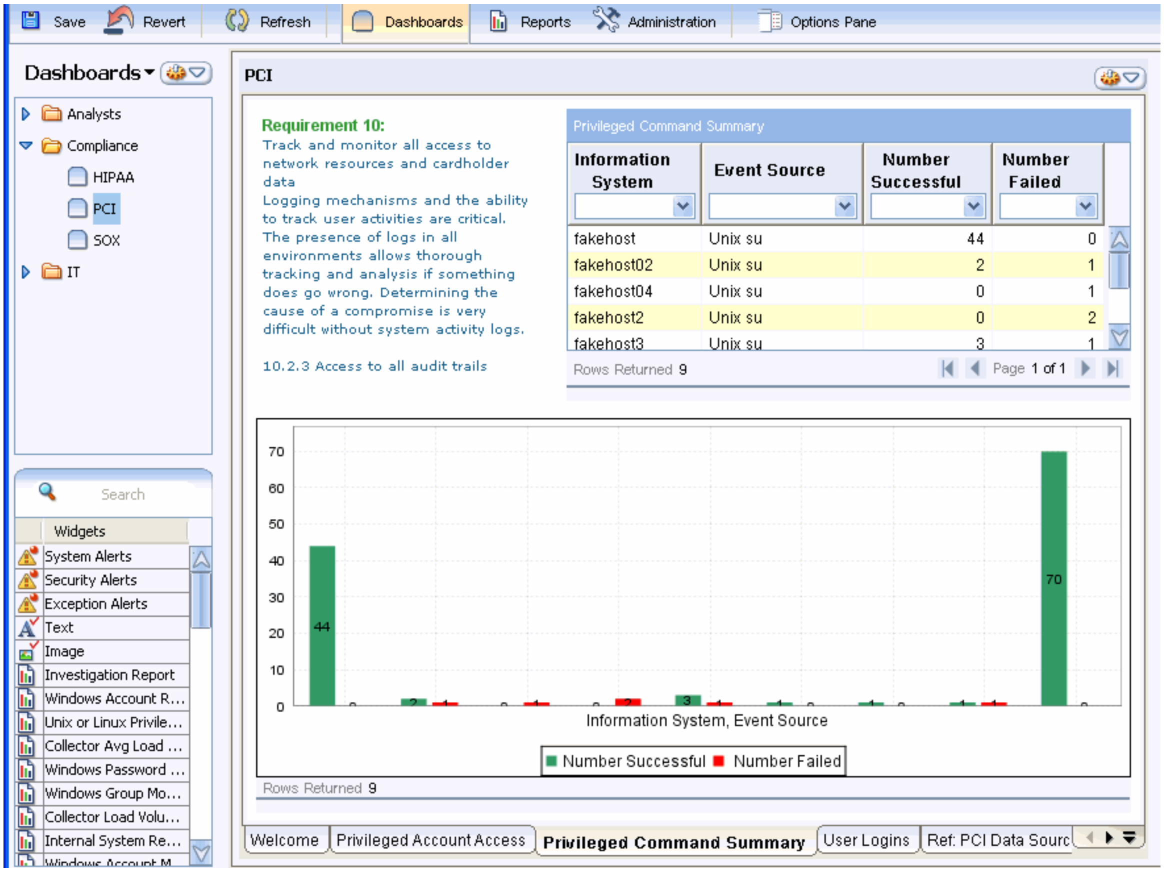
This chapter contains the following sections:

OVERVIEW

Dashboards provide clean and powerful access to event and system data that HawkEye AP collects. Report and alert widgets present and illustrate the data. Typically a dashboard contains multiple pages, each of which groups report widgets by relevance or audience. For those who monitor correlated alerts about enterprise security data and HawkEye AP system assets, a dashboard can also include a page or pages of alert widgets. A dashboard typically also contains text and graphics that complement or explain the displayed data.

Figure 2-1 illustrates the Privileged Command Summary page of the PCI dashboard. As shown in the list of folders on the left, PCI is one of the Compliance dashboards. There are also several dashboards in the IT folder, and, as indicated by the icon to the left of Analysts folder, there are also dashboards specific to analysts. To document dashboard usage, this chapter examines the example PCI dashboard in the Compliance folder and the two dashboards in the IT folder.

Figure 2-1: Example Dashboards



Requirement 10:
Track and monitor all access to network resources and cardholder data
Logging mechanisms and the ability to track user activities are critical. The presence of logs in all environments allows thorough tracking and analysis if something does go wrong. Determining the cause of a compromise is very difficult without system activity logs.

10.2.3 Access to all audit trails

Privileged Command Summary

Information System	Event Source	Number Successful	Number Failed
fakehost	Unix su	44	0
fakehost02	Unix su	2	1
fakehost04	Unix su	0	1
fakehost2	Unix su	0	2
fakehost3	Unix su	3	1

Rows Returned 9 Page 1 of 1

Rows Returned 9

For information about HawkEye AP Console components common to all three modes and how to use these components, see “Introduction to the Interface”, on page 27.

VIEWING REPORTS

This section contains the following topics:

- “About HawkEye AP Reports”, next
- “Cached Data: Making Stored Data Quickly Available”, on page 43
- “Namespaces: Using a Single Report or Dashboard to Access Different Data”, on page 43
- “Viewing and Changing Display of Report Data and Metadata”, on page 44
- “Manipulating and Expanding Report Results”, on page 51

About HawkEye AP Reports

A HawkEye AP report organizes the raw event data collected by your HawkEye AP system and presents the data in tabular or chart format. HawkEye AP runs each report from a *report definition*, which defines the data source, the manipulation and ordering of data, and data display.

HawkEye AP Analytics provides many ready-to-run report definitions for common types of information systems and events. HawkEye AP Console provides tools that enable you to create and edit your own report definitions and to edit the Analytics reports. You can develop a report graphically through the *Report Wizard*, which steps you through report creation. Alternately, you can enter HawkEye AP SQL statements directly into a query window and then graphically manipulate report display and variables. HawkEye AP SQL is a variation of standard Structured Query Language (SQL) that has been extended for manipulating event data.

After a report definition has been created, you can run the report manually or by a schedule. Because running a report can be time consuming, the HawkEye AP system saves the output of a report each time it runs as a report *cache entry* and makes it available for quick viewing.

Cached Data: Making Stored Data Quickly Available

As illustrated in Figure 2-1, report data can display in table or chart format. The two reports illustrated above are actually the same report displayed in two formats. The data in both of these widgets is identical in that it represents the results of the same query run over the same time period and using the same interval (such as day or month or year).

Whenever a report is run, the Application Manager automatically saves the results in a *report cache*. Each report cache contains data stored in the EDW for a specific time range and interval. The saved data precludes the need to query the EDW repeatedly for the same result set. A report cache improves access time, particularly if the query computes or otherwise manipulates the data.

HawkEye AP reports always display data from a report cache. You typically run report definitions on the same data set over multiple time periods and intervals. For example, to facilitate tracking login anomalies, your site might track login data on a daily, weekly, and monthly basis. If you discover an anomaly in today's daily report, you can easily view cache entries for the previous weeks and months to investigate the scope of today's issue. If a report cache is not available to meet your reporting needs, you can manually run the report to save the required event data in the cache. By default, the most recent report cache entry displays. For information on how you can view a specific cache entry for a report widget, see "Viewing and Changing the Time Range and Namespace", on page 51.

If a report runs frequently, such as every 15 or 30 minutes, the data you view in a report widget may not be the most current data. A scheduled run of the report may have generated new cache entries since you logged into the dashboard. In other words, although the most recent cache displayed when you logged into the dashboard, a more recent cache may exist. To enable you to view the most recent cache, HawkEye AP Console provides a Refresh option. Because a page or dashboard can contain multiple report widgets, several of which might have a newer cache than is currently displayed, the Refresh option enables you to specify its scope: you can refresh a single report or all report widgets on the page or all reports on the dashboard. For more information, see "Refreshing Dashboards and Running Items", on page 85.

Namespaces: Using a Single Report or Dashboard to Access Different Data

HawkEye AP organizes tables and views into *namespaces*. A namespace is like a file-system directory or folder, but it contains tables and views rather than files or documents. Namespaces also behave like file system folders in the following ways:

- Access to a namespace is determined by Namespace permissions associated with roles.

Just as access to financial information on a file system is limited only to employees with appropriate permissions, so too is access to a namespace determined by HawkEye AP-granted permissions.

- The names of the tables and views within a namespace are unique to the namespace.

  Assume you store tax information on your file system in folders named for the relevant year. For example, assume you have created folders named `2006` and `2007`. Assume further that you stored a file named `Taxes` below each of these folders. Although both folders contain an identically named file, the Taxes file is unique to the folder that contains it.

Just as you can create identically named files below different file-system folders, so too can your HawkEye AP system store identically named tables and views below different namespaces. For example, your system might store a table named `hosts` in a namespace named `Eastern` and also in a namespace named `Western`. The data in the `hosts` table is unique to the namespace that contains it.

Namespaces allow HawkEye AP administrators to create identically named tables with identical structures that store different data depending on their location. For example, the `hosts` table contains information about your computer hosts on either the east or west coast. Additionally, because access to a namespace is determined by permissions, some users might have access only to east coast host data while others have access only to west coast data. Only the IT department has access to all data.

To further simplify data access, you can create a single report definition that retrieves host information from either the east or west coast namespace. The data the report returns depends upon which namespace it runs against. Furthermore, a dashboard can be defined that contains reports that run against identically named and structured tables in different namespaces.

Viewing and Changing Display of Report Data and Metadata

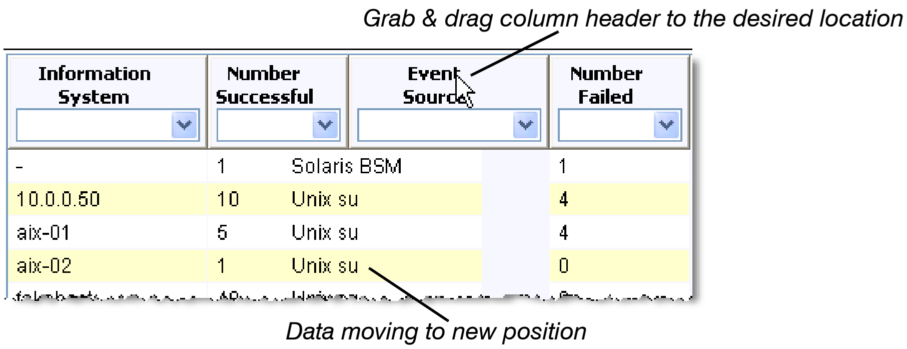
This section contains the following topics:

- “Changing Column Order”, next
- “Widening Column Display in a Report”, on page 45
- “Filtering and Sorting Report Data”, on page 46
- “Showing and Hiding Report Columns and Metadata”, on page 48
- “Changing Between Table and Chart Formats”, on page 49
- “Viewing the SQL Query and Other Properties”, on page 50

Changing Column Order

Figure 2-2 illustrates the process to change column order.

Figure 2-2: Changing Column Order: Illustrating the Move



Grab & drag column header to the desired location

Information System	Number Successful	Event Source	Number Failed
-	1	Solaris BSM	1
10.0.0.50	10	Unix su	4
aix-01	5	Unix su	4
aix-02	1	Unix su	0

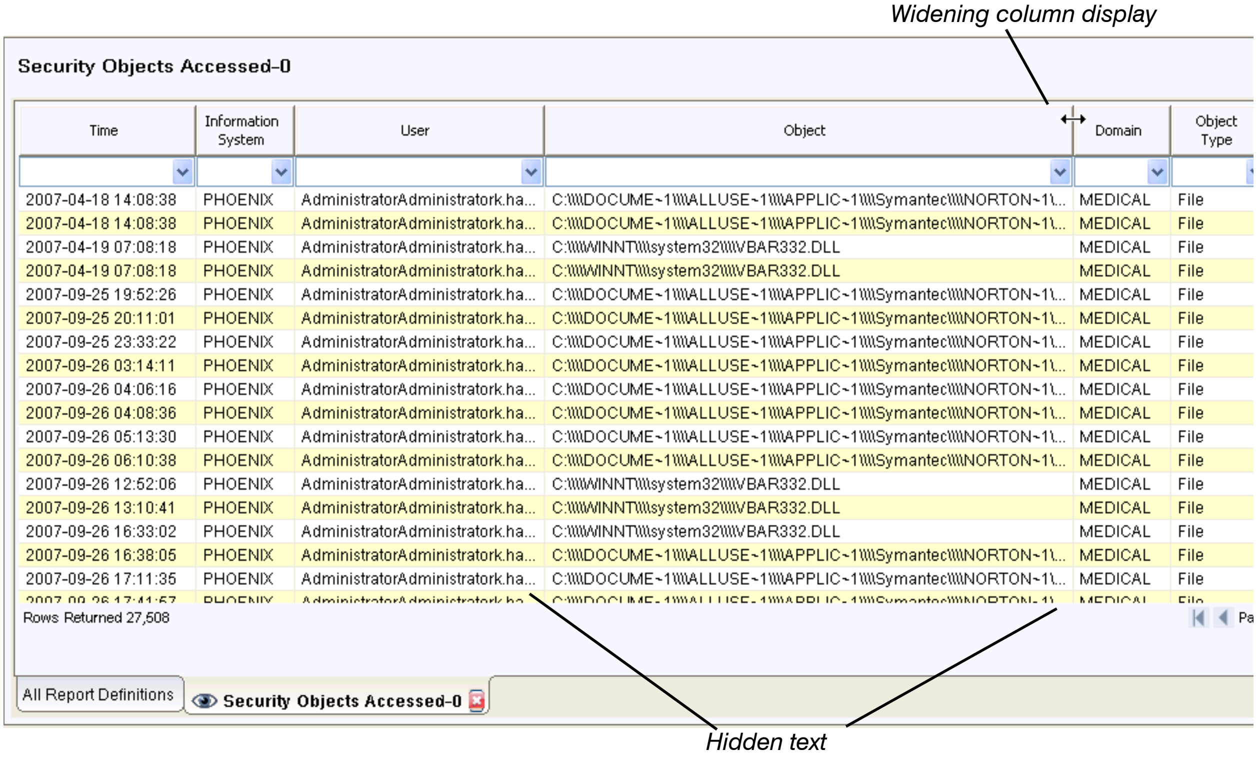
Data moving to new position

The graphic above illustrates the process to reorder two columns. The data from the **Event Source** column is moving into the new position but has not fully arrived yet.

Widening Column Display in a Report

Sometimes a report has more columns than can fully display. When column width is so narrow that some header text or column data is hidden, the report indicates hidden text with three dots, as illustrated in Figure 2-3. This figure also illustrates the sizing cursor, which allows you to expand the width of a column.

Figure 2-3: Widening Columns



Widening column display

Security Objects Accessed-0

Time	Information System	User	Object	Domain	Object Type
2007-04-18 14:08:38	PHOENIX	AdministratorAdministratork.ha...	C:\\\\DOCUME~1\\\\ALLUSE~1\\\\APPLIC~1\\\\Symantec\\\\NORTON~1\...	MEDICAL	File
2007-04-18 14:08:38	PHOENIX	AdministratorAdministratork.ha...	C:\\\\DOCUME~1\\\\ALLUSE~1\\\\APPLIC~1\\\\Symantec\\\\NORTON~1\...	MEDICAL	File
2007-04-19 07:08:18	PHOENIX	AdministratorAdministratork.ha...	C:\\\\WINNT\\\\system32\\\\VBAR332.DLL	MEDICAL	File
2007-04-19 07:08:18	PHOENIX	AdministratorAdministratork.ha...	C:\\\\WINNT\\\\system32\\\\VBAR332.DLL	MEDICAL	File
2007-09-25 19:52:26	PHOENIX	AdministratorAdministratork.ha...	C:\\\\DOCUME~1\\\\ALLUSE~1\\\\APPLIC~1\\\\Symantec\\\\NORTON~1\...	MEDICAL	File
2007-09-25 20:11:01	PHOENIX	AdministratorAdministratork.ha...	C:\\\\DOCUME~1\\\\ALLUSE~1\\\\APPLIC~1\\\\Symantec\\\\NORTON~1\...	MEDICAL	File
2007-09-25 23:33:22	PHOENIX	AdministratorAdministratork.ha...	C:\\\\DOCUME~1\\\\ALLUSE~1\\\\APPLIC~1\\\\Symantec\\\\NORTON~1\...	MEDICAL	File
2007-09-26 03:14:11	PHOENIX	AdministratorAdministratork.ha...	C:\\\\DOCUME~1\\\\ALLUSE~1\\\\APPLIC~1\\\\Symantec\\\\NORTON~1\...	MEDICAL	File
2007-09-26 04:06:16	PHOENIX	AdministratorAdministratork.ha...	C:\\\\DOCUME~1\\\\ALLUSE~1\\\\APPLIC~1\\\\Symantec\\\\NORTON~1\...	MEDICAL	File
2007-09-26 04:08:36	PHOENIX	AdministratorAdministratork.ha...	C:\\\\DOCUME~1\\\\ALLUSE~1\\\\APPLIC~1\\\\Symantec\\\\NORTON~1\...	MEDICAL	File
2007-09-26 05:13:30	PHOENIX	AdministratorAdministratork.ha...	C:\\\\DOCUME~1\\\\ALLUSE~1\\\\APPLIC~1\\\\Symantec\\\\NORTON~1\...	MEDICAL	File
2007-09-26 06:10:38	PHOENIX	AdministratorAdministratork.ha...	C:\\\\DOCUME~1\\\\ALLUSE~1\\\\APPLIC~1\\\\Symantec\\\\NORTON~1\...	MEDICAL	File
2007-09-26 12:52:06	PHOENIX	AdministratorAdministratork.ha...	C:\\\\WINNT\\\\system32\\\\VBAR332.DLL	MEDICAL	File
2007-09-26 13:10:41	PHOENIX	AdministratorAdministratork.ha...	C:\\\\WINNT\\\\system32\\\\VBAR332.DLL	MEDICAL	File
2007-09-26 16:33:02	PHOENIX	AdministratorAdministratork.ha...	C:\\\\WINNT\\\\system32\\\\VBAR332.DLL	MEDICAL	File
2007-09-26 16:38:05	PHOENIX	AdministratorAdministratork.ha...	C:\\\\DOCUME~1\\\\ALLUSE~1\\\\APPLIC~1\\\\Symantec\\\\NORTON~1\...	MEDICAL	File
2007-09-26 17:11:35	PHOENIX	AdministratorAdministratork.ha...	C:\\\\DOCUME~1\\\\ALLUSE~1\\\\APPLIC~1\\\\Symantec\\\\NORTON~1\...	MEDICAL	File

Rows Returned 27,508

Hidden text

NOTE:

- The maximum width for a column is 1000 pixels. The number of pixels does not correspond to an exact number of characters because HawkEye AP Console does not use fixed width fonts.
- The 1000-pixel limit for column width affects reports exported to PDF as well as reports that display in HawkEye AP Console.

To widen column display

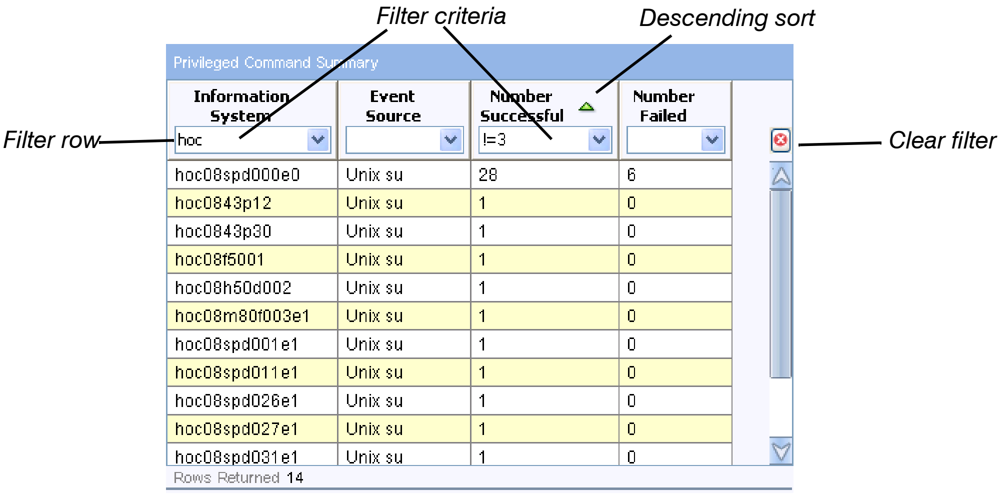
1. Position the cursor over the column header until the sizing cursor displays, as illustrated in Figure 2-3.
2. Click and drag the cursor until the column reaches the desired width.

Filtering and Sorting Report Data

By default, tabular reports contain all rows of the returned data set. You can filter the report to display only the rows that interest you. The tabular report widget displays a filter row between the column headers and the rows of data. Use this row to specify your filtering criteria.

For example, Figure 2-4 illustrates a report that is filtered on the first and third columns. It also sorts on the third column.

Figure 2-4: Filtering Report Data



Filter criteria

Descending sort

Filter row

Clear filter

Privileged Command Summary

Information System	Event Source	Number Successful	Number Failed
hoc		!=3	
hoc08spd000e0	Unix su	28	6
hoc0843p12	Unix su	1	0
hoc0843p30	Unix su	1	0
hoc08f5001	Unix su	1	0
hoc08h50d002	Unix su	1	0
hoc08m80f003e1	Unix su	1	0
hoc08spd001e1	Unix su	1	0
hoc08spd011e1	Unix su	1	0
hoc08spd026e1	Unix su	1	0
hoc08spd027e1	Unix su	1	0
hoc08spd031e1	Unix su	1	0

Rows Returned 14

NOTE:

- After you enter the filter criteria, press ENTER to activate the filter.
- Data matches the filtering criteria if the filter text appears anywhere in the data value. As illustrated in Figure 2-4, only information systems that contain "`hoc`" in their name are displayed. For more information on search criteria, see "Limiting Display by Text Search", on page 33.

In addition to entering the search text, you can enter criteria by which to exclude data from the result set. You can modify the search by using the following operators:

Table 2-1: Filter Operators

Operator	Meaning	Examples
=	equal (equal sign can be omitted)	• abc (Contains abc. The equals sign and quotes may be omitted in this syntax) • =”abc” (equals abc) • 17 (equals 17) • =17 (equals 17) • =”” (empty)
<> or !=	not equal	• <>“abc” (does not equal abc) • !=“abc” (does not equal abc) • <>5 (does not equal 5) • !=5 (does not equal 5) • !=”” (not empty)
>	greater than	>15 (greater than 15)
>=	greater than or equal	>=15 (greater than or equal to 15)
<	less than	<15 (less than 15)
<=	less than or equal	<=15 (less than or equal to 15)
like	The LIKE comparison operator yields true if the value matches a particular pattern of characters. The matching pattern can contain explicit characters and special wildcard characters. The percent sign (%) is a wildcard that matches zero or more characters of any kind; an underscore matches any single character in a particular position within the pattern.	• like”%abc%” (contains abc) • like”abc%” (begins with abc) • like“%abc”X”%” (contains abc”X”)
not like	Negates the like comparison operators	not like “%abc%” (does not contain abc)

You can use the AND and OR keywords to filter the data using multiple criteria by referencing the current column name as $C. The following are valid examples of filtering using multiple criteria:

$C=”abc” OR $C=”def” (equals abc or def)

$C>5 AND $C<20 (greater than 5 and less than 20)

NOTE: In general, text values must appear within quotes and numeric values must not appear within quotes. The one exception is when you enter a text value to filter for data containing a given value, as shown in the first example in Table 2-1.

To filter tabular report data

1 In the filtering row, click the cell associated with the column whose values you want to limit.

2 Type your filter text in the cell, and press ENTER.

The report displays only those rows that contain the matched data.

3 To add filtering criteria for additional columns, repeat Step 1 and Step 2 for each column.

If you enter filter criteria for more than one column, all criteria is evaluated. In other words, the criteria is evaluated with the `AND` keyword.

To clear the filter

- To remove all filters, click the ⊗ icon.
- To remove a subset of filters, delete the text from the desired cell or cells in the filtering row and press ENTER.

To sort data in a column

- To sort a column, click its name in the column header row. The ascending icon ▼ displays next to the header name.
- To change sort order to descending, click the header a second time. The descending icon ▲ displays.
- To remove sorting, click the header a third time.

  NOTE: Sorting an column containing an IP address or hostname results in only in an alphabetical search and may not produce the expected result.

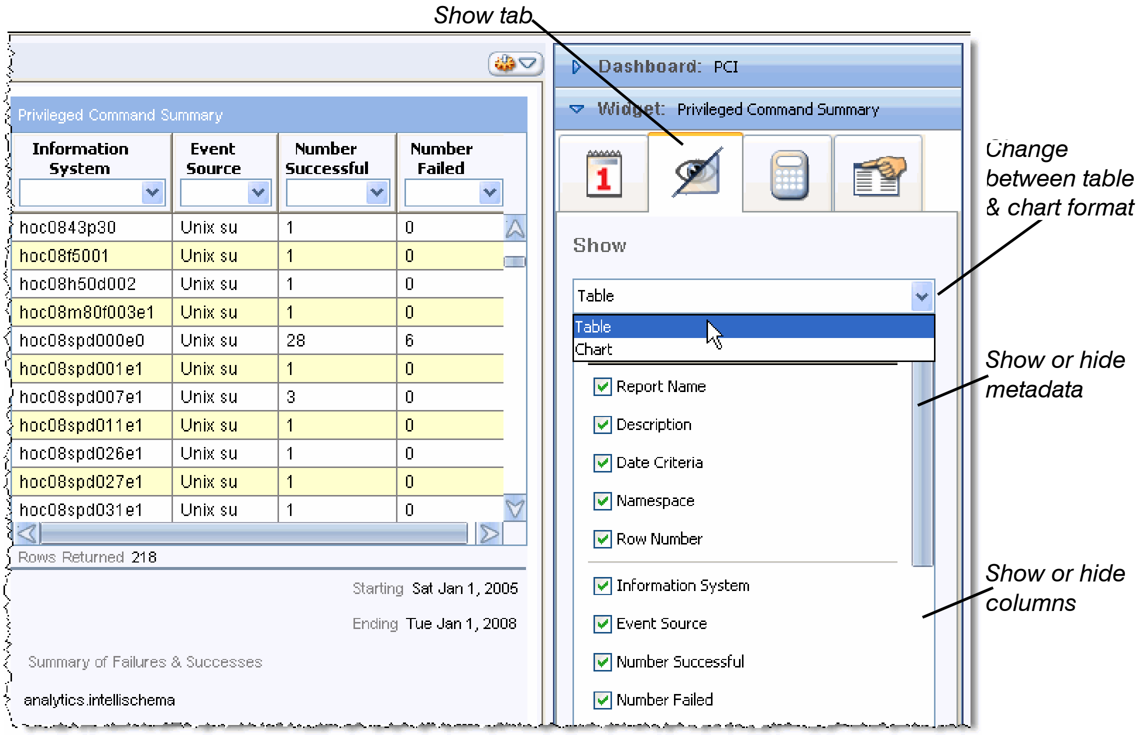
Showing and Hiding Report Columns and Metadata

If a tabular report displays columns whose data is not relevant to your investigation, you can hide those columns from display. Hiding unnecessary columns provides more space for the data that interests you. Figure 2-5 illustrates the **Show** tab, which allows you to remove specific columns from display.

In addition to displaying all columns in the result set, a report widget may display metadata. Metadata provides information about the report itself. This information includes a text description of the report, its namespace, and the date criteria that determines its time period.

From the **Show** tab you can also increase the data area by hiding information about the report, such as its description or namespace, and can toggle between table and chart format.

Figure 2-5: Showing and Hiding Tabular Report Information



Show tab

Change between table & chart format

Show or hide metadata

Show or hide columns

Dashboard: PCI

Widget: Privileged Command Summary

Privileged Command Summary

Information System	Event Source	Number Successful	Number Failed
hoc0843p30	Unix su	1	0
hoc08f5001	Unix su	1	0
hoc08h50d002	Unix su	1	0
hoc08m80f003e1	Unix su	1	0
hoc08spd000e0	Unix su	28	6
hoc08spd001e1	Unix su	1	0
hoc08spd007e1	Unix su	3	0
hoc08spd011e1	Unix su	1	0
hoc08spd026e1	Unix su	1	0
hoc08spd027e1	Unix su	1	0
hoc08spd031e1	Unix su	1	0

Rows Returned 218

Starting Sat Jan 1, 2005

Ending Tue Jan 1, 2008

Summary of Failures & Successes

analytics.intellischema

Show

Table

Table
Chart

☑ Report Name
☑ Description
☑ Date Criteria
☑ Namespace
☑ Row Number

☑ Information System
☑ Event Source
☑ Number Successful
☑ Number Failed

To remove a column or metadata item from display, deselect its property.

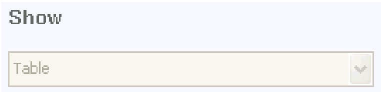
NOTE: When you show and hide report columns and metadata, only *your* view of the data changes. Other users viewing the same dashboard do not see these changes.

Changing Between Table and Chart Formats

By default, every report displays as a table. If the report's creator defined a chart format for the report, the report can display in either table or chart format. Some dashboard pages may contain two copies of the same report, one displayed as a chart and the other as a table.

If your dashboard page displays a single copy of a report widget that has been configured for chart as well as table display, you can use the Options Pane to toggle its display format.

As illustrated in Figure 2-5, when a report has been configured for both chart and table display, the **Show** tab enables you to toggle between the two display types. The name of the button changes to display the current option. When a report has been configured only for table display, this toggle field is disabled. It displays only the word "Table", as shown below.

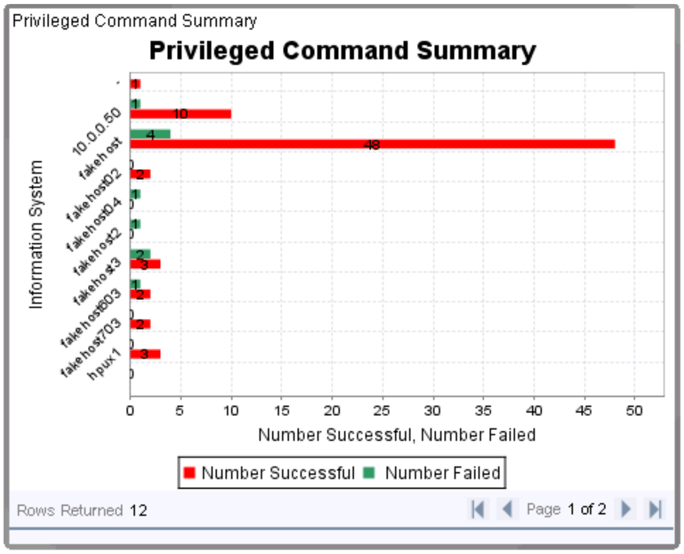


Show

Table

Although HawkEye AP provides several styles of bar and column chart options, only one chart style is available in a report widget. If you select **Chart** from the **Table** dropdown, the widget displays the chart style that was configured for the report in its definition. In other words, if the current report has been configured to include bar chart display, the dropdown allows you to toggle between these formats. Figure 2-6 illustrates the bar-chart version of the tabular report shown in Figure 2-5.

Figure 2-6: Bar Chart Display



When a report displays as a chart, you have the option of investigating data related to a specific data point in the chart. For more information, see "Investigating Data from a Line Chart", on page 60.

NOTE: When you toggle between table and chart view, only your view of the data changes. Other users viewing the same dashboard do not see these changes.

Viewing the SQL Query and Other Properties

The **SQL** tab on the Options Pane displays the SQL query that generated the report data set. This tab also displays:

- who and when the report definition was created
- who and when the report definition was last modified

Additionally, this tab includes the **View Search Criteria** button. Click this button to display the following information about the current cache entry:

- Date period and time zone over which it was run
- Column values that the user specified at run time

  This information is useful if the report provided the run-time user with parameters to limit the report results. In other words, if the user limited the result set, you can click this button to view the values used to limit the results.

- Namespace in which the report was run

This information is particularly useful if a single report runs in more than one namespace. For more information, see “Namespaces: Using a Single Report or Dashboard to Access Different Data”, on page 43

Manipulating and Expanding Report Results

This section contains the following topics:

- “Viewing and Changing the Time Range and Namespace”, next
- “Calculating Report Data”, on page 53
- “Opening an Associated Report”, on page 54
- “Browsing to Other Reports”, on page 57
- “Investigating Data from a Line Chart”, on page 60

Viewing and Changing the Time Range and Namespace

Every time a dashboard runs, a new cache entry is generated for every report the dashboard displays. When you open a dashboard and view a report, it displays the cache entry created when the dashboard ran last. If a report displays in more than one dashboard, and the different dashboards run at different intervals, the report will have a cache entry for each of those intervals. Additionally, if you or others run the report manually, the report will have a cache entry for each of these manual runs. Each of these manually run cache entries identifies the user who created it and the date created.

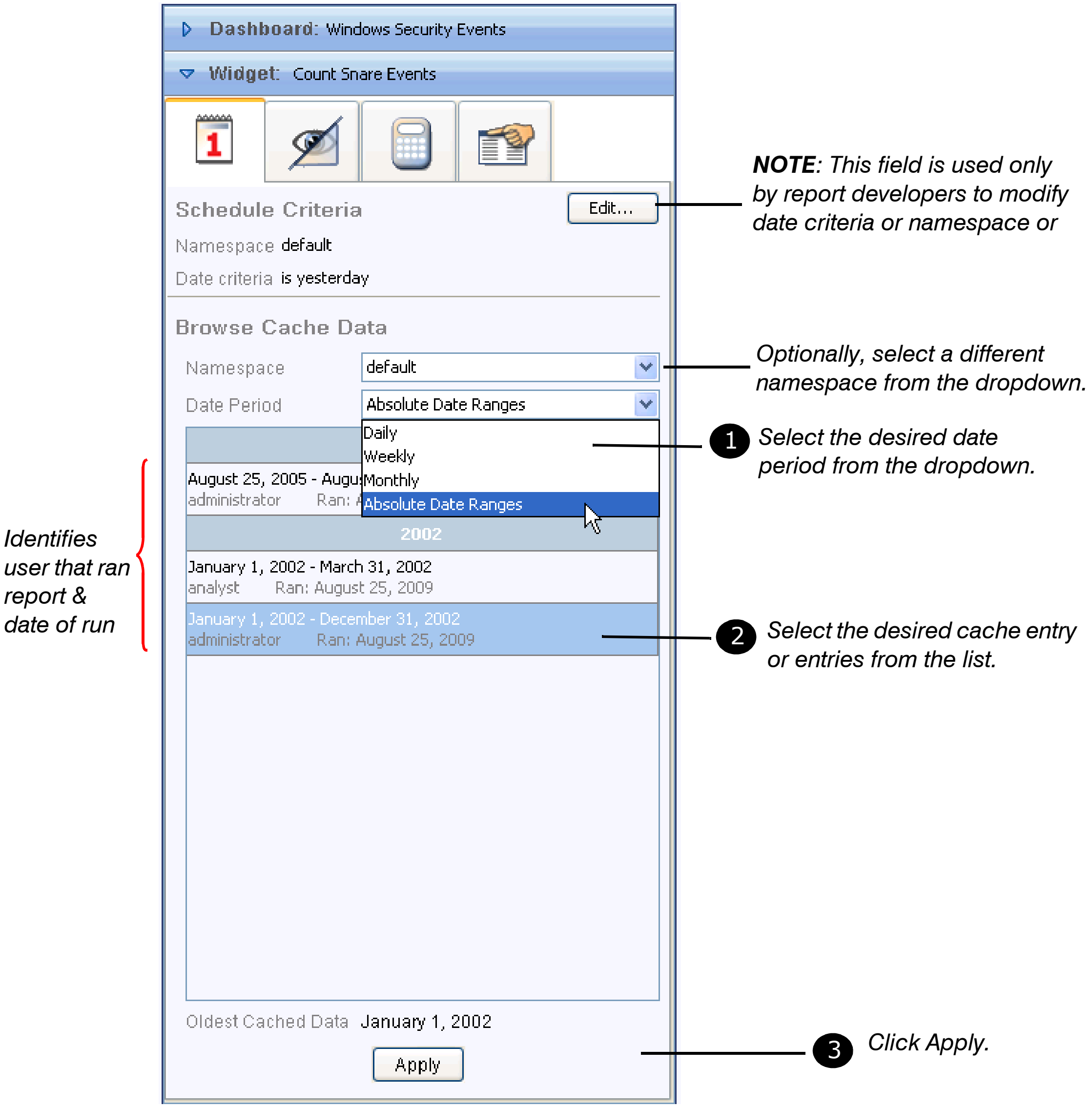
Assume one dashboard runs daily and another weekly, and that they both contain the same report. Daily and weekly cache entries will be available for that report. When you open the daily dashboard, the latest daily cache entry displays. When you open the weekly dashboard, the latest weekly cache entry displays. To switch between these cache entries, or to display an earlier cache for either interval or to merge several cache entries into a single result set, open the **Date Options** tab for the report widget.

As illustrated in Figure 2-7, if a report has been run over more than one date interval, the **Date Period** dropdown displays all cached intervals. If only one interval is available, this field displays the date period as a text label rather than in a dropdown.

NOTE: If a report has been run in more than one namespace, the **Namespace** dropdown contains all of those namespaces. You can use this dropdown to switch namespaces.

The field below the **Namespace** and **Date Period** dropdowns organizes cache entries by date and time for the current date period. The most recent entries display at the top.

Figure 2-7: Date Options Tab:



Dashboard: Windows Security Events
Widget: Count Snare Events
Edit...
Apply

Schedule Criteria

Namespace default

Date criteria is yesterday

Browse Cache Data

Namespace: default

Date Period: Absolute Date Ranges

Daily
Weekly
Monthly
Absolute Date Ranges

August 25, 2005 - Augu
administrator Ran: A

2002

January 1, 2002 - March 31, 2002
analyst Ran: August 25, 2009

January 1, 2002 - December 31, 2002
administrator Ran: August 25, 2009

Oldest Cached Data January 1, 2002

Identifies user that ran report & date of run

***NOTE**: This field is used only by report developers to modify date criteria or namespace or*

Optionally, select a different namespace from the dropdown.

1 *Select the desired date period from the dropdown.*

2 *Select the desired cache entry or entries from the list.*

3 *Click Apply.*

If a report has multiple cache entries, you can select contiguous entries to display as a single result set. Select one or more cache entries and click **Apply** to display the selected entries. To select multiple entries, select the first entry, then SHIFT-CLICK as you select the last entry

NOTE:

- You cannot combine cache entries if:
  - They represent absolute date ranges. In other words, you can combine daily cache entries into weekly reports, and weekly cache entries into monthly reports, but you cannot combine an absolute date range that represents the first week of September with one that represents the second week of September.
  - The cache entries represent different time zones. In other words, you can combine weekly cache entries into monthly reports only if all cache entries represent the same time zone.

NOTE: Merging cache entries fails if Daylight Saving Time changes in any of the entries. In other words, you can combine weekly cache entries into monthly reports, but if Daylight Savings Time changed during one of the weekly entries, the merger fails.

- The **Date Periods** dropdown displays all cache entries available at the time you opened the Options Pane. To display any cache entry created since you opened the dropdown, you must refresh the dashboard. For more information, see “Refreshing Dashboards and Running Items”, on page 85.
- When you change or combine cache entries, only your view of the data changes. Other users viewing the same dashboard do not see these changes.

If a report has run in more than one namespace, you can display cache entrie(s) for another namespace. Select the desired namespace from the **Namespace** dropdown.

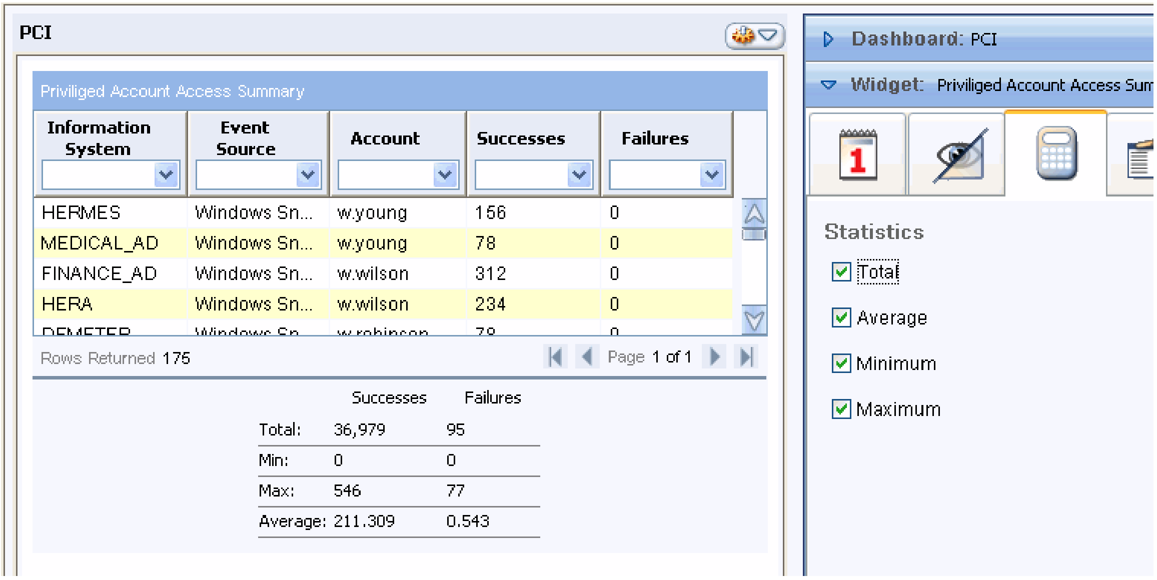
Calculating Report Data

For a tabular report widget, you can specify calculation of values in numeric columns. These calculations include totaling values, identifying the records with the minimum and maximum values, and calculating the average value in each numeric column.

Calculations are based on the current result set, regardless of the records currently displayed. For example, assume a report widget currently displays only 10 rows of a 100-row result set. If you set the report to display totals for the numeric columns, the total values displayed for those columns are based on all values in the result set.

Figure 2-8 illustrates the Statistics tab for a report widget. This tab enables calculating data. This figure also illustrates the result of setting all statistics options on a report with two numeric columns.

Figure 2-8: Calculating Report Data



PCI

Priviliged Account Access Summary

Information System	Event Source	Account	Successes	Failures
HERMES	Windows Sn...	w.young	156	0
MEDICAL_AD	Windows Sn...	w.young	78	0
FINANCE_AD	Windows Sn...	w.wilson	312	0
HERA	Windows Sn...	w.wilson	234	0
DEMETER	Windows Sn...	w.robinson	78	0

Rows Returned 175 Page 1 of 1

	Successes	Failures
Total:	36,979	95
Min:	0	0
Max:	546	77
Average:	211.309	0.543

Dashboard: PCI

Widget: Priviliged Account Access Sur

Statistics

☑ Total

☑ Average

☑ Minimum

☑ Maximum

The Statistics tab provides the calculation options shown in the table below.

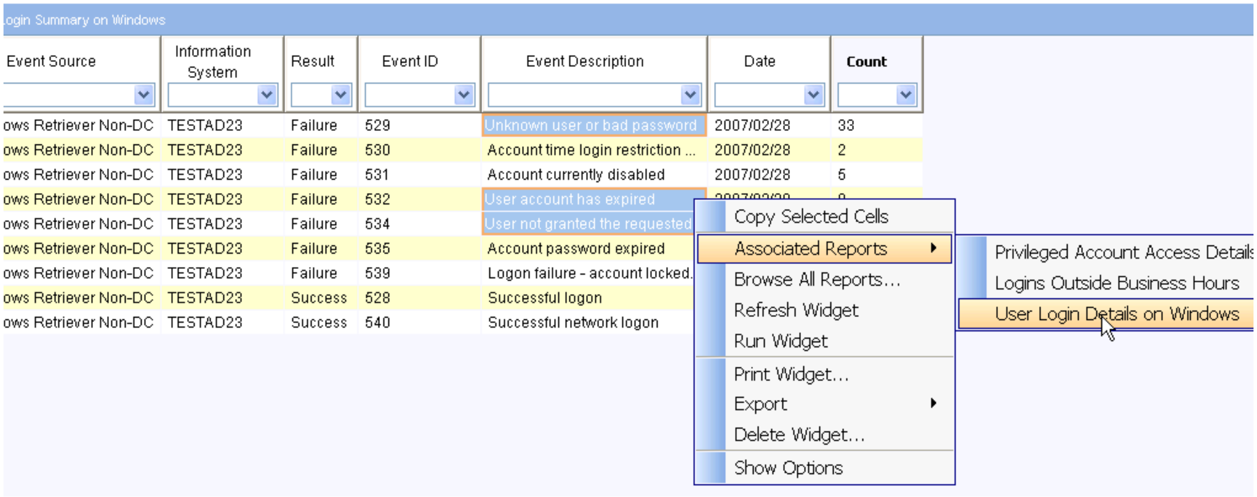
Option	Description
Total	For each numeric column, display the sum of its values for the entire result set in a row labeled "Total".
Average	For each numeric column, display the average of its values for the entire result set in a row labeled "Average"
Minimum	Display the minimum of each column's value over the entire result set in a row labeled "Min". For example, the minimum of a text column is determined alphabetically, IP address values are evaluated as IP addresses, and numeric values are determined numerically.
Maximum	Display the maximum of each column's value over the entire result set in a row labeled "Max".

Opening an Associated Report

Often when you view data in a tabular report, you want to view additional information about one or more of the displayed rows. Tabular report widgets enable you to select specific values in a column and open associated report(s) on those values. The data you select to investigate is automatically made available as criteria when you run the associated report.

Figure 2-9 illustrates a user linking from three rows in one report to a choice of associated reports.

Figure 2-9: Opening An Associated Report



ogin Summary on Windows

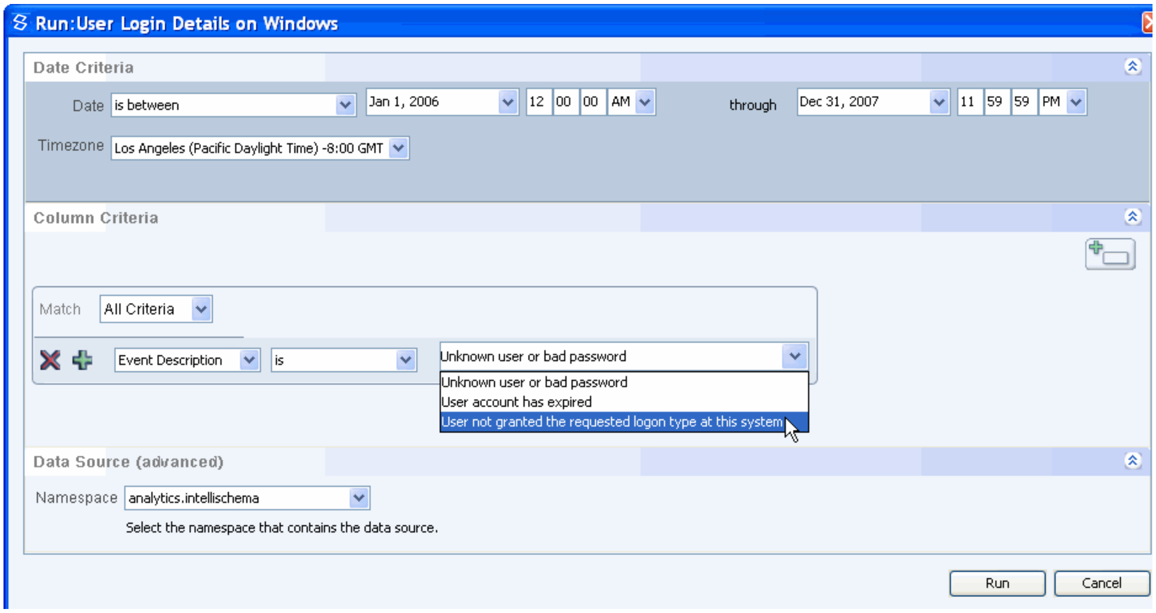
Event Source	Information System	Result	Event ID	Event Description	Date	Count
ows Retriever Non-DC	TESTAD23	Failure	529	Unknown user or bad password	2007/02/28	33
ows Retriever Non-DC	TESTAD23	Failure	530	Account time login restriction ...	2007/02/28	2
ows Retriever Non-DC	TESTAD23	Failure	531	Account currently disabled	2007/02/28	5
ows Retriever Non-DC	TESTAD23	Failure	532	User account has expired		
ows Retriever Non-DC	TESTAD23	Failure	534	User not granted the requested		
ows Retriever Non-DC	TESTAD23	Failure	535	Account password expired		
ows Retriever Non-DC	TESTAD23	Failure	539	Logon failure - account locked.		
ows Retriever Non-DC	TESTAD23	Success	528	Successful logon		
ows Retriever Non-DC	TESTAD23	Success	540	Successful network logon		

As shown above, a user has selected data in three rows of the **Event Description** column and has right-clicked to display the popup menu. From the popup, the user has clicked **Associated Reports** to display the selection of linked reports. The report’s creator determined that these reports would be useful to those who view this report and explicitly associated them to this report.

After you select the desired report, the **Run** Report dialog displays, as illustrated in Figure 2-10. All three values selected from the source report display in the dropdown of the associated report.

You select the desired value from the dropdown, as illustrated below. Then click **Run** to run the report

Figure 2-10: Running the Associated Report on Selected Data



Run:User Login Details on Windows

Date Criteria

Date: is between — Jan 1, 2006 — 12 00 00 AM — through — Dec 31, 2007 — 11 59 59 PM

Timezone: Los Angeles (Pacific Daylight Time) -8:00 GMT

Column Criteria

Match: All Criteria

Event Description — is — Unknown user or bad password

- Unknown user or bad password
- User account has expired
- User not granted the requested logon type at this system

Data Source (advanced)

Namespace: analytics.intellischema

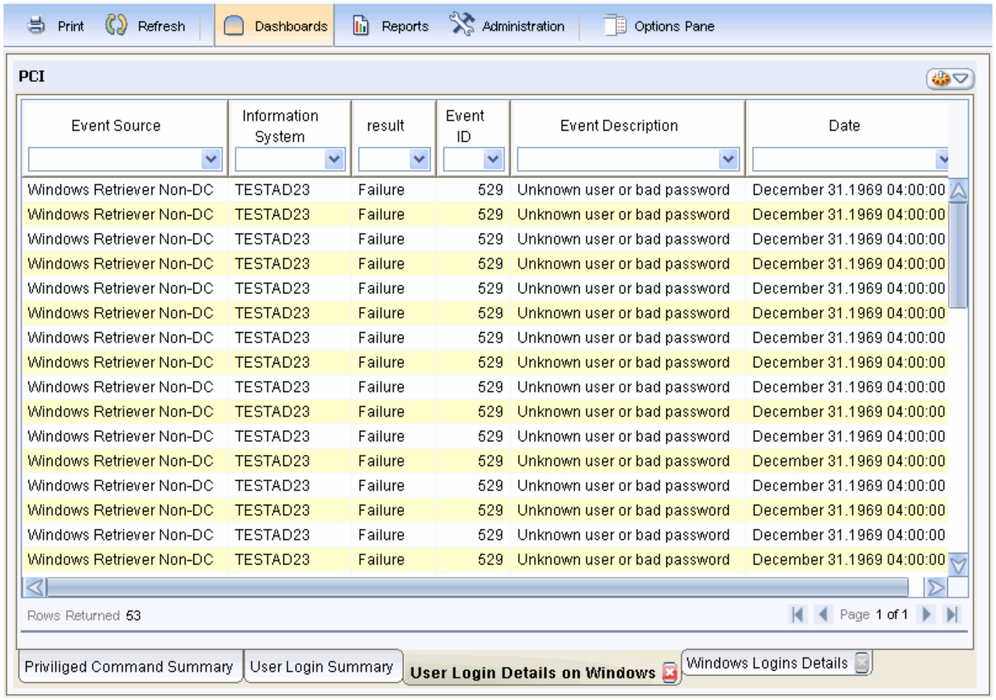
Select the namespace that contains the data source.

NOTE:

- From the **Run** dialog, you can specify a new time range or keep the default one.
- In the example above, the user selected values from the same column (**Event Description**) that displays in the **Run** dialog. If the column you are investigating does not display by default, you can select it from the left dropdown. Figure 2-14 illustrates this process.
- If the Run dialog displays additional criteria columns, you can either specify relevant values for them or delete them. Figure 2-19 illustrates this process.

- Figure 2-11 illustrates the results of running the associated reports with one of the values in the **Event Description** dropdown.

Figure 2-11: Associated Report Results



Event Source	Information System	result	Event ID	Event Description	Date
Windows Retriever Non-DC	TESTAD23	Failure	529	Unknown user or bad password	December 31.1969 04:00:00
Windows Retriever Non-DC	TESTAD23	Failure	529	Unknown user or bad password	December 31.1969 04:00:00
Windows Retriever Non-DC	TESTAD23	Failure	529	Unknown user or bad password	December 31.1969 04:00:00
Windows Retriever Non-DC	TESTAD23	Failure	529	Unknown user or bad password	December 31.1969 04:00:00
Windows Retriever Non-DC	TESTAD23	Failure	529	Unknown user or bad password	December 31.1969 04:00:00
Windows Retriever Non-DC	TESTAD23	Failure	529	Unknown user or bad password	December 31.1969 04:00:00
Windows Retriever Non-DC	TESTAD23	Failure	529	Unknown user or bad password	December 31.1969 04:00:00
Windows Retriever Non-DC	TESTAD23	Failure	529	Unknown user or bad password	December 31.1969 04:00:00
Windows Retriever Non-DC	TESTAD23	Failure	529	Unknown user or bad password	December 31.1969 04:00:00
Windows Retriever Non-DC	TESTAD23	Failure	529	Unknown user or bad password	December 31.1969 04:00:00
Windows Retriever Non-DC	TESTAD23	Failure	529	Unknown user or bad password	December 31.1969 04:00:00
Windows Retriever Non-DC	TESTAD23	Failure	529	Unknown user or bad password	December 31.1969 04:00:00
Windows Retriever Non-DC	TESTAD23	Failure	529	Unknown user or bad password	December 31.1969 04:00:00
Windows Retriever Non-DC	TESTAD23	Failure	529	Unknown user or bad password	December 31.1969 04:00:00
Windows Retriever Non-DC	TESTAD23	Failure	529	Unknown user or bad password	December 31.1969 04:00:00
Windows Retriever Non-DC	TESTAD23	Failure	529	Unknown user or bad password	December 31.1969 04:00:00

Rows Returned 53

NOTE: HawkEye AP Console opens a new page in the dashboard to display the associated report.

To run an associated or other report

1 In a column of the source report, select desired row(s) from a single column.

2 Right-click to display the popup menu.

3 Click **Associated Reports** to display the full list of associated reports.

The first three steps are illustrated in Figure 2-9.

4 Select the desired report from the list.

The **Run Report** dialog displays.

This step is illustrated in Figure 2-10.

5 If you have selected more than one value from the source report, select one to run first from the column dropdown.

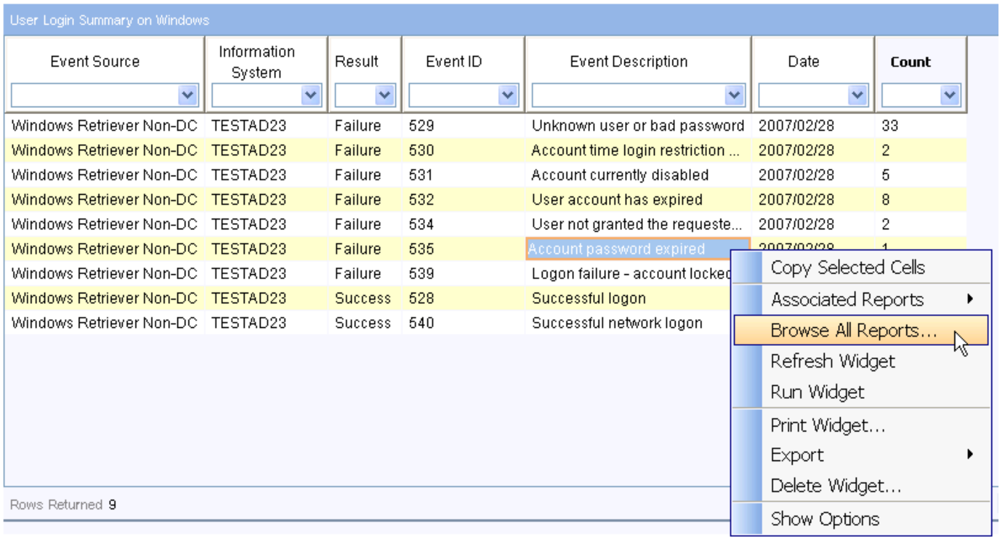
6 If desired, enter a new date period, and click **Run**.

Browsing to Other Reports

Not all reports have other reports associated to them. And not all associated reports may meet your investigation needs. HawkEye AP Console also enables you to investigate data from the current report in any other report in the system for which you have access permission.

Below the link to **Associated Reports** is a link to **Browse all Reports....**, as illustrated in Figure 2-12. As with an associated report, the data you select in the source report displays in the report you select by browsing.

Figure 2-12: Using the Browse All Reports Option



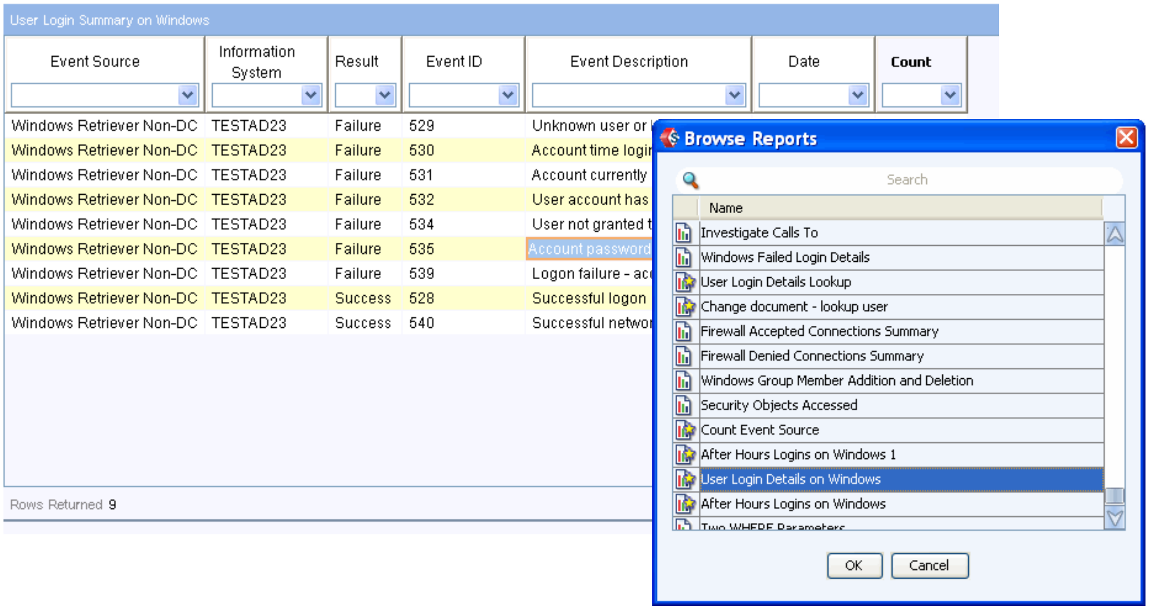
User Login Summary on Windows

Event Source	Information System	Result	Event ID	Event Description	Date	Count
Windows Retriever Non-DC	TESTAD23	Failure	529	Unknown user or bad password	2007/02/28	33
Windows Retriever Non-DC	TESTAD23	Failure	530	Account time login restriction ...	2007/02/28	2
Windows Retriever Non-DC	TESTAD23	Failure	531	Account currently disabled	2007/02/28	5
Windows Retriever Non-DC	TESTAD23	Failure	532	User account has expired	2007/02/28	8
Windows Retriever Non-DC	TESTAD23	Failure	534	User not granted the requeste...	2007/02/28	2
Windows Retriever Non-DC	TESTAD23	Failure	535	Account password expired	2007/02/28	1
Windows Retriever Non-DC	TESTAD23	Failure	539	Logon failure - account locked		
Windows Retriever Non-DC	TESTAD23	Success	528	Successful logon		
Windows Retriever Non-DC	TESTAD23	Success	540	Successful network logon		

Rows Returned 9

From the **Browse Reports** dialog, select the desired report. Figure 2-13 illustrates this process.

Figure 2-13: Selecting a Report for Browsing



User Login Summary on Windows

Event Source	Information System	Result	Event ID	Event Description
Windows Retriever Non-DC	TESTAD23	Failure	529	Unknown user or
Windows Retriever Non-DC	TESTAD23	Failure	530	Account time logir
Windows Retriever Non-DC	TESTAD23	Failure	531	Account currently
Windows Retriever Non-DC	TESTAD23	Failure	532	User account has
Windows Retriever Non-DC	TESTAD23	Failure	534	User not granted t
Windows Retriever Non-DC	TESTAD23	Failure	535	Account password
Windows Retriever Non-DC	TESTAD23	Failure	539	Logon failure - ac
Windows Retriever Non-DC	TESTAD23	Success	528	Successful logon
Windows Retriever Non-DC	TESTAD23	Success	540	Successful networ

Rows Returned 9

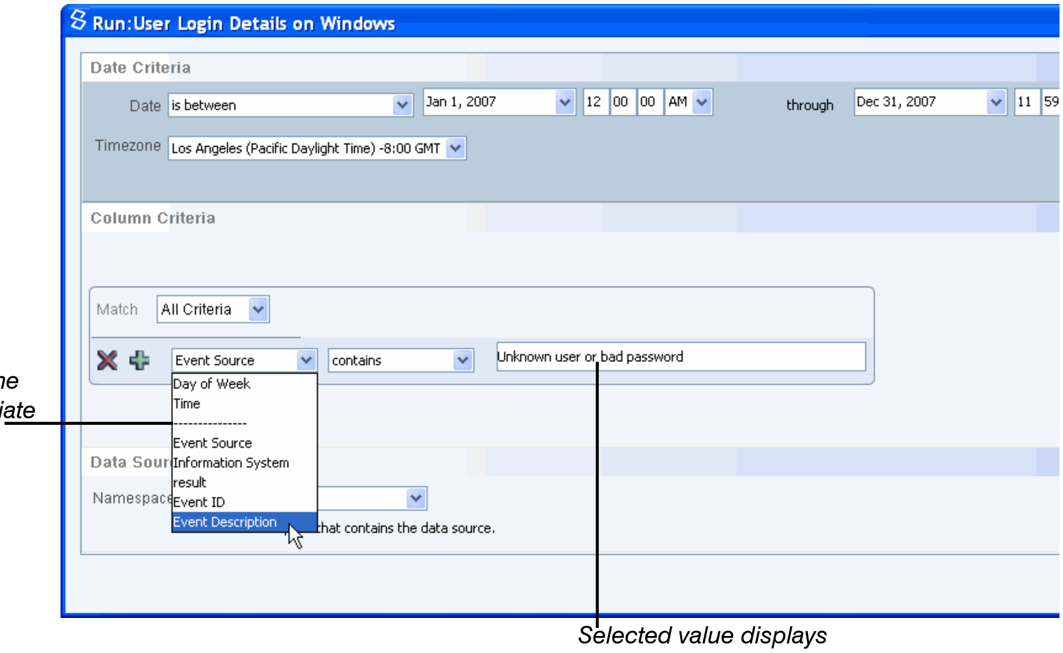
Browse Reports

Search

Name
Investigate Calls To
Windows Failed Login Details
User Login Details Lookup
Change document - lookup user
Firewall Accepted Connections Summary
Firewall Denied Connections Summary
Windows Group Member Addition and Deletion
Security Objects Accessed
Count Event Source
After Hours Logins on Windows 1
User Login Details on Windows
After Hours Logins on Windows
Two WHERE Parameters

Figure 2-14 illustrates the **Run** dialog that displays for the selected report. The example report displays one Column Criteria field, which automatically contains the value you selected in the source report. To make the run meaningful, select the relevant column from the dropdown on the left. Because the data illustrated in Figure 2-13 comes from the **Event Description** column, the user selects this column in the example illustrated below.

Figure 2-14: Selecting the Desired Column



Run:User Login Details on Windows

Date Criteria

Date: is between | Jan 1, 2007 | 12 | 00 | 00 | AM | through | Dec 31, 2007 | 11 | 59

Timezone: Los Angeles (Pacific Daylight Time) -8:00 GMT

Column Criteria

Match: All Criteria

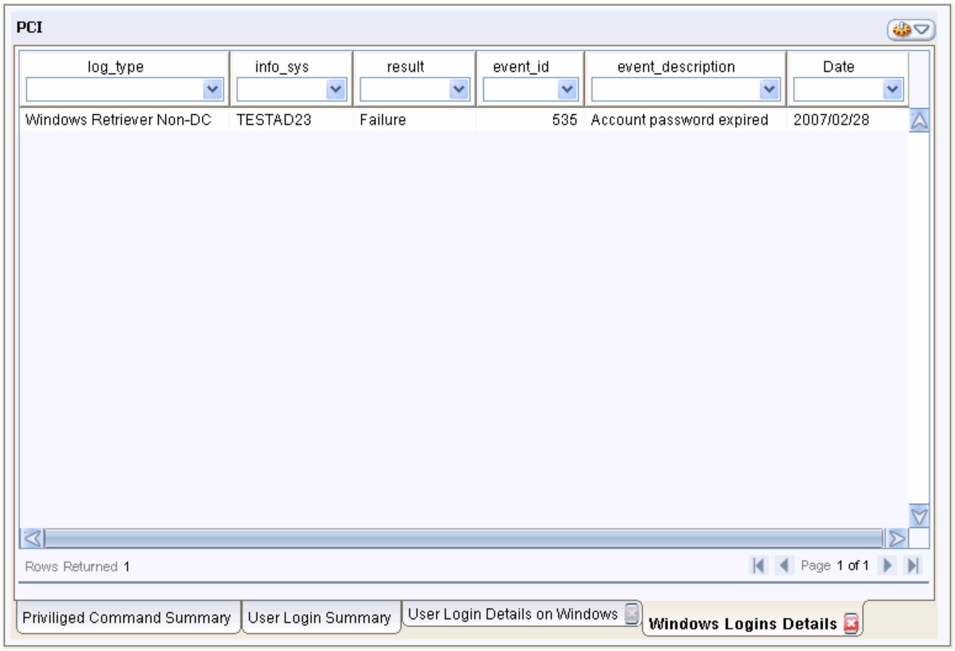
Event Source | contains | Unknown user or bad password

Select the appropriate column.

Selected value displays

Figure 2-15 illustrates the results of running the associated reports with one of the values in the **Event Description** dropdown.

Figure 2-15: Browse Reports Results



PCI

log_type	info_sys	result	event_id	event_description	Date
Windows Retriever Non-DC	TESTAD23	Failure	535	Account password expired	2007/02/28

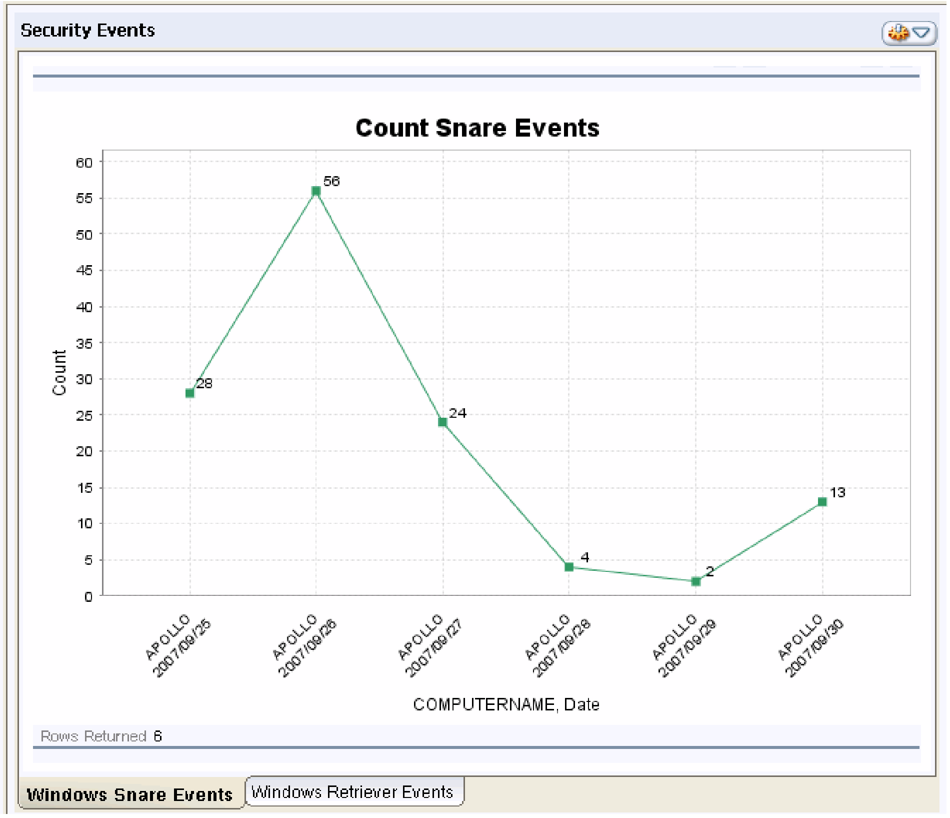
Rows Returned 1

NOTE: HawkEye AP Console opens a new page in the dashboard to display the associated report.

Investigating Data from a Line Chart

"Changing Between Table and Chart Formats", on page 49 illustrates a report that displays as a bar chart. Figure 2-16 illustrates a different report, which displays as a line chart and is part of the IT dashboard.

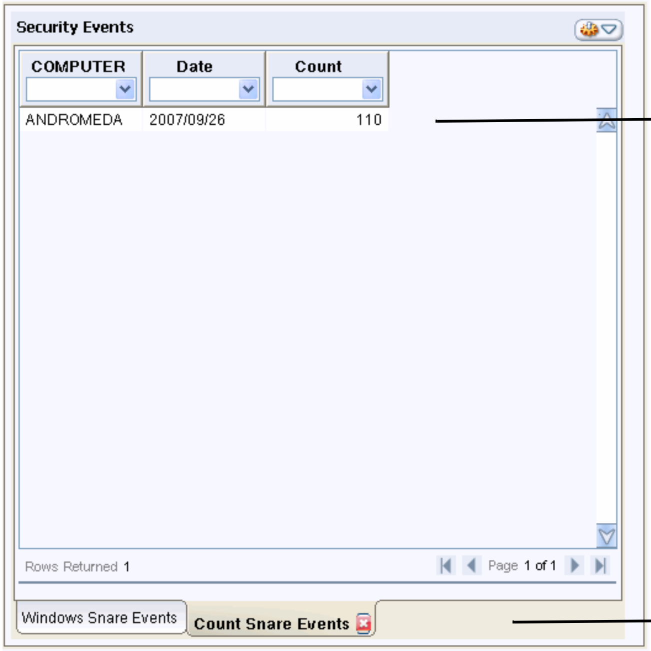
Figure 2-16: Line Chart Display of a Different Report



A line chart is particularly useful for analyzing data trends; it graphically displays how data changes over time. The chart in the example above graphically illustrates the number of snare events received by a specified computer over several days. If one data point interests you, you can investigate it further by double-clicking it.

For example, the data in the chart above spikes on September 26. To learn more about this data point, double click it. The dashboard opens a new page that displays information for the selected data point in tabular format. From the tabular report you can open an associated report or browse to any report on your system to which you have access. Figure 2-17 illustrates the tabular report that displays for the September 26 data point.

Figure 2-17: Investigating a Data Point—Displaying the Tabular Report



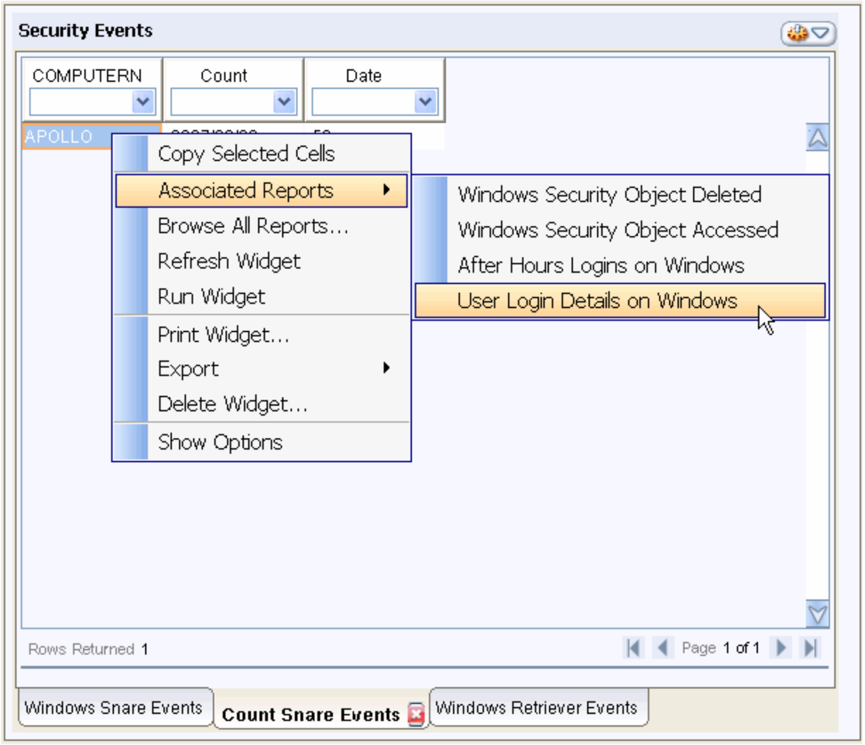
COMPUTER	Date	Count
ANDROMEDA	2007/09/26	110

A single row represents the selected data point.

The tabular report opens in a new page.

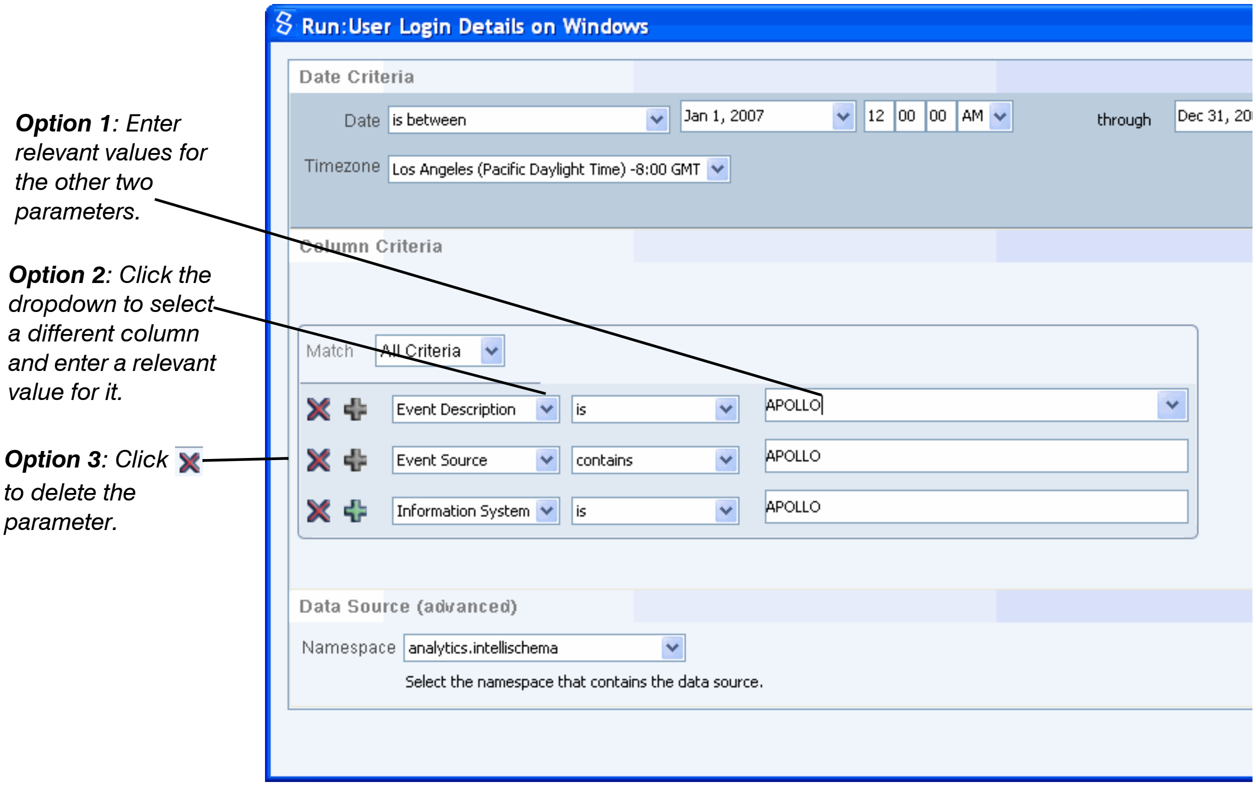
To learn more about the data point, select a displayed value and right click on it to display the popup menu. Figure 2-18 illustrates the user selecting an associated report.

Figure 2-18: Investigating a Data Point—Opening an Associated Report



As illustrated in Figure 2-18, the user has chosen the User Login Details on Windows report to open. Figure 2-19 illustrates the **Run** dialog that displays. In this example, the **Run** dialog displays more than one parameter. The value the user selected from the source report displays in all three parameters. The value is relevant only to the **Information System** parameter.

Figure 2-19: Investigating a Data Point—Specifying Values in the Run Dialog



***Option 1**: Enter relevant values for the other two parameters.*

***Option 2**: Click the dropdown to select a different column and enter a relevant value for it.*

***Option 3**: Click ✕ to delete the parameter.*

Run:User Login Details on Windows

Date Criteria

Date: is between | Jan 1, 2007 | 12 | 00 | 00 | AM | through | Dec 31, 20

Timezone: Los Angeles (Pacific Daylight Time) -8:00 GMT

Column Criteria

Match: All Criteria

Event Description	is	APOLLO
Event Source	contains	APOLLO
Information System	is	APOLLO

Data Source (advanced)

Namespace: analytics.intellischema

Select the namespace that contains the data source.

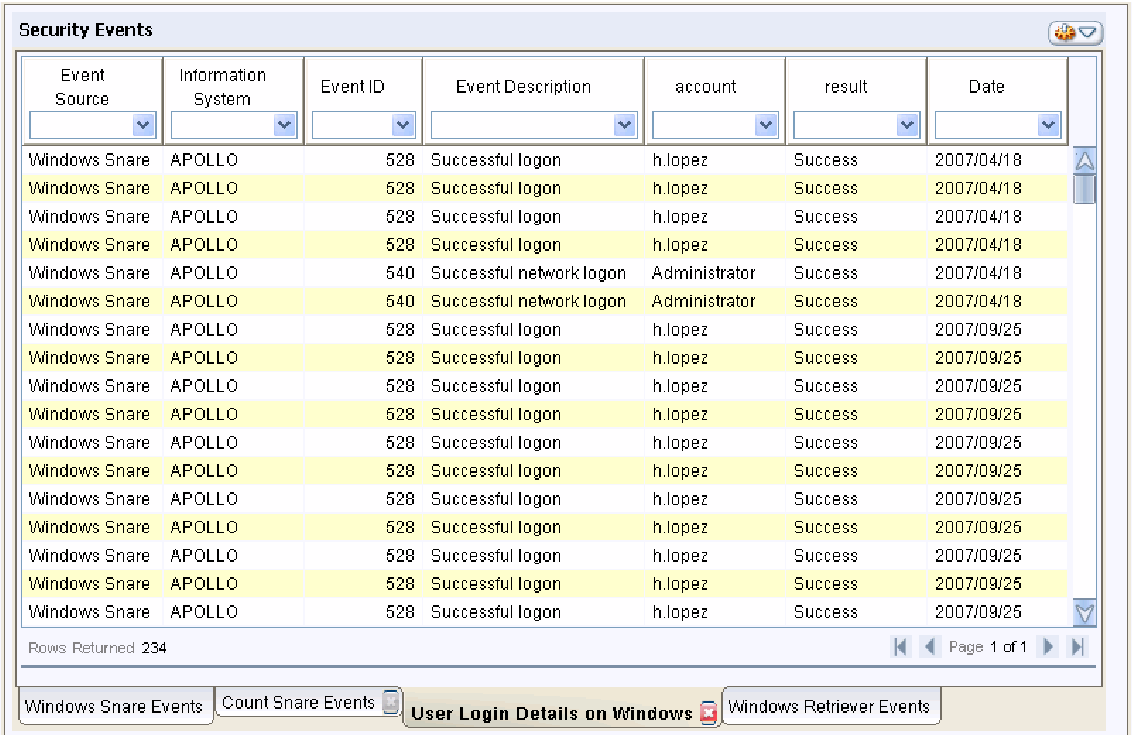
:

To further the investigation, the user has the following options before clicking **Run**:

- Specify values for the other two parameters—enter an appropriate value for the **Event Description** and **Event Source** parameters.
- Select a different column for the other parameters—click the dropdown and select a different column; then enter a relevant value for it.
- Delete the unwanted parameters—click the ✕ icon next to the unwanted parameter.

After the user sets the **Run** dialog as desired and clicks **Run**, the associated report displays on a new page, as illustrated in Figure 2-20.

Figure 2-20: Investigating a Data Point—Viewing Associated Results



Security Events

Event Source	Information System	Event ID	Event Description	account	result	Date
Windows Snare	APOLLO	528	Successful logon	h.lopez	Success	2007/04/18
Windows Snare	APOLLO	528	Successful logon	h.lopez	Success	2007/04/18
Windows Snare	APOLLO	528	Successful logon	h.lopez	Success	2007/04/18
Windows Snare	APOLLO	528	Successful logon	h.lopez	Success	2007/04/18
Windows Snare	APOLLO	540	Successful network logon	Administrator	Success	2007/04/18
Windows Snare	APOLLO	540	Successful network logon	Administrator	Success	2007/04/18
Windows Snare	APOLLO	528	Successful logon	h.lopez	Success	2007/09/25
Windows Snare	APOLLO	528	Successful logon	h.lopez	Success	2007/09/25
Windows Snare	APOLLO	528	Successful logon	h.lopez	Success	2007/09/25
Windows Snare	APOLLO	528	Successful logon	h.lopez	Success	2007/09/25
Windows Snare	APOLLO	528	Successful logon	h.lopez	Success	2007/09/25
Windows Snare	APOLLO	528	Successful logon	h.lopez	Success	2007/09/25
Windows Snare	APOLLO	528	Successful logon	h.lopez	Success	2007/09/25
Windows Snare	APOLLO	528	Successful logon	h.lopez	Success	2007/09/25
Windows Snare	APOLLO	528	Successful logon	h.lopez	Success	2007/09/25
Windows Snare	APOLLO	528	Successful logon	h.lopez	Success	2007/09/25
Windows Snare	APOLLO	528	Successful logon	h.lopez	Success	2007/09/25

Rows Returned 234

Viewing Security Alerts

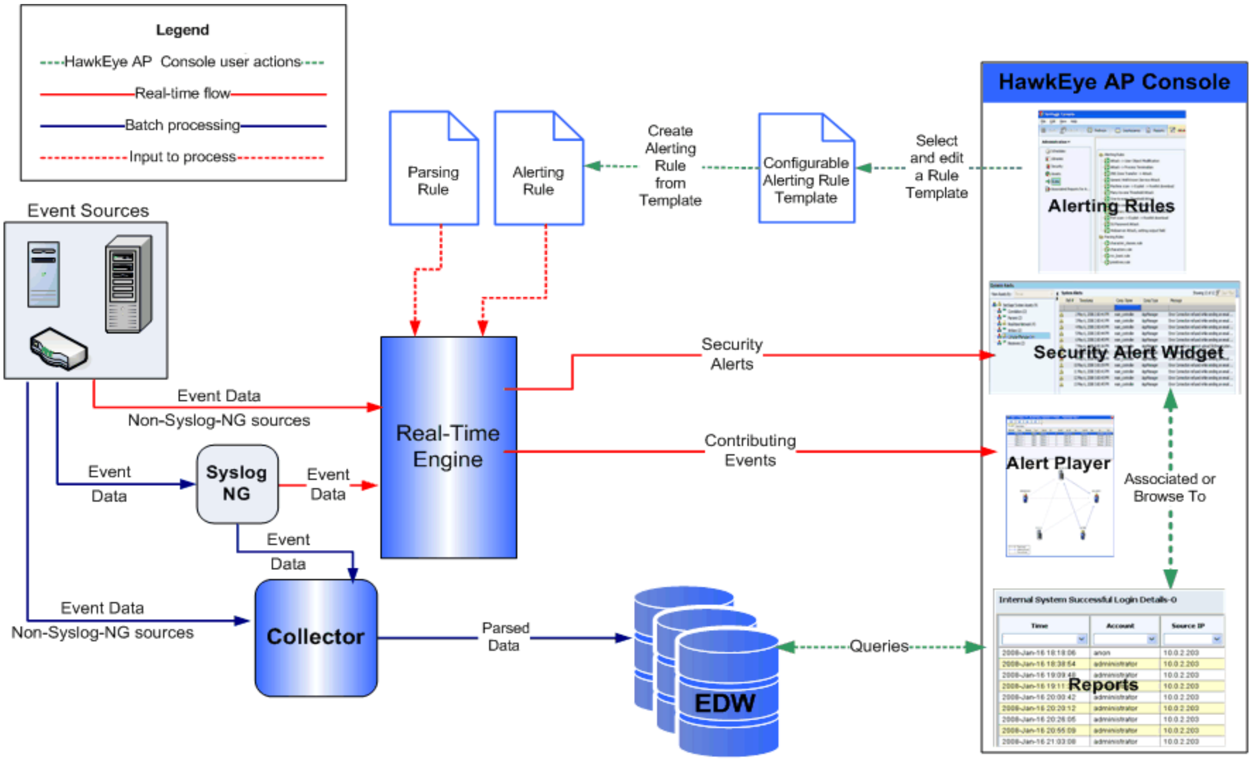
This section includes the following topics:

- "Background", next
- "About HawkEye AP Security Alerts", on page 66
- "Security Alerts: Working with the Asset Tree", on page 67
- "Security Alerts: Working with the Alerts Table", on page 68
- "Viewing the Information in a Security Alert ", on page 68
- "Launching Associated Reports", on page 69
- "Understanding and Using Threat and Risk Views", on page 71
- "Sorting Alert Data", on page 76
- "Widening Column Display in an Alert Widget", on page 77
- "Filtering Alert Data", on page 77
- "Security Alerts: Using the Alert Player", on page 78
- "Security Alerts: Understanding the Alerting Rule that Raised the Alert", on page 83
- "Viewing Exception Alerts", on page 84

Background

A HawkEye AP deployment processes event data from a variety of hardware and software systems. These sources typically create logs of their activities and save them in a variety of formats. Figure 2-21 illustrates the data flow from event sources to a HawkEye AP Console dashboard that displays the alerts triggered from the data.

Figure 2-21 Real-Time Event Data Flow



As shown in Figure 2-21, event data arrives from various event sources to the HawkEye AP Real-Time engine and to the Collector. Some of this event data arrives through syslog-ng. The Collector uses a PTL (Parse, Transform, Load) file to parse the data for storage in the EDW. The Real-Time engine uses parsing rules to parse the data for analysis and alerting rules to raise alerts when the data meets specified conditions. The Real-Time engine sends alerts and their contributing events to HawkEye AP Console where analysts use the Security Alerts Widget and the Alert Player to view them.

In addition to the parsing rules and alerting rules delivered with the product, HawkEye AP provides alerting rule templates that enable analysts to create their own alerting rules. Analysts can easily configure these rules to their specific needs.

As illustrated above, analysts use HawkEye AP Console to:

- view alerts in the Security Alerts widget

  For more information, see:

  - “About HawkEye AP Security Alerts”, next
  - “Security Alerts: Working with the Asset Tree”, on page 67
  - “Security Alerts: Working with the Alerts Table”, on page 68
  - “Viewing the Information in a Security Alert ”, on page 68
  - “Understanding and Using Threat and Risk Views”, on page 71
  - “Sorting Alert Data”, on page 76
  - “Widening Column Display in an Alert Widget”, on page 77
  - “Acknowledging Viewed Alerts”, on page 77
  - “Filtering Alert Data”, on page 77

- open reports that allow further investigation of activity highlighted in alert(s)

  For more information, see:

  - “Launching Associated Reports”, on page 69

- view events that contributed to an alert in the Alert Player

  For more information, see:

  - “Security Alerts: Using the Alert Player”, on page 78

- create new alerting rules from Alerting Rule Templates

  For more information, see:

  - Chapter 8: Creating Alerting Rules from Templates

About HawkEye AP Security Alerts

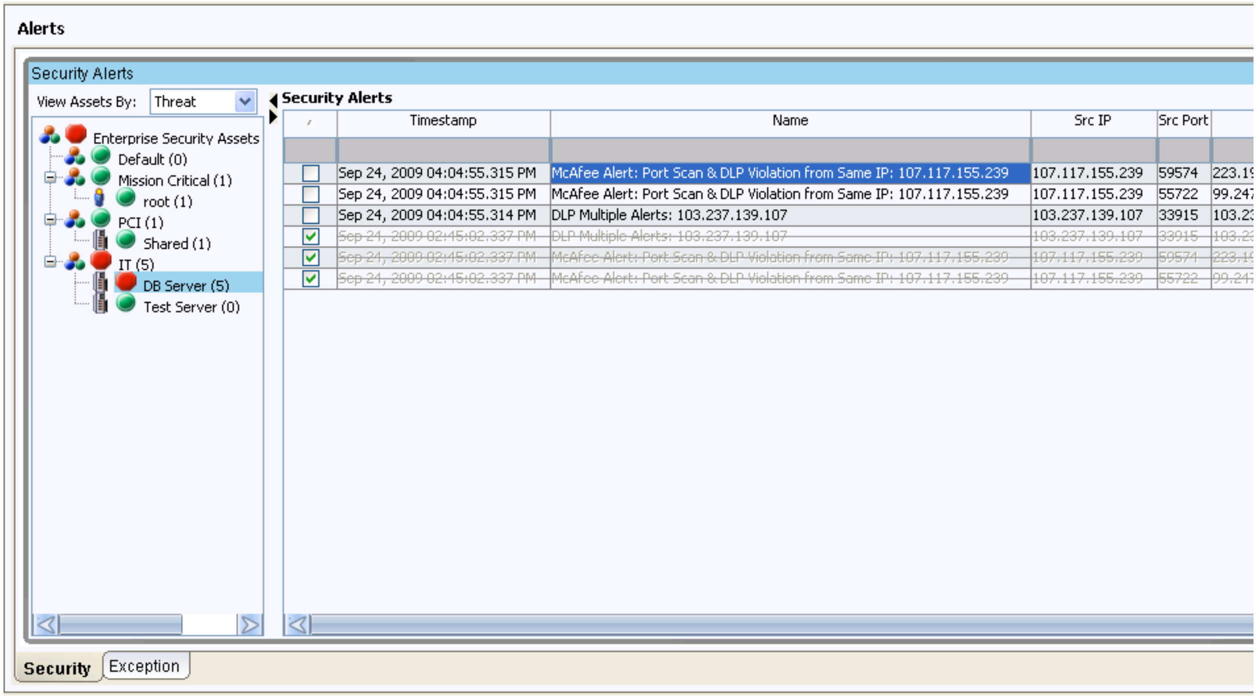
Assume your job requires you to monitor security alerts about enterprise security assets raised by the HawkEye AP Real-Time system. You can use the Security Alerts widget in a dashboard to view these alerts.

NOTE: Dashboards can also include widgets that display two other types of alert information. Dashboard widgets can display the following types of alerts:

- **Security Alerts**—display alerts about Enterprise security assets; documented in this section
- **System Alerts**—display alerts raised by HawkEye AP system assets and source-health failures; documented in Chapter 10: Administering Assets and Monitoring Alerts in the *Administration Guide*
- **Exception Alerts**—display alerts generated by an exception schedule when scheduled report(s) return one or more rows; documented in “Viewing Exception Alerts”, on page 84

Figure 2-22 illustrates an example Alerts dashboard that contains security alerts for several enterprise assets. Two of these assets display a red octagon, which indicates a high threat level.

Figure 2-22: Dashboard Display of Security Alerts



Security Alerts

	Timestamp	Name	Src IP	Src Port	
☐	Sep 24, 2009 04:04:55.315 PM	McAfee Alert: Port Scan & DLP Violation from Same IP: 107.117.155.239	107.117.155.239	59574	223.1
☐	Sep 24, 2009 04:04:55.315 PM	McAfee Alert: Port Scan & DLP Violation from Same IP: 107.117.155.239	107.117.155.239	55722	99.24
☐	Sep 24, 2009 04:04:55.314 PM	DLP Multiple Alerts: 103.237.139.107	103.237.139.107	33915	103.2
☑	~~Sep 24, 2009 02:45:02.337 PM~~	~~DLP Multiple Alerts: 103.237.139.107~~	~~103.237.139.107~~	~~33915~~	~~103.2~~
☑	~~Sep 24, 2009 02:45:02.337 PM~~	~~McAfee Alert: Port Scan & DLP Violation from Same IP: 107.117.155.239~~	~~107.117.155.239~~	~~59574~~	~~223.1~~
☑	~~Sep 24, 2009 02:45:02.337 PM~~	~~McAfee Alert: Port Scan & DLP Violation from Same IP: 107.117.155.239~~	~~107.117.155.239~~	~~55722~~	~~99.24~~

Figure 2-22 illustrates the Security Alerts widget. On the left are assets delivered with HawkEye AP and defined by the site administrator. On the right are the alerts themselves.

Security Alerts: Working with the Asset Tree

By default, all security alerts display in a single table in the Security Alerts widget. To make alert viewing more meaningful and precise, HawkEye AP recommends that your site administrator define an IP Range asset or User List asset for each server and user that you want to track. To make alert viewing even more meaningful, your site administrator can organize your IP Range assets and User List assets into Asset Groups. Whenever an alert is raised against one of your defined assets, the alert displays below the specific asset group to which the asset belongs. For information about how to create the assets in HawkEye AP Console, see Chapter 10: Administering Assets and Monitoring Alerts in the *Administration Guide*.

The Asset Tree on the left of the Security Alerts widget displays all assets configured for your system. The following assets in this tree are permanent and exist on all HawkEye AP systems:

- **Enterprise Security Assets**—contains customer-defined assets and security alerts raised against them; although this asset is permanent, you can change its name
- **Default**—contains security alerts against assets that have not been defined as assets in HawkEye AP

The table below illustrates the three types of assets that display in the tree and their associated icons.

Asset Icon	Meaning
	Asset Group—expand the group to display the assets and additional groups it contains. Figure 2-22 illustrates several asset groups, including PCI Assets and Mission Critical Assets.
	IP Range—identified by a single IP address or a range of IP addresses Figure 2-22 illustrates the PCI Assets and IT Assets groups opened to display several IP-range assets
	User List—identified by a user name Figure 2-22 illustrates one user-list asset, defined for Mission Critical Assets.

Status icons in the Security Alerts widget quickly indicate the level of threats and risks to assets that you want secured in your enterprise. Even when the asset trees are collapsed, these icons clearly indicate which alerts you should be monitoring.

For example, if a red octagon displays before an asset in the tree, you know immediately that there is a problem with one or more of your organization's assets. A red octagon that displays for a high-level asset indicates that one of that asset's children has a high-level alert. You can expand the tree to drill down to the actual asset against which the alert was raised.

Figure 2-22 illustrates a red octagon at the highest level, Enterprise Security Assets, and for a child asset (IT group), and its child asset (DB Server). For more information about the status icons, see "Understanding and Using Threat and Risk Views", on page 71.

Security Alerts: Working with the Alerts Table

The alerts table, which displays to the right of the asset tree, displays alerts raised at your site. The alerts are categorized by the assets defined at your site. An alert for which there is no defined asset displays in the **Default** asset group.

When you select an asset in the tree, the alerts table lists only the alerts raised against that asset. When you select an asset group, the table lists the alerts raised against all assets in the group.

Viewing the Information in a Security Alert

The Security Alerts table includes the following information for each security alert:

- **Acknowledged**—a checkbox indicates whether the alert has already been viewed; for more information, see "Acknowledging Viewed Alerts", on page 77
- **Timestamp**—the time the alert was raised
- **Name**—the alerting rule that generated the alert
- **Source IP address and source port**—identified by the alert
- **Destination IP address and destination port**— identified by the alert
- **Source and destination users**—identified by the alert
- **ID**—primary type of the raising rule's event

This field was used by legacy Parser and Correlator rules and has no special meaning for alerts triggered by the HawkEye AP Event Processing Language.

- **SubID**—secondary type of the raising rule's event

  This field was used by legacy Parser and Correlator rules and has no special meaning for alerts triggered by the HawkEye AP Event Processing Language.

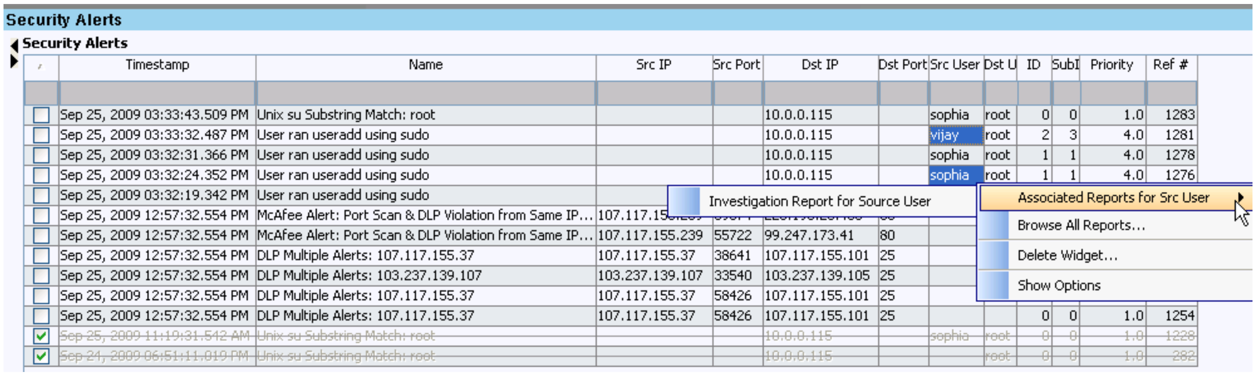
- **priority**—the threat of the alert as set by the rule or report that raised the alert
- **reference number**—unique identifier for the security alert

Launching Associated Reports

The Security Alerts widget enables you to launch associated reports to investigate element(s) of an interesting alert. You launch associated reports by selecting and right-clicking one or more values in one of six columns in the Security Alerts widget: Src IP, Dst IP, Src User, Dst User, Src Port, and Dst Port. Figure 2-23 illustrates launching an associated report on two source users (`vijay` and `sophia`) in an alert triggered by the `User ran useradd using sudo` rule.

To launch the report, right-click the desired cell(s) in one column and select the desired report from options provided.

Figure 2-23: Launching an Associated Report



Security Alerts

Security Alerts

	Timestamp	Name	Src IP	Src Port	Dst IP	Dst Port	Src User	Dst U	ID	SubI	Priority	Ref #
□	Sep 25, 2009 03:33:43.509 PM	Unix su Substring Match: root			10.0.0.115		sophia	root	0	0	1.0	1283
□	Sep 25, 2009 03:33:32.487 PM	User ran useradd using sudo			10.0.0.115		vijay	root	2	3	4.0	1281
□	Sep 25, 2009 03:32:31.366 PM	User ran useradd using sudo			10.0.0.115		sophia	root	1	1	4.0	1278
□	Sep 25, 2009 03:32:24.352 PM	User ran useradd using sudo			10.0.0.115		sophia	root	1	1	4.0	1276
□	Sep 25, 2009 03:32:19.342 PM	User ran useradd using sudo										
□	Sep 25, 2009 12:57:32.554 PM	McAfee Alert: Port Scan & DLP Violation from Same IP...	107.117.15									
□	Sep 25, 2009 12:57:32.554 PM	McAfee Alert: Port Scan & DLP Violation from Same IP...	107.117.155.239	55722	99.247.173.41	80						
□	Sep 25, 2009 12:57:32.554 PM	DLP Multiple Alerts: 107.117.155.37	107.117.155.37	38641	107.117.155.101	25						
□	Sep 25, 2009 12:57:32.554 PM	DLP Multiple Alerts: 103.237.139.107	103.237.139.107	33540	103.237.139.105	25						
□	Sep 25, 2009 12:57:32.554 PM	DLP Multiple Alerts: 107.117.155.37	107.117.155.37	58426	107.117.155.101	25						
□	Sep 25, 2009 12:57:32.554 PM	DLP Multiple Alerts: 107.117.155.37	107.117.155.37	58426	107.117.155.101	25			0	0	1.0	1254
☑	~~Sep 25, 2009 11:19:31.542 AM~~	~~Unix su Substring Match: root~~			~~10.0.0.115~~		~~sophia~~	~~root~~	~~0~~	~~0~~	~~1.0~~	~~1228~~
☑	~~Sep 24, 2009 06:51:11.019 PM~~	~~Unix su Substring Match: root~~			~~10.0.0.115~~			~~root~~	~~0~~	~~0~~	~~1.0~~	~~282~~

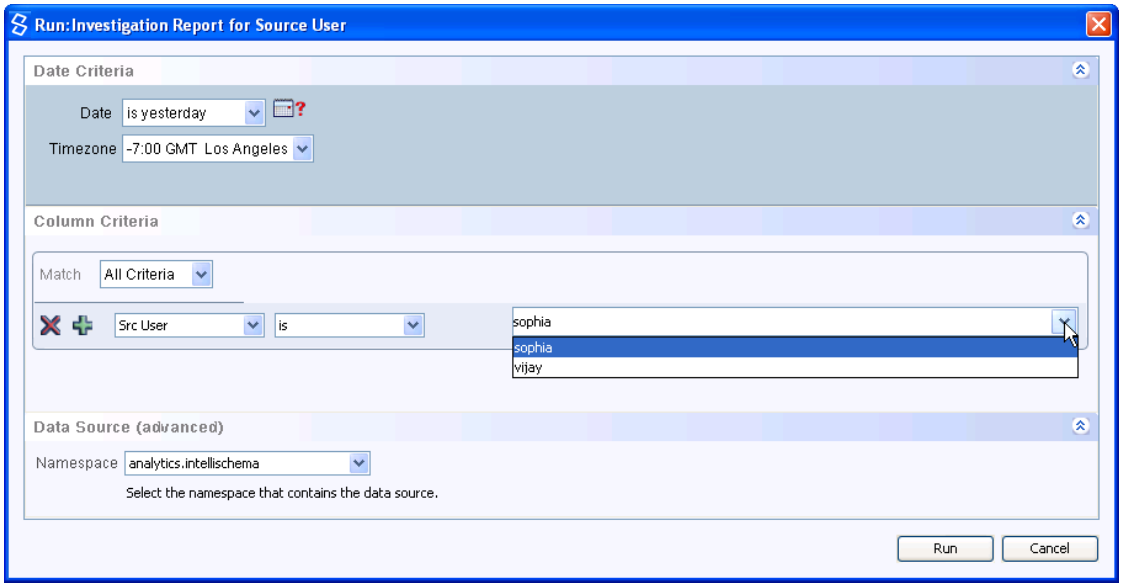
Investigation Report for Source User
Associated Reports for Src User
Browse All Reports...
Delete Widget...
Show Options

Figure 2-23 illustrates a popup menu that offers a single associated report. This report, whose title indicates that it is specific to the Src User column, is a Wizard report created by an analyst at our example site.

NOTE: When you associate a report to an alert, Wizard reports serve the purpose better than SQL reports because the column-criteria fields in a Wizard report are much more flexible than those in a SQL report. For information on the limitations of SQL report criteria fields, see “Defining Parameters for an Associated Report”, on page 194.

After you select the desired report, its **Run** dialog displays as illustrated in Figure 2-24.

Figure 2-24: Run Dialog for Associated Report



Run:Investigation Report for Source User

Date Criteria

Date: is yesterday

Timezone: -7:00 GMT Los Angeles

Column Criteria

Match: All Criteria

Src User is sophia

Data Source (advanced)

Namespace: analytics.intellischema

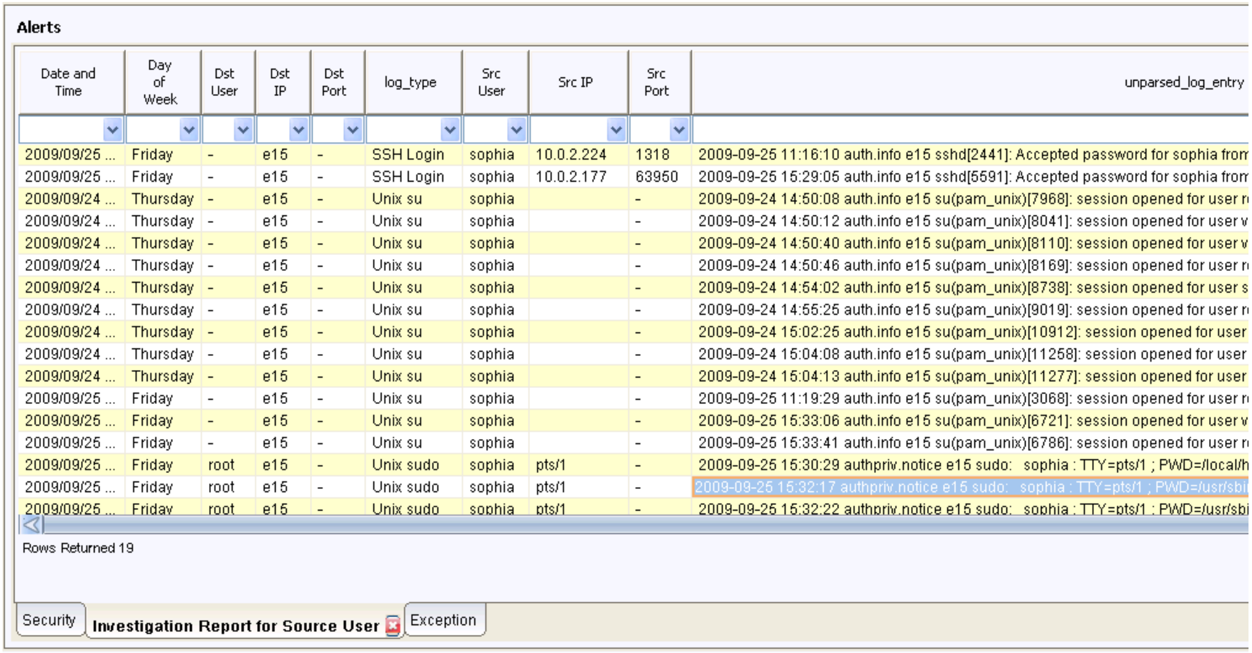
Select the namespace that contains the data source.

- If the selected report displays more than one column-criteria field, your selected column value(s) from the alert automatically display in all column-criteria fields. You can remove each irrelevant row by clicking the ✖ icon.
- If the selected report displays only one column-criteria field, your selected column value(s) from the alert automatically display in that field.
  - If you selected more than one value from the alert, all values from the selected column automatically display in the column-criteria value field. As illustrated above, you can use the value dropdown to select the desired value for each report run.
  - If the column field represents a different column from the one you have selected or you wish to investigate a different column from the one selected, you can select the desired column from the column dropdown.
  - If you want to investigate more than the one column, you can add criteria row(s) by clicking the ✚ icon.

If desired, you can change the date period and time zone. When the values are correct, click **Run**.

As illustrated in Figure 2-25, the report displays on a new page in the same dashboard as the alerts. The report contains only rows relevant to the selected column value.

Figure 2-25: Display of Associated Report



Alerts

Date and Time	Day of Week	Dst User	Dst IP	Dst Port	log_type	Src User	Src IP	Src Port	unparsed_log_entry
2009/09/25 ...	Friday	-	e15	-	SSH Login	sophia	10.0.2.224	1318	2009-09-25 11:16:10 auth.info e15 sshd[2441]: Accepted password for sophia from
2009/09/25 ...	Friday	-	e15	-	SSH Login	sophia	10.0.2.177	63950	2009-09-25 15:29:05 auth.info e15 sshd[5591]: Accepted password for sophia from
2009/09/24 ...	Thursday	-	e15	-	Unix su	sophia		-	2009-09-24 14:50:08 auth.info e15 su(pam_unix)[7968]: session opened for user r
2009/09/24 ...	Thursday	-	e15	-	Unix su	sophia		-	2009-09-24 14:50:12 auth.info e15 su(pam_unix)[8041]: session opened for user v
2009/09/24 ...	Thursday	-	e15	-	Unix su	sophia		-	2009-09-24 14:50:40 auth.info e15 su(pam_unix)[8110]: session opened for user v
2009/09/24 ...	Thursday	-	e15	-	Unix su	sophia		-	2009-09-24 14:50:46 auth.info e15 su(pam_unix)[8169]: session opened for user r
2009/09/24 ...	Thursday	-	e15	-	Unix su	sophia		-	2009-09-24 14:54:02 auth.info e15 su(pam_unix)[8738]: session opened for user s
2009/09/24 ...	Thursday	-	e15	-	Unix su	sophia		-	2009-09-24 14:55:25 auth.info e15 su(pam_unix)[9019]: session opened for user r
2009/09/24 ...	Thursday	-	e15	-	Unix su	sophia		-	2009-09-24 15:02:25 auth.info e15 su(pam_unix)[10912]: session opened for user
2009/09/24 ...	Thursday	-	e15	-	Unix su	sophia		-	2009-09-24 15:04:08 auth.info e15 su(pam_unix)[11258]: session opened for user
2009/09/24 ...	Thursday	-	e15	-	Unix su	sophia		-	2009-09-24 15:04:13 auth.info e15 su(pam_unix)[11277]: session opened for user
2009/09/25 ...	Friday	-	e15	-	Unix su	sophia		-	2009-09-25 11:19:29 auth.info e15 su(pam_unix)[3068]: session opened for user r
2009/09/25 ...	Friday	-	e15	-	Unix su	sophia		-	2009-09-25 15:33:06 auth.info e15 su(pam_unix)[6721]: session opened for user v
2009/09/25 ...	Friday	-	e15	-	Unix su	sophia		-	2009-09-25 15:33:41 auth.info e15 su(pam_unix)[6786]: session opened for user r
2009/09/25 ...	Friday	root	e15	-	Unix sudo	sophia	pts/1	-	2009-09-25 15:30:29 authpriv.notice e15 sudo: sophia : TTY=pts/1 ; PWD=/local/h
2009/09/25 ...	Friday	root	e15	-	Unix sudo	sophia	pts/1	-	2009-09-25 15:32:17 authpriv.notice e15 sudo: sophia : TTY=pts/1 ; PWD=/usr/sbi
2009/09/25 ...	Friday	root	e15	-	Unix sudo	sophia	pts/1	-	2009-09-25 15:32:22 authpriv.notice e15 sudo: sophia : TTY=pts/1 ; PWD=/usr/sbi

Rows Returned 19

HawkEye AP Console automatically creates a new dashboard page to display the associated report, as illustrated above.

Understanding and Using Threat and Risk Views

This section covers the following topics:

- “Background”, next
- “Understanding Threat and Risk Views”, on page 73
- “Using Threat View”, on page 75
- “Using Risk View”, on page 75

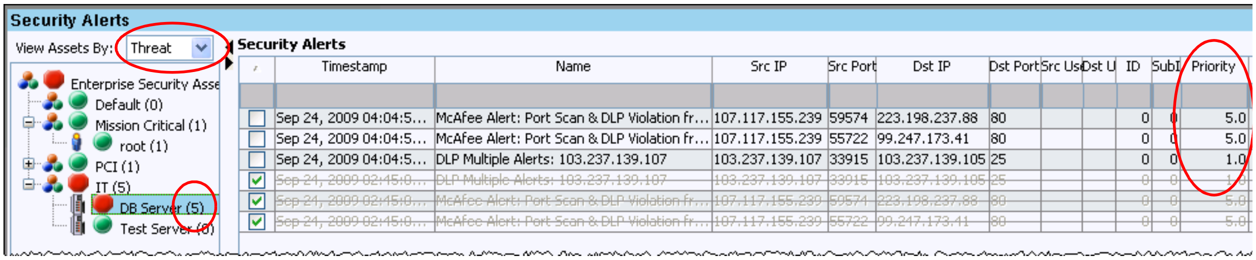
Background

As described in “Security Alerts: Working with the Asset Tree”, on page 67, the asset tree displays colored status icons that indicate the level of the security threat or risk for specific assets and their child assets.

By default, the asset tree displays **Threat** view, which indicates threat level based upon the priority of the parsing and alerting rules that triggered the alert. You can easily toggle display to **Risk** view, which indicates threat level based upon the priority level of the asset as well as the priority of the rules that triggered the alert.

Each rule that HawkEye AP delivers has a pre-set priority. Typically the priority is set low. Figure 2-26 illustrates the **Priority** column for an alert table whose rules were triggered by two different rules. One has a priority of 5.0, while the other rule a priority of 1.0.

Figure 2-26: Priority in Threat View



Security Alerts

View Assets By: Threat

Security Alerts

	Timestamp	Name	Src IP	Src Port	Dst IP	Dst Port	Src Us	Dst U	ID	SubI	Priority
☐	Sep 24, 2009 04:04:5...	McAfee Alert: Port Scan & DLP Violation fr...	107.117.155.239	59574	223.198.237.88	80			0	0	5.0
☐	Sep 24, 2009 04:04:5...	McAfee Alert: Port Scan & DLP Violation fr...	107.117.155.239	55722	99.247.173.41	80			0	0	5.0
☐	Sep 24, 2009 04:04:5...	DLP Multiple Alerts: 103.237.139.107	103.237.139.107	33915	103.237.139.105	25			0	0	1.0
☑	~~Sep 24, 2009 02:45:0...~~	~~DLP Multiple Alerts: 103.237.139.107~~	~~103.237.139.107~~	~~33915~~	~~103.237.139.105~~	~~25~~			~~0~~	~~0~~	~~1.0~~
☑	~~Sep 24, 2009 02:45:0...~~	~~McAfee Alert: Port Scan & DLP Violation fr...~~	~~107.117.155.239~~	~~59574~~	~~223.198.237.88~~	~~80~~			~~0~~	~~0~~	~~5.0~~
☑	~~Sep 24, 2009 02:45:0...~~	~~McAfee Alert: Port Scan & DLP Violation fr...~~	~~107.117.155.239~~	~~55722~~	~~99.247.173.41~~	~~80~~			~~0~~	~~0~~	~~5.0~~

As noted earlier, a high-priority alert causes a red octagon to display in the Asset Tree. As illustrated above by a red circle, a numeric priority level also displays next to each asset's name. In the example above, the DB Server asset displays alerts whose rules had a priority of 5.0 and 1.0. The value `5` displays within parentheses next to the asset name. This number does not reflect the number of alerts triggered for the asset. Instead, it indicates the highest priority alert for the asset. The parent asset, IT group, also indicates the highest priority triggered for all of its child assets.

NOTE:

- HawkEye AP parsing and alerting rules set a default priority for each alert.

  The parsing and alerting rules that HawkEye AP provides set the priority of security alerts individually. HawkEye AP expects each customer to change the priority of specific rules to fit their environments and businesses. Customers should increase this default priority for important alerts that are likely to occur and decrease the priority for alerts that are unimportant to their businesses. Before modifying alert priorities, consider your actual deployment characteristics, such as the number and type of IDs and firewalls. You need to consider varying volumes of alert messages and potentially misleading inputs.

- A developer can modify HawkEye AP alerting rule templates to enable analysts to set a specific priority for individual alerts.

  Typically all the alerting rules that you create from a single template display the same priority. This is because all rules created from the same template receive their data from the same parsing rule. However, should your site determine that individual alerting rules should have different priority settings, a developer can modify the alerting rule-template code to enable users to set priority on a rule-by-rule basis.

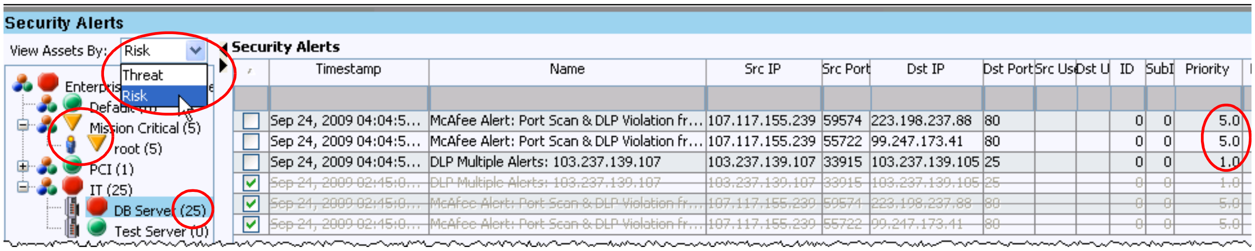
  For example, assume you create two alerting rules from the `Unix su Substring Match Rule`. Each of your rules matches a different string. Your site considers one string match to have significantly higher priority than the other. One of your developers can modify the template for the `Unix su Substring Match Rule` so that your analysts can set priority individually for the different rules that they create from the template.

For more information on modifying parsing rules, alerting rules, and alerting rule templates, see the *HawkEye Event Processing Language Developer Guide*.

Understanding Threat and Risk Views

Figure 2-26 illustrates the default display of the Security Alerts widget. The colored status icons and numeric priority values all reflect the settings displayed for **Threat** view. If you do nothing but select **Risk** view from the dropdown, the display changes considerably. Figure 2-27 illustrates the display of the same alert table but with **Risk** view selected.

Figure 2-27: Priority in Risk View



	Timestamp	Name	Src IP	Src Port	Dst IP	Dst Port	Src Us	Dst U	ID	SubI	Priority
□	Sep 24, 2009 04:04:5...	McAfee Alert: Port Scan & DLP Violation fr...	107.117.155.239	59574	223.198.237.88	80			0	0	5.0
□	Sep 24, 2009 04:04:5...	McAfee Alert: Port Scan & DLP Violation fr...	107.117.155.239	55722	99.247.173.41	80			0	0	5.0
□	Sep 24, 2009 04:04:5...	DLP Multiple Alerts: 103.237.139.107	103.237.139.107	33915	103.237.139.105	25			0	0	1.0
☑	~~Sep 24, 2009 02:45:0...~~	~~DLP Multiple Alerts: 103.237.139.107~~	~~103.237.139.107~~	~~33915~~	~~103.237.139.105~~	~~25~~			~~0~~	~~0~~	~~1.0~~
☑	~~Sep 24, 2009 02:45:0...~~	~~McAfee Alert: Port Scan & DLP Violation fr...~~	~~107.117.155.239~~	~~59574~~	~~223.198.237.88~~	~~80~~			~~0~~	~~0~~	~~5.0~~
☑	~~Sep 24, 2009 02:45:0...~~	~~McAfee Alert: Port Scan & DLP Violation fr...~~	~~107.117.155.239~~	~~55722~~	~~99.247.173.41~~	~~80~~			~~0~~	~~0~~	~~5.0~~

As illustrated above, an asset that had displayed a green circle icon in Threat view now displays a yellow triangle in Risk view. Additionally, even though the Priority column displays the same alert priority values, the values displayed within parentheses beside many assets have now increased considerably.

When you switch your view of the asset tree between Threat and Risk views, status icons and numeric values reflect priorities differently:

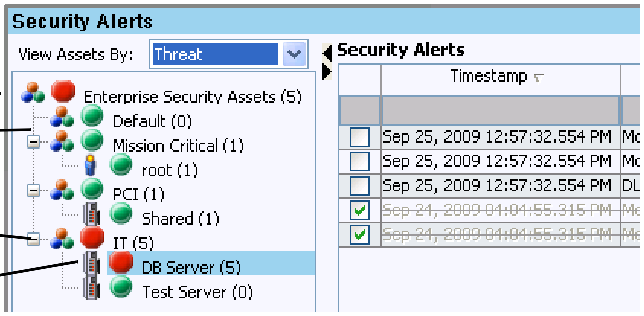
- **Threat view**—displays icons and numeric values based solely on the *priority* set in the rules that raise the alerts.
- **Risk view**—displays icons and numeric values based on the priority of alerts multiplied by the *value* of each asset as specified by your enterprise for the assets.

The table below lists the available priority values, their text equivalents, and their significance.

Priority	Meaning	Denotes
1	normal	no problem
2	error	an error condition rather than a threat level
3	warning	a low threat level
4	alert	a serious threat level
5	critical	the highest threat level

Figure 2-28 illustrates the Security Alerts widget in Threat view.

Figure 2-28: Security Alerts Widget in Threat View



Top-level node displays the highest threat found anywhere in the asset tree

Round green icons indicate no alerts for the assets in Threat view

Highest threat within a group displays by the asset group

Threatened asset displays threat icon

Security Alerts

View Assets By: Threat

- Enterprise Security Assets (5)
  - Default (0)
  - Mission Critical (1)
    - root (1)
  - PCI (1)
    - Shared (1)
  - IT (5)
    - DB Server (5)
    - Test Server (0)

Security Alerts

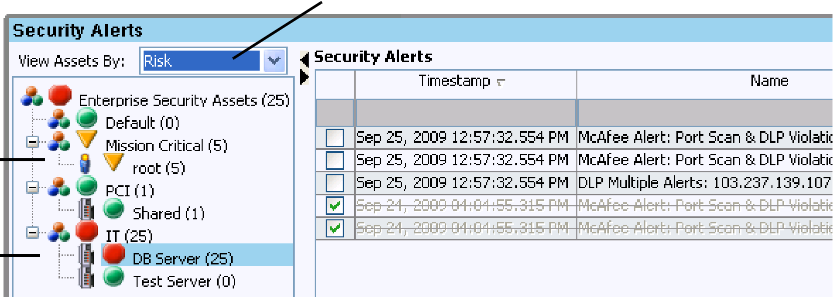
	Timestamp	
□	Sep 25, 2009 12:57:32.554 PM	Mc
□	Sep 25, 2009 12:57:32.554 PM	Mc
□	Sep 25, 2009 12:57:32.554 PM	DL
☑	~~Sep 24, 2009 04:04:55.315 PM~~	~~Mc~~
☑	~~Sep 24, 2009 04:04:55.315 PM~~	~~Mc~~

- A red octagon indicates the asset and its child assets have associated alerts with serious threat or risk.
- A yellow triangle indicates the asset and its child assets have associated alerts with moderate threat or risk.
- A green circle indicates the asset and its child assets have no alerts in Threat view and no alerts or only ones with very low risk in the Risk view.
- Threat is based solely on the priority of alerts (range 1 to 5) raised against the asset or its child assets. If multiple alerts are raised against an asset and the alerts have different priorities, the asset's threat level represents the alert with the highest priority.

Use the **View Assets By** dropdown (located at the top of the asset tree) to switch between Threat and Risk views. Figure 2-29 illustrates the Security Alerts widget in Risk view.

Figure 2-29: Security Alerts Widget in Risk View

*Use the **View Assets By** dropdown to switch between views.*



A green circle in Threat view becomes a yellow

A red octagon in Threat view remains a red octagon in

Security Alerts

View Assets By: Risk

- Enterprise Security Assets (25)
  - Default (0)
  - Mission Critical (5)
    - root (5)
  - PCI (1)
    - Shared (1)
  - IT (25)
    - DB Server (25)
    - Test Server (0)

Security Alerts

	Timestamp	Name
□	Sep 25, 2009 12:57:32.554 PM	McAfee Alert: Port Scan & DLP Violatic
□	Sep 25, 2009 12:57:32.554 PM	McAfee Alert: Port Scan & DLP Violatic
□	Sep 25, 2009 12:57:32.554 PM	DLP Multiple Alerts: 103.237.139.107
☑	~~Sep 24, 2009 04:04:55.315 PM~~	~~McAfee Alert: Port Scan & DLP Violatic~~
☑	~~Sep 24, 2009 04:04:55.315 PM~~	~~McAfee Alert: Port Scan & DLP Violatic~~

HawkEye AP Real-Time bases Risk on the priority of alerts as well as the value of the asset (range 0 to 5). The alerts widget calculates and displays risk differently for leaf assets and for asset groups:

- For a leaf asset, the risk level is the asset's value multiplied by the priority of its alerts. If multiple alerts are raised against an asset and the alerts have different priorities, the risk level is the asset's value multiplied by the highest-priority alert.

  Therefore, risk values have a range of 0 to 25 (multiplying alert priority, in the range 1 to 5, by the asset value in the range 0 to 5). In the example above, the root asset in the Mission Critical asset group displays a yellow triangle and the value `5` within parentheses. In Threat view, it displayed a green circle and the value `1` within parentheses. The increased risk that the triangle and numeric value indicate reflect the value of the root asset. The priority set by the rule has not changed.

For information about setting an asset's value, see Chapter 10: Administering Assets and Monitoring Alerts in the *Administration Guide*.

- For group assets, the risk level is the same as that of its highest risk child asset.

NOTE: The ranking of assets by risk can differ from their ranking by threat. For example, if one server is ranked a lower risk than a sibling server, their associated icons will differ in Risk view. In the Threat view, however, which ignores asset value, the two assets display the same icon and numeric value.

Using Threat View

Threat is based solely on the priority of alerts raised against the asset or its child assets. The higher the priority of an alert as set by its parsing or alerting rule, the higher the threat indicated for it. For example, one parsing rule might assign a high priority to an alert that indicates an intrusion on a security server and another parsing rule might assign a low priority to an alert that indicates failed logons.

The table below describes threat levels, based on alert priority.

Status Icon	Threat Level	Significance
🛑	5	denotes the highest threat level
	4	denotes a serious threat level
	3	denotes a low threat level
▼	2	denotes an error condition rather than a threat level
	1	denotes a possible threat
🟢		no alerts

The Security Alerts widget computes and displays the highest priority alert raised against an IP List or User asset. The browser computes and displays the threat to an asset group as the highest threat of its child assets.

Using Risk View

The higher the value placed on one asset over another, the higher the risk posed by alerts with the same priority. For example, a failed password attempt on your payroll system may pose more of risk than a failed password attempt on your email system.

The table below describes risk levels based on alert priority and asset value.

Status Icon	Significance
🛑	denotes the highest levels of risk

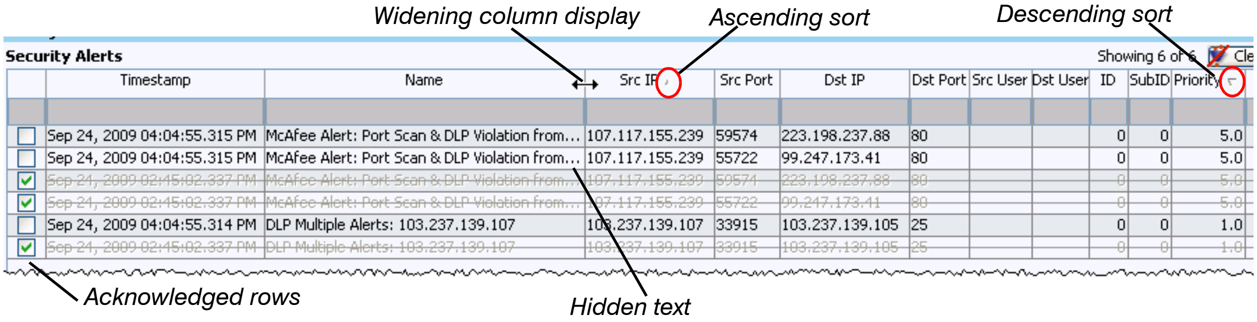
Status Icon	Significance
▼	denotes moderate levels of risk
●	denotes no alerts or alerts on assets with no value. **TIP:** To make the Risk view as meaningful as possible, assign some value to all assets.

The Security Alerts widget displays the risk to an asset group as the highest displayed risk of its child assets.

Sorting Alert Data

By default, alerts are listed in the order they are raised. You can change the order on one or more columns. For example, the alerts illustrated in Figure 2-30 are sorted on **Priority** as the primary column and **Source IP** as the secondary column. Sorting on both columns displays the source IP addresses grouped by priority. Within each group of IP addresses, alerts are sorted in the order they were raised.

Figure 2-30: Sorting and Widening Columns



Security Alerts

Showing 6 of 6

	Timestamp	Name	Src IP	Src Port	Dst IP	Dst Port	Src User	Dst User	ID	SubID	Priority
□	Sep 24, 2009 04:04:55.315 PM	McAfee Alert: Port Scan & DLP Violation from...	107.117.155.239	59574	223.198.237.88	80			0	0	5.0
□	Sep 24, 2009 04:04:55.315 PM	McAfee Alert: Port Scan & DLP Violation from...	107.117.155.239	55722	99.247.173.41	80			0	0	5.0
☑	~~Sep 24, 2009 02:45:02.337 PM~~	~~McAfee Alert: Port Scan & DLP Violation from...~~	~~107.117.155.239~~	~~59574~~	~~223.198.237.88~~	~~80~~			~~0~~	~~0~~	~~5.0~~
☑	~~Sep 24, 2009 02:45:02.337 PM~~	~~McAfee Alert: Port Scan & DLP Violation from...~~	~~107.117.155.239~~	~~55722~~	~~99.247.173.41~~	~~80~~			~~0~~	~~0~~	~~5.0~~
□	Sep 24, 2009 04:04:55.314 PM	DLP Multiple Alerts: 103.237.139.107	103.237.139.107	33915	103.237.139.105	25			0	0	1.0
☑	~~Sep 24, 2009 02:45:02.337 PM~~	~~DLP Multiple Alerts: 103.237.139.107~~	~~103.237.139.107~~	~~33915~~	~~103.237.139.105~~	~~25~~			~~0~~	~~0~~	~~1.0~~

The direction of the sort triangle indicates whether the order is ascending or descending. Columns whose headings do not display a triangle are listed in the order they were raised.

To sort alerts based on data in a single column

1 Click the column header.

The alerts are sorted in the ascending order of the values in that column.

2 Click the column header again.

The alerts are sorted in the opposite, descending order.

3 Click the column a third time

The alerts display in ascending order.

To sort alerts based on data in several columns

1. Click the column header of your primary-sort column.

   If you want the primary-sort column to be in descending order, click the column header again.

2. Hold CTRL and click the column header of your secondary-sort column.

   To sort the secondary-sort column in descending order, CTRL+CLICK the column header again.

3. To sort on additional columns, repeat Step 2.

Widening Column Display in an Alert Widget

Sometimes an alert table has more columns than can fully display. When column width is so narrow that some header text or column data is hidden, the alert table indicates hidden text with three dots, as illustrated in Figure 2-30.

To widen column display

1. Position the cursor over the column header until the sizing cursor displays, as illustrated in Figure 2-30.

2. Click and drag the cursor until the column reaches the desired width.

Acknowledging Viewed Alerts

The first column of every alert widget is the unnamed Acknowledged column. This column displays a select box for each row. Click the select box of desired row(s) to indicate that you have viewed the alert. The display of an acknowledged row changes in the following ways:

- The text is greyed.
- A line is drawn through the text.
- The row moves to the bottom of the alert-widget table (unless the rows are sorted, as illustrated in Figure 2-30).

NOTE: To return an acknowledged row to unacknowledged status, click its select box. The row returns to its original location without the line through the text and without greyed text.

Filtering Alert Data

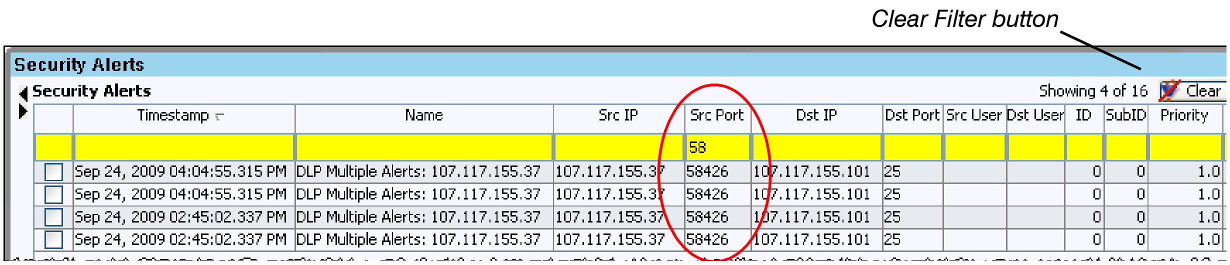
By default, HawkEye AP clears the alerts table hourly of all but the 300 most recent alerts, and every eight days it clears those alerts that have displayed for those eight days. Although alerts disappear from display after the interface has been refreshed, HawkEye AP stores them in the EDW.

NOTE: The values above are configurable; for more information, contact Hexis Cyber Solutions Technical Support.

You can filter the alerts to display only those alerts that interest you. The alert table displays a filter row between the column headers and the list of alerts. Use this row to specify your filtering criteria.

For example, the alerts table illustrated Figure 2-31 shows only alerts where the Source Port column contains the value “58”.

Figure 2-31: Filtering Alert Data



Clear Filter button

Security Alerts

Security Alerts

Showing 4 of 16 Clear

	Timestamp	Name	Src IP	Src Port	Dst IP	Dst Port	Src User	Dst User	ID	SubID	Priority
				58							
□	Sep 24, 2009 04:04:55.315 PM	DLP Multiple Alerts: 107.117.155.37	107.117.155.37	58426	107.117.155.101	25			0	0	1.0
□	Sep 24, 2009 04:04:55.315 PM	DLP Multiple Alerts: 107.117.155.37	107.117.155.37	58426	107.117.155.101	25			0	0	1.0
□	Sep 24, 2009 02:45:02.337 PM	DLP Multiple Alerts: 107.117.155.37	107.117.155.37	58426	107.117.155.101	25			0	0	1.0
□	Sep 24, 2009 02:45:02.337 PM	DLP Multiple Alerts: 107.117.155.37	107.117.155.37	58426	107.117.155.101	25			0	0	1.0

The filtering row is colored yellow when filtering has been specified. The color informs you that the alerts table is displaying a subset of alerts. The filtering row is colored grey when the alerts table is not filtered.

To filter alerts data

1. Click the cell in the filtering row below the column header for which you want to specify a criterion.

   The cell becomes shaded blue to indicate that you selected the cell.

2. Type text in the cell, and press ENTER.

   The table displays only alerts that match the criterion in the column you specified. In addition, the filtering row becomes shaded yellow and the **Clear Filter** button becomes enabled.

3. To add filtering criteria for additional columns, repeat Step 1 and Step 2.

To clear the filter on the alerts table and show all alerts

Click **Clear Filter** at the top right above the table header.

The table displays all alerts, and the filtering row becomes grey in color.

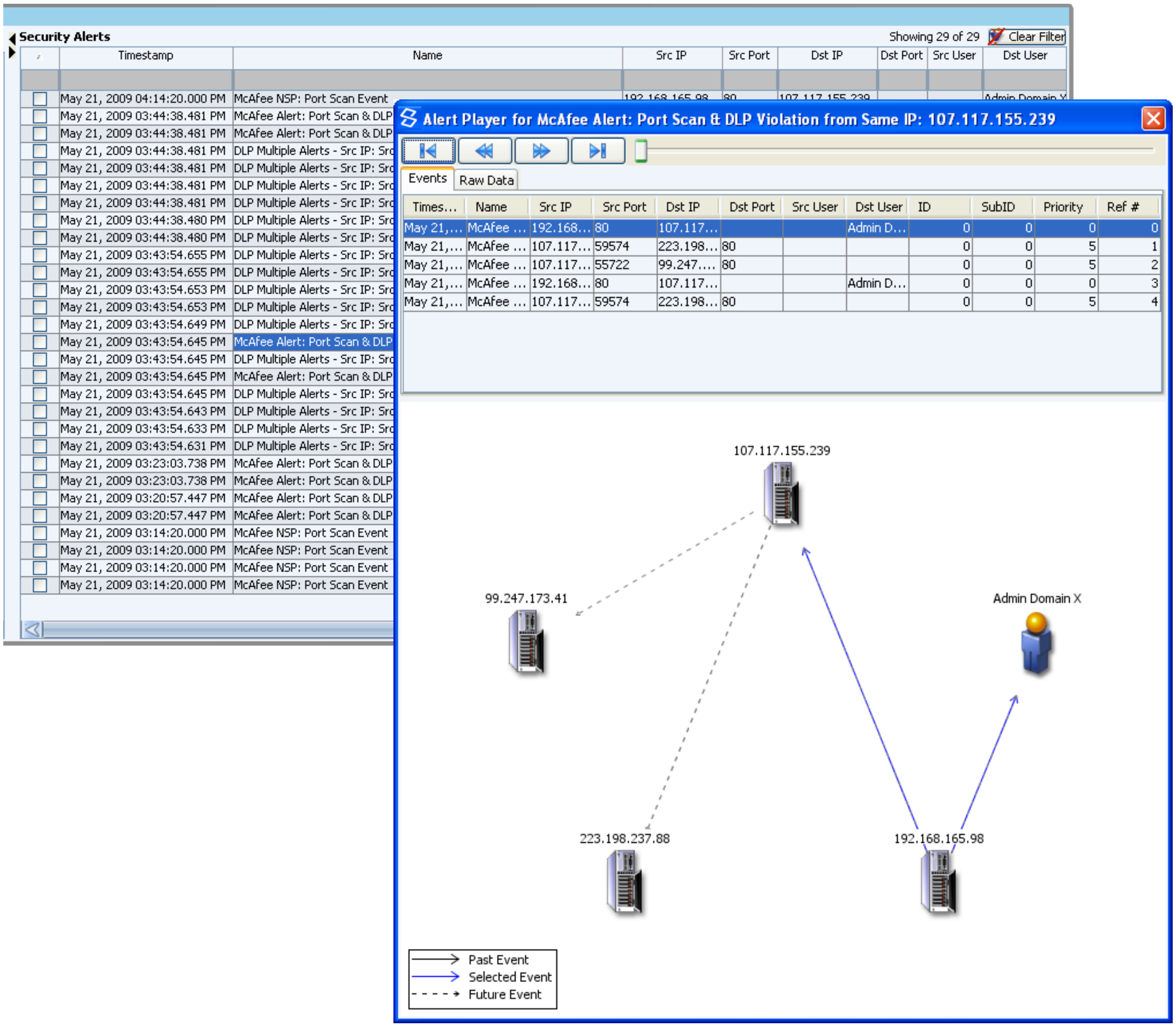
Security Alerts: Using the Alert Player

When you find an alert that requires attention, you can investigate the sequence of events that contributed to it. Double-click its row in the alert table to open the Alert Player and display the contributing events for the alert. As illustrated in Figure 2-32, the Alert Player displays a table of the events above and a graphic diagram of the events below. Use the table and associated diagrams to determine the following information:

- external source hosts or users identified as the attacker in an alert
- internal hosts or users identified as the target in an alert

- historical data about the source and destination hosts and the source and destination users

Figure 2-32: Displaying the Alert Player



Security Alerts

Showing 29 of 29 Clear Filter

	Timestamp	Name	Src IP	Src Port	Dst IP	Dst Port	Src User	Dst User
☐	May 21, 2009 04:14:20.000 PM	McAfee NSP: Port Scan Event	192.168.165.98	80	107.117.155.239			Admin Domain X
☐	May 21, 2009 03:44:38.481 PM	McAfee Alert: Port Scan & DLP						
☐	May 21, 2009 03:44:38.481 PM	McAfee Alert: Port Scan & DLP						
☐	May 21, 2009 03:44:38.481 PM	DLP Multiple Alerts - Src IP: Src						
☐	May 21, 2009 03:44:38.481 PM	DLP Multiple Alerts - Src IP: Src						
☐	May 21, 2009 03:44:38.481 PM	DLP Multiple Alerts - Src IP: Src						
☐	May 21, 2009 03:44:38.481 PM	DLP Multiple Alerts - Src IP: Src						
☐	May 21, 2009 03:44:38.480 PM	DLP Multiple Alerts - Src IP: Src						
☐	May 21, 2009 03:44:38.480 PM	DLP Multiple Alerts - Src IP: Src						
☐	May 21, 2009 03:43:54.655 PM	DLP Multiple Alerts - Src IP: Src						
☐	May 21, 2009 03:43:54.655 PM	DLP Multiple Alerts - Src IP: Src						
☐	May 21, 2009 03:43:54.653 PM	DLP Multiple Alerts - Src IP: Src						
☐	May 21, 2009 03:43:54.653 PM	DLP Multiple Alerts - Src IP: Src						
☐	May 21, 2009 03:43:54.649 PM	DLP Multiple Alerts - Src IP: Src						
☐	May 21, 2009 03:43:54.645 PM	McAfee Alert: Port Scan & DLP						
☐	May 21, 2009 03:43:54.645 PM	DLP Multiple Alerts - Src IP: Src						
☐	May 21, 2009 03:43:54.645 PM	McAfee Alert: Port Scan & DLP						
☐	May 21, 2009 03:43:54.645 PM	DLP Multiple Alerts - Src IP: Src						
☐	May 21, 2009 03:43:54.643 PM	DLP Multiple Alerts - Src IP: Src						
☐	May 21, 2009 03:43:54.633 PM	DLP Multiple Alerts - Src IP: Src						
☐	May 21, 2009 03:43:54.631 PM	DLP Multiple Alerts - Src IP: Src						
☐	May 21, 2009 03:23:03.738 PM	McAfee Alert: Port Scan & DLP						
☐	May 21, 2009 03:23:03.738 PM	McAfee Alert: Port Scan & DLP						
☐	May 21, 2009 03:20:57.447 PM	McAfee Alert: Port Scan & DLP						
☐	May 21, 2009 03:20:57.447 PM	McAfee Alert: Port Scan & DLP						
☐	May 21, 2009 03:14:20.000 PM	McAfee NSP: Port Scan Event						
☐	May 21, 2009 03:14:20.000 PM	McAfee NSP: Port Scan Event						
☐	May 21, 2009 03:14:20.000 PM	McAfee NSP: Port Scan Event						
☐	May 21, 2009 03:14:20.000 PM	McAfee NSP: Port Scan Event						

Alert Player for McAfee Alert: Port Scan & DLP Violation from Same IP: 107.117.155.239

Events | Raw Data

Times...	Name	Src IP	Src Port	Dst IP	Dst Port	Src User	Dst User	ID	SubID	Priority	Ref #
May 21,...	McAfee ...	192.168...	80	107.117...			Admin D...	0	0	0	0
May 21,...	McAfee ...	107.117...	59574	223.198...	80			0	0	5	1
May 21,...	McAfee ...	107.117...	55722	99.247....	80			0	0	5	2
May 21,...	McAfee ...	192.168...	80	107.117...			Admin D...	0	0	0	3
May 21,...	McAfee ...	107.117...	59574	223.198...	80			0	0	5	4

Expanding Alert Table Data

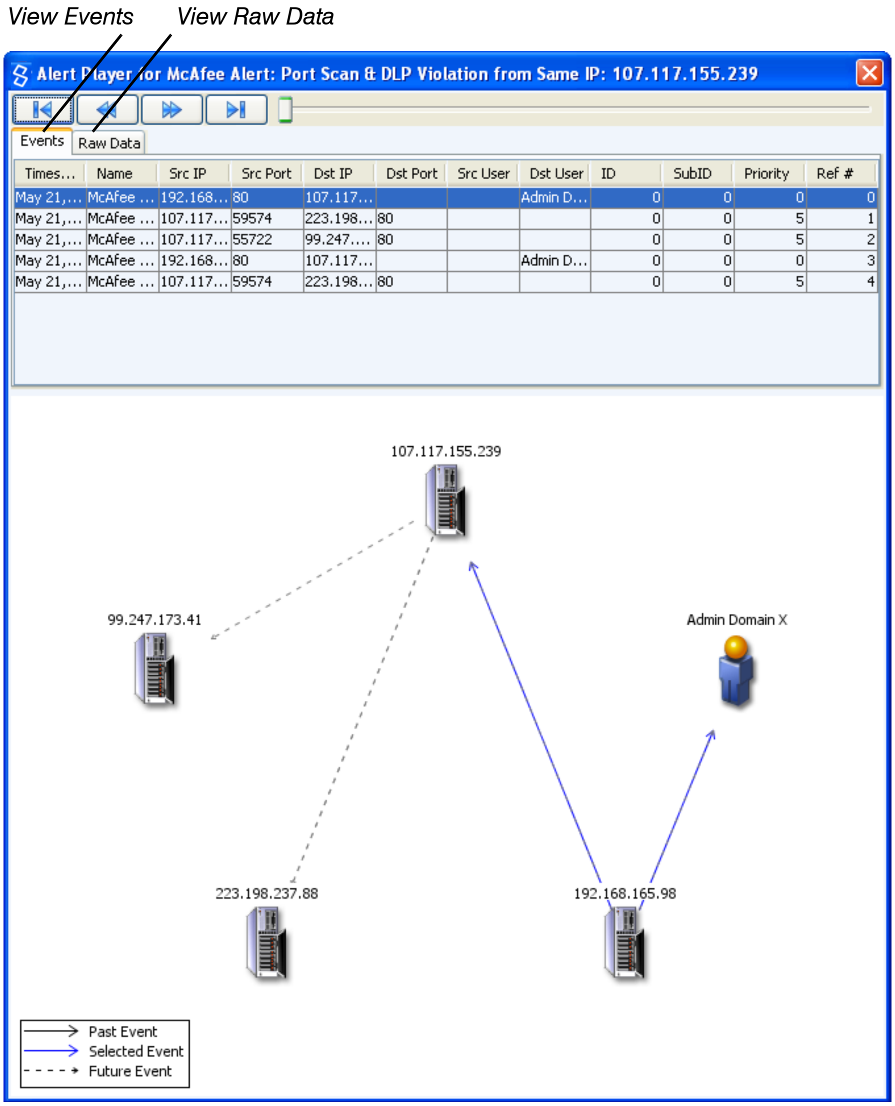
Double-click a row in the Security Alert table to launch the Alert Player in a separate window for the selected alert. Alert Player display is coordinated with row selection in the Security Alerts widget. In other words, when you select a different row in the Security Alerts widget's table or press the up or down arrow keys, the Alert Player's diagram changes to display the newly selected row.

Double-click an alert in the Alert Player's table to display the alert's contributing events in its diagram panel. Figure 2-33 illustrates an alert raised by the `McAfee Alert: Port Scan & DLP Violation from Same IP` rule. This alert has five contributing events, which display in the Alert Player table in Timestamp order. In the Alert Player, the timestamp represents the time the Parser received the event.

You cannot sort the rows in this table. To maintain the consistency of playback over time, events always appear in the order in which they contributed to the alert.

NOTE: By default, the Alert Player illustrates contributing events. Click the **Raw Data** tab to display the actual event data. For more information, see “Viewing the Raw Data”, on page 83.

Figure 2-33: Alert Raised for McAfee Alert: Port Scan & DLP Violation from Same IP



View Events *View Raw Data*

Alert Player for McAfee Alert: Port Scan & DLP Violation from Same IP: 107.117.155.239

Events | Raw Data

Times…	Name	Src IP	Src Port	Dst IP	Dst Port	Src User	Dst User	ID	SubID	Priority	Ref #
May 21,…	McAfee …	192.168…	80	107.117…			Admin D…	0	0	0	0
May 21,…	McAfee …	107.117…	59574	223.198…	80			0	0	5	1
May 21,…	McAfee …	107.117…	55722	99.247.…	80			0	0	5	2
May 21,…	McAfee …	192.168…	80	107.117…			Admin D…	0	0	0	3
May 21,…	McAfee …	107.117…	59574	223.198…	80			0	0	5	4

The Alert Player illustrated above displays all recorded contributing events for the selected alert. As indicated by the highlight in the first table row, the diagram illustrates the first contributing event.

The First Contributing Event

The diagram displays a playback of the events that led to the alert. The diagram usually includes two lines for each event: one from source user to destination user, and another from source IP to destination IP. However, if one source is missing, the other source points to both destinations; and if one destination is missing, both sources point to the other destination. In other words, as long as at least one source and one destination are known, at least one line displays.

Because the source user is unknown for the illustrated McAfee alert, the lines in the example diagram are from the source IP to destination user and destination IP.

As illustrated in Figure 2-33, the legend in the lower left of the diagram indicates:

- The blue arrows illustrate the assets involved in the current (first) contributing event. In the example, a system identified as 192.168.165.98 has broken into the machine identified as 107.117.155.239 as the Admin Domain X administrator.

  This visual information is substantiated by the values in the first and fourth rows of the Events table that displays at the top of the window. Each of these rows indicate that the system identified as 192.168.165.98 tried to break into 107.117.155.239 as the Admin Domain X administrator.

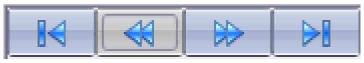
- The dashed black arrows illustrate the assets involved in future contributing events, that is, any of the contributing events below the first one in the table. In the example, a future attack originates when a system identified as 107.117.155.239 has broken into two machines identified as 99.247.173.41 and 223.198.237.88.

Because Figure 2-33 illustrates the first contributing event, the diagram does not illustrate use of solid black arrows. Such arrows represent assets involved in an event that precedes the currently displayed one. For an illustration of a solid black line, see "The Second and Subsequent Contributing Events", next.

The Second and Subsequent Contributing Events

To view the second event that contributed to the current alert, select the next row in the Alert Player table. There are several ways to move through the table:

- In Events mode table (**Events** tab selected), click the desired target row.
- In Raw Data mode (**Raw Data** tab selected), click the desired target row.
- Click the appropriate VCR button:

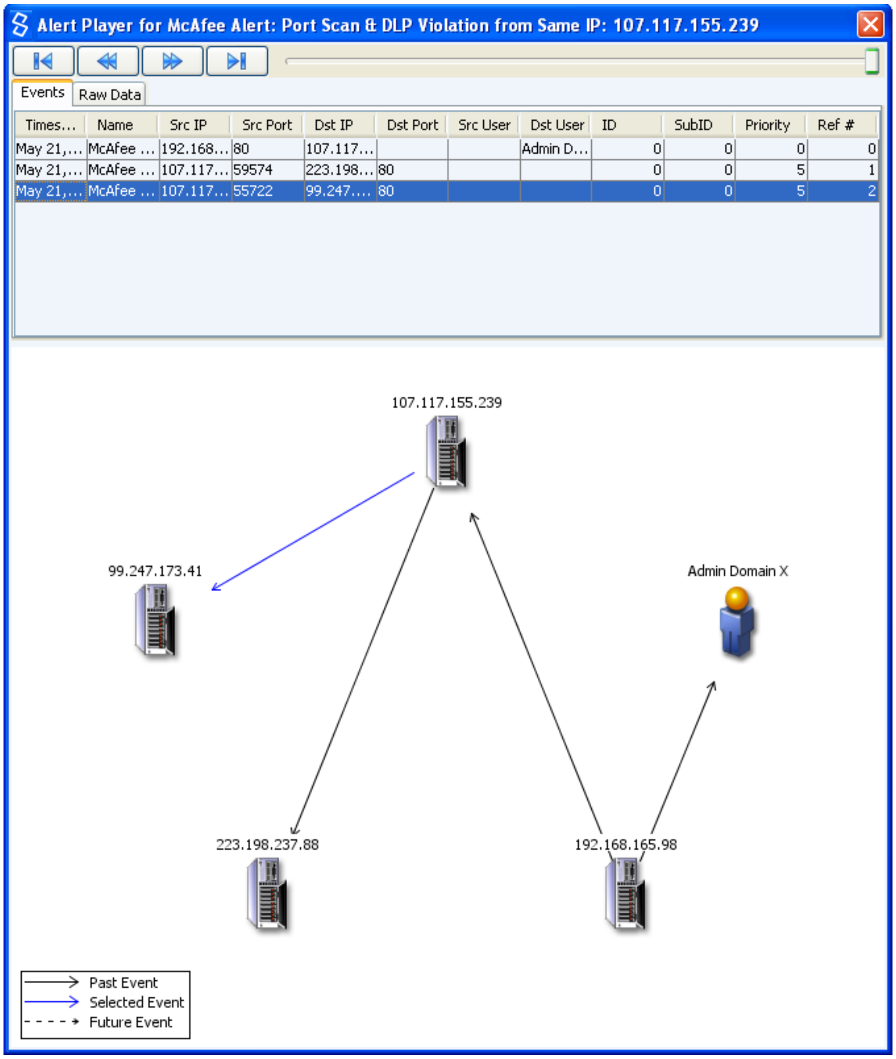


- Move the slider.
- Click an arrow in the diagram.
- Use the arrow keys (Up arrow stops moving when you reach the top of the table; Down arrow stops at the bottom).
- Use the Enter key (moves only down through the table but wraps to the first row when it reaches the bottom).
- Use Shift-Enter (moves only up through the table but wraps to the last row when it reaches the top).

As you move through events in the table, the corresponding event playback displays.

Figure 2-34 illustrates the last contributing event, which represents the destination IP from the first contributing event as the source IP for the last contributing event.

Figure 2-34: Viewing the Last Contributing Event



Alert Player for McAfee Alert: Port Scan & DLP Violation from Same IP: 107.117.155.239

Times...	Name	Src IP	Src Port	Dst IP	Dst Port	Src User	Dst User	ID	SubID	Priority	Ref #
May 21,...	McAfee ...	192.168...	80	107.117...			Admin D...	0	0	0	0
May 21,...	McAfee ...	107.117...	59574	223.198...	80			0	0	5	1
May 21,...	McAfee ...	107.117...	55722	99.247....	80			0	0	5	2

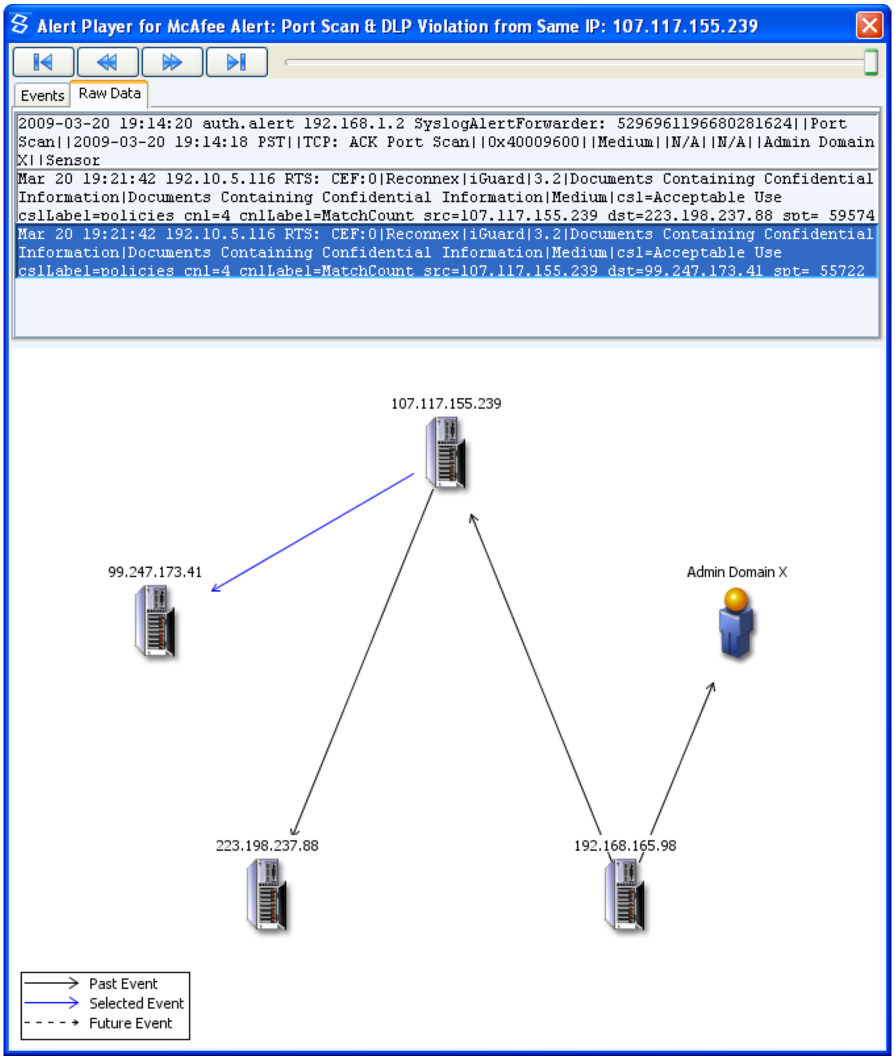
Figure 2-34 illustrates:

- The last contributing event represents a system identified as 107.117.155.239 breaking into the machine identified as 99.247.173.41
- One of the dashed black arrows shown in the first event's diagram has become solid blue in the last event’s diagram. Also, the lines that were solid blue for the first event have become solid black, which indicates the connected assets were involved in an earlier contributing event.

Viewing the Raw Data

To view the raw event data for each of the contributing events, click the **Raw Data** tab at the top of the Alert Player. Figure 2-35 illustrates display of raw data.

Figure 2-35: Viewing Raw Data



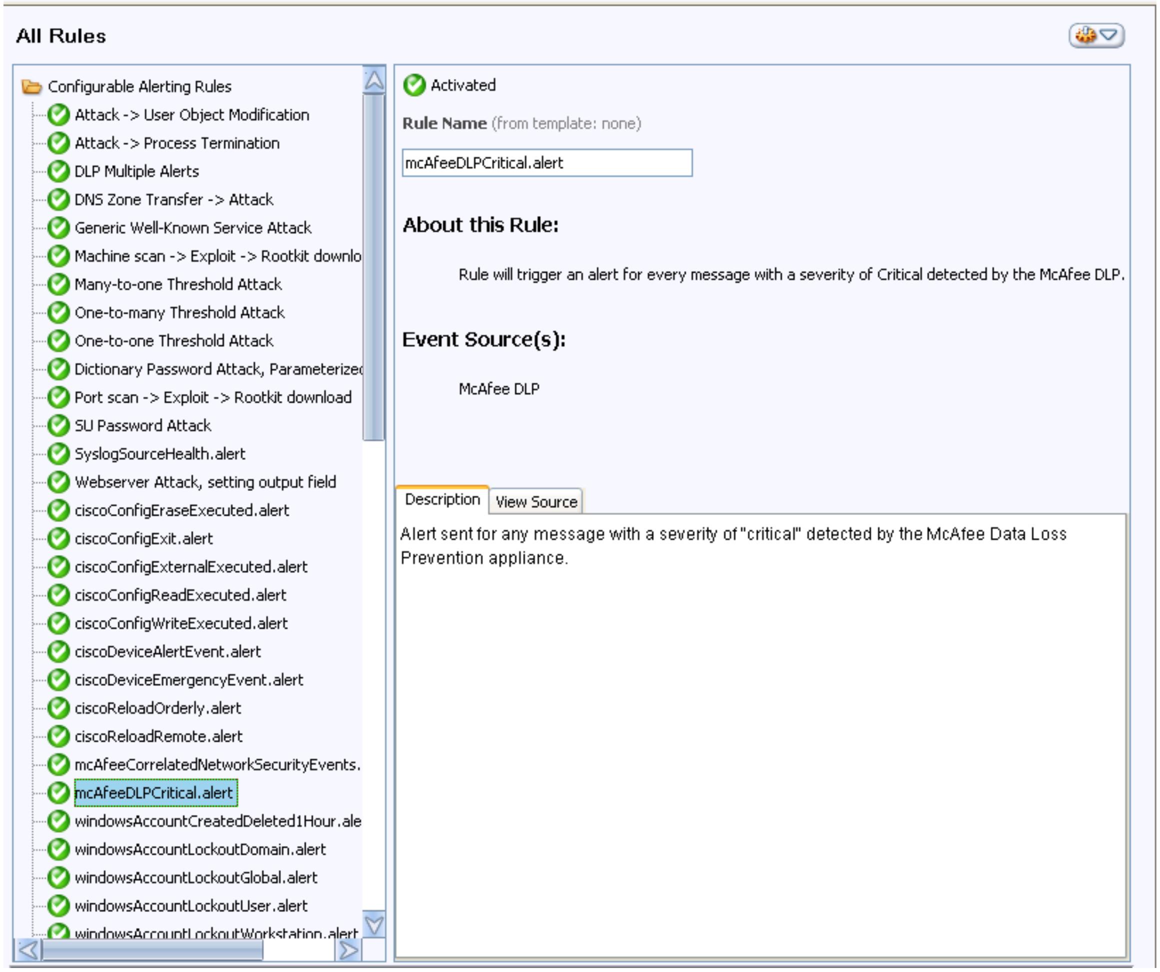
Alert Player for McAfee Alert: Port Scan & DLP Violation from Same IP: 107.117.155.239

```
2009-03-20 19:14:20 auth.alert 192.168.1.2 SyslogAlertForwarder: 5296961196680281624||Port
Scan||2009-03-20 19:14:18 PST||TCP: ACK Port Scan||0x40009600||Medium||N/A||N/A||Admin Domain
X||Sensor
Mar 20 19:21:42 192.10.5.116 RTS: CEF:0|Reconnex|iGuard|3.2|Documents Containing Confidential
Information|Documents Containing Confidential Information|Medium|csl=Acceptable Use
cslLabel=policies cnl=4 cnlLabel=MatchCount src=107.117.155.239 dst=223.198.237.88 spt= 59574
Mar 20 19:21:42 192.10.5.116 RTS: CEF:0|Reconnex|iGuard|3.2|Documents Containing Confidential
Information|Documents Containing Confidential Information|Medium|csl=Acceptable Use
cslLabel=policies cnl=4 cnlLabel=MatchCount src=107.117.155.239 dst=99.247.173.41 spt= 55722
```

Security Alerts: Understanding the Alerting Rule that Raised the Alert

To better understand the conditions that caused the **McAfee Alert: Port Scan & DLP Violation from Same IP** alert, you can read a description of the triggering rule in the **Rules** section of Administration mode. To view the description, click **Rules** in Administration mode; then select the `mcAfeeDLPCritical.alert` rule from the list of Alerting Rules. The Description panel describes the events required to raise the alert, as illustrated in Figure 2-36.

Figure 2-36: Viewing Information about the Rule that Raises an Alert



All Rules

Configurable Alerting Rules
- Attack -> User Object Modification
- Attack -> Process Termination
- DLP Multiple Alerts
- DNS Zone Transfer -> Attack
- Generic Well-Known Service Attack
- Machine scan -> Exploit -> Rootkit downlo
- Many-to-one Threshold Attack
- One-to-many Threshold Attack
- One-to-one Threshold Attack
- Dictionary Password Attack, Parameterized
- Port scan -> Exploit -> Rootkit download
- SU Password Attack
- SyslogSourceHealth.alert
- Webserver Attack, setting output field
- ciscoConfigEraseExecuted.alert
- ciscoConfigExit.alert
- ciscoConfigExternalExecuted.alert
- ciscoConfigReadExecuted.alert
- ciscoConfigWriteExecuted.alert
- ciscoDeviceAlertEvent.alert
- ciscoDeviceEmergencyEvent.alert
- ciscoReloadOrderly.alert
- ciscoReloadRemote.alert
- mcAfeeCorrelatedNetworkSecurityEvents.
- mcAfeeDLPCritical.alert
- windowsAccountCreatedDeleted1Hour.ale
- windowsAccountLockoutDomain.alert
- windowsAccountLockoutGlobal.alert
- windowsAccountLockoutUser.alert
- windowsAccountLockoutWorkstation.alert

Activated

Rule Name (from template: none)

mcAfeeDLPCritical.alert

About this Rule:

Rule will trigger an alert for every message with a severity of Critical detected by the McAfee DLP.

Event Source(s):

McAfee DLP

Description | View Source

Alert sent for any message with a severity of "critical" detected by the McAfee Data Loss Prevention appliance.

Viewing Exception Alerts

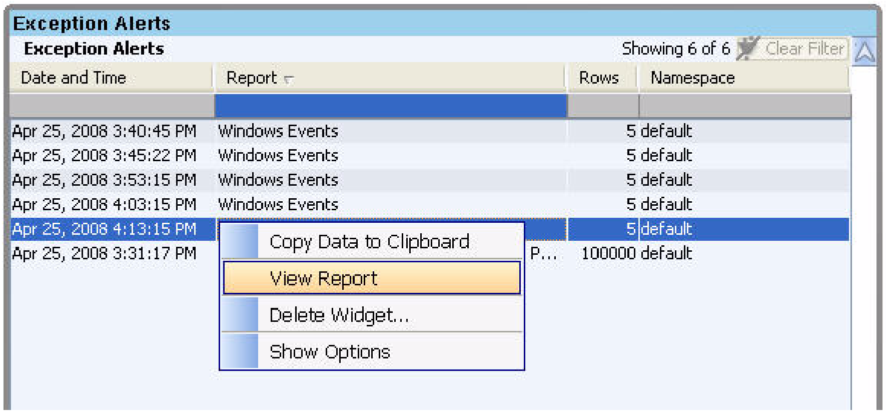
The Exception Alerts widget displays a row every time the cache entry of a scheduled report contains at least one row, which represents an exception. For example, a report that lists after-hours logins might be scheduled to trigger an alert when a user logs into your system over the weekend or after close of business. The alert displays in the Exception Alerts widget.

The alert row in the widget displays the following information about the report that triggered the alert:

- **Date and Time**—time the report ran
- **Report Name**—name of the report
- **Rows**—number of rows returned by the report
- **Namespace**—namespace against which the report ran

You can open the report to view the results by right-clicking the alert row in the widget, as shown in Figure 2-37.

Figure 2-37: Viewing the Report From the Exception Alerts Widget



Exception Alerts

Exception Alerts Showing 6 of 6 Clear Filter

Date and Time	Report	Rows	Namespace
Apr 25, 2008 3:40:45 PM	Windows Events	5	default
Apr 25, 2008 3:45:22 PM	Windows Events	5	default
Apr 25, 2008 3:53:15 PM	Windows Events	5	default
Apr 25, 2008 4:03:15 PM	Windows Events	5	default
Apr 25, 2008 4:13:15 PM		5	default
Apr 25, 2008 3:31:17 PM	P...	100000	default

Viewing the Report that Raised the Exception

To view the report that triggered an exception alert

1 In the Exception Alerts widget, right click on a row and select **View Report** from the popup.

The report displays in a new page in the dashboard.

2 If desired, right click and select **Copy Data to Clipboard**.

HawkEye AP Console copies the data in the alert row to the clipboard as plain text. You can paste it into any file that accepts text.

Using the Exception Alert Widget

Like the Security Alerts widget, you can change the appearance of the Exception Alerts widget by changing column widths, sorting rows, and filtering rows. For more information, see:

- “Widening Column Display in an Alert Widget”, on page 77
- “Sorting Alert Data”, on page 76
- “Filtering Alert Data”, on page 77.

Refreshing Dashboards and Running Items

The **Refresh** option enables you to refresh the cache entry of every report widget on a dashboard. When you refresh a dashboard, the system checks whether it contains any report widgets for which a more recent cache entry exists. If it finds a newer cache entry for any report widgets, it refreshes those widgets to display the most recent one. Figure 2-38 illustrates the three different menus that provide the **Refresh** option. You can also click the **Refresh** icon from the tool bar.

If you leave your dashboard open for a couple of days and the dashboard is scheduled to run daily, use **Refresh** to manually update all cache entries in the dashboard. Alternately, you can close and reopen the dashboard to refresh its cache.

The **Run** option differs depending on the menu from which you access it:

- **Workspace Action Menu**—The **Run** option provides a submenu to set the scope of the operation: the entire dashboard, the current page, or the selected report widget.
- **Right-Click Popup Menu from Report Widget Header**—You can run all widgets on the current page.
- **Right-Click Popup Menu from Report Widget Body**—You can run the current widget.

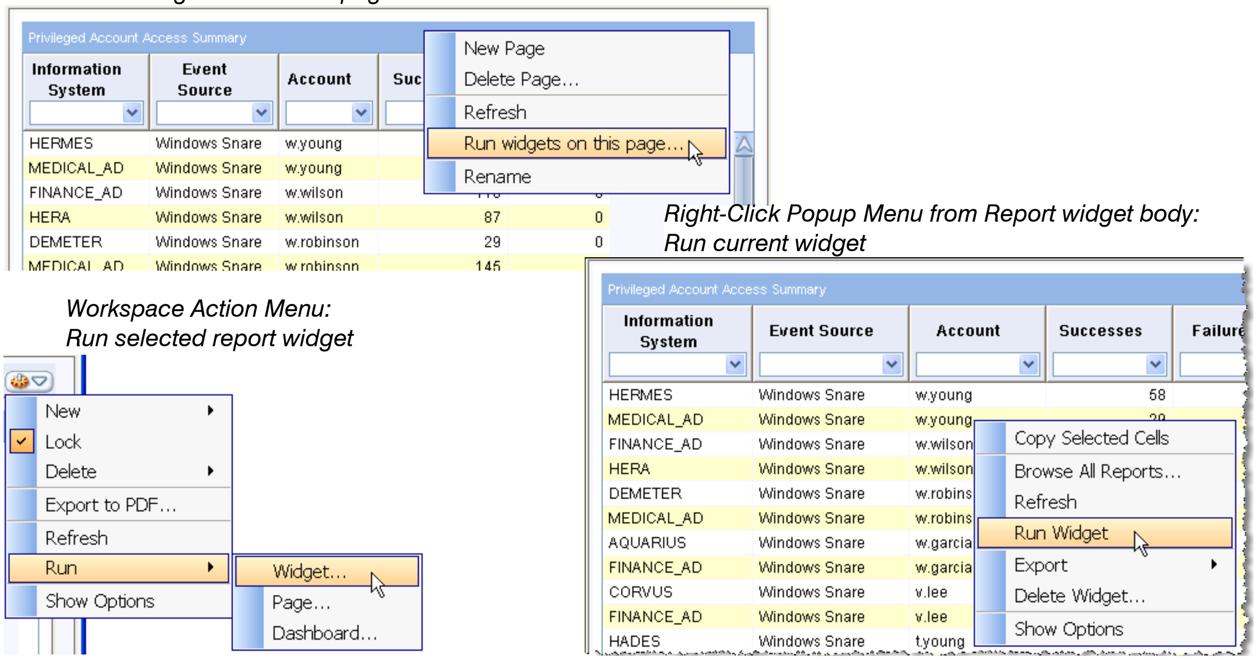
Figure 2-38 illustrates the three different menus that provide the **Run** option.

NOTE: Because alert widgets are dynamically updated, there is no need to refresh them.

Figure 2-38 illustrates both the Workspace Action Menu and two popup menus.

Figure 2-38: Running Items

Right-Click Popup Menu from Report widget header:
Run all widgets on current page



Right-Click Popup Menu from Report widget body:
Run current widget

Workspace Action Menu:
Run selected report widget

Privileged Account Access Summary

Information System	Event Source	Account	Suc	
HERMES	Windows Snare	w.young		
MEDICAL_AD	Windows Snare	w.young		
FINANCE_AD	Windows Snare	w.wilson		
HERA	Windows Snare	w.wilson	87	0
DEMETER	Windows Snare	w.robinson	29	0
MEDICAL_AD	Windows Snare	w.robinson	145	

Privileged Account Access Summary

Information System	Event Source	Account	Successes	Failure
HERMES	Windows Snare	w.young	58	
MEDICAL_AD	Windows Snare	w.young		
FINANCE_AD	Windows Snare	w.wilson		
HERA	Windows Snare	w.wilson		
DEMETER	Windows Snare	w.robins		
MEDICAL_AD	Windows Snare	w.robins		
AQUARIUS	Windows Snare	w.garcia		
FINANCE_AD	Windows Snare	w.garcia		
CORVUS	Windows Snare	v.lee		
FINANCE_AD	Windows Snare	v.lee		
HADES	Windows Snare	t.young		

When you run a specific report, the dialog displays the parameters set for the report. You can modify these parameters when you run the report, but your changes are reflected only in the current run. The report definition settings do not change.

Exporting Dashboards and Reports

HawkEye AP Console provides several options for exporting dashboards and reports:

- "Exporting a Dashboard or Report to PDF", next
- "Exporting a Report to a CSV File", on page 90
- "Exporting a Report to an HTML File", on page 91

- “Exporting a Report to an XML File”, on page 92

NOTE: Available formats differ depending on whether you are exporting a dashboard, a report widget from a dashboard, or a report from Reports mode:

- You export a dashboard to PDF format.
- When you export a report widget from a dashboard, you have a choice of HTML, XML, or CSV formats.
- When you export a report from Reports mode, you have a choice of PDF, HTML, XML, or CSV formats.

IMPORTANT:

By default, each tabular report exports no more than 10,000 rows. This limit is useful when you export a report with wide rows. If you want to export more rows, you can increase the maximum number on a report-by-report basis. The maximum number of rows depends on the size of each row in a given report, your system resources, and your export format.

If you want to export a report to PDF, Sensage strongly recommends that you set the maximum to no more than 10,000 rows. However, if you want to export the output to CSV, HTML, or XML, you can increase the maximum rows returned.

To change the default, maximum, open the report for editing and select the Properties tab. Enter the desired value in the “Export Limit” “Maximum rows per document" field. You must make this change separately for each report. For more information, see “Specifying Print Options and Maximum Rows”, on page 166.

The maximum-rows setting applies regardless of whether the report is exported manually from a dashboard or by a schedule.

Exporting a Dashboard or Report to PDF

You can export an entire dashboard or report to a PDF file. This section covers the following topics:

- “Exporting a Dashboard”, next
- “Exporting from Reports Mode”, on page 88
- “PDF Cover Sheet”, on page 88
- “Instructions for Exporting to PDF”, on page 88

Exporting a Dashboard

When you export a dashboard, the PDF file contains all reports in the dashboard as well as all images and text widgets.

Report display in the PDF file depends on its display in the dashboard:

- If a report is displayed in table format, the PDF contains every row in the table.
- If a report is displayed in chart format, the PDF displays a snapshot of the report as a chart.

You cannot export alerts from Dashboards in PDF format. You may create a custom report based on system tables for Security Alerts (not Exception Alerts) that results in the same data set and you can export that report in PDF format.

You cannot export a single report widget to PDF. To save a single report to PDF, use Reports mode.

IMPORTANT: Do not export dashboards containing more than 200,000 rows of data.

Exporting from Reports Mode

When you export a report from Reports mode, its PDF display depends on whether a chart has been defined for the report:

- If a chart has been defined, the PDF contains every row in the table as well as a snapshot of the report as a chart.
- If a chart has *not* been defined, the PDF contains every row in the table.

PDF Cover Sheet

The PDF file for a dashboard and report contains a cover sheet that includes the name of the dashboard or report, and the name of the users that created and last modified the dashboard or report. For a report, the cover sheet also displays the report start date and end date and the namespace in which the report ran.

Every page in the PDF file, including the cover sheet, contains a footer that displays the date and time of file creation and the page number.

The name of the PDF file automatically includes the name of the dashboard or report and the date of creation. On a Windows system, HawkEye AP defaults to saving the file in the **My Documents** folder; however, you can browse to another folder.

Except for the file extension, the naming convention for the CSV file is the same as for the PDF. However, when you save a report as CSV, the system displays the **Export to CSV** dialog. This dialog provides a dropdown of available encoding formats, from which you select the desired one.

Instructions for Exporting to PDF

This topic describes how to export a dashboard and a report from Reports mode to PDF format.

Exporting a Dashboard

To export a dashboard to a PDF file

1. Either select **Export Dashboard to PDF...** from the Workspace Action Menu or right click anywhere in the dashboard to display the popup menu and select **Export > Dashboard to PDF...**.

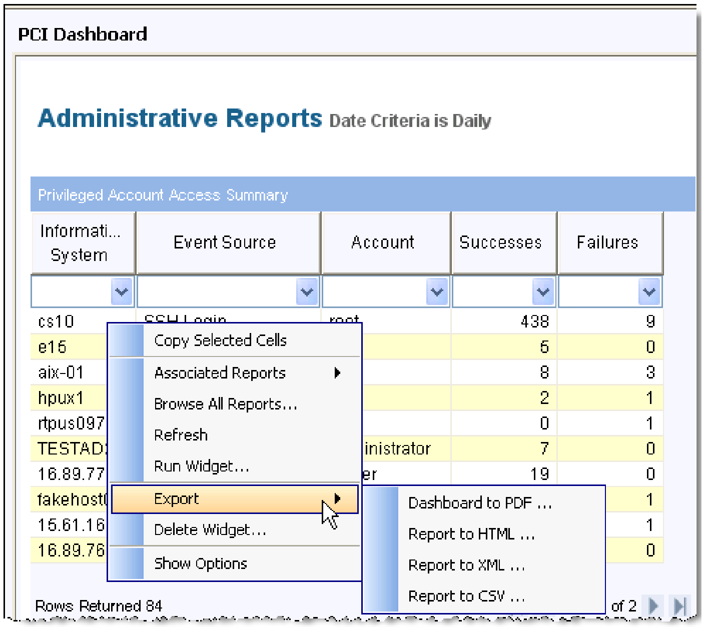
   A dialog displays that allows you to browse a location and specify a name for the file.

2. Browse to the desired destination folder and either specify the file name or keep the default name.

3. Save the file.

Figure 2-39 illustrates the right-click menu from Dashboards mode.

Figure 2-39: Exporting from a Dashboard



PCI Dashboard

Administrative Reports Date Criteria is Daily

Privileged Account Access Summary

Informati... System	Event Source	Account	Successes	Failures
cs10	SSH Login	root	438	9
e15			5	0
aix-01			8	3
hpux1			2	1
rtpus097			0	1
TESTAD:		inistrator	7	0
16.89.77		er	19	0
fakehost(				1
15.61.16				1
16.89.76				0

Rows Returned 84

Exporting a Report from Reports Mode

To export a report to a PDF file

1 Select the report from the list of All Report Definitions and open the report cache for viewing.

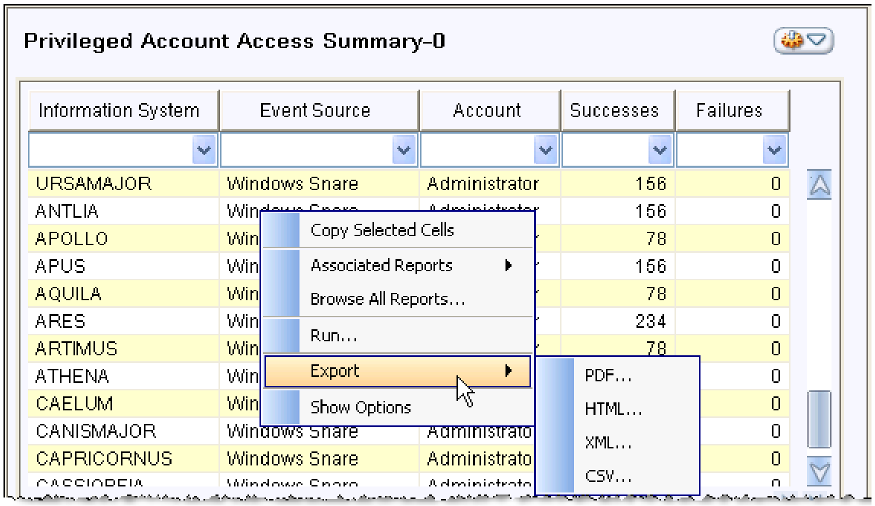
2 Either right click to display the popup menu or use the Action Menu to select **Export** > **PDF...**, as illustrated in Figure 2-40.

A dialog displays that allows you to browse a location and specify a name for the file.

3 Browse to the desired destination folder and either specify the file name or keep the default name.

4 Save the file.

Figure 2-40: Exporting a Single Report from Reports Mode



Privileged Account Access Summary-0

Information System	Event Source	Account	Successes	Failures
URSAMAJOR	Windows Snare	Administrator	156	0
ANTLIA	Win		156	0
APOLLO	Win		78	0
APUS	Win		156	0
AQUILA	Win		78	0
ARES	Win		234	0
ARTIMUS	Win		78	0
ATHENA	Win			0
CAELUM	Win			0
CANISMAJOR	Windows Snare	Administrato		0
CAPRICORNUS	Windows Snare	Administrato		0

Exporting a Report to a CSV File

This topic describes how to export a report widget from a dashboard and a report from Reports mode to CSV format.

NOTE: When exporting reports in CSV format to Microsoft Excel on a Japanese Windows installation, the `JIS` and `SJIS` CSV export encodings work correctly. However, the `EUC-JP`, `UTF-8`, and `UTF-16` CSV export encodings do not work correctly. This problem appears to be an issue with Microsoft Excel because characters encoded with UTF-8 display correctly when exported to the Notepad application.

To export a report widget to a CSV file from a dashboard

1 In Dashboards mode, right click the report widget to display the popup menu.

2 Select **Export** > **Report to CSV...** .

Figure 2-39 illustrates the popup menu.

The **Export to CSV** dialog displays.

3 From the **Export to CSV** dialog, select the desired encoding.

A dialog displays that allows you to browse a location and specify a name for the file.

4 Browse to the desired destination folder and either specify the file name or keep the default name.

5 Save the file.

To export a report to a CSV file from Reports mode

1 From Reports mode, select the report from the list of All Report Definitions, open the report cache for viewing, and either click the Action Menu or right click the report widget to display the popup menu.

2 Select **Export** > **Report to CSV...** .

Figure 2-40 illustrates the popup menu.

The **Export to CSV** dialog displays.

3 From the **Export to CSV** dialog, select the desired encoding.

A dialog displays that allows you to browse a location and specify a name for the file.

4 Browse to the desired destination folder and either specify the file name or keep the default name.

5 Save the file.

Exporting a Report to an HTML File

This topic describes how to export a report widget from a dashboard and a report from Reports mode to HTML format.

To export a report widget to an HTML file from a dashboard

1 In Dashboards mode, right click the report widget to display the popup menu.

2 Select **Export** > **Report to HTML...** from the submenu, as illustrated in Figure 2-39.

The **Export HTML** dialog displays.

3 From the **Export HTML** dialog, select the desired encoding.

A dialog displays that allows you to browse a location and specify a name for the file.

4 Browse to the desired destination folder and either specify the file name or keep the default name.

5 Save the file.

6 Double click the HTML file to display the report in a browser.

To export a report to an HTML file from Reports mode

1 From Reports mode, select the report from the list of All Report Definitions, open the report cache for viewing, and either click the Action Menu or right click the report widget to display the popup menu.

2 Select **Export** > **HTML...**.

Figure 2-40 illustrates the popup menu.

The **Export HTML** dialog displays.

3 From the **Export HTML** dialog, select the desired encoding.

A dialog displays that allows you to browse a location and specify a name for the file.

4 Browse to the desired destination folder and either specify the file name or keep the default name.

5 Save the file.

6 Double click the HTML file to display the report in a browser.

NOTE: The export process saves the report in a zip file that contains:

- the tabular report in HTML format
- its css style sheet
- if the report includes a chart, a png image file that illustrates a snapshot of the chart

Exporting a Report to an XML File

This topic describes how to export a report widget from a dashboard and a report from Reports mode to XML format.

To export a report to an XML file from a dashboard

1 In Dashboards mode, right click the report widget to display the popup menu.

2 Select **Export** > **Report to XML...** from the submenu, as illustrated in Figure 2-39.

A dialog displays that allows you to browse a location and specify a name for the file.

3 Browse to the desired destination folder and either specify the file name or keep the default name.

4 Save the file.

To export a report to an XML file from Reports mode

1 From Reports mode, select the report from the list of All Report Definitions, open the report cache for viewing, and either click the Action Menu or right click the report widget to display the popup menu.

2 Select **Export** > **XML...**.

Figure 2-40 illustrates the popup menu.

A dialog displays that allows you to browse a location and specify a name for the file.

3 Browse to the desired destination folder and either specify the file name or keep the default name.

4 Save the file.

NOTE:

- The export process saves the report in a zip file that contains:

  - the report in XML format
  - an xsd file (its schema)
  - if the report has been saved as a chart, a png image file that illustrates a snapshot of the report as a chart

- If the report contains many rows, the XML file will contain all lines and might take a long time to process. Because the data is saved in chunks, processing should not fill memory.

CHAPTER 3

Running, Viewing, and Managing Reports

This chapter contains the following sections:

Overview of Reports Mode and the All Report Definitions List

This chapter introduces Reports mode and the **All Report Definitions** list, which displays all report definitions that you have permission to access. A *report definition* is a specification that determines the actual data to display in a report and the way the data is formatted and arranged. A report definition differs from a *report cache entry* as a recipe differs from a meal. The definition provides a set of instructions. The report cache entry provides the formatted data.

After a report definition is created, a user or schedule runs it to return results. The results are saved in a report *cache entry*, which is a quickly accessible version of data stored in the EDW for a specific date period. Typically the report definition has aggregated or otherwise manipulated the stored data, and the cached data reflects these manipulations. For more information about the report cache, see "Cached Data: Making Stored Data Quickly Available", on page 43.

Every run of a report definition creates a new cache entry. Each cache entry represents a single report run and contains the data set from that run for the specific date range. The associated caches for a report definition are available from the Options Pane when you view a report widget in the dashboard or from Reports mode.

Typically, a report displays the most recent cache for the smallest interval. However, you can select any associated cache or set of caches for display. You can use the Options Pane to combine multiple contiguous caches into a single result set. For example, assume you run a report weekly on Sundays. You can combine 4 weekly caches into a single report that represents an entire month. For more information, see "Viewing and Changing the Time Range and Namespace", on page 51.

Schedules usually determine when a report definition runs, but users with appropriate permissions can also run them manually. In addition to being associated to report caches, a report definition is also associated with all schedules and permissions assigned to it.

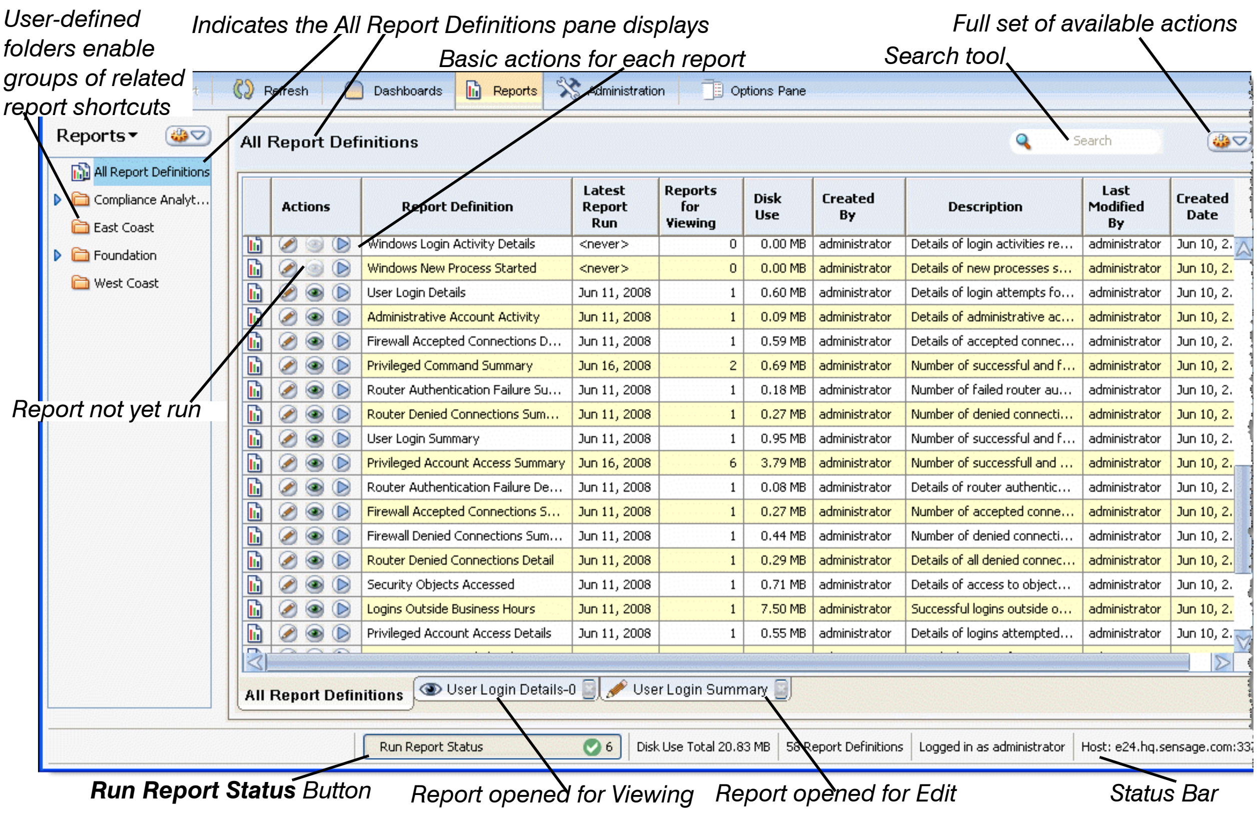
The columns in the **All Report Definitions** window list each report definition's type (Wizard or SQL), name, description, date last run, number of reports for viewing (saved caches), disk usage, creator and creation date, and the last user to modify the report. The list also displays the **Actions** column, which provides the three icons that enable you to edit, view, and run the report.

This chapter describes how to use Reports mode and the **All Report Definitions** list to perform the following tasks:

- View all reports in your system that you have permission to access, and view information about those reports
- Create folders to organize report shortcuts by audience, usage, or other need
- View report results
- View report cache entries and the amount of disk space each consumes
- Run reports manually
- Delete report definitions
- View and assign permissions to each report
- Delete unwanted caches

Figure 3-1 illustrates the **All Report Definitions** pane in the Reports workspace. This pane displays a list of all reports in the system that you have permission to access.

Figure 3-1: Reports Mode Example



	Actions	Report Definition	Latest Report Run	Reports for Viewing	Disk Use	Created By	Description	Last Modified By	Created Date
		Windows Login Activity Details	<never>	0	0.00 MB	administrator	Details of login activities re...	administrator	Jun 10, 2.
		Windows New Process Started	<never>	0	0.00 MB	administrator	Details of new processes s...	administrator	Jun 10, 2.
		User Login Details	Jun 11, 2008	1	0.60 MB	administrator	Details of login attempts fo...	administrator	Jun 10, 2.
		Administrative Account Activity	Jun 11, 2008	1	0.09 MB	administrator	Details of administrative ac...	administrator	Jun 10, 2.
		Firewall Accepted Connections D...	Jun 11, 2008	1	0.59 MB	administrator	Details of accepted connec...	administrator	Jun 10, 2.
		Privileged Command Summary	Jun 16, 2008	2	0.69 MB	administrator	Number of successful and f...	administrator	Jun 10, 2.
		Router Authentication Failure Su...	Jun 11, 2008	1	0.18 MB	administrator	Number of failed router au...	administrator	Jun 10, 2.
		Router Denied Connections Sum...	Jun 11, 2008	1	0.27 MB	administrator	Number of denied connecti...	administrator	Jun 10, 2.
		User Login Summary	Jun 11, 2008	1	0.95 MB	administrator	Number of successful and f...	administrator	Jun 10, 2.
		Privileged Account Access Summary	Jun 16, 2008	6	3.79 MB	administrator	Number of successfull and ...	administrator	Jun 10, 2.
		Router Authentication Failure De...	Jun 11, 2008	1	0.08 MB	administrator	Details of router authentic...	administrator	Jun 10, 2.
		Firewall Accepted Connections S...	Jun 11, 2008	1	0.27 MB	administrator	Number of accepted conne...	administrator	Jun 10, 2.
		Firewall Denied Connections Sum...	Jun 11, 2008	1	0.44 MB	administrator	Number of denied connecti...	administrator	Jun 10, 2.
		Router Denied Connections Detail	Jun 11, 2008	1	0.29 MB	administrator	Details of all denied connec...	administrator	Jun 10, 2.
		Security Objects Accessed	Jun 11, 2008	1	0.71 MB	administrator	Details of access to object...	administrator	Jun 10, 2.
		Logins Outside Business Hours	Jun 11, 2008	1	7.50 MB	administrator	Successful logins outside o...	administrator	Jun 10, 2.
		Privileged Account Access Details	Jun 11, 2008	1	0.55 MB	administrator	Details of logins attempted...	administrator	Jun 10, 2.

User-defined folders enable groups of related report shortcuts

Indicates the All Report Definitions pane displays

Basic actions for each report

Search tool

Full set of available actions

Report not yet run

***Run Report Status** Button*

Report opened for Viewing

Report opened for Edit

Status Bar

As illustrated in Figure 3-1, you can group reports in your own folders. The **All Report Definitions** window displays all the reports that you have permission to access. You create a folder to display only a subset of report definitions, which are actually shortcuts to the master definition. When you select a folder, the workspace displays only the report shortcuts in that folder. For more information, see “Managing Report Shortcuts in Folders”, on page 110.

The icons in the **Actions** column in Figure 3-1 provide quick access to the most common operations that you can perform on a report definition:

- **Edit** —Open for editing an existing report definition

  For more information, see Chapter 4: Creating and Editing Wizard Reports and Chapter 5: Creating and Editing SQL Reports.

- **View** —Open for viewing a report that has already run

  NOTE: A report must be run before you can view its results. The results are saved in a report cache, which you open for viewing. If the report has never been run, the view icon displays as disabled, as illustrated in Figure 3-1.

  For more information, see “Viewing Report Results and Managing Report Cache Entries”, on page 114.

- **Run** —Run a report definition; you can use the **Run** dialog to change the date period and may have the opportunity to affect the output in other ways, such as specifying values for column criteria.

  For more information, see “Running a Report”, on page 100.

The icons to the left of the All Report Definitions list differentiate between the two types of report definitions:

- Wizard Reports —Creating and editing a wizard report requires knowledge of your data and a clear understanding of the report’s purpose but no understanding of SQL (Structured Query Language). For more information, see Chapter 4: Creating and Editing Wizard Reports.

- SQL Reports .—Creating and editing a SQL report requires knowledge of SQL and HawkEye AP extensions to SQL and allows for more advanced queries than those enabled by the wizard. For more information, see Chapter 5: Creating and Editing SQL Reports.

What Does a Report Definition Specify?

A report definition identifies the report data source, the columns that display, and the date range of the data to include in the report. It may also define column-criteria rows that allow a run-time user to set value(s) for specific column(s).

Each definition also identifies the process used to create it—a definition is created either as a Wizard report or a SQL report and can be edited only in the style by which it was created. You can, however, copy the SQL generated for a Wizard report and use it as the basis for a SQL report. For information on viewing and copying the generated SQL, see “Viewing the Report Query”, on page 116.

The definition also includes the following manipulations that can be specified on the report data:

- criteria that limits the returned data by time range and column values
- maximum number of rows to display
- associated reports
- column display
- chart display

Additionally, a Wizard report contains:

- aggregation that summarizes and groups the data by column values
- threshold that eliminates aggregated groups of data
- sort order

NOTE: A SQL report definition also enables aggregation, thresholds, and sorting, but these are part of the SQL query. A Wizard report defines these separately in the user interface. A SQL report definition also allows for additional manipulations. For more information, see "Overview of SQL Reports", on page 177.

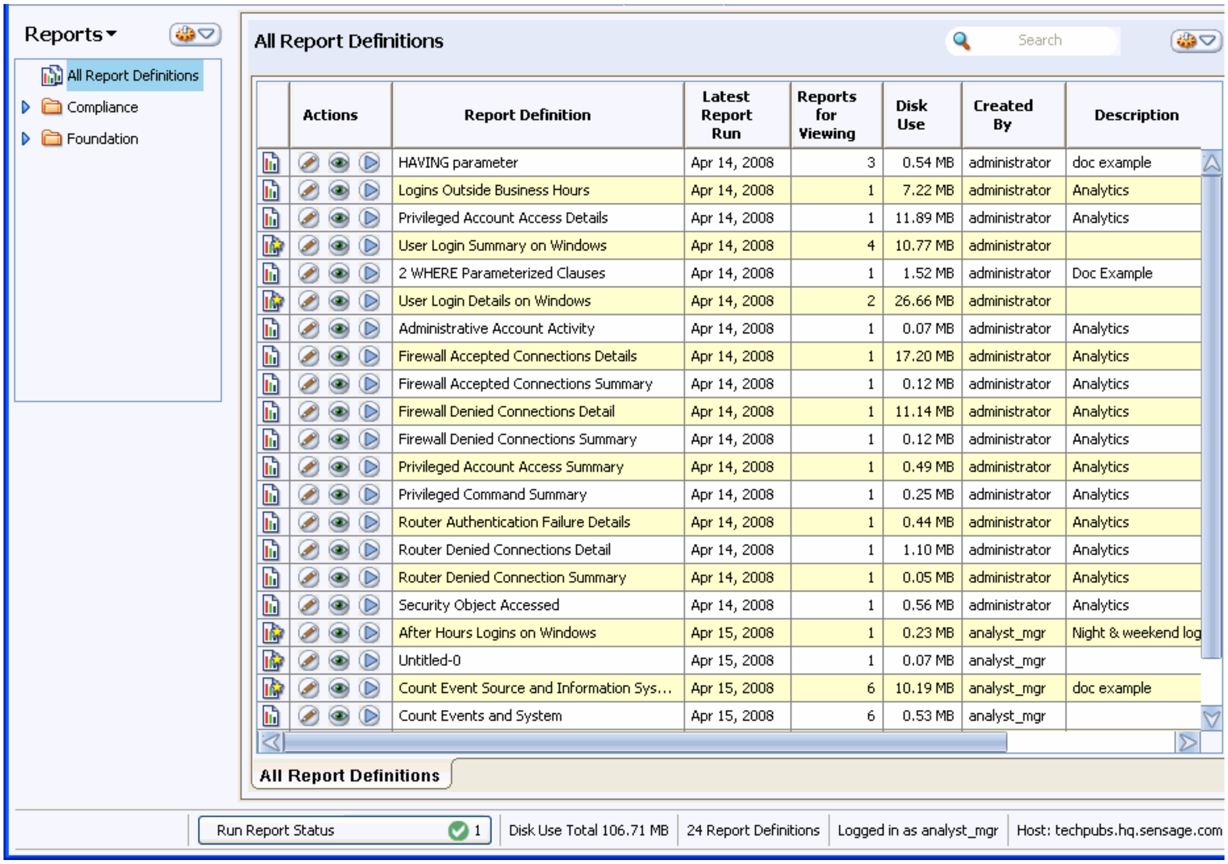
The following sections describe how to run, view, and manage Wizard and SQL reports.

Finding Specific Reports in the All Report Definitions List

The workspace for the **All Report Definitions** window contains a powerful Search field. This field accepts text from any of the columns as criteria to limit display of the report-definition rows. For example, you can enter text that limits the rows by report name. You can enter more text that limits the rows further by metadata text. The next few graphics illustrate how you can limit display from 24 rows to a single row.

Figure 3-2 illustrates an **All Report Definitions** list that contains 24 report definitions.

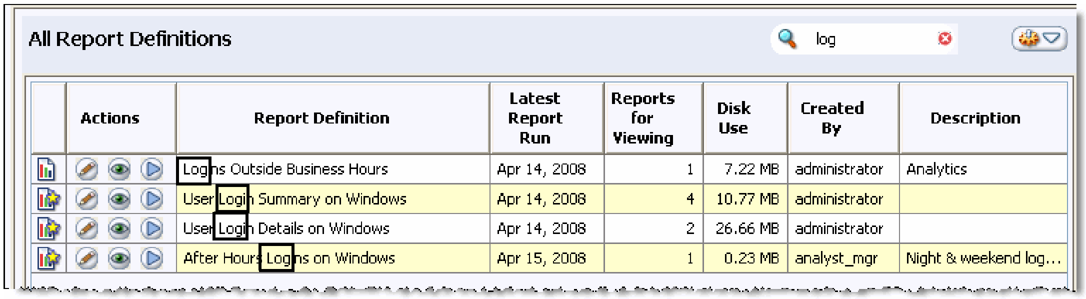
Figure 3-2: Displaying All Rows in the All Report Definitions List



	Actions	Report Definition	Latest Report Run	Reports for Viewing	Disk Use	Created By	Description
		HAVING parameter	Apr 14, 2008	3	0.54 MB	administrator	doc example
		Logins Outside Business Hours	Apr 14, 2008	1	7.22 MB	administrator	Analytics
		Privileged Account Access Details	Apr 14, 2008	1	11.89 MB	administrator	Analytics
		User Login Summary on Windows	Apr 14, 2008	4	10.77 MB	administrator	
		2 WHERE Parameterized Clauses	Apr 14, 2008	1	1.52 MB	administrator	Doc Example
		User Login Details on Windows	Apr 14, 2008	2	26.66 MB	administrator	
		Administrative Account Activity	Apr 14, 2008	1	0.07 MB	administrator	Analytics
		Firewall Accepted Connections Details	Apr 14, 2008	1	17.20 MB	administrator	Analytics
		Firewall Accepted Connections Summary	Apr 14, 2008	1	0.12 MB	administrator	Analytics
		Firewall Denied Connections Detail	Apr 14, 2008	1	11.14 MB	administrator	Analytics
		Firewall Denied Connections Summary	Apr 14, 2008	1	0.12 MB	administrator	Analytics
		Privileged Account Access Summary	Apr 14, 2008	1	0.49 MB	administrator	Analytics
		Privileged Command Summary	Apr 14, 2008	1	0.25 MB	administrator	Analytics
		Router Authentication Failure Details	Apr 14, 2008	1	0.44 MB	administrator	Analytics
		Router Denied Connections Detail	Apr 14, 2008	1	1.10 MB	administrator	Analytics
		Router Denied Connection Summary	Apr 14, 2008	1	0.05 MB	administrator	Analytics
		Security Object Accessed	Apr 14, 2008	1	0.56 MB	administrator	Analytics
		After Hours Logins on Windows	Apr 15, 2008	1	0.23 MB	analyst_mgr	Night & weekend log
		Untitled-0	Apr 15, 2008	1	0.07 MB	analyst_mgr	
		Count Event Source and Information Sys…	Apr 15, 2008	6	10.19 MB	analyst_mgr	doc example
		Count Events and System	Apr 15, 2008	6	0.53 MB	analyst_mgr	

Figure 3-3 illustrates search text ("`log`") that limits rows to only four reports, each of whose name includes "`Login`".

Figure 3-3: Entering Text That Limits Rows by Report Definition Name

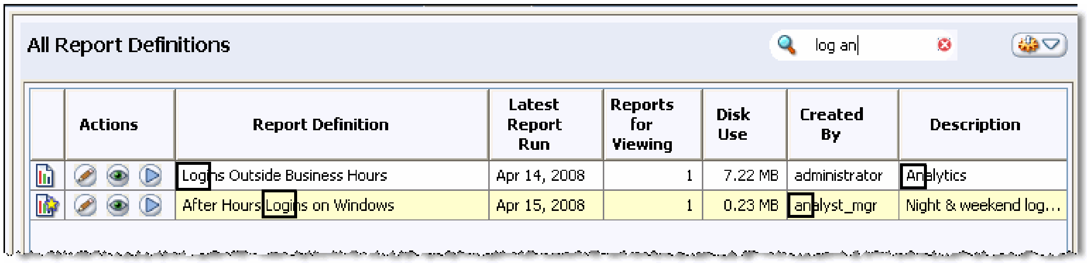


	Actions	Report Definition	Latest Report Run	Reports for Viewing	Disk Use	Created By	Description
		Logins Outside Business Hours	Apr 14, 2008	1	7.22 MB	administrator	Analytics
		User Login Summary on Windows	Apr 14, 2008	4	10.77 MB	administrator	
		User Login Details on Windows	Apr 14, 2008	2	26.66 MB	administrator	
		After Hours Logins on Windows	Apr 15, 2008	1	0.23 MB	analyst_mgr	Night & weekend log…

Figure 3-4 illustrates an expansion of the search to include two additional letters ("`an`"). These letters do not appear in the report names. Instead they appear in two of the metadata columns:

Created By and **Description**. Note that the search is case insensitive: the value in one column contains an uppercase "`A`".

Figure 3-4: Entering Text That Limits Rows by Metadata



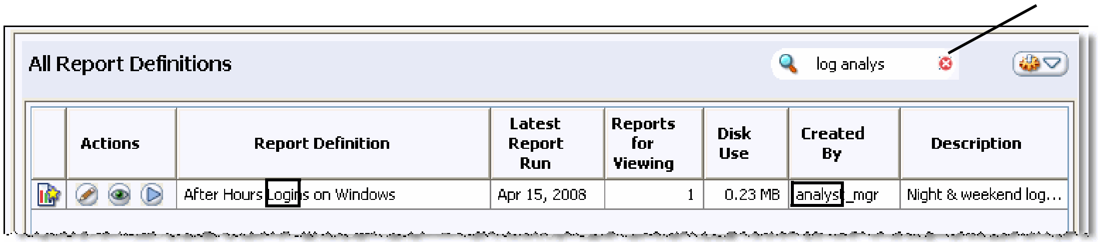
All Report Definitions

log an

	Actions	Report Definition	Latest Report Run	Reports for Viewing	Disk Use	Created By	Description
		Logins Outside Business Hours	Apr 14, 2008	1	7.22 MB	administrator	Analytics
		After Hours Logins on Windows	Apr 15, 2008	1	0.23 MB	analyst_mgr	Night & weekend log...

Figure 3-5 illustrates an expansion of the search to include four additional letters ("`alys`") that appear only in the **Created By** column.The search has now limited report definition display to a single row.

Figure 3-5: Limiting Rows to a Single Row



Clear-search button

All Report Definitions

log analys

	Actions	Report Definition	Latest Report Run	Reports for Viewing	Disk Use	Created By	Description
		After Hours Logins on Windows	Apr 15, 2008	1	0.23 MB	analyst_mgr	Night & weekend log...

The three graphics above illustrate how you can limit the displayed rows by searching for values in both the report name column and metadata columns.

When you are done searching, click the Clear-Search button (⊗) to re-display all report definition rows.

Running a Report

- "Background", next
- "Run Dialog for a Wizard Report", on page 102
- "Run Dialog for a SQL Report", on page 105
- "Run Report Status Dialog", on page 106
- "Running Multiple Reports Simultaneously", on page 109

Background

Running a report is simple, and there are several steps:

1 Find the report definition in the **All Report Definitions** list.

2 Click the Run (▷) icon.

3. In the **Run** dialog:
    - Enter date criteria.
    - Enter or change column criteria, if available and desired.
    - Change the namespace, if desired.
4. Click **Run**.
5. When the Run Report Status dialog indicates that the run has completed, click the View (👁) icon to view the report.

NOTE: The default maximum number of rows returned in a report is 100,000. You can reset the maximum in the `controller.prop` file. Refer to “Administering HawkEye AP Console and the Application Manager”, on page 183," in the *Administration Guide*.

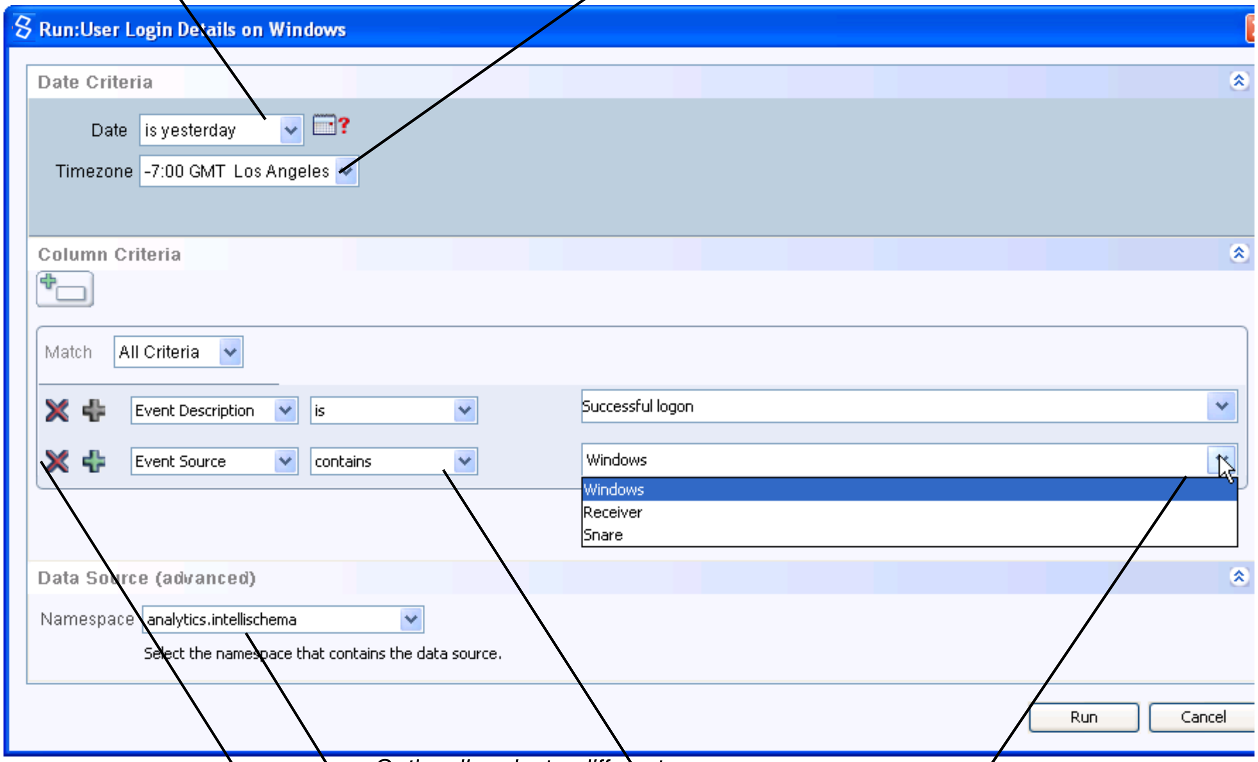
The **Run** dialog for a Wizard report differs from the one for a SQL report. The next sections describe these **Run** dialogs and how to use them.

Run Dialog for a Wizard Report

The **Run** dialog for a Wizard report offers several options, as illustrated in Figure 3-6 and Figure 3-7.

Figure 3-6: Run Dialog Options for Wizard Reports

Change the time period as desired. *Change the time zone if desired.*



Run:User Login Details on Windows

Date Criteria

Date: is yesterday

Timezone: -7:00 GMT Los Angeles

Column Criteria

Match: All Criteria

Event Description | is | Successful logon

Event Source | contains | Windows

Windows
Receiver
Snare

Data Source (advanced)

Namespace: analytics.intellischema

Select the namespace that contains the data source.

Click to delete unwanted criteria row(s).

Optionally select a different operator from the dropdown.

Select the desired column value.

Optionally, select a different namespace.

As Figure 3-6 illustrates, the **Run** dialog enables you to change the date and time range that the report data represents. You can also select a different time zone from the dropdown. For more information, see "Modifying the Time Range", next.

Modifying the Time Range

There are several options for specifying the time range:

- **is**—all events occur between 12:00:00 am and 11:59:59 pm
- **is between**—all events occur between two user-specified dates and times
- **is yesterday**—all events occur *yesterday* between 12:00:00 am and 11:59:59 pm
- **is last week**—all events occur between Monday through Sunday

  NOTE: You can set the week to begin on Sunday by editing your preferences. For more information, see "Setting Preferences", on page 35.

- **is last month**—all events occur between the first and last days of the month
- **is last (custom)**

For detailed information about changing the date range, see "Date Criteria: Specifying Time Range", on page 134.

Changing the Namespace

Figure 3-6 also illustrates that you can select a different namespace from a dropdown of available namespaces. For example, if your system stores the `hosts` table in both the `Eastern` and `Western` namespaces, and a single report definition retrieves host information from either of these namespaces, you can select the desired namespace at the time you run the report.

Modifying the Operator

The **Run** dialog for a Wizard report may provide one or more criteria rows. Each criteria row is specific to a column in the report and is associated with an operator and a value field. Depending on the type of value field, you can either select from a dropdown of values or enter your own value for each column.

Each criteria row specifies the relationship of the selected value to the specific column. For example, the criteria row might specify that the column exactly matches the selected value, contains the selected value, or does not contain the value. You can change this relationship from the criteria dropdown. The table below describes each of the operators.

Operator	Explanation	Example
contains	matches values that include the specified text	`Windows` would match: • Windows Retriever Non-DC • Windows Retriever DC • Windows Snare Non-DC • Windows Snare DC • My Windows server
does not contain	matches values that do NOT include the specified text	`Non-DC` would match: • Windows Retriever DC • Windows Snare DC
is	exactly matches values that include ONLY the specified text	`Windows Retriever Non-DC` would match: • Windows Retriever Non-DC
is not	matches values that do NOT exactly match the specified text	`Windows Retriever Non-DC` would match: • Windows Retriever DC • Windows Snare Non-DC • Windows Snare DC • Unix • Linux • Candy Bars
starts with	matches values that begin with the specified text	`Windows` would match: • Windows Retriever Non-DC • Windows Retriever DC • Windows Snare Non-DC • Windows Snare DC

Operator	Explanation	Example
ends with	matches values that end with the specified text	`DC` would match: • Windows Retriever Non-DC • Windows Retriever DC • Windows Snare Non-DC • Windows Snare DC

Regular Expression	matches the specified regular expression
	NOTE
	• A regular expression is written in a formal language that interprets such text as specific characters, words, or patterns of characters.
	• If the search pattern includes a slash (/) character, you must precede it with a backslash (\) character.
	• HawkEye AP supports Perl regular expressions
		To locate the following text:
		`default/:Report`
		Enter the following text:
		`default\/:Report`

Adding, Deleting, and Modifying Criteria Rows

The **Run** dialog for a Wizard report also displays the Add Criteria (✚) and the Delete Criteria (✖) icons. These icons allow you to add additional criteria row(s) or remove unwanted criteria row(s). For example, if you do not want to limit the report by Event Source, you could click the Delete Criteria icon to remove this row from the **Run** dialog. If you want to limit the report by a third column's value, you could click the Add Criteria icon to add another row to the dialog.

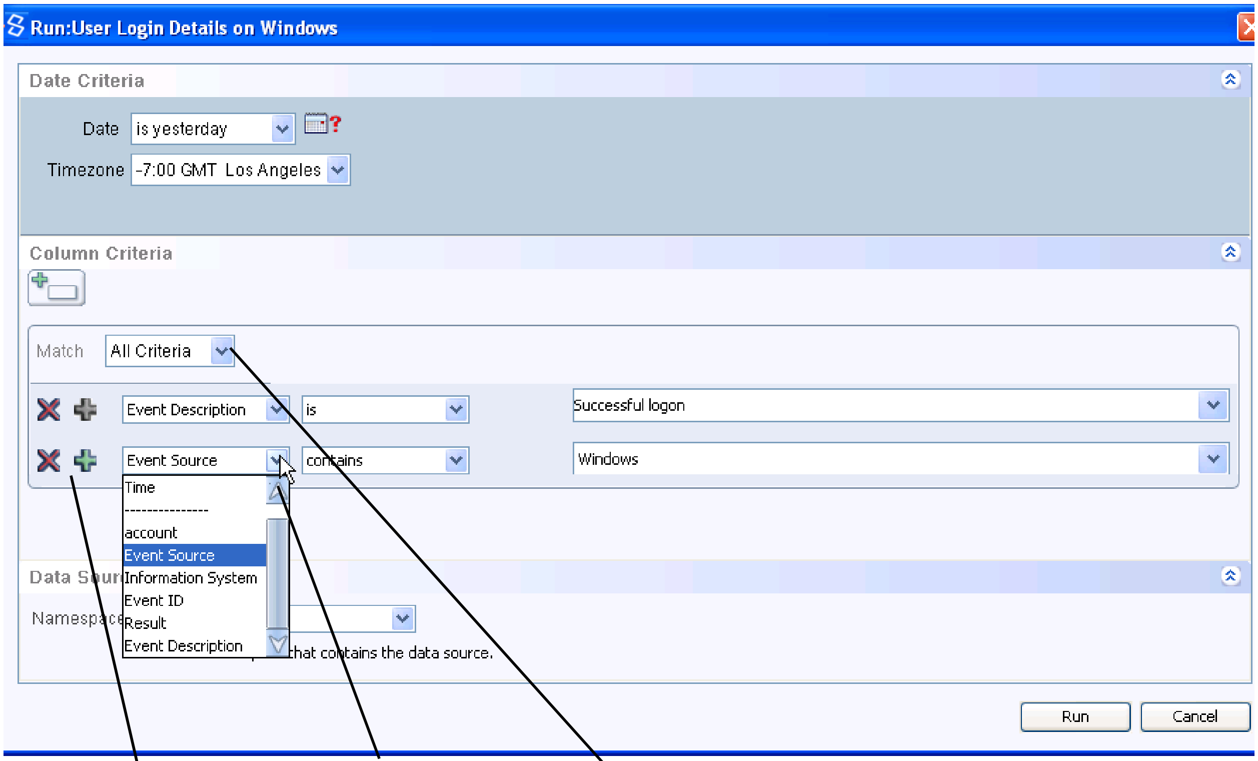
Figure 3-7 illustrates additional options that the **Run** dialog provides for a Wizard report. This figure shows that you can specify whether each row in the report should represent the specified value for every column or whether each report row should represent the specified value for only one of the columns:

- If the **Match** column is set to All Criteria, the example report returns data only when the **Event Description** is `Successful logon` AND the **Event Source** contains `Windows`.

- If the **Match** column is set to Any Criteria, the example report returns data when the **Event Description** is `Successful logon` OR the **Event Source** contains `Windows`.

Figure 3-7 also illustrates that you can change the column represented by a criteria row without deleting the row and adding a new row for the desired column. Simply select the desired column from the column dropdown in the existing criteria row and enter an appropriate value for it in the value field. You may also want to change the operator.

Figure 3-7: Run Dialog: Additional Options



Run:User Login Details on Windows

Date Criteria

Date: is yesterday

Timezone: -7:00 GMT Los Angeles

Column Criteria

Match: All Criteria

Event Description	is	Successful logon
Event Source	contains	Windows

Data Sour

Namespace

that contains the data source.

Click to select a different column. *Click to switch to Any Criteria.*

Click to add a criteria row.

After you set all values and criteria as desired, click **Run** to generate the report. The **Run Report Status** dialog displays, as described in “Run Report Status Dialog”, on page 106.

Run Dialog for a SQL Report

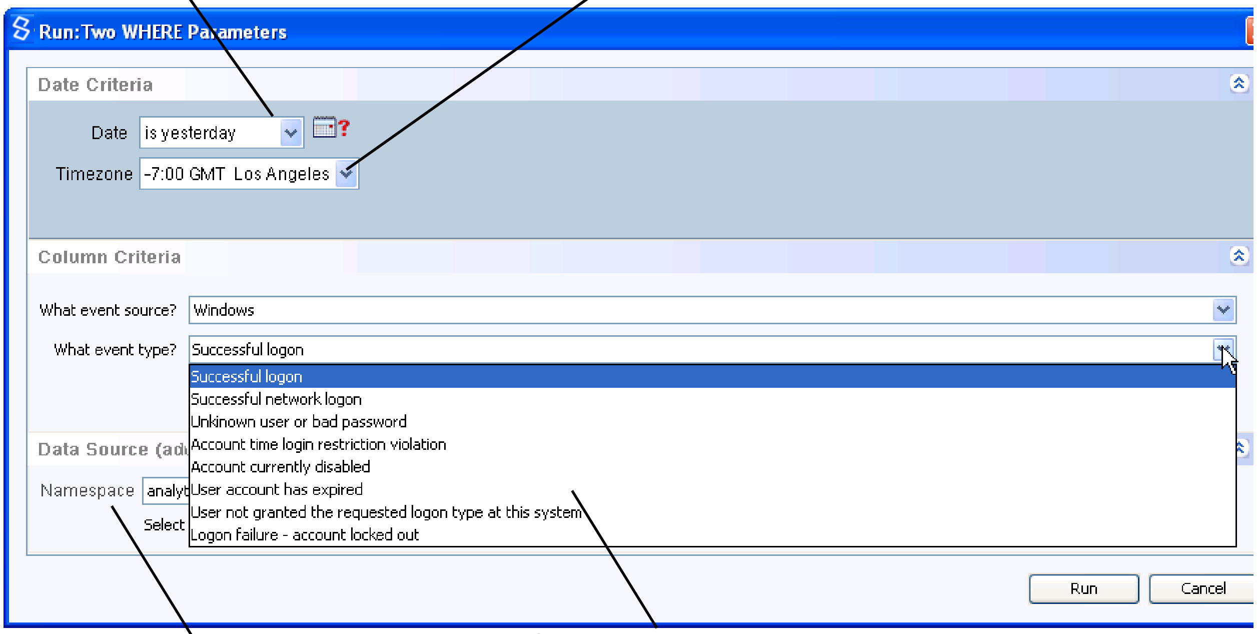
The **Run** dialog for a SQL report is not as flexible as the one for a Wizard report. As Figure 3-8 illustrates, the **Run** dialog for a SQL report also enables you to change the time period and time zone for the report. It also enables you to select a value for column criteria and to change the namespace.

However, the **Run** dialog for a SQL report does not provide flexibility in removing or adding a column-criteria row, or changing the column represented by a a column-criteria row, or changing the operator that specifies the relationship between the column and the selected column value.

Figure 3-8: Run Dialog Options for SQL Reports

Change the time period as desired.

Change the time zone if desired.



Run: Two WHERE Parameters

Date Criteria

Date: is yesterday

Timezone: -7:00 GMT Los Angeles

Column Criteria

What event source? Windows

What event type? Successful logon

Successful logon
Successful network logon
Unknown user or bad password
Account time login restriction violation
Account currently disabled
User account has expired
User not granted the requested logon type at this system
Logon failure - account locked out
Data Source (ad
Namespace
analyt
Select
Run
Cancel

Optionally, select a different namespace.

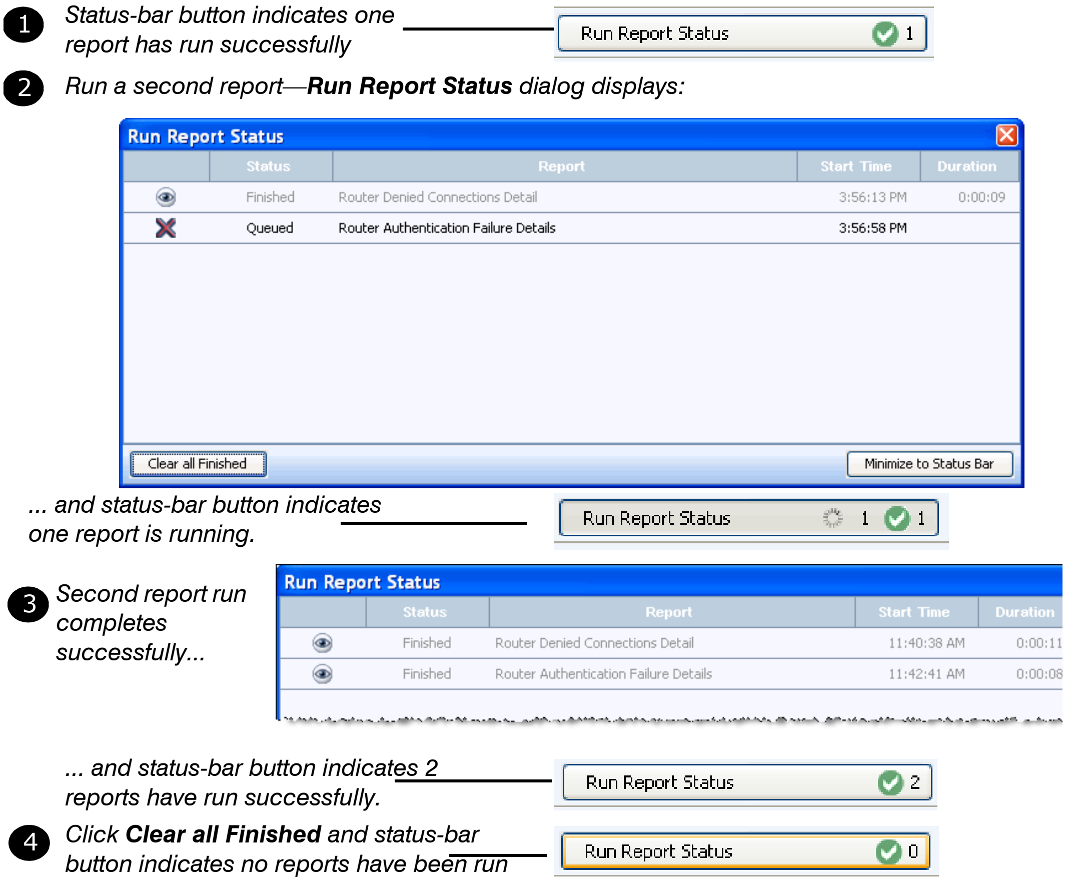
Click to select a column. value

After you set all values and criteria as desired, click **Run** to generate the report. The **Run Report Status** dialog displays, as described next.

Run Report Status Dialog

When you run a Wizard or SQL report, the **Run Report Status** dialog displays and the **Run Report Status** button changes. This button displays in the status bar of Reports and Dashboards mode. Figure 3-9 illustrates how the **Run Report Status** dialog and the status-bar button change as your run several reports.

Figure 3-9: Run Report Status Dialog and Button: Explained



1 *Status-bar button indicates one report has run successfully*

2 *Run a second report—**Run Report Status** dialog displays:*

Run Report Status

	Status	Report	Start Time	Duration
👁	Finished	Router Denied Connections Detail	3:56:13 PM	0:00:09
✕	Queued	Router Authentication Failure Details	3:56:58 PM	

... and status-bar button indicates one report is running.

3 *Second report run completes successfully...*

Run Report Status

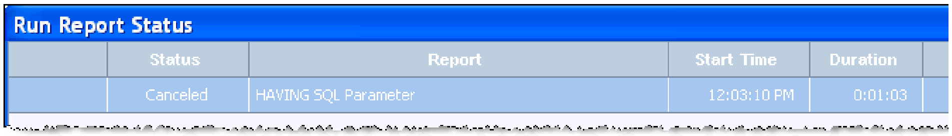
	Status	Report	Start Time	Duration
👁	Finished	Router Denied Connections Detail	11:40:38 AM	0:00:11
👁	Finished	Router Authentication Failure Details	11:42:41 AM	0:00:08

... and status-bar button indicates 2 reports have run successfully.

4 *Click **Clear all Finished** and status-bar button indicates no reports have been run*

As the report runs, the **Run Report Status** dialog displays the Cancel ✕ icon, which allows you to abort processing. While the report is cancelling, the status dialog displays the word `Cancelling`. When cancelling is complete, the dialog indicates that the run was cancelled, as illustrated below in Figure 3-10.

Figure 3-10: Run Report Status Dialog: Cancelling a Run

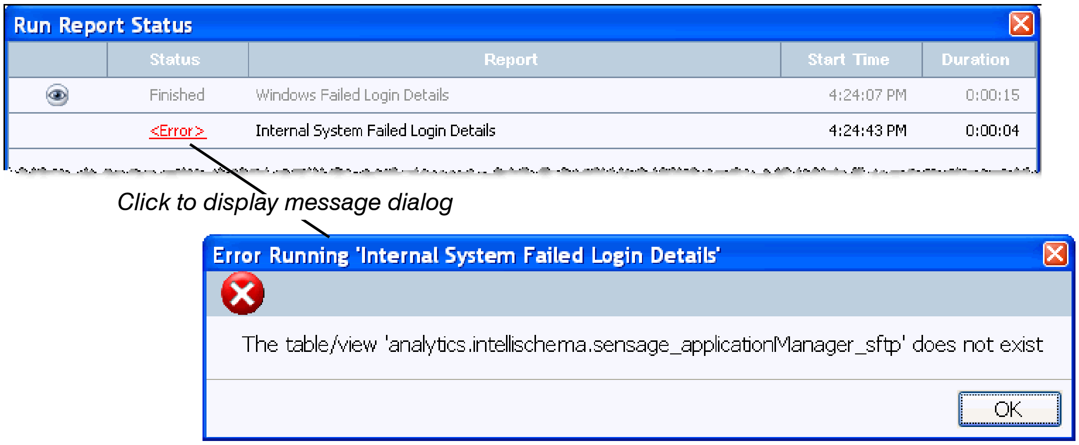


Run Report Status

	Status	Report	Start Time	Duration
	Canceled	HAVING SQL Parameter	12:03:10 PM	0:01:03

If the report fails to run, the **Run Report Status** dialog an <error> link. Click the link for error information, as illustrated in Figure 3-11.

Figure 3-11: Run Report Status Dialog: Encountering a Run Error



	Status	Report	Start Time	Duration
👁	Finished	Windows Failed Login Details	4:24:07 PM	0:00:15
	<Error>	Internal System Failed Login Details	4:24:43 PM	0:00:04

Click to display message dialog

The table/view 'analytics.intellischema.sensage_applicationManager_sftp' does not exist

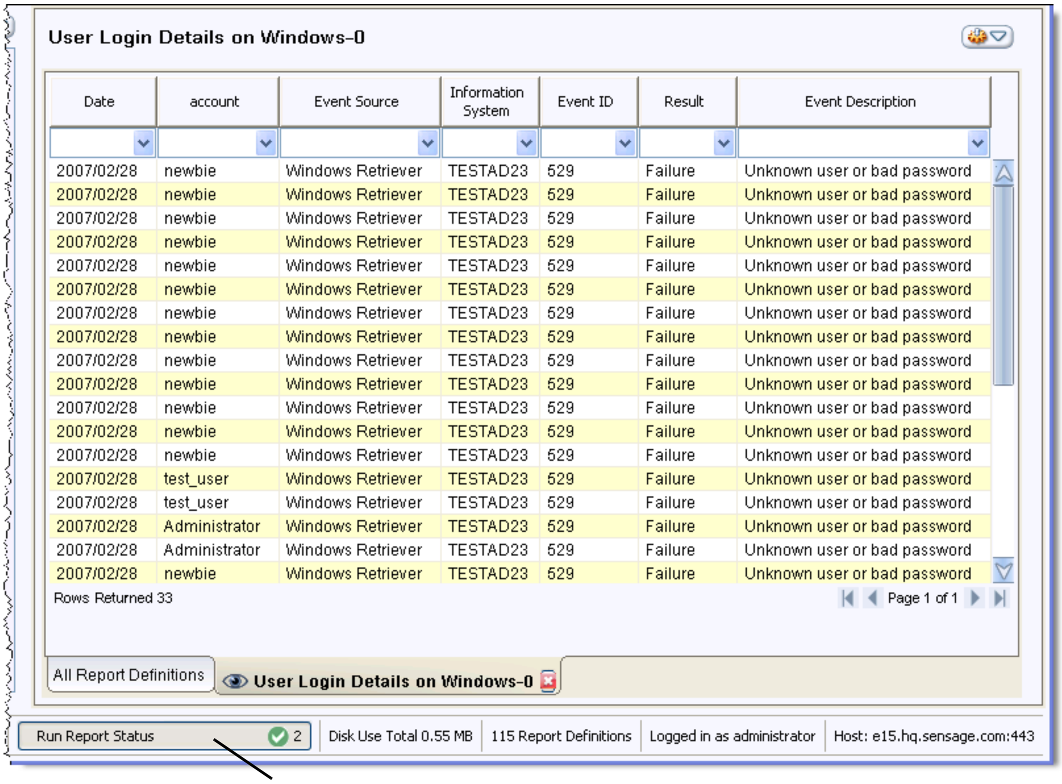
If running your report takes a long time, you can click **Minimize** or the standard Close icon to minimize the **Run Report Status** dialog. Regardless of which way you minimize the window, processing continues. Even if you close HawkEye AP Console, processing continues. You can use the status-bar **Run Report Status** button to determine when processing completes.

As soon as processing completes, the status-bar button increments the number of successfully run reports. You can reopen the dialog by clicking this button, as illustrated in Figure 3-12.

The status bar shows zero (0) reports run after you click **Clear all Finished** in the Run Report Status dialog or close and reopen HawkEye AP Console.

Figure 3-12 shows the *User Login Details on Windows* report displayed in Reports mode. The user selected Unknown User or bad password as the value for the **Event Description** column.

Figure 3-12: Viewing Your Report in the All Report Definitions List


User Login Details on Windows-0
Rows Returned 33
Page 1 of 1
All Report Definitions
User Login Details on Windows-0
Run Report Status
2
Disk Use Total 0.55 MB
115 Report Definitions
Logged in as administrator
Host: e15.hq.sensage.com:443

Date	account	Event Source	Information System	Event ID	Result	Event Description
2007/02/28	newbie	Windows Retriever	TESTAD23	529	Failure	Unknown user or bad password
2007/02/28	newbie	Windows Retriever	TESTAD23	529	Failure	Unknown user or bad password
2007/02/28	newbie	Windows Retriever	TESTAD23	529	Failure	Unknown user or bad password
2007/02/28	newbie	Windows Retriever	TESTAD23	529	Failure	Unknown user or bad password
2007/02/28	newbie	Windows Retriever	TESTAD23	529	Failure	Unknown user or bad password
2007/02/28	newbie	Windows Retriever	TESTAD23	529	Failure	Unknown user or bad password
2007/02/28	newbie	Windows Retriever	TESTAD23	529	Failure	Unknown user or bad password
2007/02/28	newbie	Windows Retriever	TESTAD23	529	Failure	Unknown user or bad password
2007/02/28	newbie	Windows Retriever	TESTAD23	529	Failure	Unknown user or bad password
2007/02/28	newbie	Windows Retriever	TESTAD23	529	Failure	Unknown user or bad password
2007/02/28	newbie	Windows Retriever	TESTAD23	529	Failure	Unknown user or bad password
2007/02/28	newbie	Windows Retriever	TESTAD23	529	Failure	Unknown user or bad password
2007/02/28	newbie	Windows Retriever	TESTAD23	529	Failure	Unknown user or bad password
2007/02/28	test_user	Windows Retriever	TESTAD23	529	Failure	Unknown user or bad password
2007/02/28	test_user	Windows Retriever	TESTAD23	529	Failure	Unknown user or bad password
2007/02/28	Administrator	Windows Retriever	TESTAD23	529	Failure	Unknown user or bad password
2007/02/28	Administrator	Windows Retriever	TESTAD23	529	Failure	Unknown user or bad password
2007/02/28	newbie	Windows Retriever	TESTAD23	529	Failure	Unknown user or bad password

Click to open or close the Run Report Status dialog

If you run several different reports at the same time, you can minimize the **Run Report Status** dialog while they run. The status bar automatically increments the number of reports run as each report completes processing.

Running Multiple Reports Simultaneously

Reports mode makes it easy to run several reports simultaneously, provided you do not want to change any default values. In other words, if you select and run multiple reports from the **All Reports Definitions** list, all selected reports display in the **Run Report Status** dialog and begin running immediately.

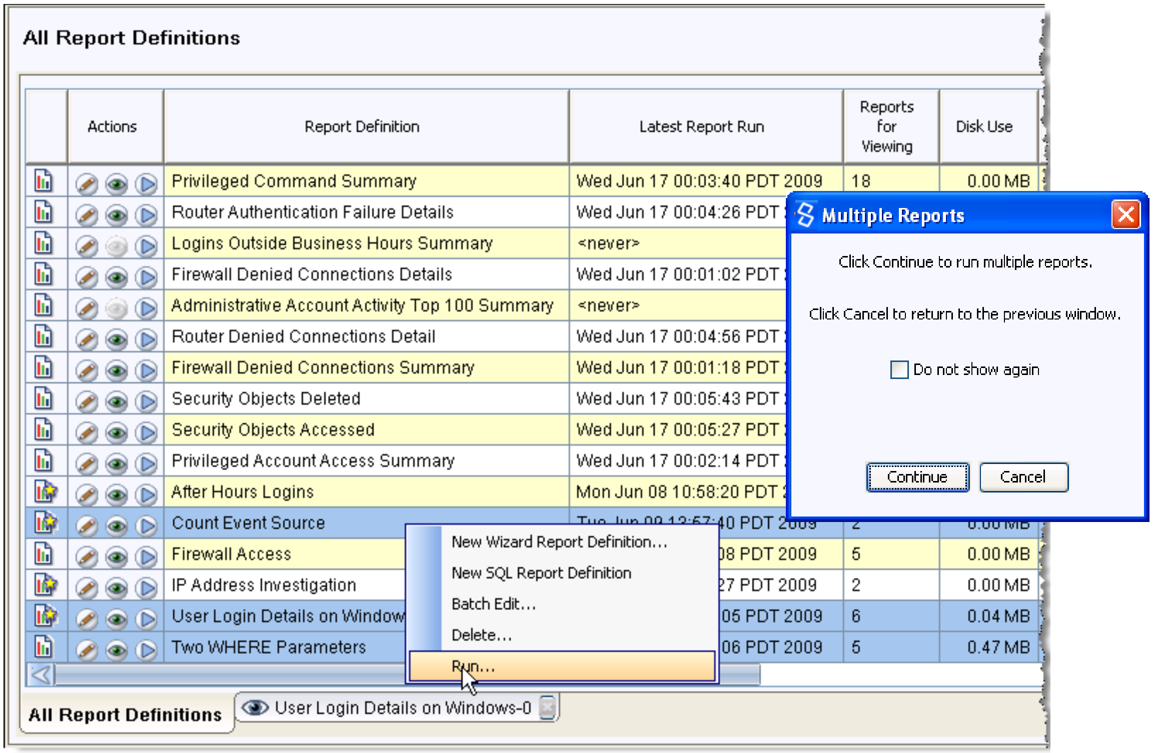
To select multiple definitions, use the standard interface keys:

- **Contiguous definitions**—select the first definition; SHIFT-CLICK as you select the last definition.
- **Non-contiguous definitions**—select the first definition; CTRL-CLICK as you select each additional definition.

Figure 3-13 illustrates selection of three reports in the **All Reports Definitions** list. The user has right-clicked to open the popup menu and has selected the **Run...** menu option. HawkEye AP

Console responds by displaying the **Multiple Reports** dialog, which prompts whether you really want to run multiple reports.

Figure 3-13: Running Multiple Reports Simultaneously



All Report Definitions

	Actions	Report Definition	Latest Report Run	Reports for Viewing	Disk Use
		Privileged Command Summary	Wed Jun 17 00:03:40 PDT 2009	18	0.00 MB
		Router Authentication Failure Details	Wed Jun 17 00:04:26 PDT		
		Logins Outside Business Hours Summary	<never>		
		Firewall Denied Connections Details	Wed Jun 17 00:01:02 PDT		
		Administrative Account Activity Top 100 Summary	<never>		
		Router Denied Connections Detail	Wed Jun 17 00:04:56 PDT		
		Firewall Denied Connections Summary	Wed Jun 17 00:01:18 PDT		
		Security Objects Deleted	Wed Jun 17 00:05:43 PDT		
		Security Objects Accessed	Wed Jun 17 00:05:27 PDT		
		Privileged Account Access Summary	Wed Jun 17 00:02:14 PDT		
		After Hours Logins	Mon Jun 08 10:58:20 PDT		
		Count Event Source	Tue Jun 09 12:57:40 PDT 2009	2	0.00 MB
		Firewall Access	08 PDT 2009	5	0.00 MB
		IP Address Investigation	27 PDT 2009	2	0.00 MB
		User Login Details on Window	05 PDT 2009	6	0.04 MB
		Two WHERE Parameters	06 PDT 2009	5	0.47 MB

Multiple Reports

Click Continue to run multiple reports.

Click Cancel to return to the previous window.

□ Do not show again

Continue
Cancel
New Wizard Report Definition...
New SQL Report Definition
Batch Edit...
Delete...
Run...
All Report Definitions
User Login Details on Windows-0

To run the three reports illustrated above, the user would click **Continue** in the **Multiple Reports** dialog. All three reports run with their default settings, including the default time period.

Managing Report Shortcuts in Folders

Reports mode provides folders, which enable you to group report shortcuts by usage, department, accessibility, or any other need. Grouping report definitions in folders allows you or an administrator to apply schedules and permissions to the definitions as a group, and to send a set of report cache entries to a department.

NOTE: Setting permissions on a folder does not effect the permission of the existing shortcuts in the folder nor the permissions of the report definitions that the shortcuts reference. The permissions set on a folder are copied to any new shortcuts or folders that you create in the folder. When you move a shortcut to another folder, permissions defined by the source folder are removed from the shortcut and permissions defined in the destination folder are copied to the shortcut. For more information, see:

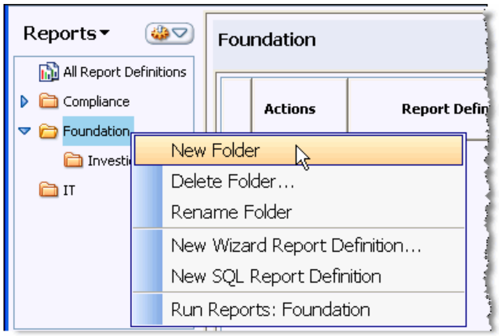
- “Viewing and Assigning Report, Dashboard, and Folder Permissions”, on page 122
- Managing Access to HawkEye AP Console Reports, Dashboards, and Folders in Chapter 8, “Administering Users and Authentication” of the *Administration Guide*.

Dragging a report definition into a folder from the **All Report Definition**s list creates a shortcut to the definition. In other words, the report definitions that display in your folders do not exist separately from their master report definitions. They only point to the actual definitions that display in the **All Report Definitions** list.

NOTE: If you delete a report definition from a folder, you delete only the shortcut. The master definition remains unchanged. If you modify a report definition from a folder, you modify all instances of the definition in every folder as well as the master definition.

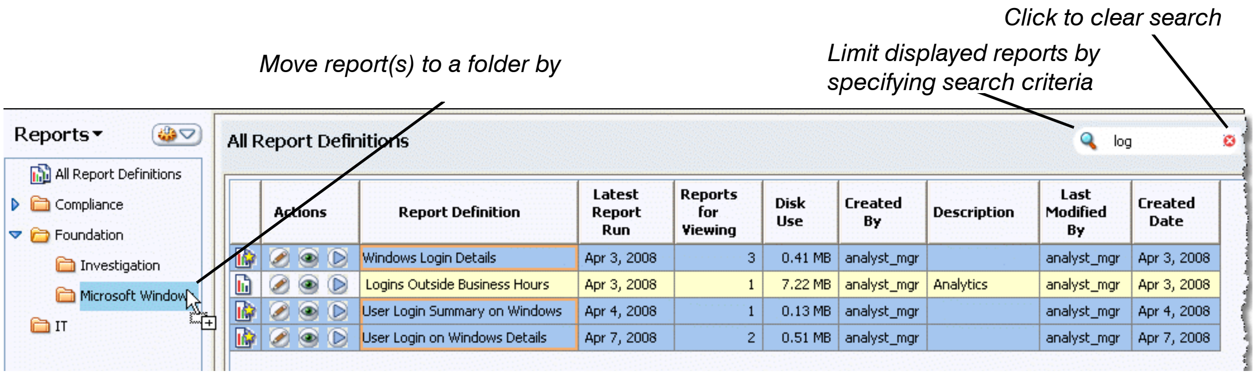
As illustrated in Figure 3-14, you can create a hierarchy of folders by creating one folder below another. In the graphic below, the user has right-clicked on the **Foundation** folder to display the **New Folder** menu. Alternately, you can create folders at the same level and drag one folder below the other.

Figure 3-14: Creating a Folder



After you create a folder, you populate it by dragging desired report definition(s) into it. Figure 3-14 illustrates the process of dragging three report definitions simultaneously to the **Microsoft Windows** folder. Before selecting the definitions to move, the user limited the number of displayed reports by entering criteria in the **Search** field.

Figure 3-15: Populating a Folder


Click to clear search

Move report(s) to a folder by

Limit displayed reports by specifying search criteria

Actions	Report Definition	Latest Report Run	Reports for Viewing	Disk Use	Created By	Description	Last Modified By	Created Date
	Windows Login Details	Apr 3, 2008	3	0.41 MB	analyst_mgr		analyst_mgr	Apr 3, 2008
	Logins Outside Business Hours	Apr 3, 2008	1	7.22 MB	analyst_mgr	Analytics	analyst_mgr	Apr 3, 2008
	User Login Summary on Windows	Apr 4, 2008	1	0.13 MB	analyst_mgr		analyst_mgr	Apr 4, 2008
	User Login on Windows Details	Apr 7, 2008	2	0.51 MB	analyst_mgr		analyst_mgr	Apr 7, 2008

The operation illustrated above drags three report definitions simultaneously to the **Microsoft Windows** folder.

NOTE:

- You can drag the same report definition to multiple folders.
- There is an alternate way to create and populate a folder.

Without first creating a folder, select the report definition(s) in the **All Report Definition**s list and drag them to the Navigator. HawkEye AP Console automatically creates a folder named "*`user_name.number`*", where *`user_name`* is the name of the user logged into HawkEye AP Console and the *`number`* is long enough to make the name unique. This folder contains shortcuts to all definitions you dragged to it. Rename the folder by right-clicking and selecting **Rename Folder** from the popup.

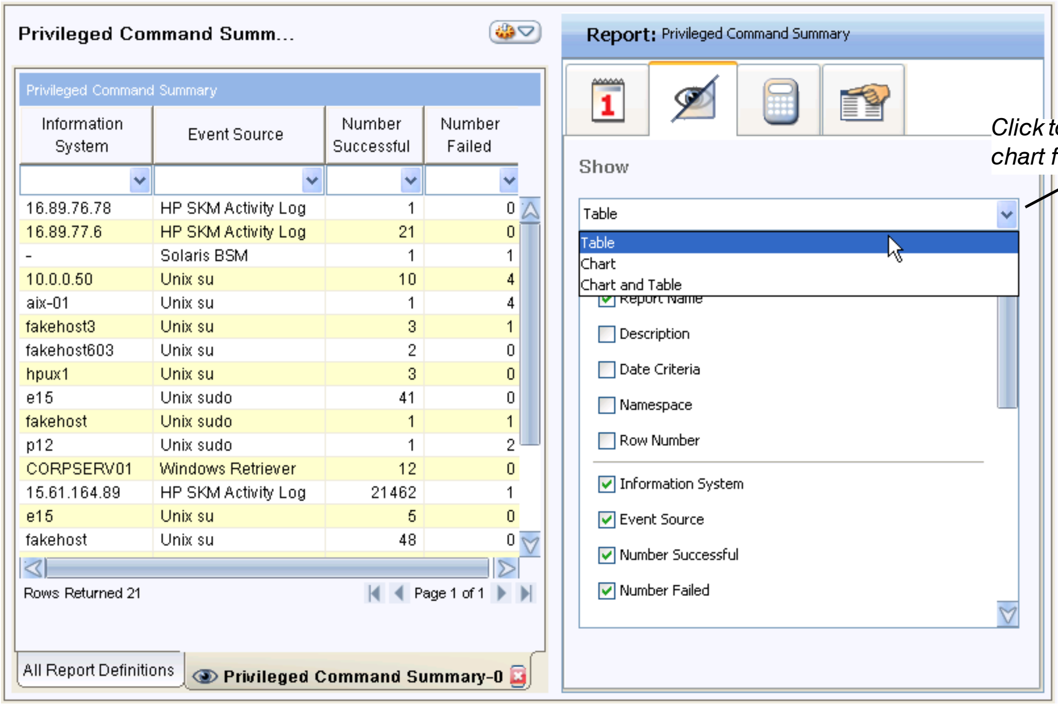
Modifying Display of a Report

If a tabular report displays unnecessary columns and you want to decrease the width of the report, you can use the Workspace Options Pane to hide specific columns. To open this pane, either click the Options Pane icon Options Pane above the workspace or select **Show Options** from the Workspace Action Menu.

The Options Pane provides the **Show** tab, which enables you to show and hide specific columns in a report and to show and hide its metadata, which provides information about the report. The metadata includes the report name, description, and the date criteria that determines its time period. From this tab, you can also select the **Row Number** option, which inserts a column of row numbers to the far left of the report.

The **Show** tab also enables you to change report display between table, chart, and table and chart for a report that has been configured as a chart. When you select **Table and Chart**, the report displays in both formats in the same window. Figure 3-16 illustrates how you can use this tab to change table and chart display.

Figure 3-16: Show/Hide Report Metadata and Data for Table Display



Privileged Command Summ...

Privileged Command Summary

Information System	Event Source	Number Successful	Number Failed
16.89.76.78	HP SKM Activity Log	1	0
16.89.77.6	HP SKM Activity Log	21	0
-	Solaris BSM	1	1
10.0.0.50	Unix su	10	4
aix-01	Unix su	1	4
fakehost3	Unix su	3	1
fakehost603	Unix su	2	0
hpux1	Unix su	3	0
e15	Unix sudo	41	0
fakehost	Unix sudo	1	1
p12	Unix sudo	1	2
CORPSERV01	Windows Retriever	12	0
15.61.164.89	HP SKM Activity Log	21462	1
e15	Unix su	5	0
fakehost	Unix su	48	0

Rows Returned 21 Page 1 of 1

All Report Definitions | Privileged Command Summary-0

Report: Privileged Command Summary

Show

Table

- Table
- Chart
- Chart and Table

☑ Report Name
□ Description
□ Date Criteria
□ Namespace
□ Row Number

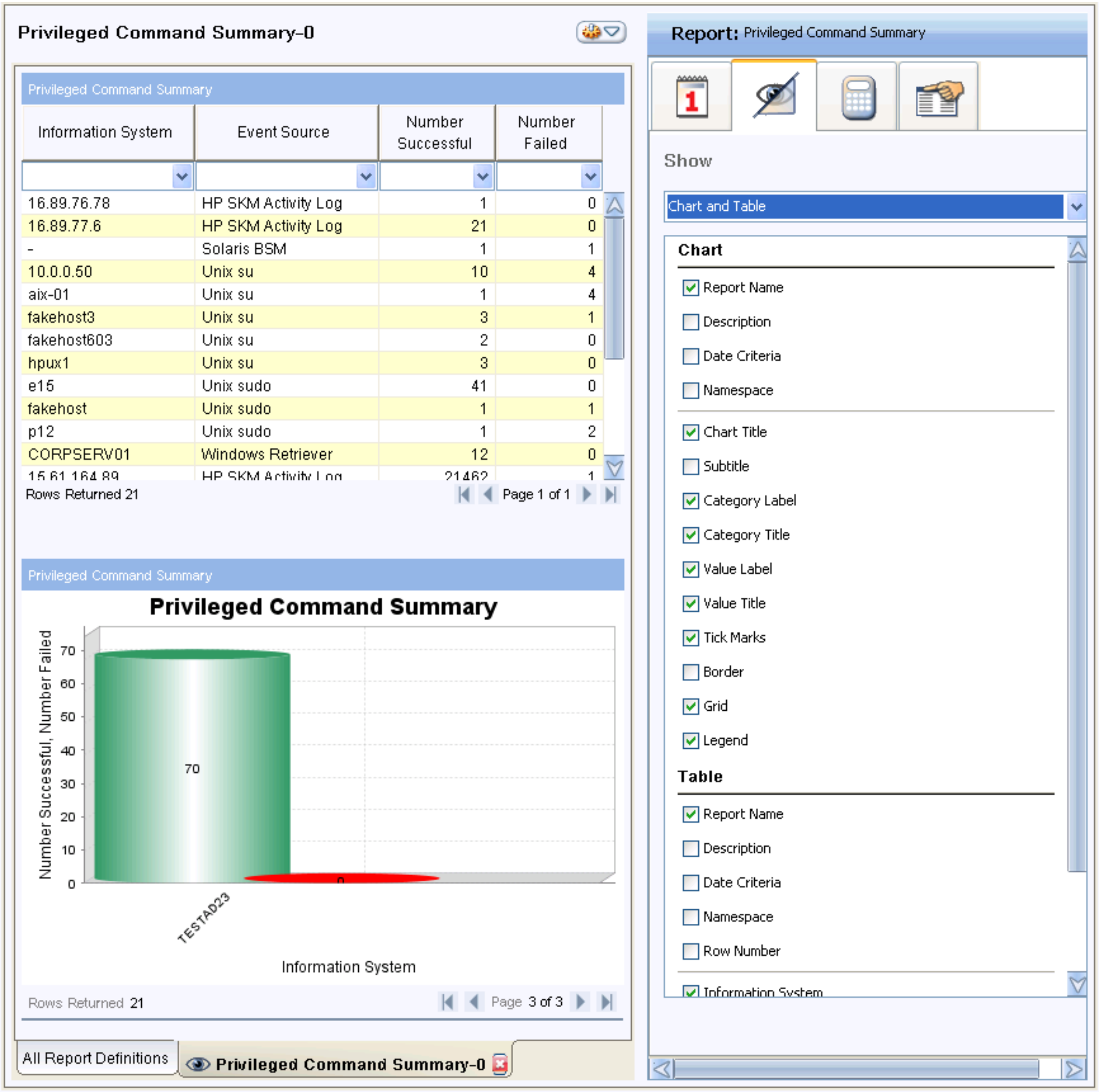
☑ Information System
☑ Event Source
☑ Number Successful
☑ Number Failed

Click to display chart format.

As illustrated above, the fields above the horizontal line enable you to show or hide metadata information. Currently the only metadata value selected is the report name. The fields below the horizontal line enable you to show or hide data columns. Currently all data columns have been selected for display.

When you display the report in both chart and table format, the set of options expands considerably. As illustrated in Figure 3-17, the set of fields expands to include metadata and chart values for chart display at the top. Below these are the fields for table metadata and columns.

Figure 3-17: Show/Hide Report Metadata and Data for Table and Chart Display



Privileged Command Summary-0

Privileged Command Summary

Information System	Event Source	Number Successful	Number Failed
16.89.76.78	HP SKM Activity Log	1	0
16.89.77.6	HP SKM Activity Log	21	0
-	Solaris BSM	1	1
10.0.0.50	Unix su	10	4
aix-01	Unix su	1	4
fakehost3	Unix su	3	1
fakehost603	Unix su	2	0
hpux1	Unix su	3	0
e15	Unix sudo	41	0
fakehost	Unix sudo	1	1
p12	Unix sudo	1	2
CORPSERV01	Windows Retriever	12	0
15.61.164.89	HP SKM Activity Log	21462	1

Rows Returned 21 Page 1 of 1

Privileged Command Summary

Rows Returned 21 Page 3 of 3

Report: Privileged Command Summary

Show: Chart and Table

Chart

- ☑ Report Name
- □ Description
- □ Date Criteria
- □ Namespace
- ☑ Chart Title
- □ Subtitle
- ☑ Category Label
- ☑ Category Title
- ☑ Value Label
- ☑ Value Title
- ☑ Tick Marks
- □ Border
- ☑ Grid
- ☑ Legend

Table

- ☑ Report Name
- □ Description
- □ Date Criteria
- □ Namespace
- □ Row Number
- ☑ Information System

NOTE: Although you can display both chart and tabular formats simultaneously when you view the report in Reports mode, Dashboards mode displays only one format style at a time. For more information, see “Defining Charts”, on page 171.

VIEWING REPORT RESULTS AND MANAGING REPORT CACHE ENTRIES

To view a report that has already run, click the View icon 👁 to the left of its name or select the report in the **All Report Definitions** list or folder list and then select **View** from the Workspace Action Menu. The report displays in a new tab. If it has been run multiple times over different time periods or intervals, the results of each run have been saved in a *report cache entry*.

As described in "Cached Data: Making Stored Data Quickly Available", on page 43, each report cache contains data for specific date criteria, column criteria, and namespace. The saved data, which precludes the need to query the EDW repeatedly for the same result set, improves access time.

Viewing and Changing Cache Entries

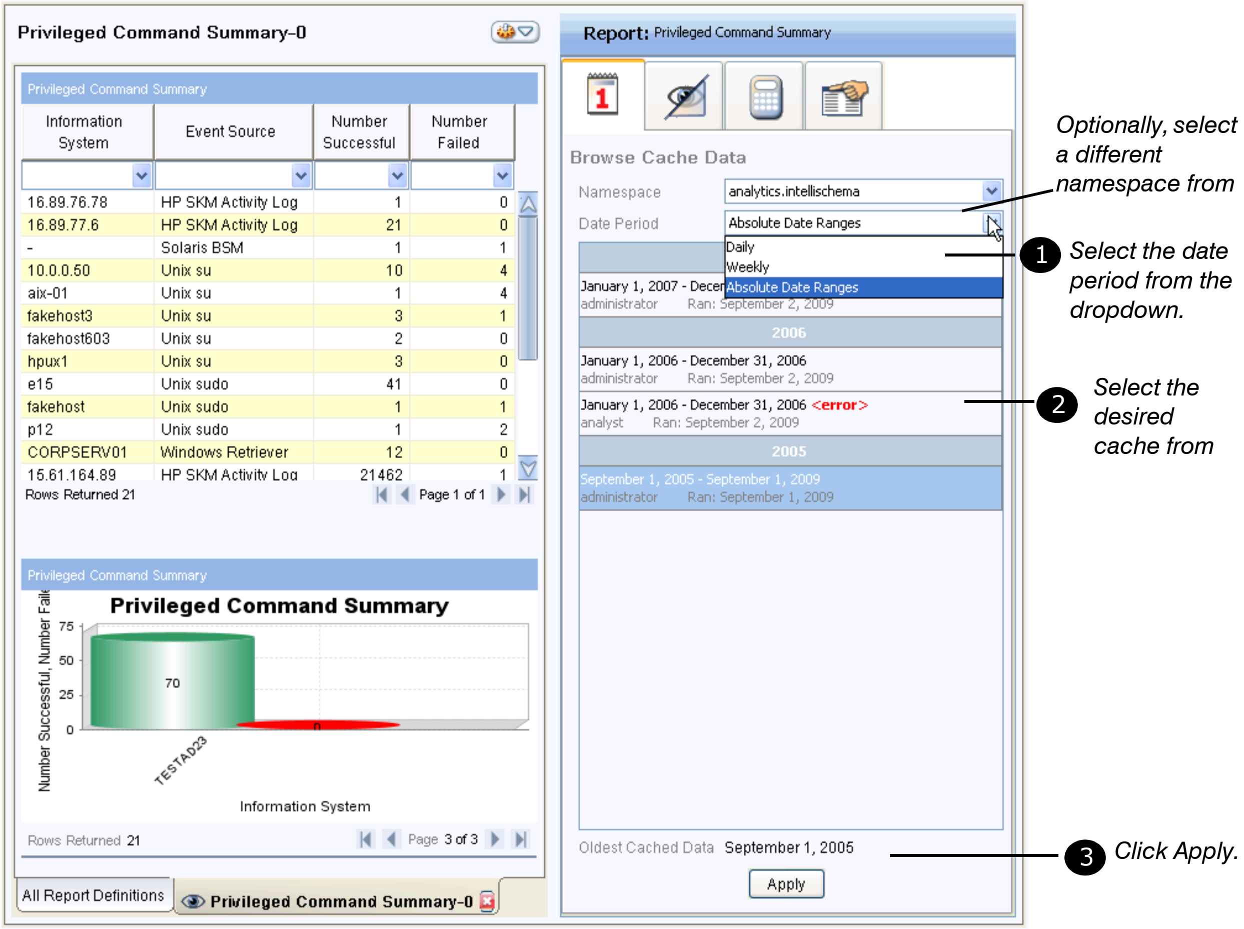
To display all cache entries associated with a report, use the **Browse Cache Date** tab of the Options Pane. Figure 3-18 illustrates this tab, which displays three sets of cache entries available to the Privileged Command Summary report:

- **Daily**—The report has daily runs; the **Date Criteria** field is set to `is yesterday`.
- **Weekly**—The report has weekly runs; the **Date Criteria** field is set to `is last week`.
- **Absolute Date range**—The report has been run manually multiple times.

If a report has multiple saved cache entries, you can access them all and choose the one(s) to display; select the desired period from the **Date Period** dropdown. If a report has run in more than one namespace, you can display cache(s) for another namespace; select the desired namespace from the **Namespace** dropdown.

Figure 3-18 illustrates how to change the displayed cache in the current namespace.

Figure 3-18: Selecting a Cache



Privileged Command Summary-0

Privileged Command Summary

Information System	Event Source	Number Successful	Number Failed
16.89.76.78	HP SKM Activity Log	1	0
16.89.77.6	HP SKM Activity Log	21	0
-	Solaris BSM	1	1
10.0.0.50	Unix su	10	4
aix-01	Unix su	1	4
fakehost3	Unix su	3	1
fakehost603	Unix su	2	0
hpux1	Unix su	3	0
e15	Unix sudo	41	0
fakehost	Unix sudo	1	1
p12	Unix sudo	1	2
CORPSERV01	Windows Retriever	12	0
15.61.164.89	HP SKM Activity Log	21462	1

Rows Returned 21 Page 1 of 1

Privileged Command Summary

Rows Returned 21 Page 3 of 3

All Report Definitions | Privileged Command Summary-0

Report: Privileged Command Summary

Browse Cache Data

Namespace: analytics.intellischema

Date Period: Absolute Date Ranges

January 1, 2007 - Decem
administrator Ran: September 2, 2009

2006

January 1, 2006 - December 31, 2006
administrator Ran: September 2, 2009

January 1, 2006 - December 31, 2006 <error>
analyst Ran: September 2, 2009

2005

September 1, 2005 - September 1, 2009
administrator Ran: September 1, 2009

Oldest Cached Data September 1, 2005

Optionally, select a different namespace from

1 *Select the date period from the dropdown.*

2 *Select the desired cache from*

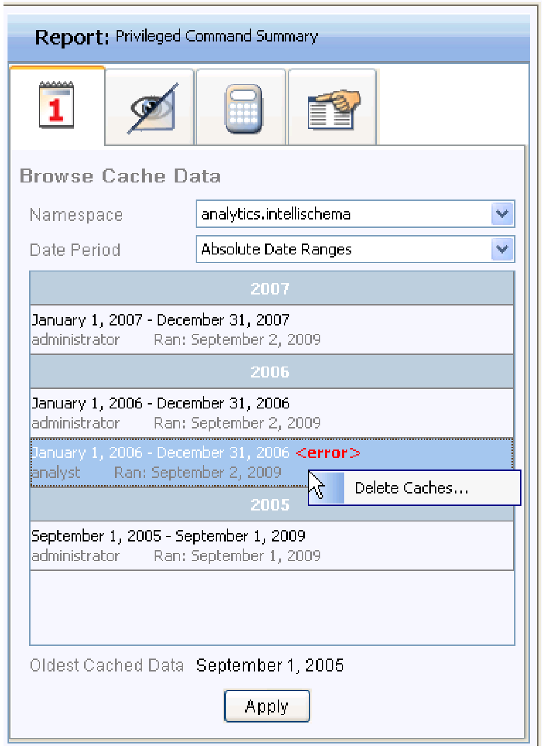
3 *Click Apply.*

Select one or more contiguous caches and click **Apply** to display the selected cache(s) in the workspace. For information on displaying multiple cache entries, see “Viewing and Changing the Time Range and Namespace”, on page 51.

Deleting Cache Entries

Over time, the Options Pane will display old cache entries that no longer interest you. To delete them, select contiguous one(s) you no longer want, right-click, and select **Delete Caches**. You will be prompted to confirm or cancel the deletion. Figure 3-19 illustrates this process.

Figure 3-19: Deleting Cache Entries



Report: Privileged Command Summary

Browse Cache Data

Namespace: analytics.intellischema

Date Period: Absolute Date Ranges

2007

January 1, 2007 - December 31, 2007
administrator Ran: September 2, 2009

2006

January 1, 2006 - December 31, 2006
administrator Ran: September 2, 2009

January 1, 2006 - December 31, 2006 <error>
analyst Ran: September 2, 2009

2005

September 1, 2005 - September 1, 2009
administrator Ran: September 1, 2009

Oldest Cached Data: September 1, 2005

Viewing the Report Query

The Options Pane also enables you to view and copy the query that retrieves and manipulates the report data.

- If the report is a Wizard report, the **SQL** tab displays a generated query.

  A user defines the report graphically. The SQL query is generated from the user's definition.

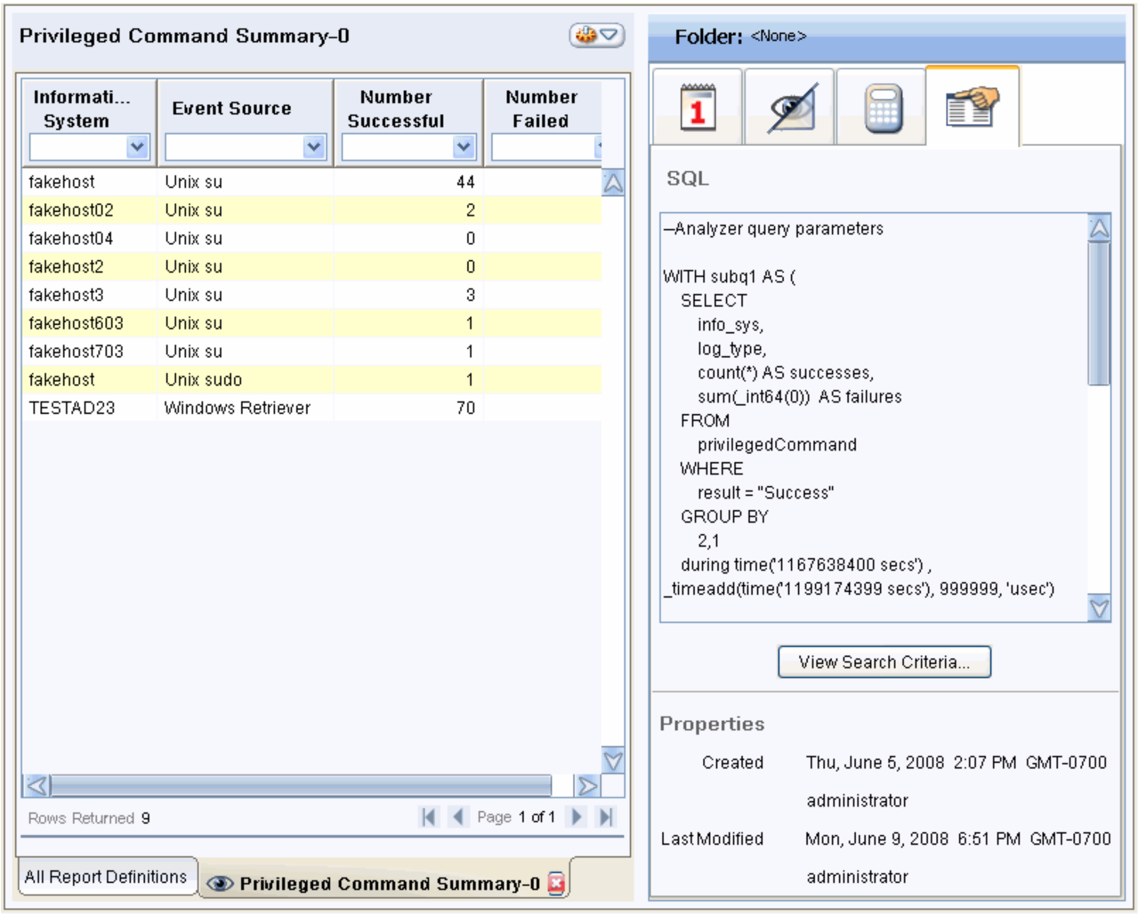
  If the report definition provides criteria that enable the run-time user to specify value(s), the query displayed in the **SQL** tab shows the user-specified value for each criterion. Locating the specified value does not require reading the SQL query, however. To display a disabled version of the actual **Run** dialog that displays the user-specified criteria values, click **View Search Criteria**.

- If the report is a SQL report, the **SQL** tab displays the SQL statement that a user entered to create the definition.

  If the report definition provides criteria that enable the run-time user to specify value(s), the top of the query displayed in the **SQL** tab shows the user-specified value for each criterion. The value displays as an OVERRIDE because the user has overridden the default value for the criterion. For more information, see “Viewing and Manipulating the SQL Query”, on page 200.

Figure 3-20 illustrates a SQL report that does not provide criteria.

Figure 3-20: Viewing the SQL Query and Report Properties



Privileged Command Summary-0

Informati... System	Event Source	Number Successful	Number Failed
fakehost	Unix su	44	
fakehost02	Unix su	2	
fakehost04	Unix su	0	
fakehost2	Unix su	0	
fakehost3	Unix su	3	
fakehost603	Unix su	1	
fakehost703	Unix su	1	
fakehost	Unix sudo	1	
TESTAD23	Windows Retriever	70	

Rows Returned 9

Page 1 of 1

```
—Analyzer query parameters

WITH subq1 AS (
   SELECT
      info_sys,
      log_type,
      count(*) AS successes,
      sum(_int64(0))  AS failures
   FROM
      privilegedCommand
   WHERE
      result = "Success"
   GROUP BY
      2,1
   during time('1167638400 secs') ,
_timeadd(time('1199174399 secs'), 999999, 'usec')
```

Properties

Created	Thu, June 5, 2008 2:07 PM GMT-0700
	administrator
Last Modified	Mon, June 9, 2008 6:51 PM GMT-0700
	administrator

NOTE:

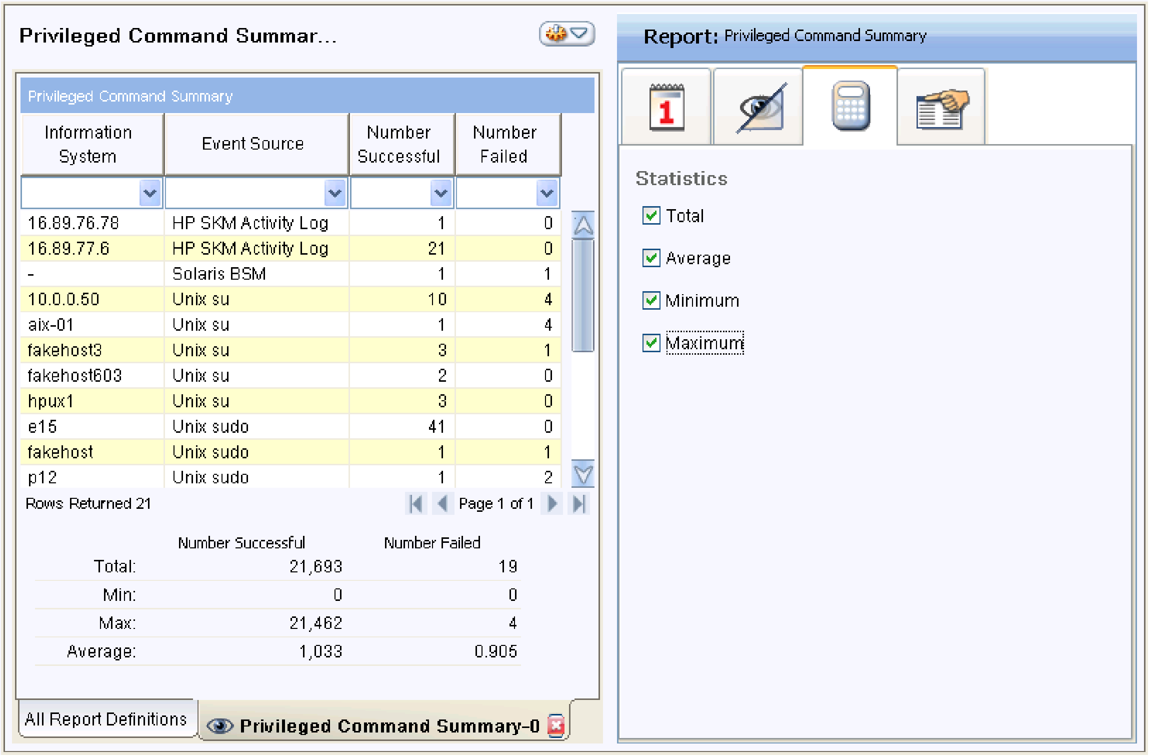
- You do not need to know any SQL to create a Wizard report. As you create and edit a Wizard report visually in the interface, HawkEye AP translates all your selected options into a HawkEye AP SQL query. When you run the saved Wizard report, HawkEye AP uses the underlying query to retrieve and manipulate the data according to your visual specifications.
- You can copy the query and paste it into any application that accepts text.

Viewing Report Statistics

The Options Pane enables you to calculate statistics on the report data. For each numeric column, you can total the values, calculate their average, and determine the minimum and maximum value. You can specify all of these calculations or only a subset.

Figure 3-21 illustrates the results when all four calculations have been selected.

Figure 3-21: Viewing Report Statistics



Privileged Command Summar...

Privileged Command Summary

Information System	Event Source	Number Successful	Number Failed
16.89.76.78	HP SKM Activity Log	1	0
16.89.77.6	HP SKM Activity Log	21	0
-	Solaris BSM	1	1
10.0.0.50	Unix su	10	4
aix-01	Unix su	1	4
fakehost3	Unix su	3	1
fakehost603	Unix su	2	0
hpux1	Unix su	3	0
e15	Unix sudo	41	0
fakehost	Unix sudo	1	1
p12	Unix sudo	1	2

Rows Returned 21 Page 1 of 1

	Number Successful	Number Failed
Total:	21,693	19
Min:	0	0
Max:	21,462	4
Average:	1,033	0.905

Report: Privileged Command Summary

Statistics

☑ Total

☑ Average

☑ Minimum

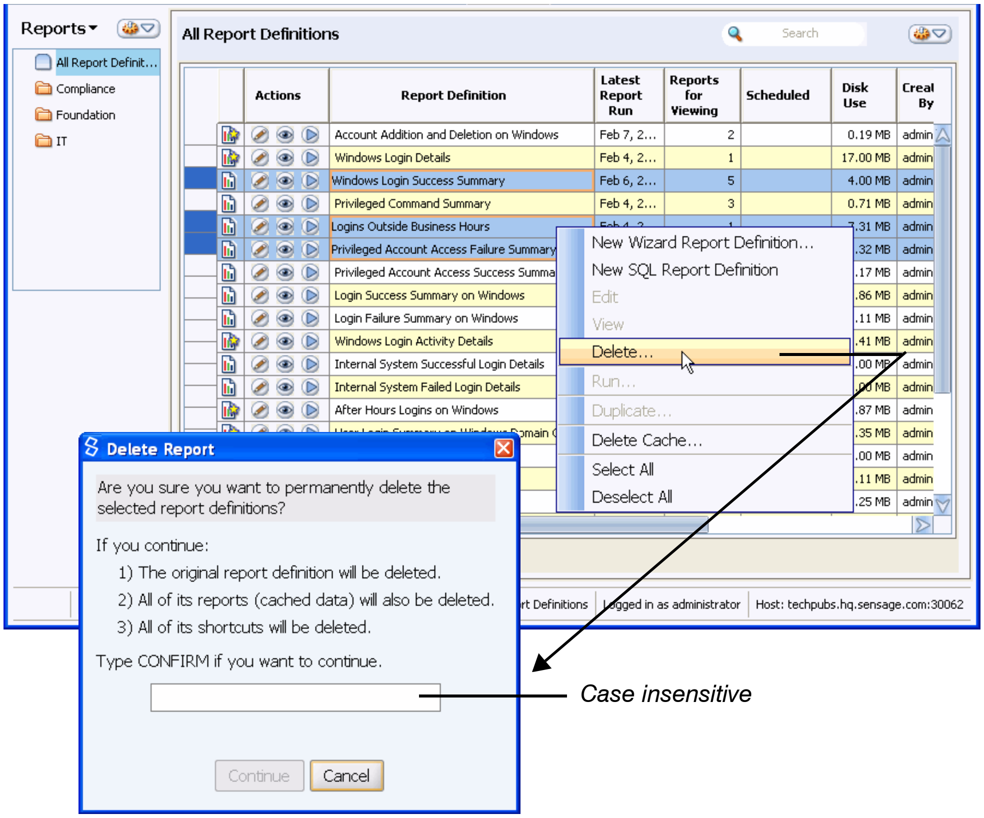
☑ Maximum

Deleting Report Definitions

You delete report definitions from the **All Report Definitions** list. Deleting a report definition also deletes all shortcuts to it. However, deleting a report shortcut does not affect its associated report definition.

When you delete a report or reports, a dialog displays that prompts you to verify the deletion by typing “`confirm`” in an entry field and clicking **Continue**. Figure 3-22 illustrates this process.

Figure 3-22: Deleting Reports



Actions	Report Definition	Latest Report Run	Reports for Viewing	Scheduled	Disk Use	Creal By
	Account Addition and Deletion on Windows	Feb 7, 2…	2		0.19 MB	admin
	Windows Login Details	Feb 4, 2…	1		17.00 MB	admin
	Windows Login Success Summary	Feb 6, 2…	5		4.00 MB	admin
	Privileged Command Summary	Feb 4, 2…	3		0.71 MB	admin
	Logins Outside Business Hours				.31 MB	admin
	Privileged Account Access Failure Summary				.32 MB	admin
	Privileged Account Access Success Summa				.17 MB	admin
	Login Success Summary on Windows				.86 MB	admin
	Login Failure Summary on Windows				.11 MB	admin
	Windows Login Activity Details				.41 MB	admin
	Internal System Successful Login Details				.00 MB	admin
	Internal System Failed Login Details				MB	admin
	After Hours Logins on Windows				.87 MB	admin
					.35 MB	admin
					.00 MB	admin
					.11 MB	admin
					.25 MB	admin

rt Definitions | Logged in as administrator | Host: techpubs.hq.sensage.com:30062

Are you sure you want to permanently delete the selected report definitions?

If you continue:

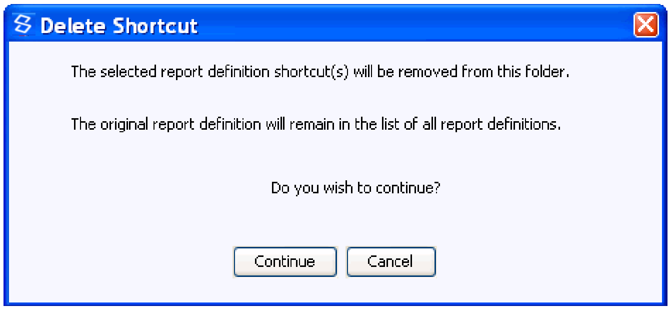
1) The original report definition will be deleted.
2) All of its reports (cached data) will also be deleted.
3) All of its shortcuts will be deleted.

Type CONFIRM if you want to continue.

Case insensitive

If you delete a report from the **All Report Definitions** list, you permanently delete the report from the system. However, if you delete a report from a folder, you simply remove the shortcut to the report. Figure 3-23 illustrates the dialog that displays when you delete a report shortcut.

Figure 3-23: Deleting a Shortcut



The selected report definition shortcut(s) will be removed from this folder.

The original report definition will remain in the list of all report definitions.

Do you wish to continue?

NOTE: If you mistakenly delete the master report definition, you must recreate it in the same way that is was first created.

Viewing and Assigning Report Schedules

A report can be associated to multiple schedules, such as daily, weekly, or monthly. Moreover, one schedule can contain multiple reports and dashboards. After you create a report or a folder of reports, a user assigned to the **administrator** role can schedule them. The scheduler is one of the options available from Administration mode. For more information, see Chapter 7: Creating and Editing Schedules.

Using Distribution Filters to Limit Viewable Data by Role

Distribution Filters allow a report creator to limit the rows a user can view in a report depending on the roles assigned to the person viewing the report. For example, you could create a distribution filter that enables a user who is a associated with an Accounting role to view data only from Accounting Department servers while a user who is associated with an IT role could view all of the data.

A Distribution Filter contains a filter expression that limits the report's output to rows that match the filter expression. The filter also associates one or more roles with the filter. Users who belong to these roles will only be able to view rows that match the filter expression.

An user must be associated with the `administrator` or `analyzer.admin` role to create Distribution filters using the Administration Mode of HawkEye AP Console. For more information, see Command Line: Creating and Managing Users, Roles, and Permissions in Chapter 8, “Administering Users and Authentication” in the *Administration Guide*.

A user who creates a report can then apply the filter to their report. Users who view the report must belong to a role associated with the filter to see rows that match the filter expression. Other users will not be able to view rows in the report that match the filter.

Applying a Distribution Filter to a Report

After a Distribution Filter is created, you can associate them to reports in Reports mode. You can choose from a list of existing reports, or you can associate a Distribution Filter to a new report you create in HawkEye AP Console.

To associate a Distribution Filter with an *existing* report

1 Open HawkEye AP Console and navigate to Reports mode.

2 In the Navigator tree, select the Reports folder that contains the report you want to associate with a Distribution Filter.

3 Select one or more reports from the Reports Definition list that you want to associate with a Distribution Filter.

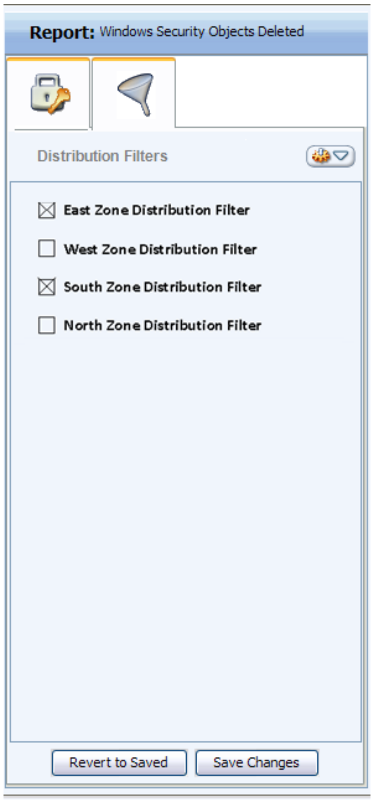
4 Click the **Options Pane** button in the Toolbar.

The Options Pane opens to the right.

5 Select the Distribution Filters Tab

A list of defined Distribution Filters displays in the Options Pane, as shown in Figure 3-24.

Figure 3-24: Distribution Filters Tab in the Options Pane



Report: Windows Security Objects Deleted

Distribution Filters

☒ East Zone Distribution Filter

☐ West Zone Distribution Filter

☒ South Zone Distribution Filter

☐ North Zone Distribution Filter

6 Select the Distribution Filter(s) you want to associate with this report.

7 Click the **Save Changes** button at the bottom of the Options Pane.

To associate a Distribution Filter with a *new* report

1 Begin creating a new SQL or Wizard report (see "Creating and Editing SQL Reports", on page 177 or "Creating and Editing Wizard Reports", on page 129).

2 Click the **Options Pane** button in the Toolbar.

The Options Pane opens to the right.

3 Select the Distribution Filters Tab

A list of defined Distribution Filters displays in the Options Pane, as shown in Figure 3-24.

4 Select the Distribution Filter(s) you want to associate with this report.

5 Click the **Save Changes** button at the bottom of the Options Pane.

6 Continue creating your report.

VIEWING AND ASSIGNING REPORT, DASHBOARD, AND FOLDER PERMISSIONS

This section describes how to use roles and permissions to control access to reports, dashboards, and folders in HawkEye AP Console and contains the following sections:

- “Overview of Roles and Permissions”, next
- “Default Roles and Permissions”, on page 123
- “Setting Roles and Permissions in HawkEye AP Console”, on page 125

Overview of Roles and Permissions

Users are associated with roles. Their membership in roles determines the types of action they can perform on specific items, such as whether they can view a dashboard or modify a report or open a folder. Administrators give users access to reports, dashboards, and folders by giving specific roles permissions to those items.

Creating New Reports, Shortcuts, Folders or Dashboards

- To create a folder, or dashboard, you must belong to a role that has **Edit** permission on the folder where the report, dashboard or folder is created. All report shortcuts, dashboards, or sub-folders subsequently created in a folder will take on the permissions associated with the folder. If a report shortcut, dashboard, or folder is moved from another folder into the current folder, the report, dashboard, or folder loses all permissions associated with the source folder, and takes on all permissions of the destination folder.

- To create a report, you must belong to the `analyzer.reports.creator` role. When you create a new report, the report is automatically granted "All" permissions for the `analyzer.report.creator` role.

- If you move a master report definition to a folder to create a shortcut, the shortcut takes on the permissions of the master report definition.

- A report, dashboard, or folder can also have permissions associated only with the report, dashboard, or folder. If such an object is moved into another folder, these additional permissions remain with object.

- To give a user access to a report, either create a new role or use an existing role, add the user to the role, select a report, and grant permissions to the role.

The roles to which a user is assigned as well as the permissions granted to the roles determines whether a user can edit a report or only run and view it or only run it or have no access to it. Users who have no permission to view a report will not see the report listed in the **All Report Definitions** list. They will also not see the report shortcut in a folder, even if they have full permissions on the folder. A shortcut has its own set of permissions, which are different than permissions associated with the report. For example, permissions on a shortcut may allow a user to see the report’s *listing* in the Report Definitions list, but that same user may not have permission to view the report’s *output*.

NOTE: Users gain the cumulative set of permissions from all roles to which they are assigned. In other words, if you are assigned to the Human Resources role, which has no permission to access an item, and to the Investigations role, which has permission to run and view a report, you have permission to run and view the report.

Special Roles

HawkEye AP provides the following three special roles:

- `administrator`–allows access to all HawkEye AP functionality
- `analyzer.alerts`–allows access to Security, Exception, and System Alerts (users must also have **View** permission for the dashboard containing the Security, Exception, or System Alert Widgets.
- `analyzer.admin`—allows access to HawkEye AP Console Administration mode
- `analyzer.reports`—allows access to Reports mode and grants Read permission to view folders within the All Reports Definition folder, however, a user having this role can only view the folders and cannot view the reports or report links.
- `analyzer.reports.creator`—allows the user to create a report. When a user creates a new report, Write permission is assigned to this role.

For more information, see:

- Managing Access to HawkEye AP Console Reports, Dashboards, and Folders in Chapter 8, “Administering Users and Authentication” in the *Administration Guide*.

Default Roles and Permissions

You can use folders to establish default permissions for reports, dashboards, and folders. For each folder you can define the following permissions for each available role:

- View—allows the user to view a report or dashboard, or to view a list of items contained in the folder
- Edit—allows the user to modify a report definition, dashboard, or folder
- Run—allows the user to run a report or dashboard

The initial permissions for a report, dashboard, or folder are determined by the permissions set on the folder where the new report, dashboard, or folder is created. In a new installation of HawkEye AP software, the roles and permissions are defined as shown in Table 3-1.

Table 3-1: Initial roles and permissions

Folder	Role	Permission
Dashboard folders (created at the top level of the Navigator tree)	administrator	View, Edit, Run
	analyzer.admin	View, Edit, Run
	analyzer.dashboards	View
Report folders (created at the top level of the Navigator tree)	administrator	View, Edit, Run
	analyzer.admin	View, Edit, Run
	analyzer.reports.creator	View, Edit, Run
	analyzer.reports	View
All Report Definitions folder	administrator	View, Edit, Run
	analyzer.admin	View, Edit, Run
	analyzer.reports.creator	View, Edit, Run
	analyzer.reports	View

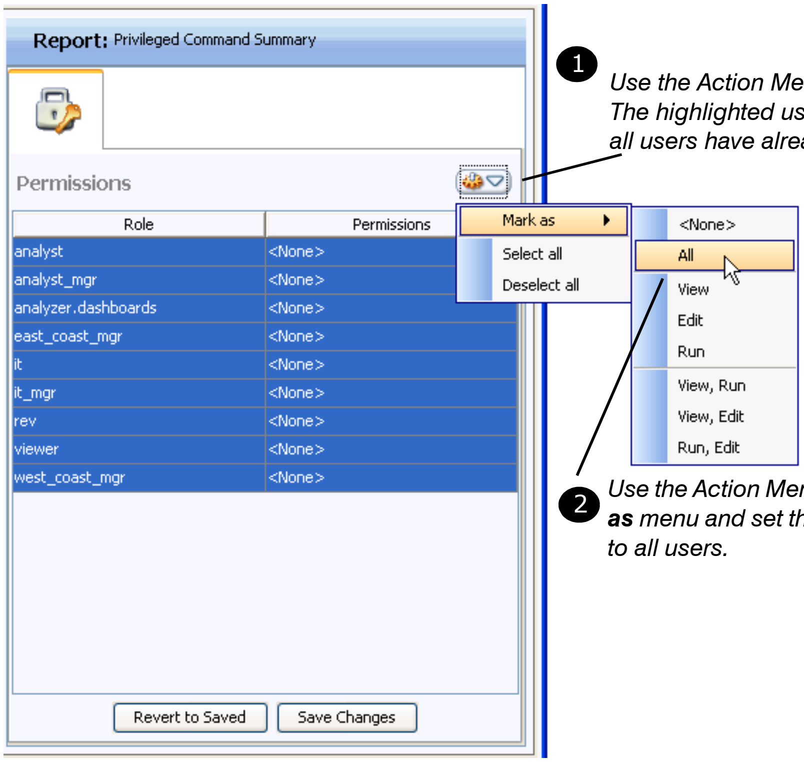
Any new reports, dashboards, or folders you create in the above folders will take on the roles and permissions indicated in Table 3-1. If you want a different set of initial roles and permissions, a HawkEye AP administrator can create new folders under these folders that have a different set of permissions where users can create new reports, dashboards, or folders that have the desired set of initial permissions.

Figure 3-25 and Figure 3-26 illustrate the **Permissions** tab for a report: Privileged Command Summary. This tab, which functions identically for folder permissions, enables you to set report permissions to specific roles and to view the assigned permissions. A user with administration permission creates and modifies the permissions in Administration mode.

Setting Roles and Permissions in HawkEye AP Console

As illustrated below, you can select all roles and then apply the same permission to all roles.

Figure 3-25: Assigning Permissions to a Report Collectively



Report: Privileged Command Summary

Permissions

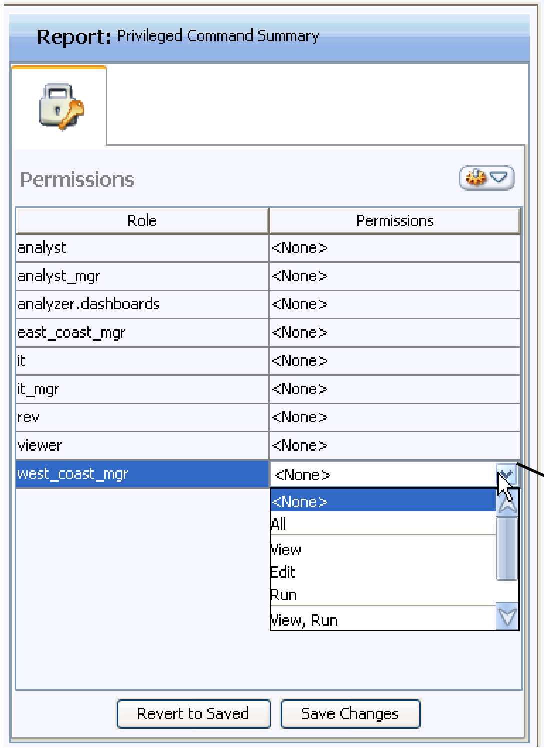
Role	Permissions
analyst	<None>
analyst_mgr	<None>
analyzer.dashboards	<None>
east_coast_mgr	<None>
it	<None>
it_mgr	<None>
rev	<None>
viewer	<None>
west_coast_mgr	<None>

1 *Use the Action Menu to* ***Select all*** *users. The highlighted users here indicate that all users have already been selected.*

2 *Use the Action Menu to select the* ***Mark as*** *menu and set the same permissions to all users.*

As illustrated below, you can specify permissions separately for each role from the dropdown next to each role.

Figure 3-26: Assigning Permissions to Roles for a Report



Report: Privileged Command Summary

Permissions

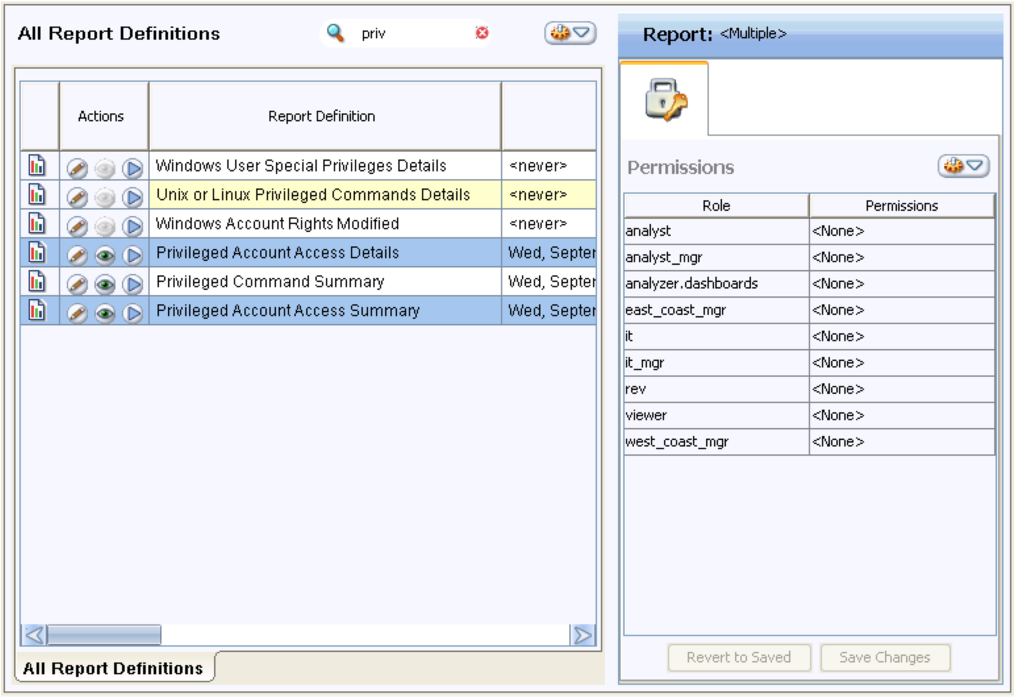
Role	Permissions
analyst	<None>
analyst_mgr	<None>
analyzer.dashboards	<None>
east_coast_mgr	<None>
it	<None>
it_mgr	<None>
rev	<None>
viewer	<None>
west_coast_mgr	<None>

Use the dropdown to set permissions for each role

As illustrated above, you can assign the same permissions to all roles or can assign specific permissions to specific roles. In both examples above, the user is assigning permissions to a single report. If you have several reports and you want to assign the same permissions to all reports, you can do so with a single operation.

To assign permissions to multiple reports simultaneously, select the desired reports and perform the same operations illustrated above. Figure 3-27 illustrates a user assigning permissions to two reports at the same time.

Figure 3-27: Assigning Permissions to Multiple Reports



	Actions	Report Definition	
		Windows User Special Privileges Details	<never>
		Unix or Linux Privileged Commands Details	<never>
		Windows Account Rights Modified	<never>
		Privileged Account Access Details	Wed, Septer
		Privileged Command Summary	Wed, Septer
		Privileged Account Access Summary	Wed, Septer

Role	Permissions
analyst	<None>
analyst_mgr	<None>
analyzer.dashboards	<None>
east_coast_mgr	<None>
it	<None>
it_mgr	<None>
rev	<None>
viewer	<None>
west_coast_mgr	<None>

As illustrated above, the two selected reports already have assigned permissions. The presence of `<Mixed>` in most of the Permissions rows indicates that each report assigned different permissions to these roles. Only the permissions assigned to the analyst_mgr role are identical for the two selected reports.

The user can easily make all settings the same for both reports by clicking in each `<Mixed>` row and selecting the desired permission for each role. Alternately, the user can use the Action Menu to select all roles and assign the same permissions to all roles.

NOTE: The **Permissions** tab lists all roles defined for your EDW instance, except the `administrator` role.

CHAPTER 4

Creating and Editing Wizard Reports

This chapter contains the following sections:

- “Creating Wizard Reports”, next
- “Running and Viewing the Wizard Report”, on page 149
- “Editing a Wizard Report”, on page 151

CREATING WIZARD REPORTS

The Report Wizard enables you to graphically select desired columns from a desired table or view and to specify operations on the selected data. You can start this wizard from any Action Menu or right-click menu in Reports mode.

This section includes the following topics:

- “Step 1: Specifying Where to Get the Data”, next
- “Step 2: Specifying What Data to Return”, on page 133
- “Step 3: Specifying Further Refinements”, on page 146
- “Running and Viewing the Wizard Report”, on page 149

Step 1: Specifying Where to Get the Data

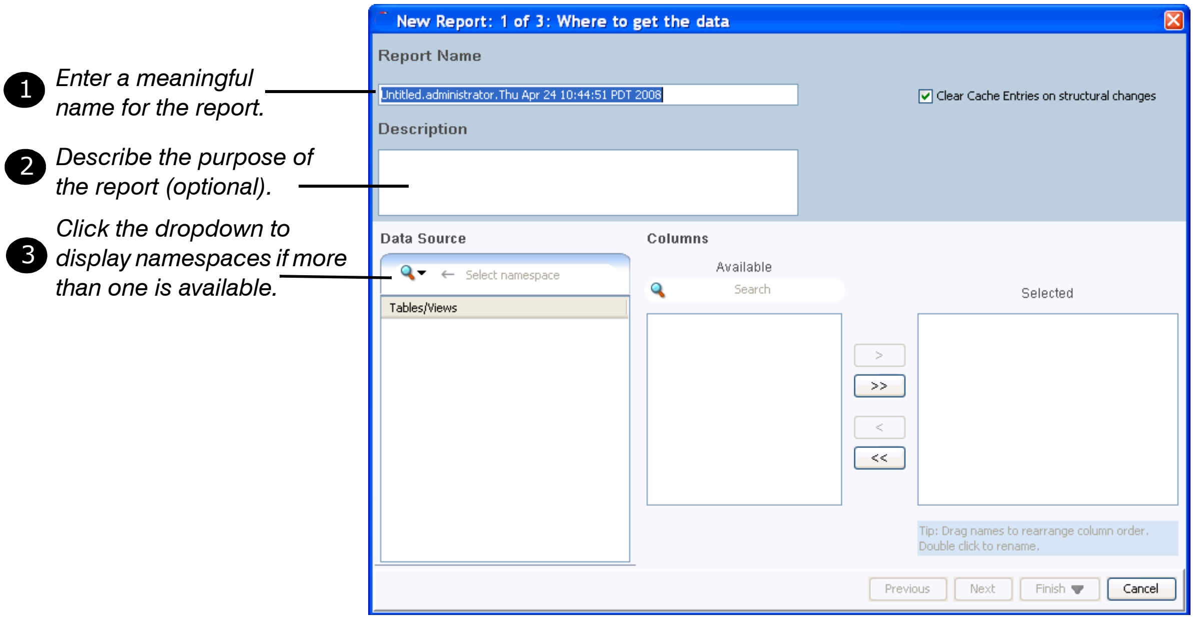
This topic includes the following headings:

- “Specifying Report Name and Data Source”, next
- “Understanding How HawkEye AP Displays Date and Time”, on page 132
- “Selecting, Renaming, and Ordering Columns for Display”, on page 132

Specifying Report Name and Data Source

Select **New Wizard Report Definition...** to display the first step of the Report Wizard. This step enables you to name and describe the report and to specify the data on which to base the report. Figure 4-1 illustrates this window as it first displays.

Figure 4-1: Naming and Describing the Report and Specifying the Data



1 *Enter a meaningful name for the report.*

2 *Describe the purpose of the report (optional).*

3 *Click the dropdown to display namespaces if more than one is available.*

New Report: 1 of 3: Where to get the data

Report Name: Untitled.administrator.Thu Apr 24 10:44:51 PDT 2008

☑ Clear Cache Entries on structural changes

Tip: Drag names to rearrange column order. Double click to rename.

- **Report Name**—Every report must have a unique name, which can be a combination of any characters except the slash (/).

  NOTE: Depending on your permissions, you might not see all reports in the All Report Definitions list. However, you cannot name a report identically to one that already exists in your system. The system notifies you if the name you enter is a duplicate.

- **Description**—You can provide a short description for each report. Your description displays in the **All Report Definitions** pane and can display when a user views the report. The description can be helpful to users who have access to many reports.

- **Data Source**—As described in "Namespaces: Using a Single Report or Dashboard to Access Different Data", on page 43, HawkEye AP organizes tables and views into namespaces. Namespaces allow your site to limit access to data by user permissions and to run the same report against identically named tables that store different data.

  - If you have access to more than one namespace, the **Data Source** dropdown prompts you to select a namespace. After you click the dropdown to select a namespace, all tables and views for which you have access permission display below the **Data Source** field.
  - If you have access to ten or more namespaces, the **Select Namespace** dialog displays. This dialog enables you to choose from all available namespaces.
  - If you have access to only one namespace, its name displays in the **Data Source** field and all tables and views for which you have access permission display below it.

- **Clear Cache Entries on structural changes**—When this box is checked, all cache entries for this report are deleted any time you make changes to this report definition and save those changes. If you leave this box unchecked and change the report definition, you will not be able to save those changes until you first delete all cache entries for the report. If you try to save changes to a report definition that has existing cache entries, you will see a warning message but there is no way to keep any changes you have made.

It may be useful to check this box when iteratively developing and debugging a report, so that you do not have to manually delete cache entries after each test run of the report. When the report is ready for use in a production environment, Sensage recommends that you uncheck this box so that cache entries are not unexpectedly deleted.

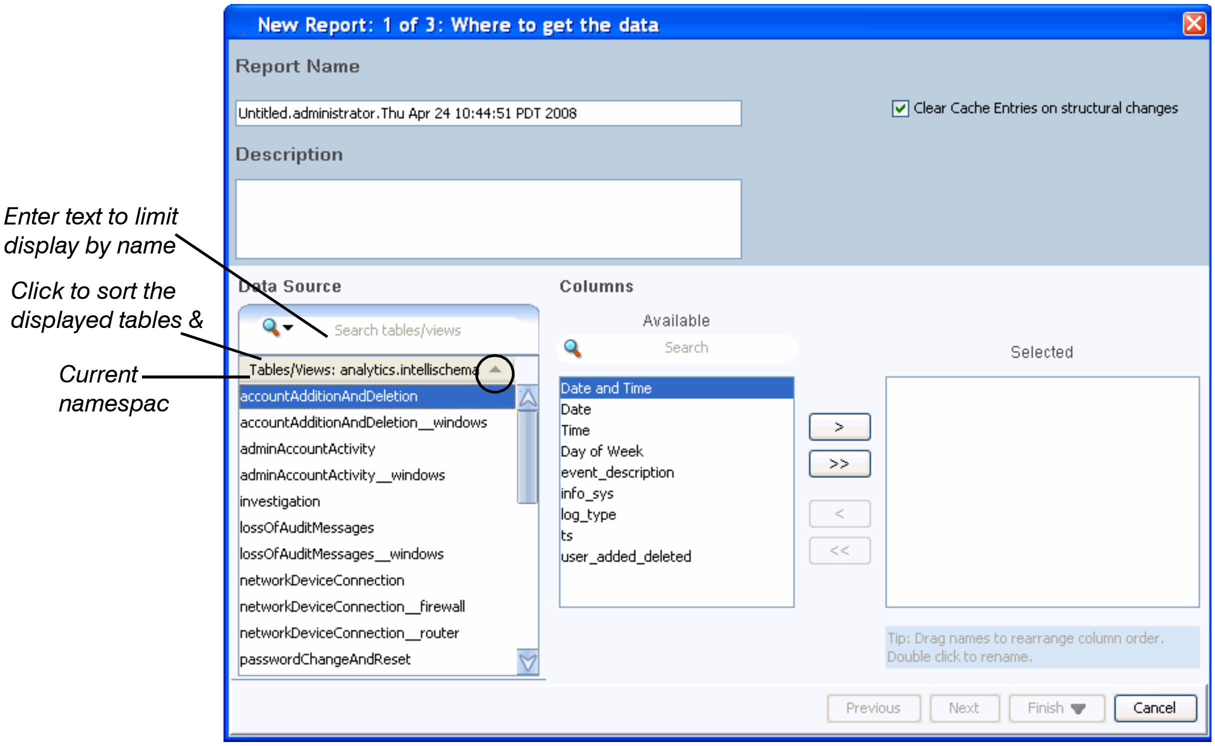
NOTE: Opening the Data-Source dropdown may take a while the first time you open HawkEye AP Console.

NOTE: Depending on how preferences have been set at your site, only views may display on your system.

TIP: HawkEye AP Analytics views simplify report creation by presenting data in a more meaningful and consistent way than tables.

As illustrated in Figure 4-2, after the tables and views display, you can click the namespace field to sort the data sources. The triangle icon ▼ or ▲ indicates a ascending or descending sort order, respectively. The columns for the selected table display in the **Available** pane.

Figure 4-2: Specifying the Report Data Source



Enter text to limit display by name

Click to sort the displayed tables &

Current namespac

New Report: 1 of 3: Where to get the data

Report Name

Untitled.administrator.Thu Apr 24 10:44:51 PDT 2008

☑ Clear Cache Entries on structural changes

Description

Data Source

Search tables/views

Tables/Views: analytics.intellischema ▲

accountAdditionAndDeletion
accountAdditionAndDeletion__windows
adminAccountActivity
adminAccountActivity__windows
investigation
lossOfAuditMessages
lossOfAuditMessages__windows
networkDeviceConnection
networkDeviceConnection__firewall
networkDeviceConnection__router
passwordChangeAndReset

Columns

Available

Search

Date and Time
Date
Time
Day of Week
event_description
info_sys
log_type
ts
user_added_deleted

Selected

Tip: Drag names to rearrange column order. Double click to rename.

TIP: For Power Users—After you first display all your namespaces and the tables and views they contain, you will not see changes to these objects until the next time you log into HawkEye AP Console. In other words, if another user adds or removes namespaces or tables or views, these changes do not display dynamically during your current session.

Understanding How HawkEye AP Displays Date and Time

Every table that HawkEye AP creates and every view that it delivers contains a **ts** (timestamp) column. HawkEye AP uses this column to index and manage the data it stores. A record with a **ts** column is called an *event.* The EDW is optimized to handle event data.

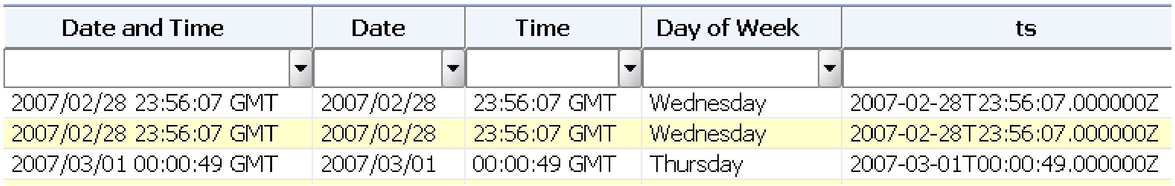
The value of the **ts** column contains accuracy down to the microsecond, and GMT time zone. Because the **ts** column contains so much information, it can be a challenge to determine where the data for one interval ends and the next begins from the raw **ts** value. It can also change the way your data displays. For more information, see "Summarizing Data", on page 154.

NOTE: The EDW stores all **ts** data in GMT.

To streamline the report creation process, the Report Wizard provides four columns that are derived from the **ts** column: **Date and Time**, **Date**, **Time**, and **Day of Week**. This isolation of day and time data enables you to create reports that examine time intervals; for example, you can create a report to identify activity that occurs outside ordinary business hours or to display only time intervals during a single day. For an example of such a report, see "Column Criteria: Limiting the Number of Returned Rows", on page 139.

Figure 4-3 illustrates the granularity of data displayed in the derived date columns as well as in the **ts** column.

Figure 4-3: Illustrating Options for Date and Time Column Data



Date and Time	Date	Time	Day of Week	ts
2007/02/28 23:56:07 GMT	2007/02/28	23:56:07 GMT	Wednesday	2007-02-28T23:56:07.000000Z
2007/02/28 23:56:07 GMT	2007/02/28	23:56:07 GMT	Wednesday	2007-02-28T23:56:07.000000Z
2007/03/01 00:00:49 GMT	2007/03/01	00:00:49 GMT	Thursday	2007-03-01T00:00:49.000000Z

As an example, this chapter documents how to create a Wizard report that identifies potentially suspect user logins during off-hours activity. To track the off-hours activity, the example report takes advantage of the **Day of Week** and **Date** columns. For more information, see "Column Criteria: Limiting the Number of Returned Rows", on page 139.

NOTE: The dates above use GMT as the time zone. You can change the default time zone in your preference settings. For more information, see "Setting Preferences", on page 35.

Selecting, Renaming, and Ordering Columns for Display

After you select the desired table or view, your next task is to select the desired columns by moving them from the **Available** into the **Selected** field. To locate specific available columns, you can either use the scroll bar or enter text in the **Search** field. For information about entering search text, see "Limiting Display by Text Search", on page 33.

Use the arrow buttons to move all columns or selected columns between the **Available** and **Selected** fields. To move multiple columns simultaneously, use the standard interface keys:

- **Contiguous columns**—select the first column; SHIFT-CLICK as you select the last column
- **Non-contiguous columns**—select the first column; CTRL-CLICK as you select each additional column

To rearrange the order of columns for display, drag and reposition them in the **Selected** field.

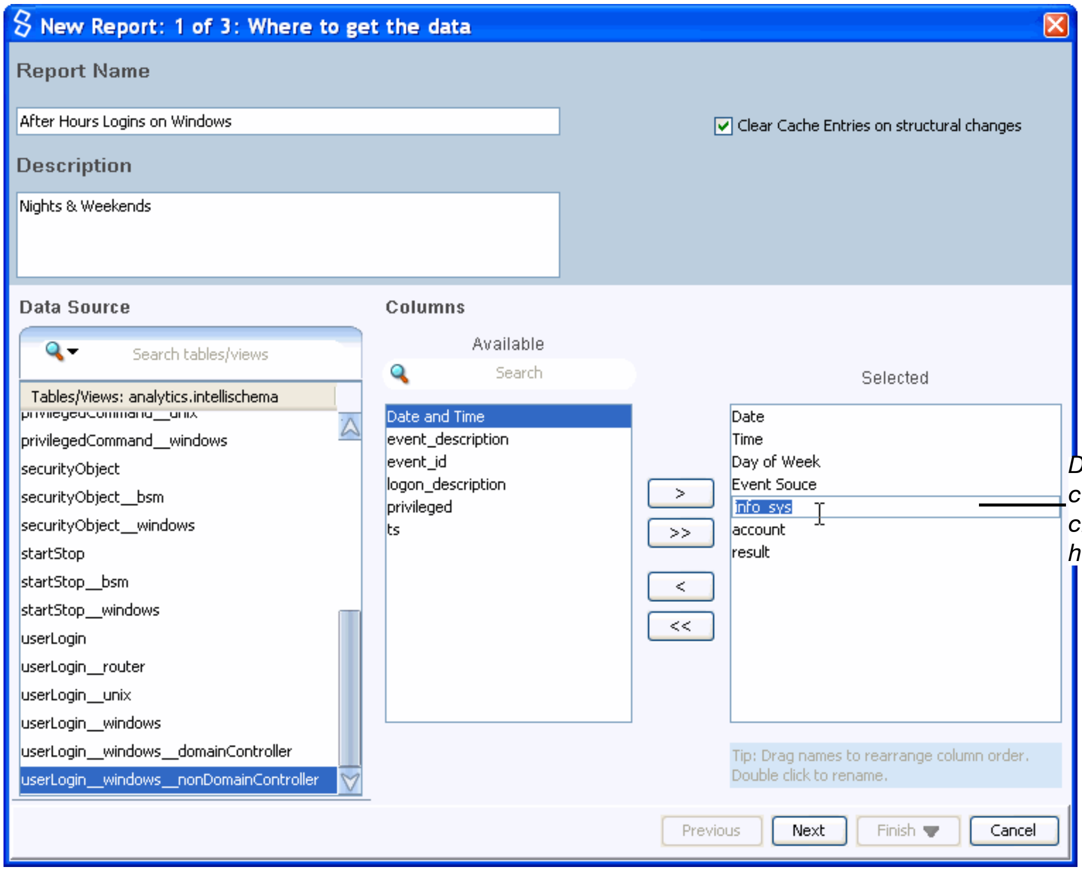
NOTE: You can use this field to change column order only when you first create the report. After you save and run it, you use a different window to change column order. For more information, see "Formatting Columns", on page 167.

To rename columns for display, double click them in the **Selected** field and enter the desired name.

NOTE: After you save and run the report, you use a different window to rename columns. For more information, see "Formatting Columns", on page 167.

Figure 4-4 illustrates creation of a wizard report based on a view named **userLogin**.

Figure 4-4: Specifying Columns for Display



New Report: 1 of 3: Where to get the data

Report Name: After Hours Logins on Windows

☑ Clear Cache Entries on structural changes

Description: Nights & Weekends

Data Source

Tables/Views: analytics.intellischema

privilegedCommand__unix
privilegedCommand__windows
securityObject
securityObject__bsm
securityObject__windows
startStop
startStop__bsm
startStop__windows
userLogin
userLogin__router
userLogin__unix
userLogin__windows
userLogin__windows__domainController
userLogin__windows__nonDomainController

Columns

Available:
Date and Time
event_description
event_id
logon_description
privileged
ts

Selected:
Date
Time
Day of Week
Event Souce
info_sys
account
result

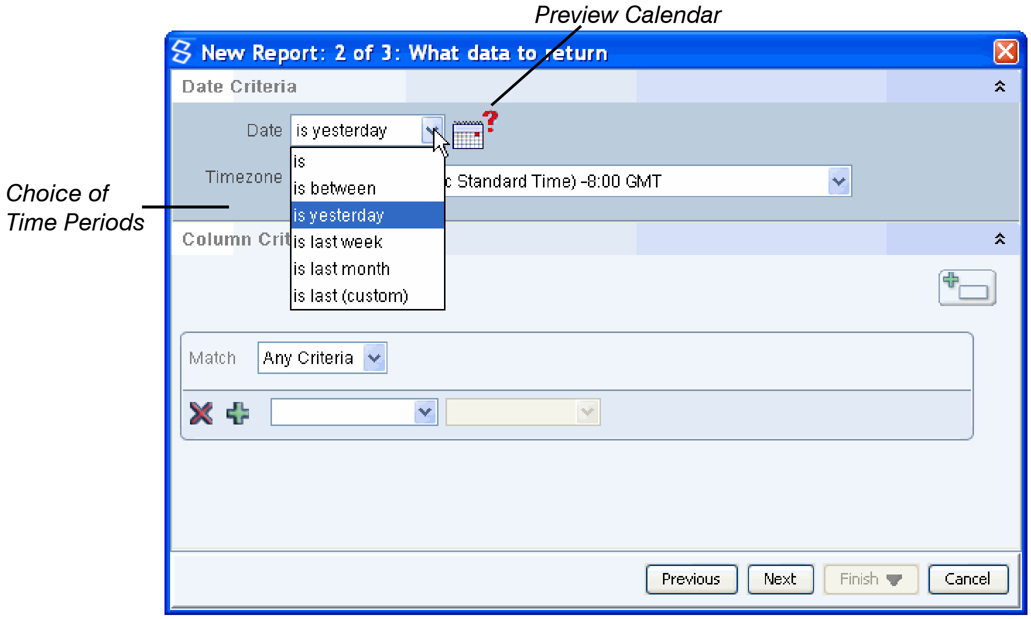
Tip: Drag names to rearrange column order. Double click to rename.

Double click a column name to change its report heading.

Step 2: Specifying What Data to Return

In the first Wizard step, illustrated in Figure 4-4, you selected the report data source and specified which of its columns to display. Perhaps you also renamed the report columns and changed their order. In the second Wizard step, illustrated in Figure 4-5, you have the opportunity to limit the number of returned rows by setting report criteria.

Figure 4-5: Limiting Results by Specifying Date Criteria



Preview Calendar

Choice of Time Periods

Each row stored in the EDW represents a single incoming event. The EDW stores all events that enter the system. When you create a report, you limit its results to a desired subset of events. You can use this step to limit the rows returned from the EDW by specifying:

- **Date Criteria**—a period of time over which to run your query; for example, you can limit data to a specific week or month or year.
- **Column Criteria**—column value(s); for example, you can limit results to a specific IP address.

Date Criteria: Specifying Time Range

As illustrated in Figure 4-5, there are several options for limiting the time range:

- **is**—all events occur between 12:00:00 am and 11:59:59 pm
- **is between**—all events occur between two user-specified dates and times
- **is yesterday**—all events occur *yesterday* between 12:00:00 am and 11:59:59 pm
- **is last week**—all events occur between Monday through Sunday

  NOTE: You can set the week to begin on Sunday by editing your preferences. For more information, see "Setting Preferences", on page 35.
- **is last month**—all events occur between the first and last days of the month
- **is last (custom)**

By default, the criteria window displays the **is yesterday** keywords because reports commonly cover the 24-hour period that ended at midnight yesterday in the time zone set in your preferences. When you specify a relative date period for a report, such as **is yesterday** or **is last**

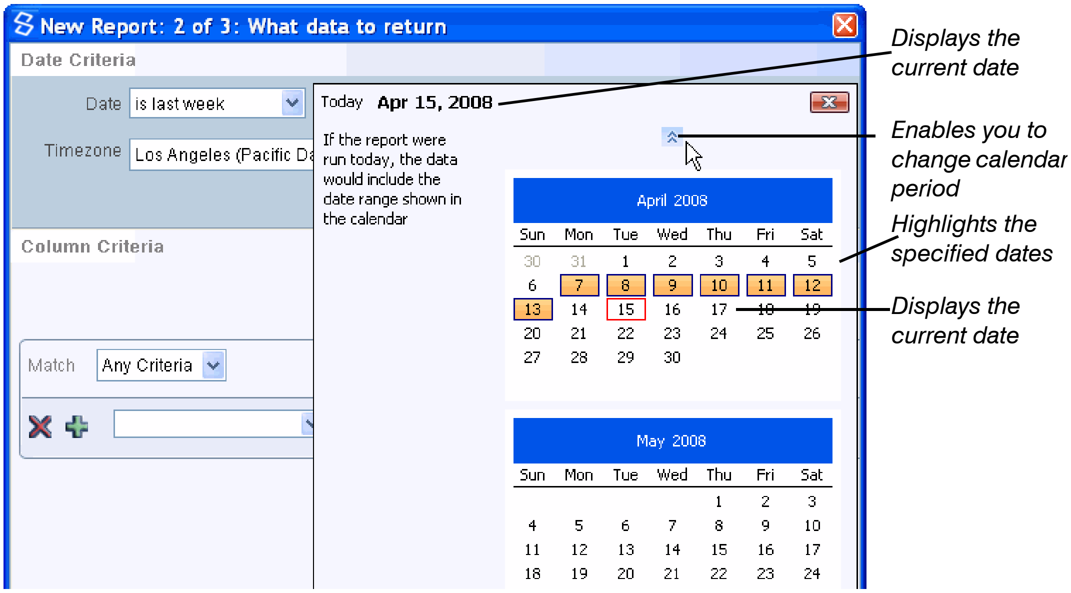
week, that period is returned every time a schedule runs the report. The value you specify here sets the default date range for the report. A user who runs the report manually can change the period, as can the person who schedules the report.

NOTE: If a user in New York has his time zone set to his local time and he sends an **is yesterday** report to a user in San Francisco, the San Francisco recipient views events that occurred in New York at the times the events occurred in New York.

When you choose relative date criteria, the Wizard displays a preview calendar icon, as illustrated in Figure 4-6.

Optionally, click the icon to validate your selection. Clicking this icon displays a calendar that illustrates the selected time range. Use this calendar to verify your specified date(s). As illustrated in Figure 4-6, the preview calendar identifies the current date and highlights the specified date(s).

Figure 4-6: Displaying the Preview Calendar



New Report: 2 of 3: What data to return

Date Criteria

Date: is last week

Timezone: Los Angeles (Pacific D

Column Criteria

Match: Any Criteria

Today Apr 15, 2008

If the report were run today, the data would include the date range shown in the calendar

April 2008

Sun	Mon	Tue	Wed	Thu	Fri	Sat
30	31	1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30			

May 2008

Sun	Mon	Tue	Wed	Thu	Fri	Sat
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24

Displays the current date

Enables you to change calendar period

Highlights the specified dates

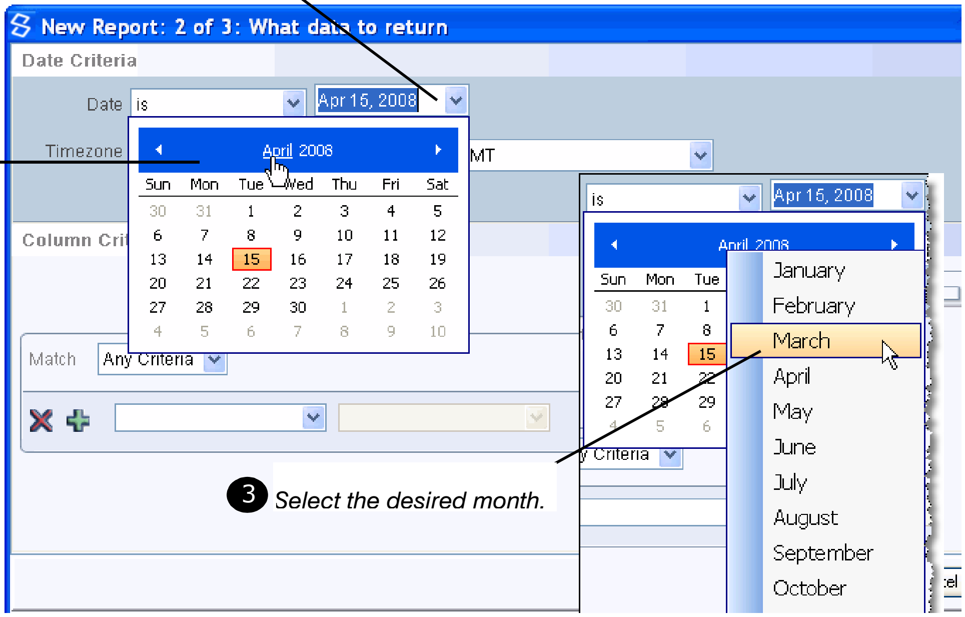
Displays the current date

Figure 4-6 illustrates that the period for last week begins on a Monday. You can set your system so that each week begins instead on a Sunday. For more information, see "Setting Preferences", on page 35.

If you select the **is** keyword from the **Date** dropdown, the Wizard provides a single date field that displays the current date by default. You can directly modify the text in the **Date** entry field.

Alternately, you can click its dropdown to display the calendar and change the date graphically, as illustrated in Figure 4-7.

Figure 4-7: Selecting the Date from the Calendar



1 *Click the down arrow to display the calendar.*

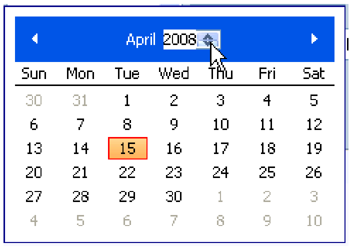
2 *Click the month to display a*

3 *Select the desired month.*

NOTE: To change the default date from the calendar, you must select a date before you move focus from the calendar.

TIP: There are other ways to specify the date from the calendar popup:

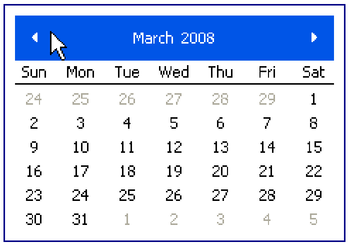
- Click the year to select the year.



April 2008

Sun	Mon	Tue	Wed	Thu	Fri	Sat
30	31	1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	1	2	3
4	5	6	7	8	9	10

- Click the left or right arrows in the header to move backward or forward by months.

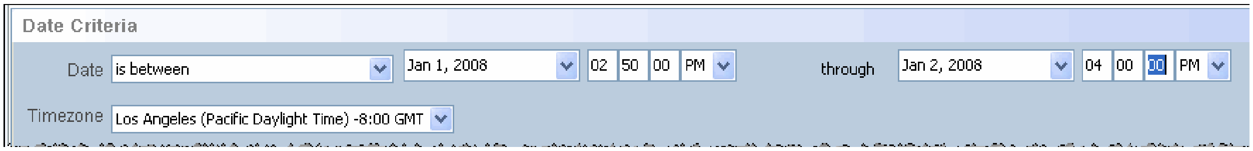


March 2008

Sun	Mon	Tue	Wed	Thu	Fri	Sat
24	25	26	27	28	29	1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31	1	2	3	4	5

When you select **is between** from the **Date** dropdown, the Wizard displays a second date entry field and each date entry field includes a time field, as illustrated in Figure 4-8. Use the same tools to modify the values in each of these fields.

Figure 4-8: Specifying a Range of Dates and Times



Date Criteria

Date: is between | Jan 1, 2008 | 02 | 50 | 00 | PM | through | Jan 2, 2008 | 04 | 00 | 00 | PM

Timezone: Los Angeles (Pacific Daylight Time) -8:00 GMT

NOTE: By default, the time range is `00 00 00 AM` through `11 59 59 PM`, which represents midnight of the earlier date to midnight of the later date.

To understand the dates and times in the example above, assume the system contains the following records:

Jan 1, 2008 01:00 AM

Jan 1, 2008 04:00 AM

Jan 1, 2008 03:50 PM

Jan 1, 2008 11:59 PM

Jan 2, 2008 02:00 AM

Jan 2, 2008 05:00 PM

Given the example dates and times, the report returns records for the following dates:

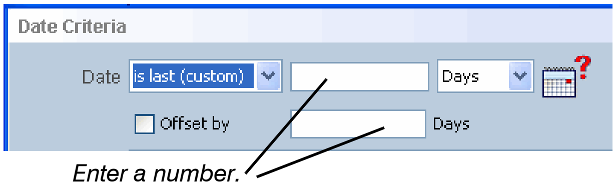
Jan 1, 2008 03:50 PM

Jan 1, 2008 11:59 PM

Jan 2, 2008 02:00 AM

When you choose a custom date range, the Wizard displays several new fields, as illustrated in Figure 4-9.

Figure 4-9: Specifying a Custom Date Range



The custom date-range option provides additional fields that enable you to configure the interval and an offset for your report. Enter a numeric value in the first field and use the dropdown to specify the interval (**Seconds**, **Minutes**, **Hours**, **Days**, **Weeks**, **Months**, or **Years**) that the number represents.

NOTE: Each interval includes midnight of the first day through midnight of the last day. For example, the **is yesterday** selection indicates an interval that begins midnight of the day before yesterday and ends at midnight yesterday for the specified time zone. The **is last month** selection indicates an interval that begins midnight of the first day of the month and ends at midnight of the last day of the month for the specified time zone. For example, assume that on April 17 you run a last-month report. The results reflect 12:00:00 am on March 1 through 11:59:59 pm on March 31.

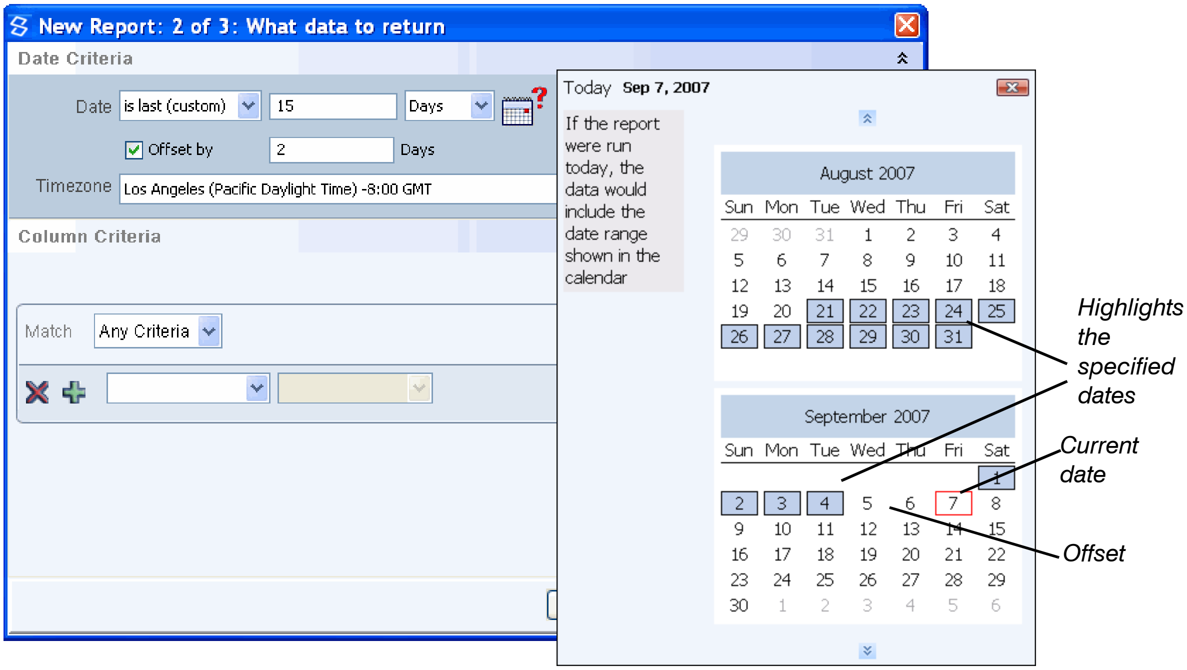
The custom date-range option also provides an offset field that enables you to specify an interval by which to move the reporting period backward. This feature allows you to run the report at times that are meaningful to your data collection. For example, assume you want to report on data gathered over the past 7 days. Assume further that you know that the data for yesterday is incomplete. A report that includes yesterday's data would be misleading. You would set the interval to 7 days and, to cause the report to skip yesterday's data, set the offset to "`1`". The offset shifts the collection backward by one day as desired.

To verify that the time range and offset have been specified correctly, click the 📅 icon to display a preview calendar that illustrates the selected time range.

To display previous months as well as the current month, click the previous icon ⌃ that displays above the top calendar.

Figure 4-10 illustrates a 15-day time range that is offset by 2 days from the current date.

Figure 4-10: Displaying the Preview Calendar for a Custom Time Range



New Report: 2 of 3: What data to return

Date Criteria

Date: is last (custom) | 15 | Days

☑ Offset by | 2 | Days

Timezone: Los Angeles (Pacific Daylight Time) -8:00 GMT

Column Criteria

Match: Any Criteria

Today Sep 7, 2007

If the report were run today, the data would include the date range shown in the calendar

August 2007

Sun	Mon	Tue	Wed	Thu	Fri	Sat
29	30	31	1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	31	

September 2007

Sun	Mon	Tue	Wed	Thu	Fri	Sat
						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	1	2	3	4	5	6

Highlights the specified dates

Current date

Offset

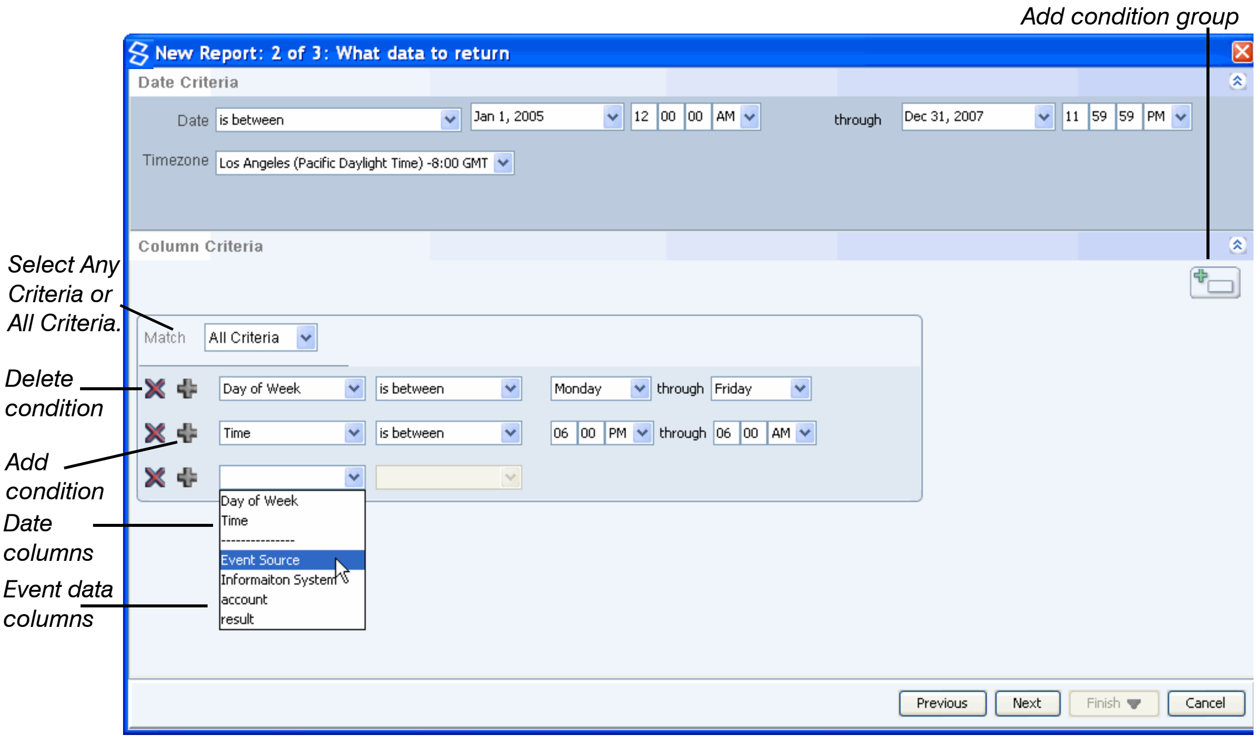
Column Criteria: Limiting the Number of Returned Rows

In addition to limiting returned rows by limiting the time range, you can limit rows by restricting data values in specific columns. This topic describes how to add individual column criteria and sets of column criteria to your report.

ADDING CONDITIONS AND SETS OF CONDITIONS

Figure 4-11 illustrates specification of criteria for three columns. Two of the columns are date columns derived from the stored timestamp. The third column, which is being created in the illustration, will represent stored event data.

Figure 4-11: Step 1 of Specifying Column Criteria



Add condition group

Select Any Criteria or All Criteria.

Delete condition

Add condition

Date columns

Event data columns

New Report: 2 of 3: What data to return

Date Criteria

Date: is between · Jan 1, 2005 · 12 : 00 : 00 AM · through · Dec 31, 2007 · 11 : 59 : 59 PM

Timezone: Los Angeles (Pacific Daylight Time) -8:00 GMT

Column Criteria

Match: All Criteria

Day of Week · is between · Monday · through · Friday

Time · is between · 06 : 00 PM · through · 06 : 00 AM

Figure 4-10 illustrates a single empty row of column criteria. Figure 4-11 illustrates three rows, two of which have already been configured with the `is between` operator. These rows represent derived date values. They have been configured to return data between 6:00 PM and 6:00 AM from Monday through Friday. If the **Match** field is set to `Any Criteria`, the report will return data if *either* of these conditions is true. If the **Match** field is set to `All Criteria`, the report will return data only if *all* conditions in this group are true.

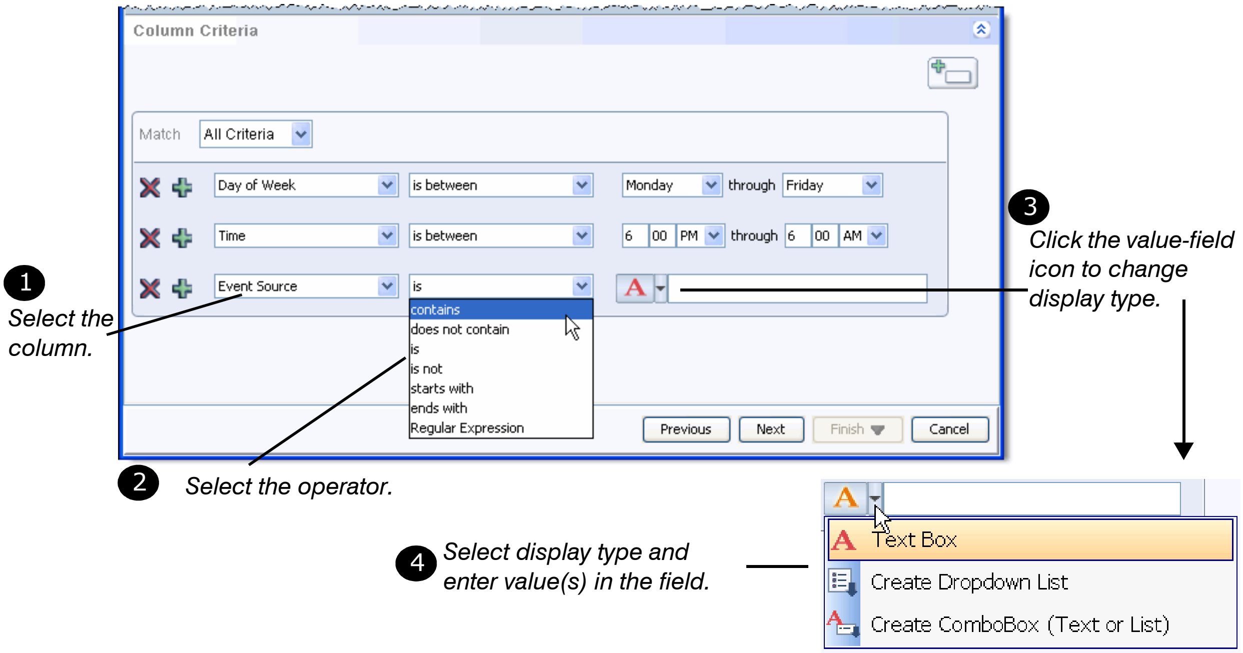
The third row illustrates selection of the **Event Source** data column. As described next, you can configure this field to return all values by default but also provide users with a choice of values from which they can select at runtime or to enter a value at runtime. These options enable users to investigate a specific event source.

You can specify conditions for any number of columns in the report. You can also specify conditions for the same column multiple times, each with a different operator. To add more conditions, click the ✚ icon. To delete a condition, click the ✖ icon. To add a separate set of conditions, click the ⊞ icon. To delete a group, remove all conditions within it.

PROVIDING A SET OF VALUES FOR A COLUMN

Figure 4-12 begins to illustrate how to configure a condition that provides users with a choice of values for a column. The figure below shows all available condition operators.

Figure 4-12: Specifying Conditions for a Column



Column Criteria

Match: All Criteria

Day of Week | is between | Monday through Friday

Time | is between | 6 00 PM through 6 00 AM

Event Source | is

1 Select the column.

2 Select the operator.

3 Click the value-field icon to change display type.

4 Select display type and enter value(s) in the field.

As illustrated above, after you select a column from the column dropdown, you can select an operator from the operator dropdown. By default, this field displays the `is` keyword. The table below describes each of the operators.

Operator	Explanation	Example
contains	matches values that include the specified text	`Windows` matches: • Windows Retriever Non-DC • Windows Retriever DC • Windows Snare Non-DC • Windows Snare DC • My Windows server
does not contain	matches values that do NOT include the specified text	`Non-DC` matches: • Windows Retriever DC • Windows Snare DC
is	exactly matches values that include ONLY the specified text	`Windows Retriever Non-DC` matches: • Windows Retriever Non-DC
is not	matches values that do NOT exactly match the specified text	`Windows Retriever Non-DC` matches: • Windows Retriever DC • Windows Snare Non-DC • Windows Snare DC • Unix • Linux • Candy Bars

Operator	Explanation	Example
starts with	matches values that begin with the specified text	`Windows` matches: • Windows Retriever Non-DC • Windows Retriever DC • Windows Snare Non-DC • Windows Snare DC
ends with	matches values that end with the specified text	`DC` matches: • Windows Retriever Non-DC • Windows Retriever DC • Windows Snare Non-DC • Windows Snare DC

Regular Expression	matches the specified regular expression
	NOTE
	• A regular expression is written in a formal language that interprets such text as specific characters, words, or patterns of characters.
	• If the search pattern includes a slash (/) character, you must precede it with a backslash (\) character.
	• HawkEye AP supports Perl regular expressions
		To locate the following text:
		`default/:Report`
		Enter the following text:
		`default\/:Report`

Selecting an event-data column from the column dropdown causes a third field to display. This third field enables you to specify text values relevant to the selected column. Click the A icon or dropdown to change the field type to one of the following:

- **Text Box—**Allows run-time users to enter a value
  - ■ You can leave the field empty.

    Users can enter text to restrict the data returned or keep the field empty to return all values.
  - ■ You can specify a default value.

    Users can keep your value, change it, or clear the field to return all values.
- **Dropdown List—**Provides a fixed list of values to run-time users
  - ■ You specify multiple values and select one to display by default.

    Users can keep your default value or select a different value from the dropdown.
- **ComboBox**—Provides an editable list of values to run-time users
  - ■ You specify multiple values and select one to display by default.

    Users can keep your default value, overwrite your default value to specify their own value, or select a value from the dropdown.
  - ■ You specify multiple values and leave the field empty.

    Users can enter valid text or select a value from the dropdown.

NOTE: You cannot delete a value that you enter accidentally in a Dropdown or ComboBox. You must delete the entire criteria row and recreate it. To delete a criteria row, click the ✕ icon.

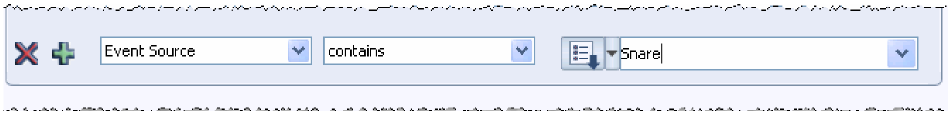
To provide values in the condition-value field

1 Click the Text Field icon or dropdown to display the value-field display options.

2 Select the desired display type, enter text in the field, and press ENTER.

If the field is a Text Box, you are done. If the field is a Dropdown or ComboBox, the field clears.

The graphic below illustrates the first value entered in a Dropdown List.



Event Source

contains

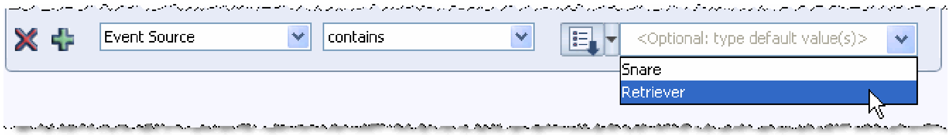
Snare

NOTE:

- In the example above, the report creator has changed the operator value from the default `is` keyword to `contains`. If the report creator had kept the default operator, she must enter exact values in the value-field. For example, to provide the run-time user with the value of `Windows Snare Non-DC`, she must enter the full name. However, by using the `contains` operator, the report creator can enter text that uniquely identifies the value without entering the full text.
- In addition to enabling the creator to enter less text, the `contains` operator provides more flexibility for the report. For example, because the value `Snare` matches both `Windows Snare Non-DC` and `Windows Snare DC`, this same report could be run against systems that are part of a Windows domain and those that are not.

3 If you are defining values for a Dropdown or ComboBox, enter a second text value into the field, and press ENTER.

The field clears again. To view the text you have already entered, click the dropdown icon, as shown below.



Event Source

contains

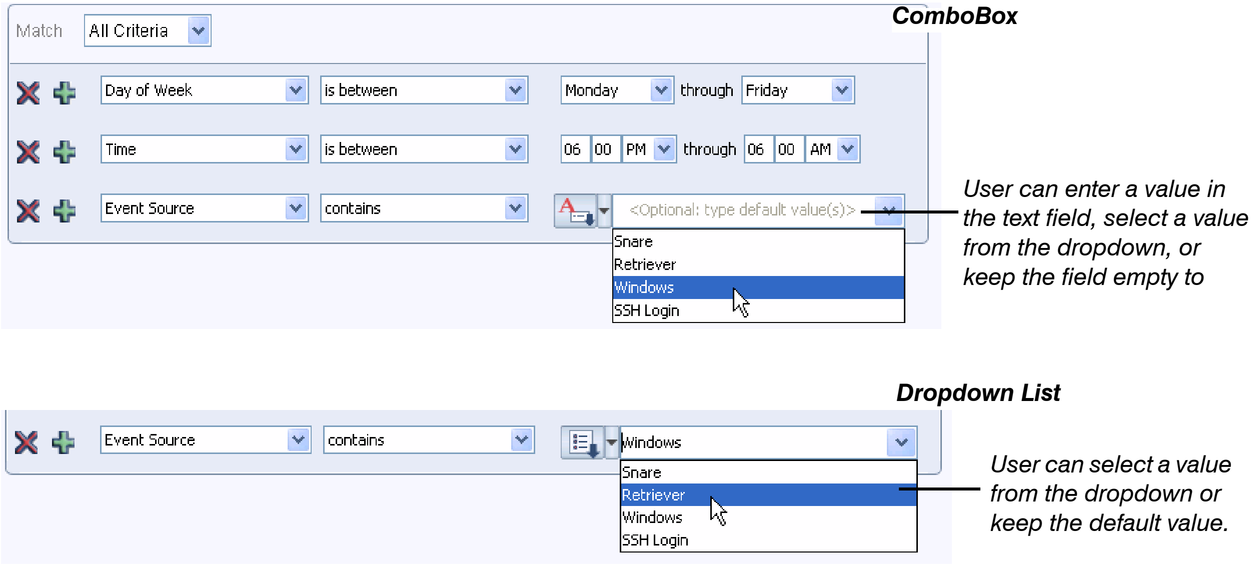
<Optional: type default value(s)>

4 Repeat for each value to display in the dropdown.

NOTE: To create a dropdown list, display the desired default value in the text field. To create a ComboBox, you can either leave the text field empty or display the desired default value. The value displayed in the field determines default report results.

Figure 4-13 illustrates two options that the user might see at runtime.

Figure 4-13: Runtime Options



ComboBox

Match All Criteria

Day of Week is between Monday through Friday

Time is between 06 00 PM through 06 00 AM

Event Source contains <Optional: type default value(s)>

User can enter a value in the text field, select a value from the dropdown, or keep the field empty to

Dropdown List

Event Source contains Windows

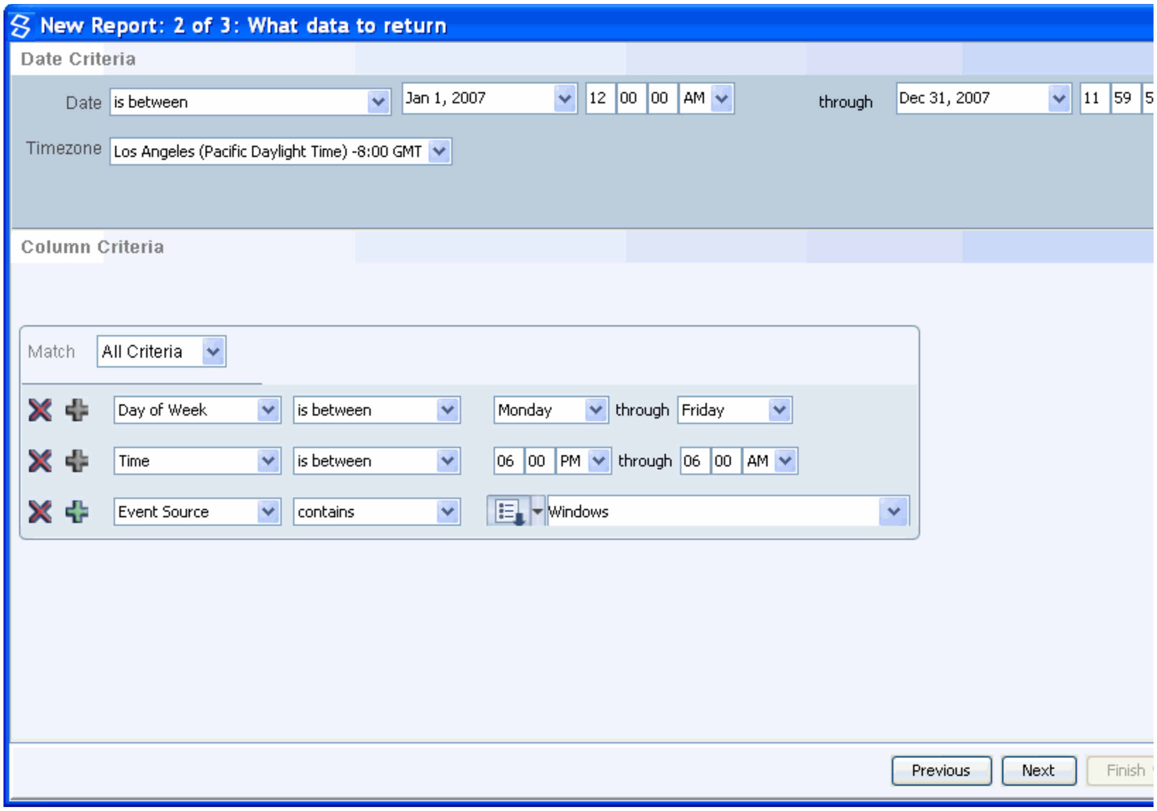
User can select a value from the dropdown or keep the default value.

Adding a Second Group of Column Criteria

Figure 4-14 illustrates an after-hours report that locates users who logged into the system after the close of business during the work week. Because the conditions have been set to match *All Criteria*, the report returns data only if all three conditions are met—the user logged in between

Monday and Friday AND did so between 6:00 p.m. and 6:00 a.m AND the event source matched the specified type (if one is specified).

Figure 4-14: One Condition Group that Matches All Criteria



Date: is between Jan 1, 2007 12 00 00 AM through Dec 31, 2007 11 59 5

Timezone: Los Angeles (Pacific Daylight Time) -8:00 GMT

Match: All Criteria

Day of Week | is between | Monday through Friday

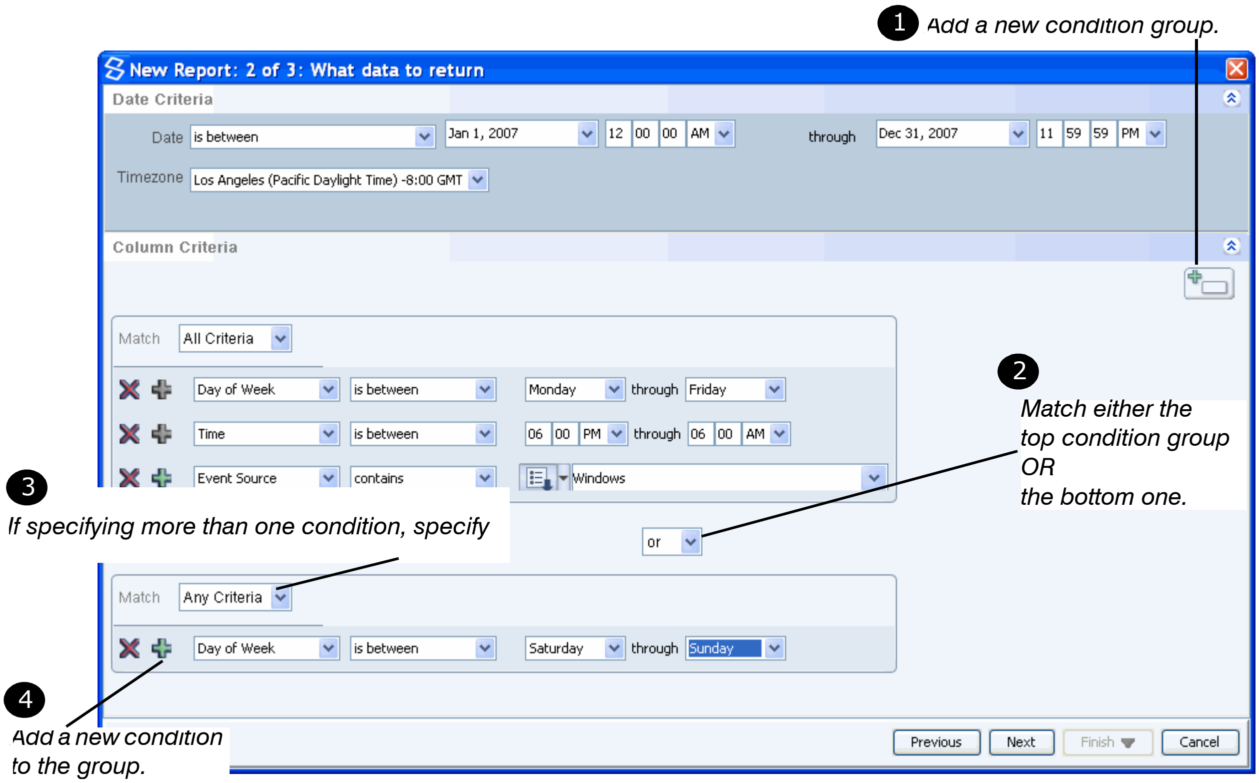
Time | is between | 06 00 PM through 06 00 AM

Event Source | contains | Windows

To also return users who log into the system any time on Saturday or Sunday, you must specify another condition. If you include this condition with the first group, the report will return results only if *all* conditions are true. To cause the report to examine weekday data separately from weekend data, you must add a second condition group. To do so, click the Add Group icon illustrated in Figure 4-15

The figure below illustrates how to add a second condition group. The second group locates users who log in only during the weekend.

Figure 4-15: Adding a Condition Group



New Report: 2 of 3: What data to return

Date Criteria

Date: is between — Jan 1, 2007 12 00 00 AM through Dec 31, 2007 11 59 59 PM

Timezone: Los Angeles (Pacific Daylight Time) -8:00 GMT

Column Criteria

Match: All Criteria

Day of Week | is between | Monday through Friday

Time | is between | 06 00 PM through 06 00 AM

Event Source | contains | Windows

or

Match: Any Criteria

Day of Week | is between | Saturday through Sunday

1 Add a new condition group.

2 Match either the top condition group OR the bottom one.

3 If specifying more than one condition, specify

4 Add a new condition to the group.

The window above specifies criteria that returns users who logged into the system after the close of business during the work week OR any time on either Saturday or Sunday. Because the match dropdown between the two sets of criteria specifies **or**, the report returns data even if only one of the two sets of conditions is true.

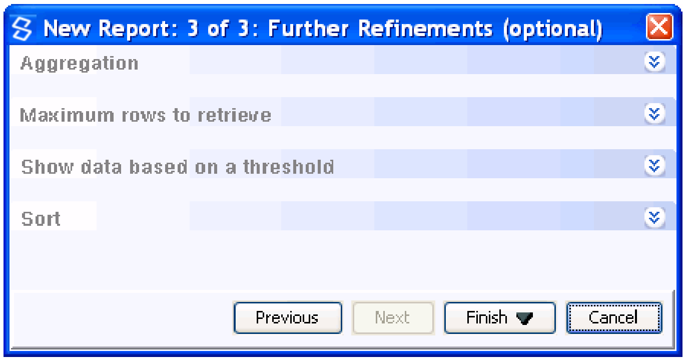
Step 3: Specifying Further Refinements

After you specify criteria that limits the data returned, you can further refine the data by:

- Aggregating values in specific column(s) to create a summary report
- Specifying the maximum number of rows to retrieve
- Specifying a threshold for data display
- Sorting the result set

Figure 4-16 illustrates the default Report Wizard window for refining data.

Figure 4-16: Options for Further Refining Data


New Report: 3 of 3: Further Refinements (optional)
Aggregation
Maximum rows to retrieve
Show data based on a threshold
Sort
Previous
Next
Finish
Cancel

The next topics describe how to specify the maximum number of rows to return and how to sort the returned data. Typically you will get familiar with your data by running your report and viewing the output before you summarize its data and specify a threshold. Therefore, although you can perform these operations from the Wizard, they are described in the section on editing the report.

For more information, see:

- “Specifying Maximum Number of Rows to Retrieve”, next
- “Specifying Column Sort”, on page 148
- “Running and Viewing the Wizard Report”, on page 149
- “Creating Summary Reports”, on page 154
- “Specifying a Threshold”, on page 159
- “Selecting, Renaming, and Ordering Columns for Display”, on page 132

Specifying Maximum Number of Rows to Retrieve

The number of rows that a report returns depends upon:

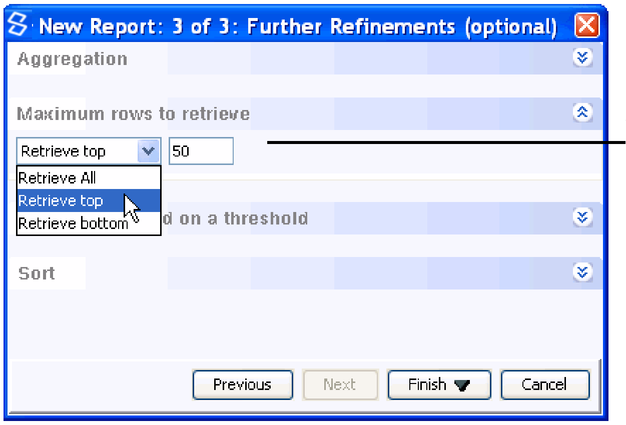
- amount of data in its data source
- selection criteria (date and column)
- aggregation criteria; for more information, see “Creating Summary Reports”, on page 154
- threshold; for more information, see “Specifying a Threshold”, on page 159

There are times you may want to view only a very small subset of data. For example, rather than view all rows in a large data set, you might want to preview the kind of data returned. In this case, you would want to return a small subset of the rows. Limiting the rows retrieved can be very helpful when you test your report’s behavior.

The second of the refinements available from step 3 of the Report Wizard enables you to specify the maximum number of rows retrieved by the report.

As illustrated in Figure 4-17, you can specify the number of rows from the top or bottom of the data set.

Figure 4-17: Specifying the Maximum Number of Rows to Retrieve



After you select top or bottom, specify how many rows to retrieve.

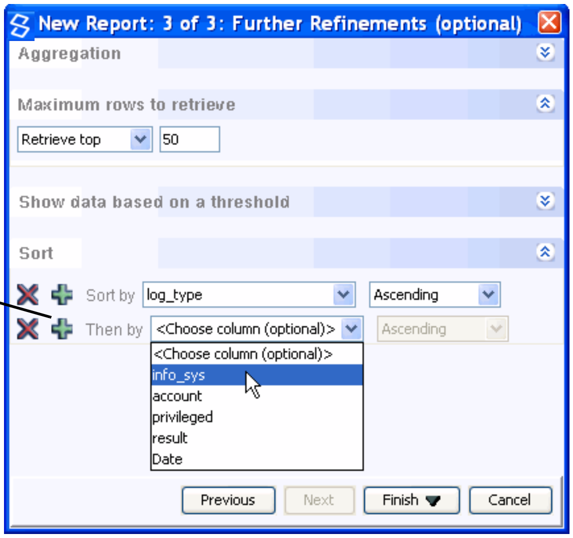
IMPORTANT: If your EDW instance comprises multiple hosts and you configure the query to return only the top or bottom subset of data, your query results may differ if you run the query multiple times. To ensure that the query returns the same subset of data each time, sort the data on at least one column. When the data is not sorted, the EDW might retrieve data in a different order from different hosts in your EDW instance. In this case, the top or bottom values returned are not consistent. For more information on a multi-host EDW instance, see EDW Architecture in Chapter 1, “Introduction” in the *Administration Guide*.

Specifying Column Sort

Sorting provides another way to make report data meaningful. As mentioned above, sorting also ensures consistent data when you run the report on a multi-host EDW instance. The last of the refinements available from step 3 of the Report Wizard enables you to specify column(s) for sorting.

As illustrated in Figure 4-18, you select the desired column(s) from the column dropdown. You also select desired sort order from the order dropdown. You can select up to three columns for sorting. Click the ✚ icon to add sort rows.

Figure 4-18: Specifying Sort Order



Add a new sort row.

New Report: 3 of 3: Further Refinements (optional)

Aggregation

Maximum rows to retrieve

Retrieve top: 50

Show data based on a threshold

Sort

Sort by: log_type — Ascending

Then by: <Choose column (optional)> — Ascending

NOTE: At run time, users can select their own column for sorting. However, they can sort on only one column at a time. For more information, see "Filtering and Sorting Report Data", on page 46.

Running and Viewing the Wizard Report

After you name your report, identify its data, specify its criteria, and refine it as desired, it is a good time to test your report by running it. To run the report, click **Finish** or **Finish and Run**:

- **Finish and Run**—HawkEye AP Console saves and closes the report definition, the **Run Report Status** dialog opens, and the report begins running.

  When the run completes, you can view your new report by clicking the 👁 icon from the **All Report Definitions** window or the **Run Report Status** dialog.

- **Finish**—HawkEye AP Console opens the report definition in Edit mode so that you can specify additional formatting, such as changing column display, associating reports, or specifying a chart display. After you complete this round of edits and save the report, you can run it from the **All Report Definitions** window.

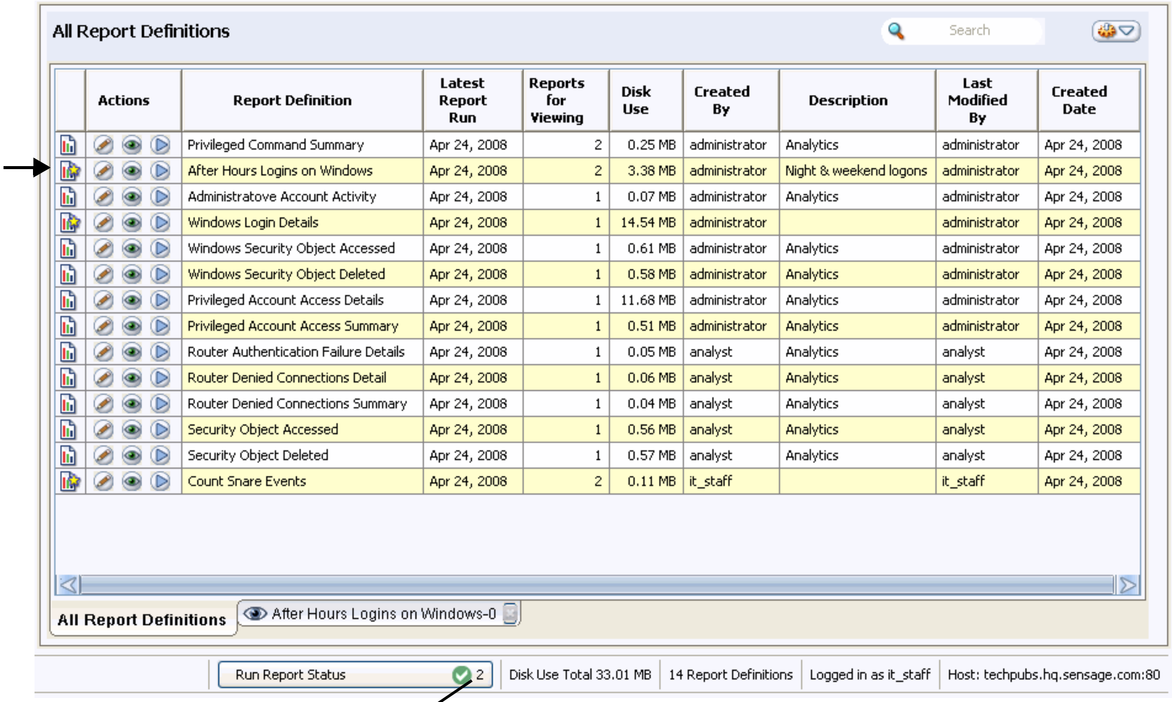
  NOTE: The **All Report Definitions** tab remains open as the leftmost tab while you are in Reports mode. If this tab remains hidden because multiple tabs display at the bottom of your window, use the page-scroll ◂ ▸ ▾ icon, as documented in "Changing Focus to a Page", on page 32.

  To run your new report, click the ▶ icon. The **Run** dialog displays, which enables you to change the time range, column criteria, and namespace. After you make your changes in this dialog, click **Run**. When the run completes, you can view your new report by clicking the 👁 icon from the **All Report Definitions** window or the **Run Report Status** dialog.

For more information, see "Running a Report", on page 100.

Figure 4-19 shows the *After Hours Logins on Windows* report displayed in the **All Report Definitions** list. The **Latest Report Run** column indicates that the report has been run three times.

Figure 4-19: Viewing Your Report in the All Report Definitions List



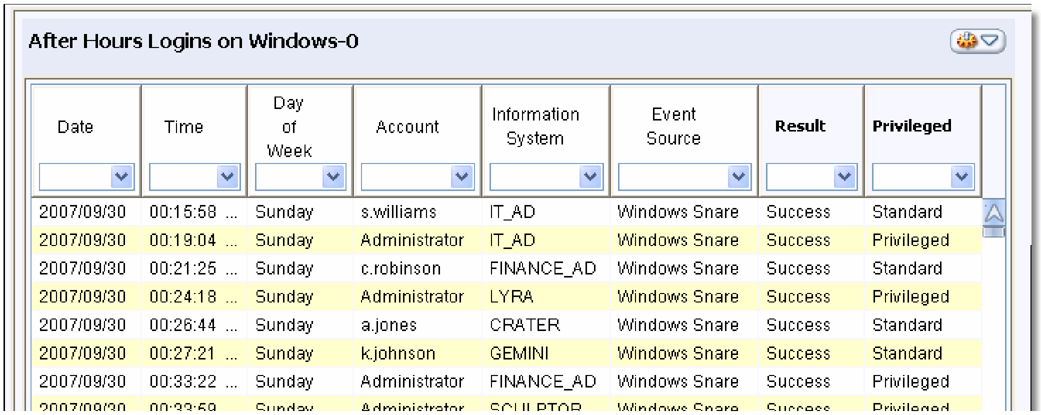
All Report Definitions

	Actions	Report Definition	Latest Report Run	Reports for Viewing	Disk Use	Created By	Description	Last Modified By	Created Date
		Privileged Command Summary	Apr 24, 2008	2	0.25 MB	administrator	Analytics	administrator	Apr 24, 2008
		After Hours Logins on Windows	Apr 24, 2008	2	3.38 MB	administrator	Night & weekend logons	administrator	Apr 24, 2008
		Administratove Account Activity	Apr 24, 2008	1	0.07 MB	administrator	Analytics	administrator	Apr 24, 2008
		Windows Login Details	Apr 24, 2008	1	14.54 MB	administrator		administrator	Apr 24, 2008
		Windows Security Object Accessed	Apr 24, 2008	1	0.61 MB	administrator	Analytics	administrator	Apr 24, 2008
		Windows Security Object Deleted	Apr 24, 2008	1	0.58 MB	administrator	Analytics	administrator	Apr 24, 2008
		Privileged Account Access Details	Apr 24, 2008	1	11.68 MB	administrator	Analytics	administrator	Apr 24, 2008
		Privileged Account Access Summary	Apr 24, 2008	1	0.51 MB	administrator	Analytics	administrator	Apr 24, 2008
		Router Authentication Failure Details	Apr 24, 2008	1	0.05 MB	analyst	Analytics	analyst	Apr 24, 2008
		Router Denied Connections Detail	Apr 24, 2008	1	0.06 MB	analyst	Analytics	analyst	Apr 24, 2008
		Router Denied Connections Summary	Apr 24, 2008	1	0.04 MB	analyst	Analytics	analyst	Apr 24, 2008
		Security Object Accessed	Apr 24, 2008	1	0.56 MB	analyst	Analytics	analyst	Apr 24, 2008
		Security Object Deleted	Apr 24, 2008	1	0.57 MB	analyst	Analytics	analyst	Apr 24, 2008
		Count Snare Events	Apr 24, 2008	2	0.11 MB	it_staff		it_staff	Apr 24, 2008

Click to open Run Report Status dialog

Figure 4-20 illustrates some of the results displayed by the *After Hours Windows Login* report.

Figure 4-20: Viewing the Report Results



After Hours Logins on Windows-0

Date	Time	Day of Week	Account	Information System	Event Source	Result	Privileged
2007/09/30	00:15:58 ...	Sunday	s.williams	IT_AD	Windows Snare	Success	Standard
2007/09/30	00:19:04 ...	Sunday	Administrator	IT_AD	Windows Snare	Success	Privileged
2007/09/30	00:21:25 ...	Sunday	c.robinson	FINANCE_AD	Windows Snare	Success	Standard
2007/09/30	00:24:18 ...	Sunday	Administrator	LYRA	Windows Snare	Success	Privileged
2007/09/30	00:26:44 ...	Sunday	a.jones	CRATER	Windows Snare	Success	Standard
2007/09/30	00:27:21 ...	Sunday	k.johnson	GEMINI	Windows Snare	Success	Standard
2007/09/30	00:33:22 ...	Sunday	Administrator	FINANCE_AD	Windows Snare	Success	Privileged
2007/09/30	00:33:59 ...	Sunday	Administrator	SCULPTOR	Windows Snare	Success	Privileged

The report above illustrates how the derived columns, **Date** and **Day of Week**, facilitate creation of a report that highlights after-hours activity.

If you run several different reports at the same time, you can minimize the **Run Report Status** dialog while they run. The status bar automatically increments the number of reports run as each report completes processing.

If your report does not display as desired or if you are ready to refine the report, open the edit window for the report and make the necessary changes. For more information, see "Editing a Wizard Report", next.

Editing a Wizard Report

After you run and test your report, you can refine it further by editing it. HawkEye AP Console provides two ways to edit a report:

- Batch editing

  Use this mode when you want to make the same change to the default time range, time zone, and/or namespace for multiple reports. For more information, see "Batch Editing Multiple Reports for Time Range, Time Zone, and Namespace", next.

- Single editing

  Use this mode to make changes to a report's content or display, or to associate a report to another report or to a security alert. For more information, see "Editing a Single Report", on page 153.

This section covers the following topics:

- "Batch Editing Multiple Reports for Time Range, Time Zone, and Namespace", next
- "Editing a Single Report", on page 153
- "Summarizing Data", on page 154
- "Associating Reports to a Report or Security Alert", on page 161
- "Formatting Columns", on page 167
- "Defining Charts", on page 171

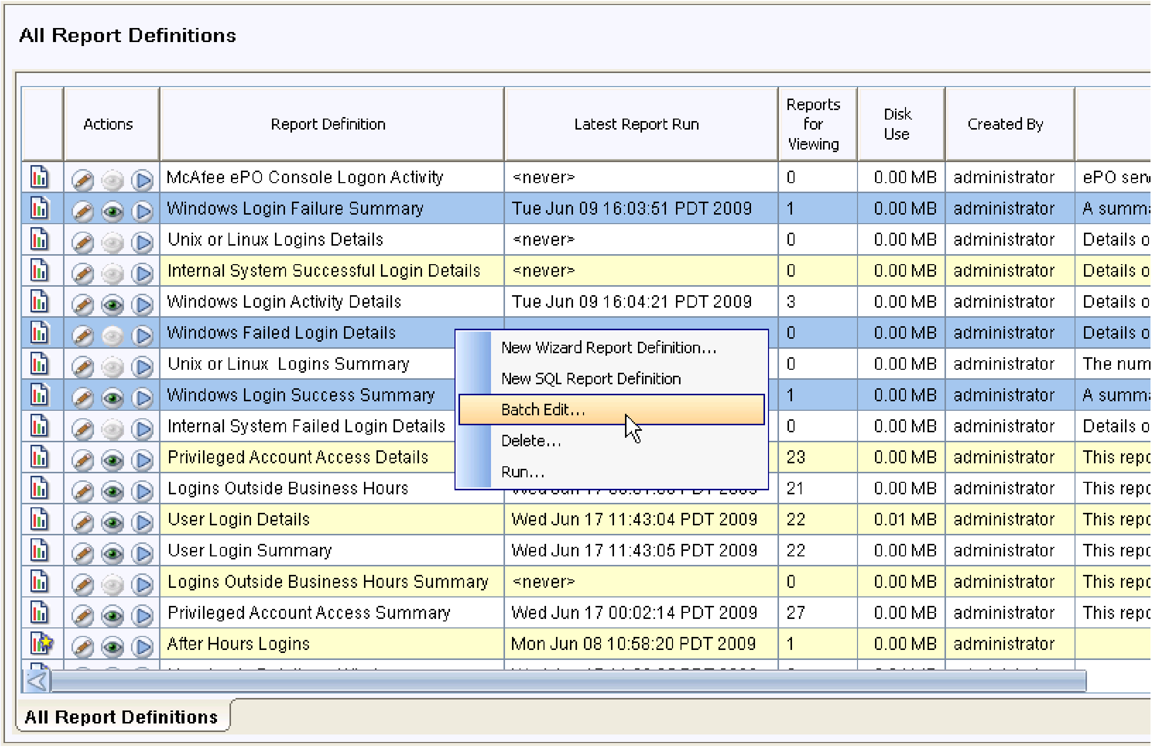
Batch Editing Multiple Reports for Time Range, Time Zone, and Namespace

Typically the date range of a report defaults to "is yesterday". If you want to change this setting for several reports and you want to save the new setting with the report definitions, you can open multiple selected reports for batch editing.

Batch editing also enables you to save a different time zone and/or namespace for a selected set of report definitions. For example, if your system stores the `hosts` table in both the `Eastern` and `Western` namespaces, and a set of report definitions specifies `Eastern` as the default namespace, you can use batch editing to change this default for a group of report definitions.

Figure 4-21 illustrates a user selecting the **Batch Edit ...** menu option on three selected reports.

Figure 4-21: Selecting Batch Edit on Multiple Reports

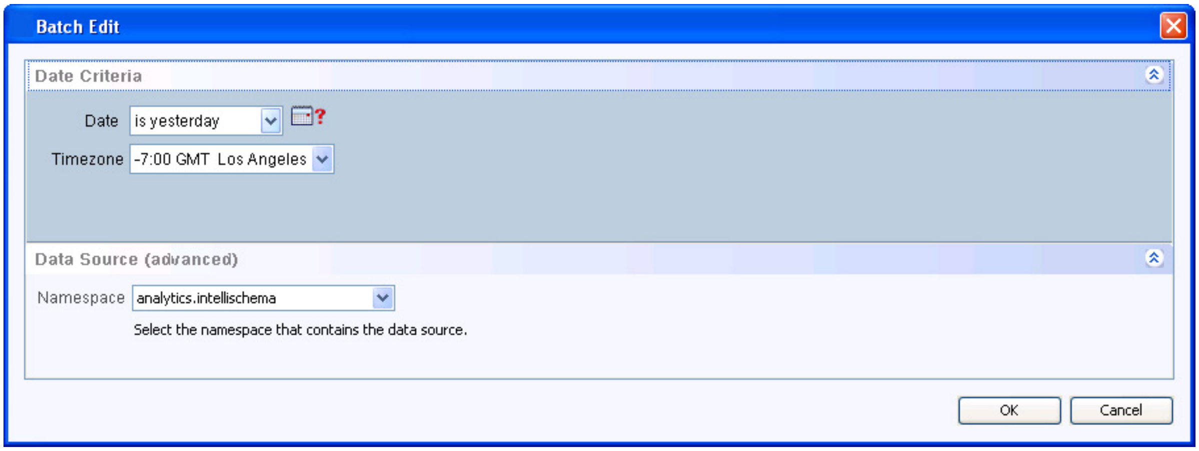


All Report Definitions

	Actions	Report Definition	Latest Report Run	Reports for Viewing	Disk Use	Created By	
		McAfee ePO Console Logon Activity	<never>	0	0.00 MB	administrator	ePO serv
		Windows Login Failure Summary	Tue Jun 09 16:03:51 PDT 2009	1	0.00 MB	administrator	A summ
		Unix or Linux Logins Details	<never>	0	0.00 MB	administrator	Details o
		Internal System Successful Login Details	<never>	0	0.00 MB	administrator	Details o
		Windows Login Activity Details	Tue Jun 09 16:04:21 PDT 2009	3	0.00 MB	administrator	Details o
		Windows Failed Login Details		0	0.00 MB	administrator	Details o
		Unix or Linux Logins Summary		0	0.00 MB	administrator	The num
		Windows Login Success Summary		1	0.00 MB	administrator	A summ
		Internal System Failed Login Details		0	0.00 MB	administrator	Details o
		Privileged Account Access Details		23	0.00 MB	administrator	This repo
		Logins Outside Business Hours		21	0.00 MB	administrator	This repo
		User Login Details	Wed Jun 17 11:43:04 PDT 2009	22	0.01 MB	administrator	This repo
		User Login Summary	Wed Jun 17 11:43:05 PDT 2009	22	0.00 MB	administrator	This repo
		Logins Outside Business Hours Summary	<never>	0	0.00 MB	administrator	This repo
		Privileged Account Access Summary	Wed Jun 17 00:02:14 PDT 2009	27	0.00 MB	administrator	This repo
		After Hours Logins	Mon Jun 08 10:58:20 PDT 2009	1	0.00 MB	administrator	

Figure 4-22 illustrates the **Batch Edit** dialog box that displays.

Figure 4-22: Batch Edit Dialog



Batch Edit

Date Criteria

Date: is yesterday

Timezone: -7:00 GMT Los Angeles

Data Source (advanced)

Namespace: analytics.intellischema

Select the namespace that contains the data source.

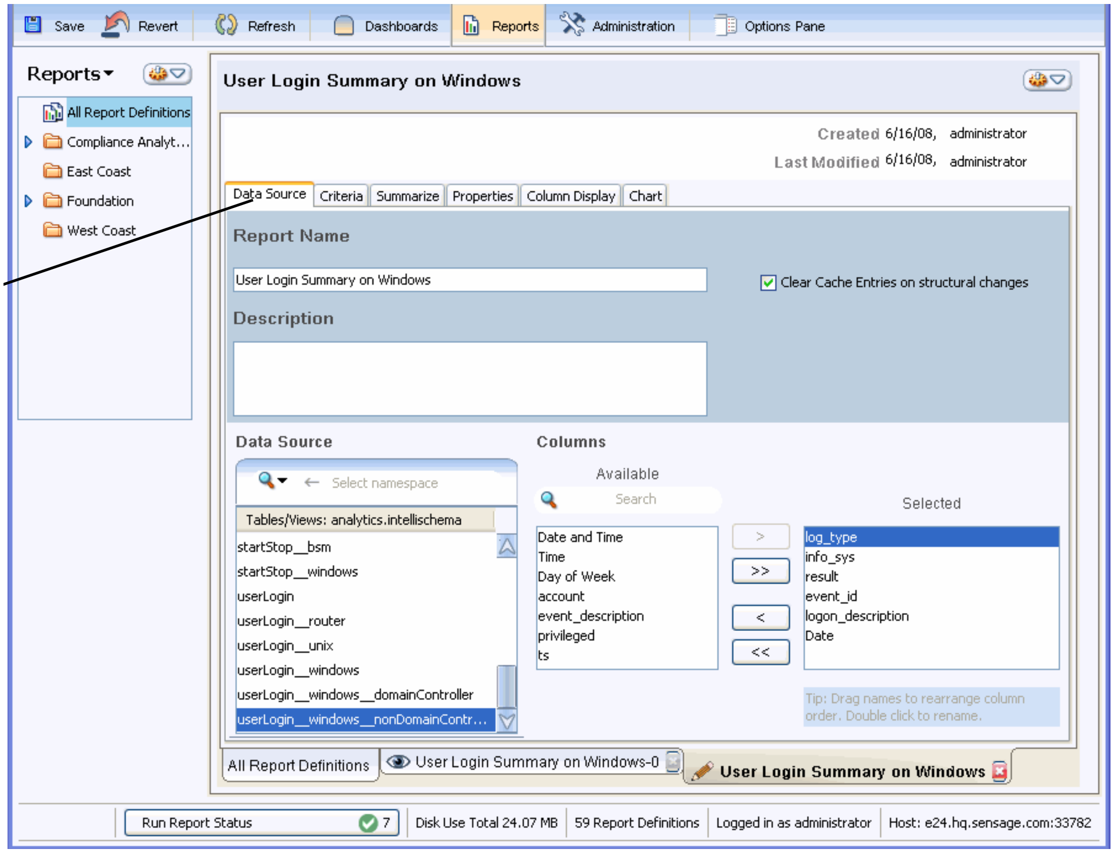
Note that the **Batch Edit** dialog box displays only three fields: **Date**, **Timezone**, and **Namespace**. Even reports that have been defined to display column-criteria rows display only these three fields when opened for batch editing. To edit any other aspect of a report, you must edit one report at a time. To do so, you select the report in the All Report Definitions list and either click the pencil icon or right click and select **Edit** from the popup menu. The remaining topics in this section describe the edits you can perform when you edit one report at a time.

Editing a Single Report

Figure 4-23 illustrates the Edit window for a wizard report named **User Login Summary on Windows**, which tracks all Windows logins and summarizes the results. This report displays six columns from the `userLogin_windows_nonDomainController` view.

Figure 4-23: Edit Window for New Wizard Report

Edit options are grouped in tabs



Created 6/16/08, administrator
Last Modified 6/16/08, administrator

Report Name: User Login Summary on Windows

☑ Clear Cache Entries on structural changes

Tables/Views: analytics.intellischema
startStop__bsm
startStop__windows
userLogin
userLogin__router
userLogin__unix
userLogin__windows
userLogin__windows__domainController
userLogin__windows__nonDomainContr…

Available: Date and Time, Time, Day of Week, account, event_description, privileged, ts

Selected: log_type, info_sys, result, event_id, logon_description, Date

Tip: Drag names to rearrange column order. Double click to rename.

Run Report Status ✓ 7 | Disk Use Total 24.07 MB | 59 Report Definitions | Logged in as administrator | Host: e24.hq.sensage.com:33782

The tabs at the top of the window enable you to view and modify the following attributes of a report:

- **Data source**—Allows you to view the report's data source and the specific columns selected for display.

  You can use this tab to change the report description, the namespace, the columns displayed, and the name and order of the columns displayed. You can also change the data source here.

  NOTE:

  - If you change selected columns, your report definition may no longer run. At a minimum, you will lose all your column-display settings and your chart definition. However, column changes can also affect your column-criteria and summarize settings.
  - Unless you are selecting the same table or view from a different namespace, Sensage recommends that you create a new report if you want to change the data source. For information on using this tab, see: "Step 1: Specifying Where to Get the Data", on page 129 and "Selecting, Renaming, and Ordering Columns for Display", on page 132.

- **Criteria**—Allows you to view or change the time range and the column criteria to limit the data returned. For information on using this tab, see "Step 2: Specifying What Data to Return", on page 133 and "Column Criteria: Limiting the Number of Returned Rows", on page 139.
- **Summarize**—Allows you to aggregate, group, and sort the report data. For more information, see "Summarizing Data", next.
- **Properties**—Allows you to associate reports to your report, and to specify layout and display options. For more information, see "Associating a Report to Another Report", on page 161, "Associating a Report to a Security Alert", on page 164, and "Specifying Print Options and Maximum Rows", on page 166.
- **Column Display**—Allows you to format column width and column header text, and to change column data type. For more information, see "Formatting Columns", on page 167.
- **Chart**—Allows you to change output format from table to chart and to select the desired chart style. For more information, see "Defining Charts", on page 171

NOTE: Click **Revert** to undo all changes made since the last time you saved the report. When you revert changes, all changes made in any or all tabs revert to their status as of the last save.

Summarizing Data

This section includes the following topics:

- "Creating Summary Reports", next
- "Specifying a Threshold", on page 159

Creating Summary Reports

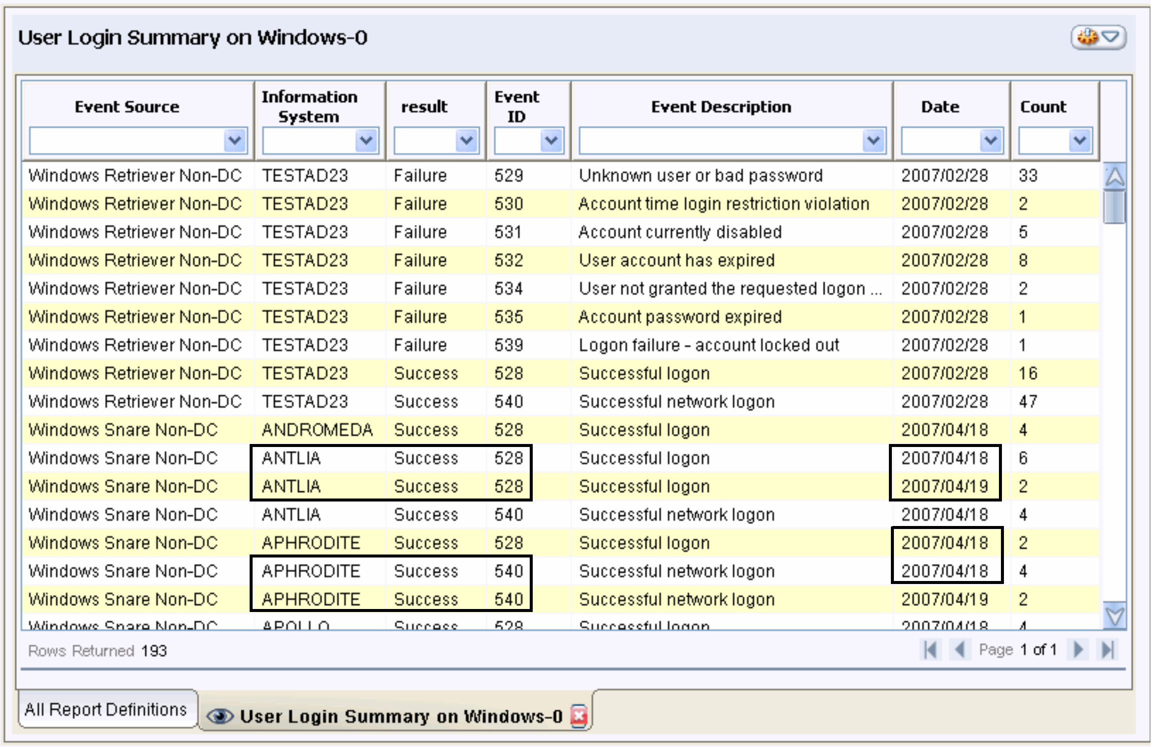
You can add meaning to a report by aggregating data in one or more of its columns. Aggregating data groups multiple rows as a single row and can dramatically reduce the number of rows returned.

BACKGROUND: UNDERSTANDING AGGREGATION

Figure 4-24 illustrates a report that aggregates the data in the result set. The aggregation summarizes the number of user logins on a Windows system that is not part of a domain. The report groups the columns in the result set as part of the aggregation process. When the EDW aggregates the data, it examines all rows in the result set and returns one row for each unique combination.

The report below returns only 193 rows. When run against the same example data set without the calculation, the same report returns 1583 rows.

Figure 4-24: Summarizing and Grouping Data



User Login Summary on Windows-0

Event Source	Information System	result	Event ID	Event Description	Date	Count
Windows Retriever Non-DC	TESTAD23	Failure	529	Unknown user or bad password	2007/02/28	33
Windows Retriever Non-DC	TESTAD23	Failure	530	Account time login restriction violation	2007/02/28	2
Windows Retriever Non-DC	TESTAD23	Failure	531	Account currently disabled	2007/02/28	5
Windows Retriever Non-DC	TESTAD23	Failure	532	User account has expired	2007/02/28	8
Windows Retriever Non-DC	TESTAD23	Failure	534	User not granted the requested logon ...	2007/02/28	2
Windows Retriever Non-DC	TESTAD23	Failure	535	Account password expired	2007/02/28	1
Windows Retriever Non-DC	TESTAD23	Failure	539	Logon failure - account locked out	2007/02/28	1
Windows Retriever Non-DC	TESTAD23	Success	528	Successful logon	2007/02/28	16
Windows Retriever Non-DC	TESTAD23	Success	540	Successful network logon	2007/02/28	47
Windows Snare Non-DC	ANDROMEDA	Success	528	Successful logon	2007/04/18	4
Windows Snare Non-DC	ANTLIA	Success	528	Successful logon	2007/04/18	6
Windows Snare Non-DC	ANTLIA	Success	528	Successful logon	2007/04/19	2
Windows Snare Non-DC	ANTLIA	Success	540	Successful network logon	2007/04/18	4
Windows Snare Non-DC	APHRODITE	Success	528	Successful logon	2007/04/18	2
Windows Snare Non-DC	APHRODITE	Success	540	Successful network logon	2007/04/18	4
Windows Snare Non-DC	APHRODITE	Success	540	Successful network logon	2007/04/19	2
Windows Snare Non-DC	APOLLO	Success	528	Successful logon	2007/04/18	4

Rows Returned 193

Page 1 of 1

NOTE: The aggregation process groups unique values in the order that the columns display, from the left to the right. In the example above, the aggregation process first groups unique values in the **Event Source** column. The grouped data determines the first data break. Because this data represents two different event sources (Windows Snare Non-DC and Windows Retriever Non-DC), the data first breaks on this column.

The aggregation process next groups the data in **Information System**, the second displayed column. Because Windows Retriever Non-DC received all its events from the same server (`TESTAD23`), the value of **Information System** does not affect the data break. A data break occurs only when a value changes. Therefore, because Windows Snare Non-DC received events from many information systems, the **Information System** values for this event source break into multiple rows.

The aggregation process looks at the next column, which is the **Result** column. Because only one value displays for each event source (`Failure` or `Success`), the data does not break further for this column.

The grouping of the data becomes clear as we examine the **Event ID** column. The value of this column determines the final data breaks for most of the rows. Note that Windows Retriever breaks into one row for each event ID. However, Windows Snare displays multiple rows for the 528 event ID because each row is unique to the information system as well as to the event ID.

Note also that the **Event ID** column does not determine the final data break for all Windows Snare Non-DC events. Two rows represent successful logon (528) on the ANTLIA system and another two rows represent successful network logon (540) on the APHRODITE system. Because the

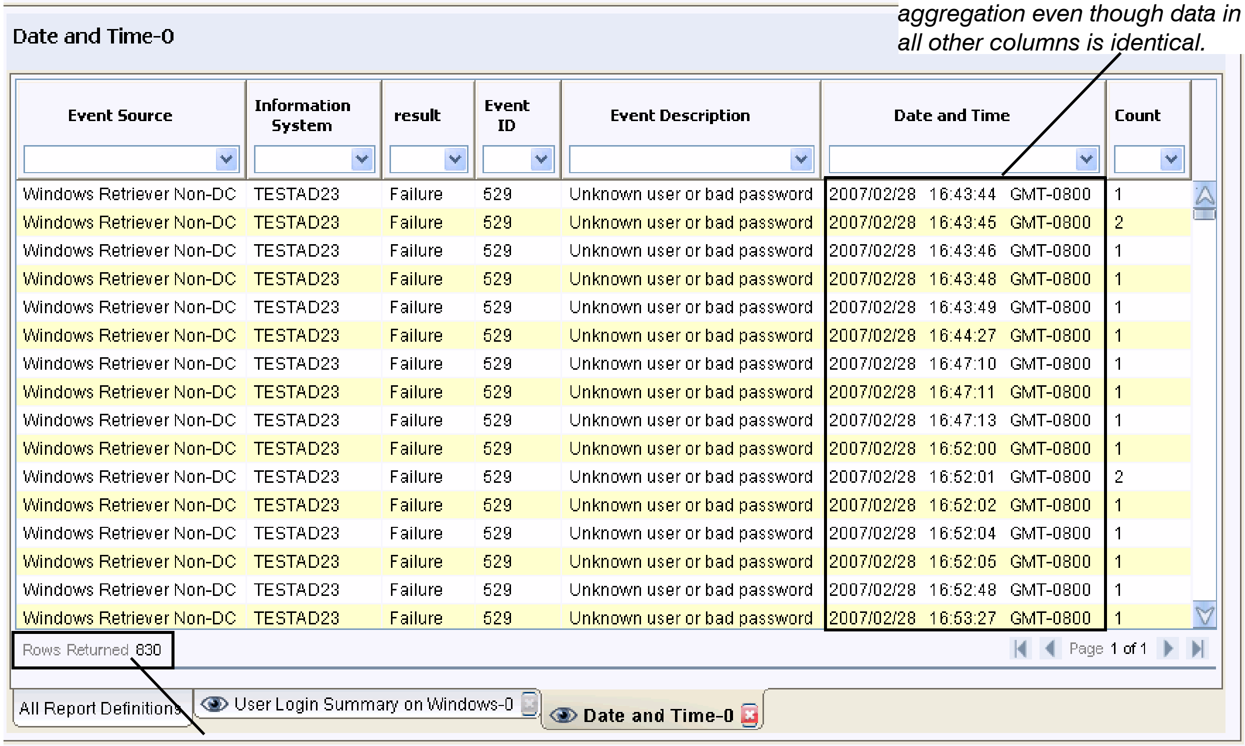
Event Description column represents the same data as the **Event ID** column, data does not break further based on event description. However, the Windows Snare rows that did *not* break on the value of their **Event ID** column *do* break on the value of their **Date** column. Figure 4-24 illustrates the significant values in these four rows by enclosing them in red boxes.

NOTE: The date column you select in Step 1 of the Wizard can dramatically affect your report result set. As illustrated in Figure 4-3 on page 132, the **Date and Time** column represents data down to the second whereas the **Date** column represents only whole days. Therefore, a report that contains the **Date and Time** column represents a granularity of data very different from one that contains the **Date** column. If you summarize data in these two reports, the granularity of the data significantly changes the results.

For example, if this report had used the **Date and Time** column instead of the **Date** column and had been run against the same example data, the report would have returned many more rows. Instead of 33 rows of unknown user or bad password, there would have been multiple rows, most with a count of `1`. The different results depend on the granularity of the stored data. The **Date** column differentiates February from March. The **Date and Time** column differentiates this minute and second from the last minute and second and the next minute and second. When seconds are part of the grouped data, most rows are unique; thus the count of `1`.

Figure 4-25 illustrates a report that uses the **Date and Time** column instead of the **Date** column. The report below displays a small subset of the rows returned.

Figure 4-25: Granularity of Data Determines Result Set



Data in this column affects the aggregation even though data in all other columns is identical.

Date and Time-0

Event Source	Information System	result	Event ID	Event Description	Date and Time	Count
Windows Retriever Non-DC	TESTAD23	Failure	529	Unknown user or bad password	2007/02/28 16:43:44 GMT-0800	1
Windows Retriever Non-DC	TESTAD23	Failure	529	Unknown user or bad password	2007/02/28 16:43:45 GMT-0800	2
Windows Retriever Non-DC	TESTAD23	Failure	529	Unknown user or bad password	2007/02/28 16:43:46 GMT-0800	1
Windows Retriever Non-DC	TESTAD23	Failure	529	Unknown user or bad password	2007/02/28 16:43:48 GMT-0800	1
Windows Retriever Non-DC	TESTAD23	Failure	529	Unknown user or bad password	2007/02/28 16:43:49 GMT-0800	1
Windows Retriever Non-DC	TESTAD23	Failure	529	Unknown user or bad password	2007/02/28 16:44:27 GMT-0800	1
Windows Retriever Non-DC	TESTAD23	Failure	529	Unknown user or bad password	2007/02/28 16:47:10 GMT-0800	1
Windows Retriever Non-DC	TESTAD23	Failure	529	Unknown user or bad password	2007/02/28 16:47:11 GMT-0800	1
Windows Retriever Non-DC	TESTAD23	Failure	529	Unknown user or bad password	2007/02/28 16:47:13 GMT-0800	1
Windows Retriever Non-DC	TESTAD23	Failure	529	Unknown user or bad password	2007/02/28 16:52:00 GMT-0800	1
Windows Retriever Non-DC	TESTAD23	Failure	529	Unknown user or bad password	2007/02/28 16:52:01 GMT-0800	2
Windows Retriever Non-DC	TESTAD23	Failure	529	Unknown user or bad password	2007/02/28 16:52:02 GMT-0800	1
Windows Retriever Non-DC	TESTAD23	Failure	529	Unknown user or bad password	2007/02/28 16:52:04 GMT-0800	1
Windows Retriever Non-DC	TESTAD23	Failure	529	Unknown user or bad password	2007/02/28 16:52:05 GMT-0800	1
Windows Retriever Non-DC	TESTAD23	Failure	529	Unknown user or bad password	2007/02/28 16:52:48 GMT-0800	1
Windows Retriever Non-DC	TESTAD23	Failure	529	Unknown user or bad password	2007/02/28 16:53:27 GMT-0800	1

Rows Returned 830

Many more rows are returned.

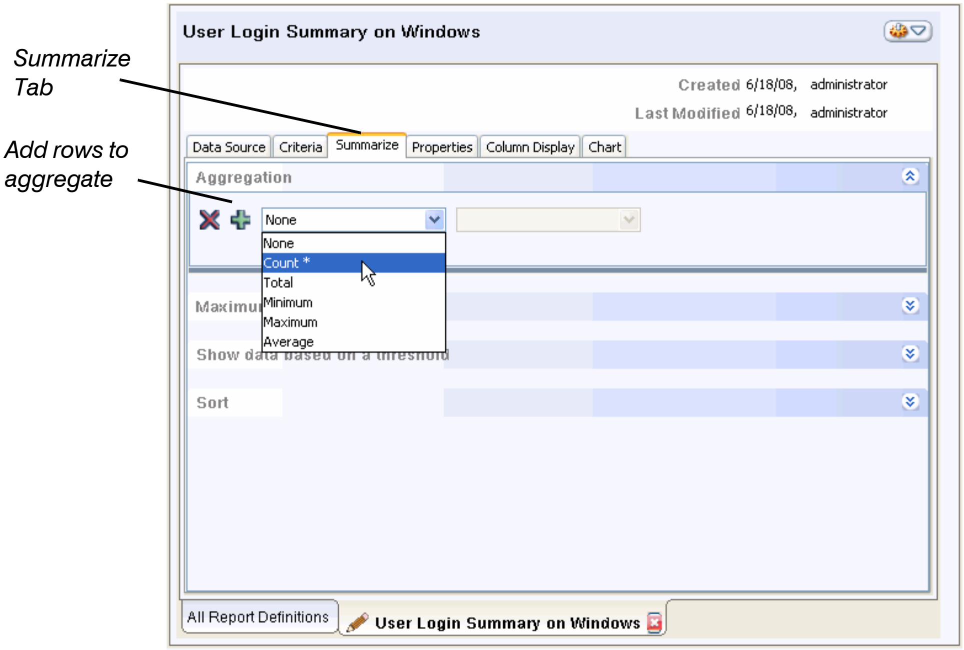
As illustrated in Figure 4-24, the example data set has 33 rows when Windows Retriever Non-DC is the source and the **Event Description** column represents `Unknown user or bad password`. In the report illustrated in Figure 4-24, these 33 rows are grouped into a single row because all of these events occurred on February 28, 2007. When the report uses the **Date and Time** column instead of the **Date** column, the aggregated report contains a separate row for each of these events; each row displays a count of `1`.

IMPORTANT: When you aggregate the data in one or more columns, all data *not* in the aggregated column or columns is grouped in relation to the aggregated data. The granularity of data in all non-aggregated columns determines the groups that result from the aggregation. For an example of how the derived date columns display data, see "Step 1: Specifying Where to Get the Data", on page 129.

SUMMARIZE TAB: AGGREGATING DATA

The **Summarize** tab facilitates the creation of a report that performs a count like the one whose results display in Figure 4-24. Figure 4-26 illustrates the window that enables you to configure the report to count the values returned by the report.

Figure 4-26: Specifying a Count and Grouping Remaining Columns

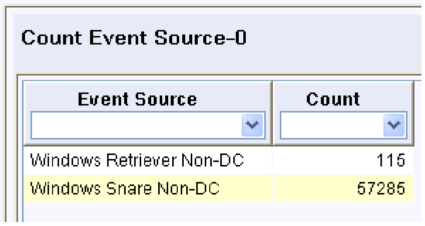


As illustrated above, the `Count *` operation has been specified for the User Login Summary on Windows report. As described in "Background: Understanding Aggregation", on page 154, the report groups the columns in the result set as part of the aggregation process. When you select the `Count *` operation, the report returns a count for each group of rows.

For example, assume the report returns only the count and the data from the **Event Source** column. In this case, it groups all values in the **Event Source** column before it performs the count. In other words, it would group all rows that contained `Windows Retriever` as the event source and all rows that contained `Windows Snare` and then it would count the instances of

`Windows Retriever` and `Windows Snare`. The result set would display one row for each event source, each with a unique count. Figure 4-27 illustrates such a report.

Figure 4-27: Example of a Summary Report that Displays One Event-Data Column

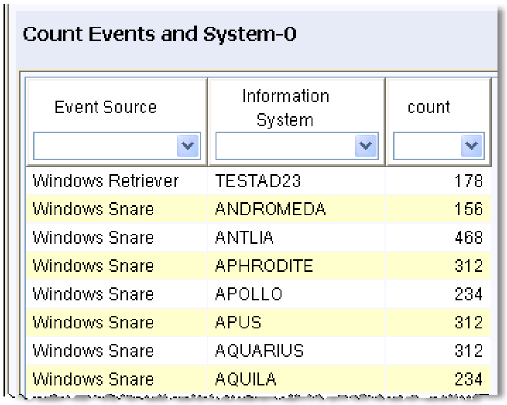


Count Event Source-0

Event Source	Count
Windows Retriever Non-DC	115
Windows Snare Non-DC	57285

If you modify the report to also return data from the **Information System** column, the report groups values by both the **Event Source** and the **Information System** columns before it performs the count. In other words, it groups the `Windows Snare` rows by each information system and then it counts the groups.The result set displays one row for each event source/ information system group, each with a unique count. Figure 4-28 illustrates such a report.

Figure 4-28: Example of a Summary Report that Displays Two Event-Data Columns



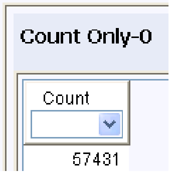
Count Events and System-0

Event Source	Information System	count
Windows Retriever	TESTAD23	178
Windows Snare	ANDROMEDA	156
Windows Snare	ANTLIA	468
Windows Snare	APHRODITE	312
Windows Snare	APOLLO	234
Windows Snare	APUS	312
Windows Snare	AQUARIUS	312
Windows Snare	AQUILA	234

Because the example data shows only one system sending data through Windows Retriever, this event source is grouped into a single row with a count of 78. However, because many systems send data through Windows Snare, this event source is grouped into multiple rows, each with their own count.

If you create a report that returns only the count, it returns a single column with a single row of data. Figure 4-29 illustrates such a report.

Figure 4-29: Example of a Summary Report that Displays Only a Count



Count Only-0

Count
57431

Because a summary report that contains multiple columns provides meaningful data only when the non-aggregated columns are grouped, the Report Wizard automatically groups every column that is not aggregated. The Report Wizard also sorts the columns in ascending order. You can specify a descending sort order for specific columns. As illustrated in Figure 4-18, you can select `Descending` from the dropdown next to each column row.

NOTE: Typically, you can keep the default grouping order. However, if your report includes the **ts** column as the first or one of the first columns, HawkEye AP recommends that you move this column lower in the list. Grouping by the **ts** column can slow performance significantly. To change the column display order, use the **Data Source** tab or the **Column Display** tab. For more information, see “Selecting, Renaming, and Ordering Columns for Display”, on page 132 and “Changing Column Width and Relative Position”, on page 167

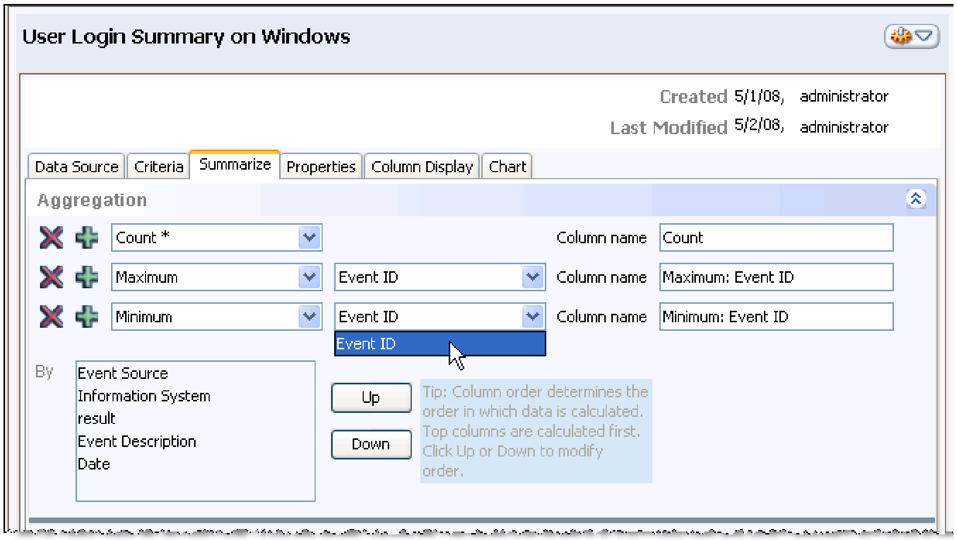
For an example of a summary report with many columns, all of which are grouped in their display order, see “Background: Understanding Aggregation”, on page 154.

The Report Wizard provides several aggregation operations besides `count *`. The other operations, which can be applied only to columns that contain numeric values, are available from the Aggregation dropdown. These are:

- Total
- Minimum
- Maximum
- Average

Although Figure 4-26 illustrates only one aggregation operation, you can click the ✚ icon to add rows and specify aggregation on multiple columns or different operations on the same column. For example, Figure 4-30 illustrates the `Minimum` and `Maximum` aggregation applied to the **Event ID** column. The Report Wizard names each aggregation column automatically. You can change the name to your liking.

Figure 4-30: Aggregating a Numeric Column



User Login Summary on Windows

Created 5/1/08, administrator

Last Modified 5/2/08, administrator

Aggregation

Count * — Column name: Count

Maximum — Event ID — Column name: Maximum: Event ID

Minimum — Event ID — Column name: Minimum: Event ID

By: Event Source, Information System, result, Event Description, Date

Tip: Column order determines the order in which data is calculated. Top columns are calculated first. Click Up or Down to modify order.

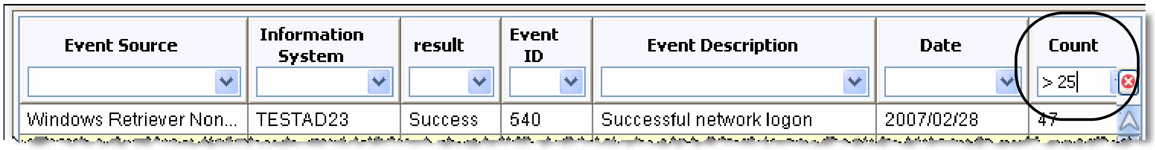
Specifying a Threshold

As discussed in “Specifying Maximum Number of Rows to Retrieve”, on page 147, there are many ways to limit the number of rows a report returns. This topic presents yet another way you can meaningfully limit the rows returned. You can set a threshold to eliminate groups that have been aggregated for your report.

For example, Figure 4-24 illustrates a report that aggregates data by counting values in each grouped row. The example report returns 9 rows that group results for Windows Retriever Non-DC. Some of these groups represent a single event, as shown by the value of 1 in the **Count** column. Some groups represent multiple events.

To display only the groups that represent more than 25 events, you could filter the **Count** column in the result set. Figure 4-31 illustrates report output that uses the filter option to eliminate groups. The figure below circles the filtered column.

Figure 4-31: Eliminating Groups by Filtering



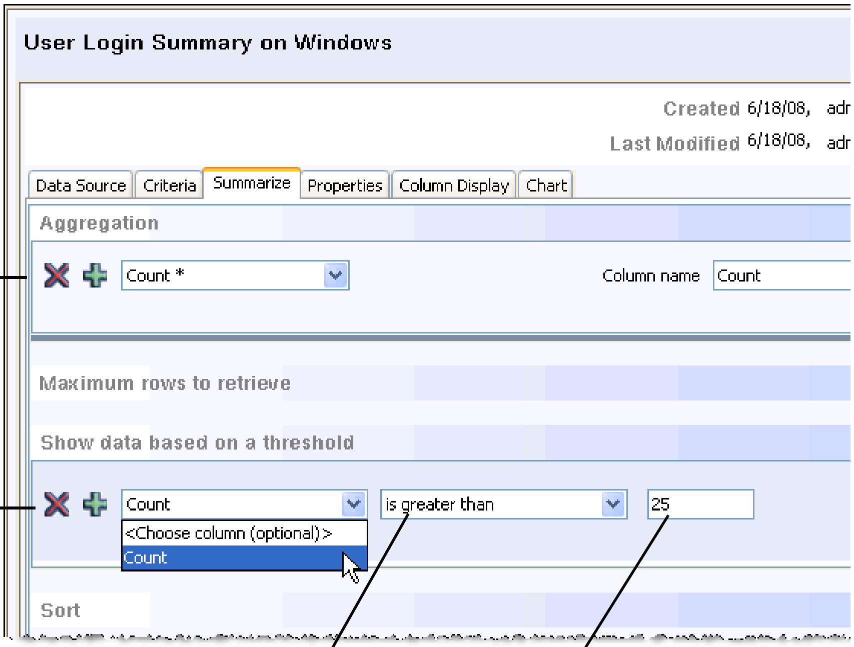
Event Source	Information System	result	Event ID	Event Description	Date	Count
						> 25
Windows Retriever Non...	TESTAD23	Success	540	Successful network logon	2007/02/28	47

For more information, see "Filtering and Sorting Report Data", on page 46.

Specifying a threshold has a similar affect as filtering. However, the threshold is saved with the report definition so the filtering occurs automatically every time the report runs. The threshold causes the report to return only the filtered rows.

Figure 4-32 illustrates how you would specify a threshold of 25 for the **Count** column, which is created when the report results are aggregated.

Figure 4-32: Specifying a Threshold



User Login Summary on Windows

Created 6/18/08, adr

Last Modified 6/18/08, adr

Aggregation

Count * — Column name: Count

Maximum rows to retrieve

Show data based on a threshold

Count — is greater than — 25

1 *Specify an aggregation.*

2 *Select numeric column(s) whose values you want to limit.*

3 *Select a comparison value.*

4 *Specify the threshold value.*

Associating Reports to a Report or Security Alert

After you are satisfied with a report that you have created, you can associate the report to another report or to a security alert. Although the association process is similar, you use the **Properties** tab in report-edit mode to associate a report to another report and you use Administration mode to associate a report to a security alert.

NOTE: As documented in“Browsing to Other Reports”, on page 57, dashboard users who view a report widget or security alert widget can open any report on the system that they have permission to access. Associated reports, however, provide a quick list of suggested reports.

This topic covers the following:

- “Associating a Report to Another Report”, next
- “Associating a Report to a Security Alert”, on page 164

Associating a Report to Another Report

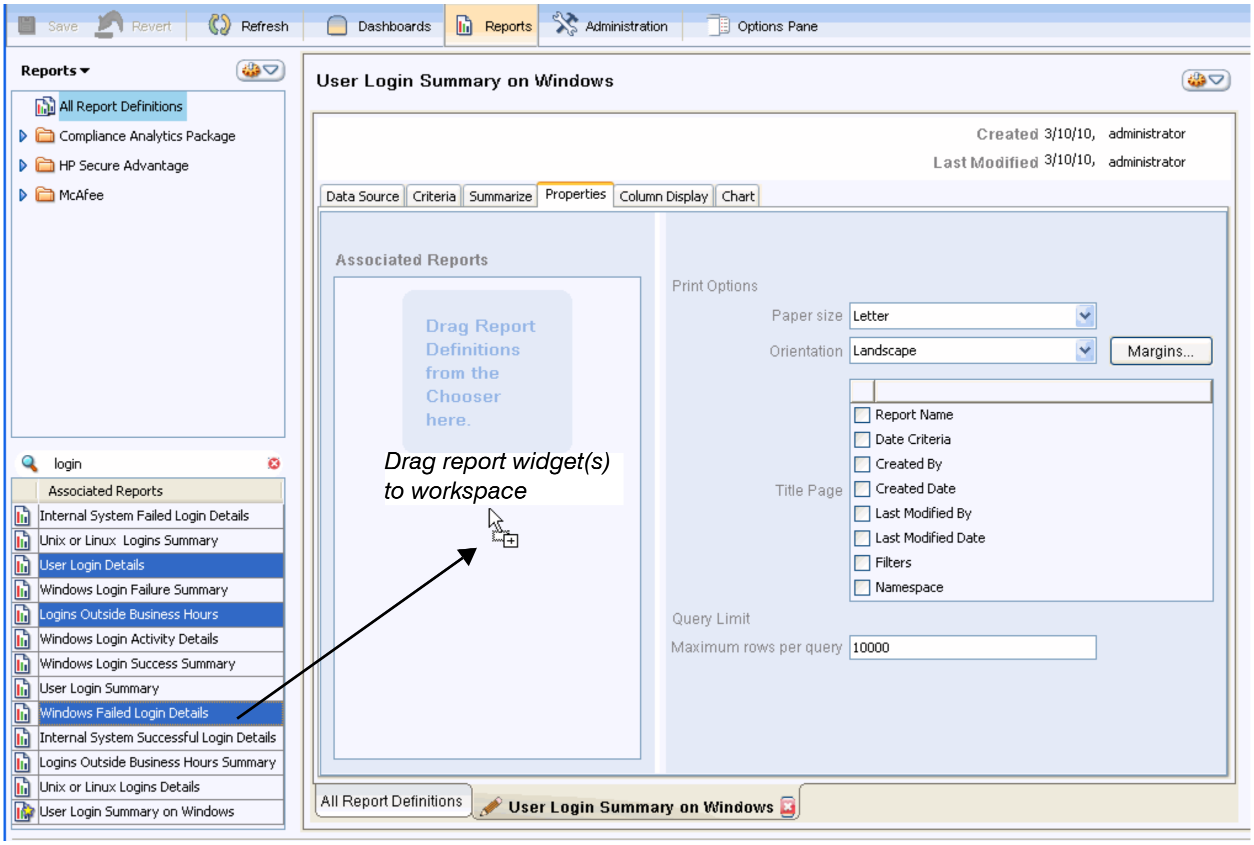
The **Properties** tab enables several operations, which include **Associate reports to your report**. Typically, detailed reports are linked to summary reports to allow users to easily drill down to specific data in the summary report. The associated reports enable users to conduct investigations that examine the same set of data in more than one report.

The user selects the data to examine from one of the columns in the summary or source report. The selected data populates the column criteria field(s) of the associated report, which returns only rows that pertain to the selected value. Because the associated report limits results to the user-selected value, each report you associate must have at least one column criteria field. The user can investigate data only in columns that have a corresponding column criteria field in the associated report.

If the user selects more than one value, all selected values display in a dropdown, regardless of the style defined for the value field. The user selects one value from the dropdown and runs the report, which returns only rows that match the selected value. The user must run the report separately for each of the other selected values.

Figure 4-33 illustrates how to use the **Properties** tab to associate reports.

Figure 4-33: Associating Reports to a Report



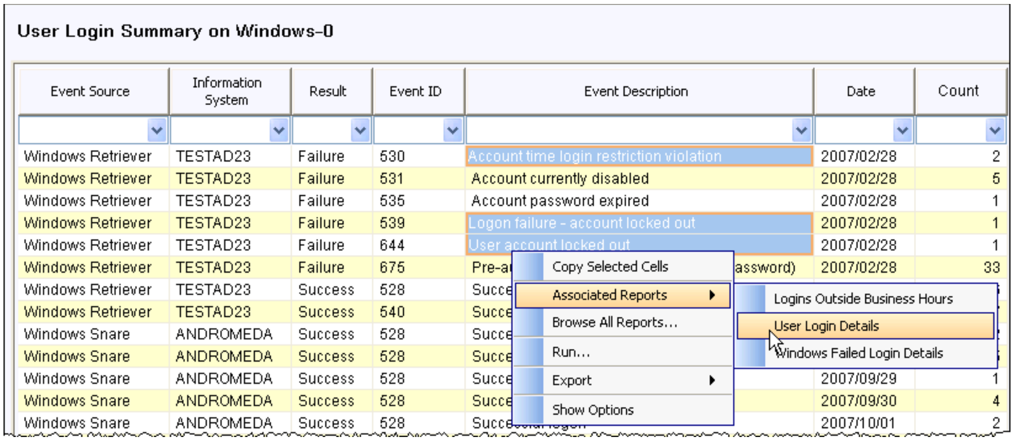
Save
Revert
Refresh
Dashboards
Reports
Administration
Options Pane
Reports
All Report Definitions
Compliance Analytics Package
HP Secure Advantage
McAfee
login
Associated Reports
Internal System Failed Login Details
Unix or Linux Logins Summary
User Login Details
Windows Login Failure Summary
Logins Outside Business Hours
Windows Login Activity Details
Windows Login Success Summary
User Login Summary
Windows Failed Login Details
Internal System Successful Login Details
Logins Outside Business Hours Summary
Unix or Linux Logins Details
User Login Summary on Windows
User Login Summary on Windows
Created 3/10/10, administrator
Last Modified 3/10/10, administrator
Data Source
Criteria
Summarize
Properties
Column Display
Chart
Associated Reports
Drag Report Definitions from the Chooser here.
Drag report widget(s) to workspace
Print Options
Paper size Letter
Orientation Landscape
Margins...
Title Page
□ Report Name
□ Date Criteria
□ Created By
□ Created Date
□ Last Modified By
□ Last Modified Date
□ Filters
□ Namespace
Query Limit
Maximum rows per query 10000
All Report Definitions
User Login Summary on Windows

The operation illustrated above associates three reports simultaneously to the User Login Summary on Windows report. To select multiple reports, use the standard interface keys:

- **Contiguous reports**—select the first report; SHIFT-CLICK as you select the last report
- **Non-contiguous reports**—select the first report; CTRL-CLICK as you select each additional report

At runtime, the user can investigate data in any or all of the reports you associate. Figure 4-34 illustrates a user selecting three values from the Event Description column and right-clicking to display the popup. From the popup, the user chooses to run one of three associated reports: the User Login Details on Windows.

Figure 4-34: Selecting Multiple Column Values for an Associated Report

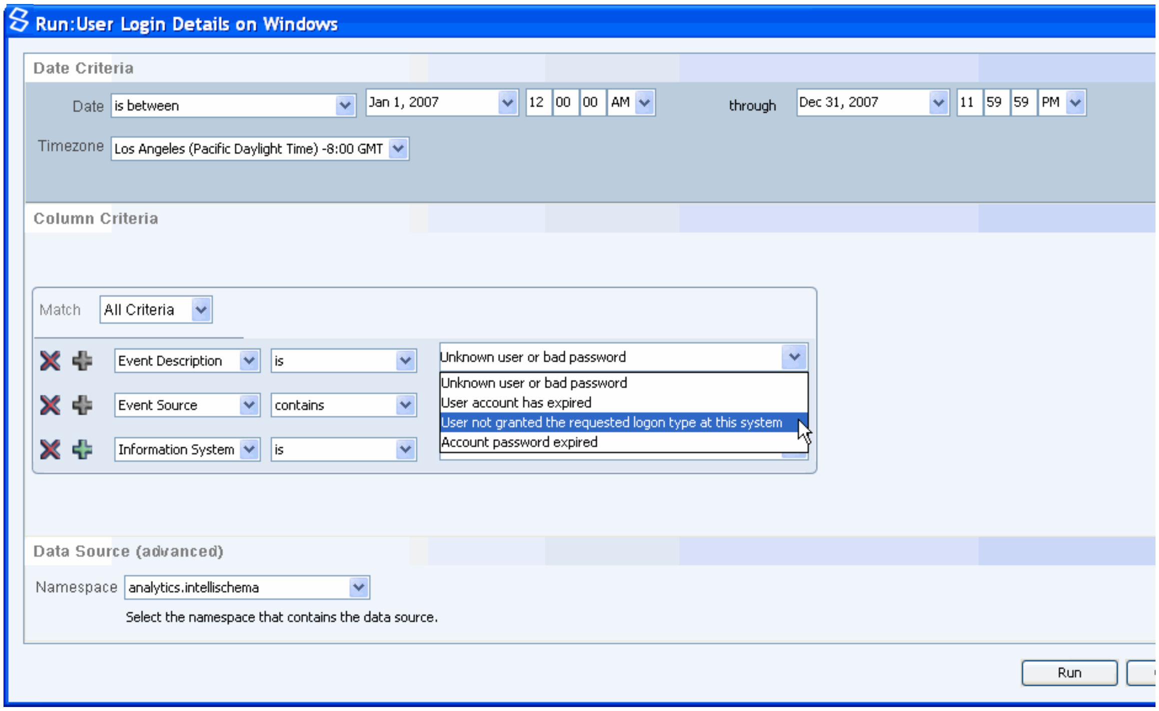


User Login Summary on Windows-0

Event Source	Information System	Result	Event ID	Event Description	Date	Count
Windows Retriever	TESTAD23	Failure	530	Account time login restriction violation	2007/02/28	2
Windows Retriever	TESTAD23	Failure	531	Account currently disabled	2007/02/28	5
Windows Retriever	TESTAD23	Failure	535	Account password expired	2007/02/28	1
Windows Retriever	TESTAD23	Failure	539	Logon failure - account locked out	2007/02/28	1
Windows Retriever	TESTAD23	Failure	644	User account locked out	2007/02/28	1
Windows Retriever	TESTAD23	Failure	675	Pre-a … assword)	2007/02/28	33
Windows Retriever	TESTAD23	Success	528	Succe		
Windows Retriever	TESTAD23	Success	540	Succe		
Windows Snare	ANDROMEDA	Success	528	Succe		
Windows Snare	ANDROMEDA	Success	528	Succe		
Windows Snare	ANDROMEDA	Success	528	Succe	2007/09/29	1
Windows Snare	ANDROMEDA	Success	528	Succe	2007/09/30	4
Windows Snare	ANDROMEDA	Success	528	Succe	2007/10/01	2

After the user selects the associated report, its **Run** dialog displays, as illustrated in Figure 4-35. All three values selected from the source report display in each criteria-column dropdown of the associated report. In other words, if the **Run** dialog provides multiple criteria rows, each representing a different column in the source report, each dropdown displays all user-selected values.

Figure 4-35: Running the Associated Report on Selected Data



Run:User Login Details on Windows

Date Criteria

Date: is between — Jan 1, 2007 12 00 00 AM — through — Dec 31, 2007 11 59 59 PM

Timezone: Los Angeles (Pacific Daylight Time) -8:00 GMT

Column Criteria

Match: All Criteria

Event Description — is — Unknown user or bad password

Event Source — contains

Information System — is

Unknown user or bad password
User account has expired
User not granted the requested logon type at this system
Account password expired

Data Source (advanced)

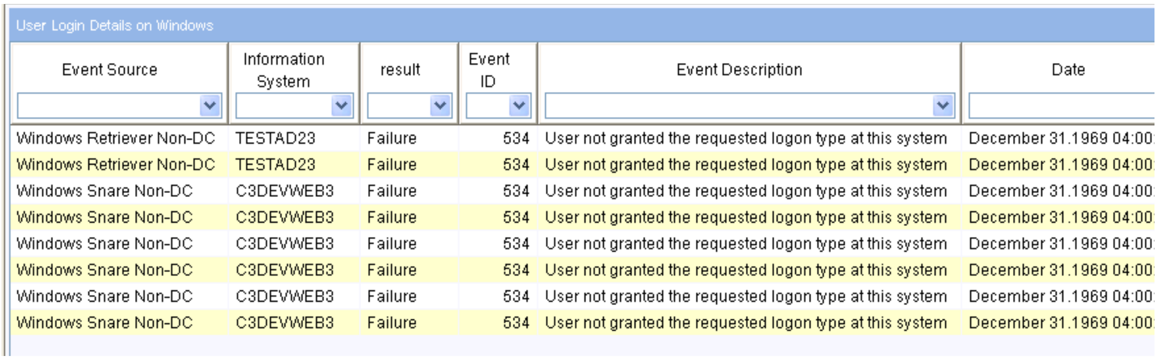
Namespace: analytics.intellischema

Select the namespace that contains the data source.

The user selects the desired value from the appropriate dropdown, which is **Event Description** in the current example, and deletes the other two column criteria rows by clicking the ✖ icon. If none of the column-criteria rows represent the desired column, the user can select the desired column from the column dropdown.

Figure 4-36 illustrates the results of running the associated reports with one of the values in the **Event Description** dropdown.

Figure 4-36: Associated Report Results



User Login Details on Windows

Event Source	Information System	result	Event ID	Event Description	Date
Windows Retriever Non-DC	TESTAD23	Failure	534	User not granted the requested logon type at this system	December 31.1969 04:00
Windows Retriever Non-DC	TESTAD23	Failure	534	User not granted the requested logon type at this system	December 31.1969 04:00
Windows Snare Non-DC	C3DEVWEB3	Failure	534	User not granted the requested logon type at this system	December 31.1969 04:00
Windows Snare Non-DC	C3DEVWEB3	Failure	534	User not granted the requested logon type at this system	December 31.1969 04:00
Windows Snare Non-DC	C3DEVWEB3	Failure	534	User not granted the requested logon type at this system	December 31.1969 04:00
Windows Snare Non-DC	C3DEVWEB3	Failure	534	User not granted the requested logon type at this system	December 31.1969 04:00
Windows Snare Non-DC	C3DEVWEB3	Failure	534	User not granted the requested logon type at this system	December 31.1969 04:00
Windows Snare Non-DC	C3DEVWEB3	Failure	534	User not granted the requested logon type at this system	December 31.1969 04:00

Associating a Report to a Security Alert

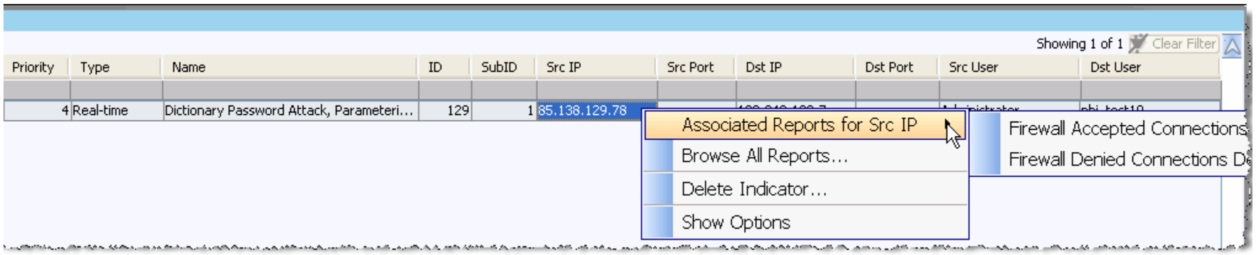
BACKGROUND

As documented in “Associating a Report to Another Report”, on page 161, you can associate detailed reports to summary reports to allow users to easily investigate specific data in the summary report. The associated reports enable users to conduct investigations that examine the same set of data in detail in more than one report.

As documented in “Launching Associated Reports”, on page 69, a dashboard user can select value(s) from a column in a security alert and open associated report(s) that enable the user to conduct investigations on that data.

Figure 4-37 illustrates a security alert as it displays in the dashboard. In the screen shot below, the user is about to launch one of two associated reports for the **Src IP** column. The user has selected data in one of the alert columns and has right-clicked to display the reports associated to that column.

Figure 4-37: Launching an Associated Report



Showing 1 of 1

Priority	Type	Name	ID	SubID	Src IP	Src Port	Dst IP	Dst Port	Src User	Dst User
4	Real-time	Dictionary Password Attack, Parameteri...	129	1	85.138.129.78		[illegible]		[illegible]	[illegible]

Clear Filter
Associated Reports for Src IP
Browse All Reports...
Delete Indicator...
Show Options
Firewall Accepted Connections
Firewall Denied Connections D

After the user selects the desired report, its **Run** dialog displays. All of its column-criteria fields contain the value that the user selected. For more information on launching the report, see “Launching Associated Reports”, on page 69.

How Do You Associate A Report to a Security Alert?

The process to associate a report to a security alert is similar to the one to associate a report to another report. There are two primary differences, however:

- Instead of using the **Properties** tab in Report edit mode to associate a report, you use Administration mode to associate a report to a security alert.

  NOTE: You must be assigned to the **analyzer.admin** role to access Administration mode.

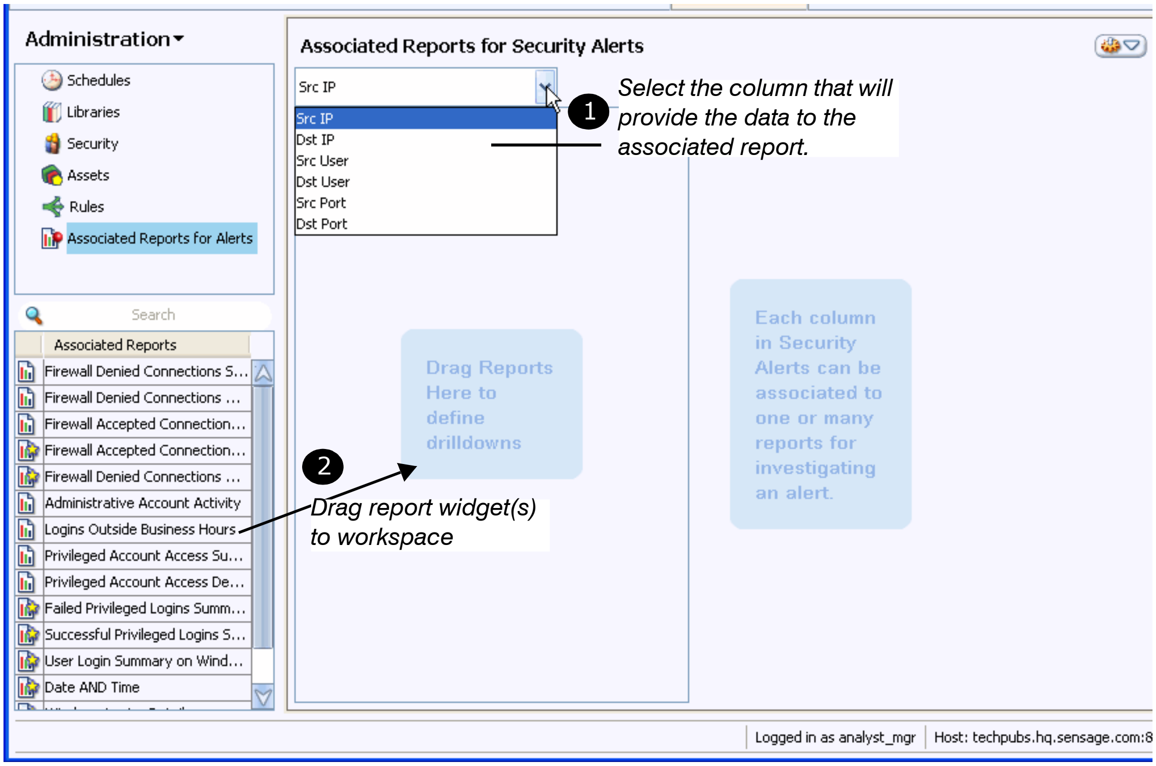
- You associate reports separately to one or more of six columns: source IP address (**Src IP**), destination IP address (**Dest IP**), source user (**Src User**), destination user (**Dest User**), source port (**Src Port**), and destination port (**Dest Port**).

The user can select the data to examine only from one of these six columns in the security alert. The selected data populates the column criteria field(s) of the associated report, which returns only rows that pertain to the selected value. Because the associated report limits results to the user-selected value, each report you associate must have at least one column criteria field that is relevant to the selected column. The user can investigate data only in columns that have a corresponding column criteria field in the associated report. For example, if you associate a report to the **Src IP** column, the associated report should have at least one column-criteria field for the **Src IP** column.

If the user selects more than one value, all selected values display in a dropdown. The user selects one value from the dropdown and runs the report, which returns only rows that match the selected value. The user must run the report separately for each of the other selected values.

Figure 4-38 illustrates the window in which you associate reports to security alerts and the steps to associate a report.

Figure 4-38: Associating Reports to a Security Alert



Firewall Denied Connections S...
Firewall Denied Connections ...
Firewall Accepted Connection...
Firewall Accepted Connection...
Firewall Denied Connections ...
Administrative Account Activity
Logins Outside Business Hours
Privileged Account Access Su...
Privileged Account Access De...
Failed Privileged Logins Summ...
Successful Privileged Logins S...
User Login Summary on Wind...
Date AND Time

1 *Select the column that will provide the data to the associated report.*

2 *Drag report widget(s) to workspace*

Drag Reports Here to define drilldowns

Each column in Security Alerts can be associated to one or many reports for investigating an alert.

The window above displays when you select the **Associated Reports for Alerts** module from the Navigator in Administration mode. As illustrated above, you select each column separately and drag only reports relevant to that column to the workspace.

For an illustration of how you can drag multiple reports simultaneously, see Figure 4-33. For an explanation of how to write a parameterized SQL query that provides criteria rows for multiple columns, see “Defining Parameters for an Associated Report”, on page 194.

NOTE: HawkEye AP Analytics includes the Investigation Event Report, which queries all tables in the EDW. This special report was designed to query the EDW for events related to security alerts. Associate this Analytics report to a security alert to maximize your users’ investigation options.

Specifying Print Options and Maximum Rows

In addition to the associate reports to a report option, the **Properties** tab enables several other operations:

- **Specify print options**:
  - **Paper size**—select the desired paper size
  - **Orientation**—either landscape or portrait orientation
  - **Title Page**—select one or more pieces of metadata to display on the title page; options include the report name, the properties associated with the report in the All Report Definitions list, and a namespace

- **Margin**—click to display a dialog that enables you to choose between inches or centimeters and to specify margin size at the top, bottom, right, and left.

- **Specify the maximum rows to display on each page for cached reports**

Figure 4-33 illustrates the **Properties** tab.

Formatting Columns

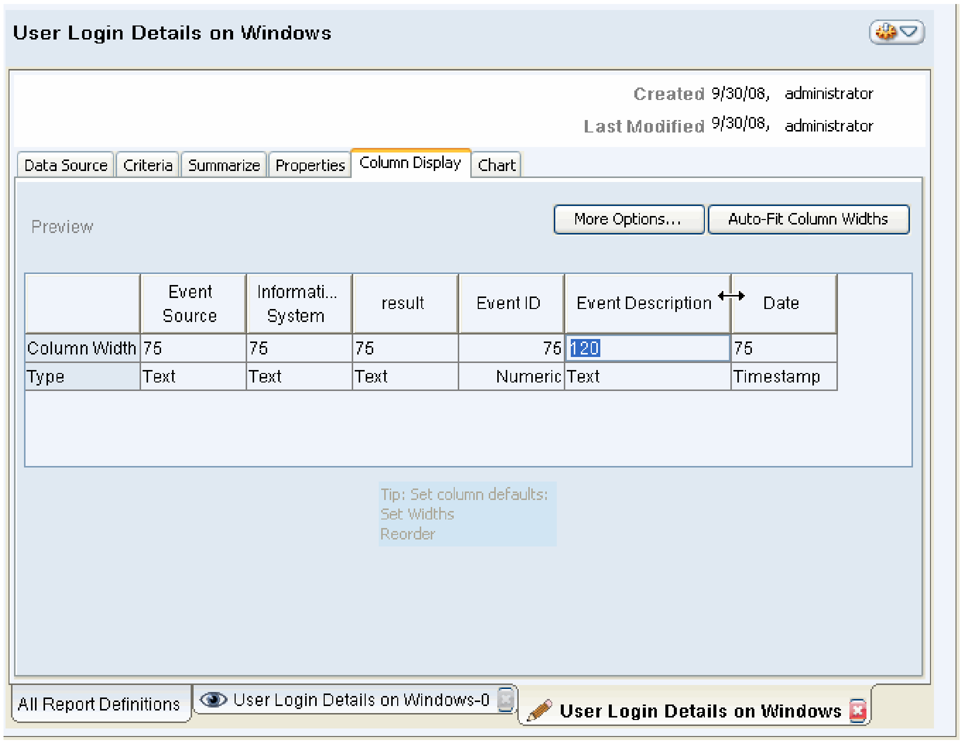
The **Column Display** tab provides several options for formatting result columns. This tab enables you to configure:

- Width of each column
- Relative order of columns
- Display name of each column
- Data type of each column
- Font style, size, and color of each heading and its data

Changing Column Width and Relative Position

Figure 4-39 illustrates two ways you can change column width from the **Column Display** tab.

Figure 4-39: Changing Column Width



User Login Details on Windows

Created 9/30/08, administrator

Last Modified 9/30/08, administrator

	Event Source	Informati... System	result	Event ID	Event Description	Date
Column Width	75	75	75	75	120	75
Type	Text	Text	Text	Numeric	Text	Timestamp

Tip: Set column defaults:
Set Widths
Reorder

All Report Definitions
User Login Details on Windows-0
User Login Details on Windows

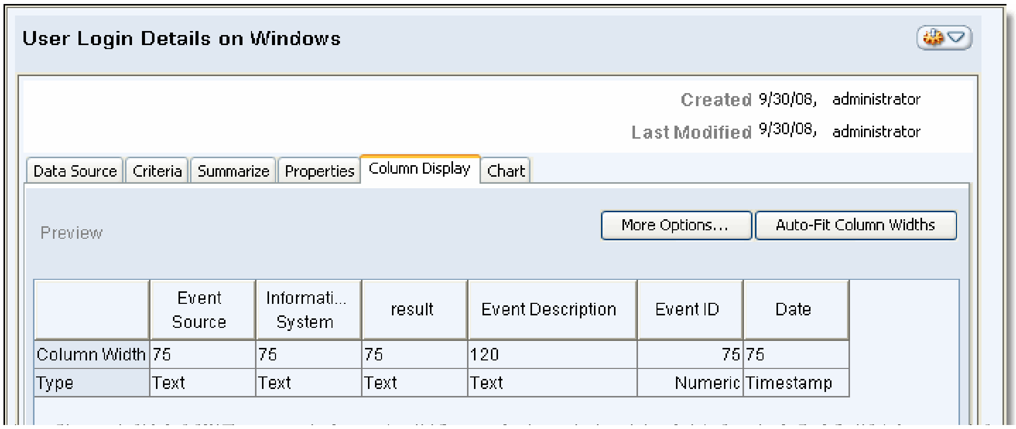
The graphic above illustrates two different ways to change column width:

- As indicated by the resize cursor to the right of the **Event Description** column header, you can resize column width by dragging the header boundary to the left or right.

- As indicated by the highlighted width units in the Column-Width row of **Event Description**, you can double-click the width value and overwrite it.

You can also reorder columns. Figure 4-40 illustrates the same report as above but the **Event Description** column now displays to the left of **Event ID**. The report below also illustrates new column sizes for the **result** and the **Event Description** columns.

Figure 4-40: Changed Column Width and Display Order



User Login Details on Windows

Created 9/30/08, administrator

Last Modified 9/30/08, administrator

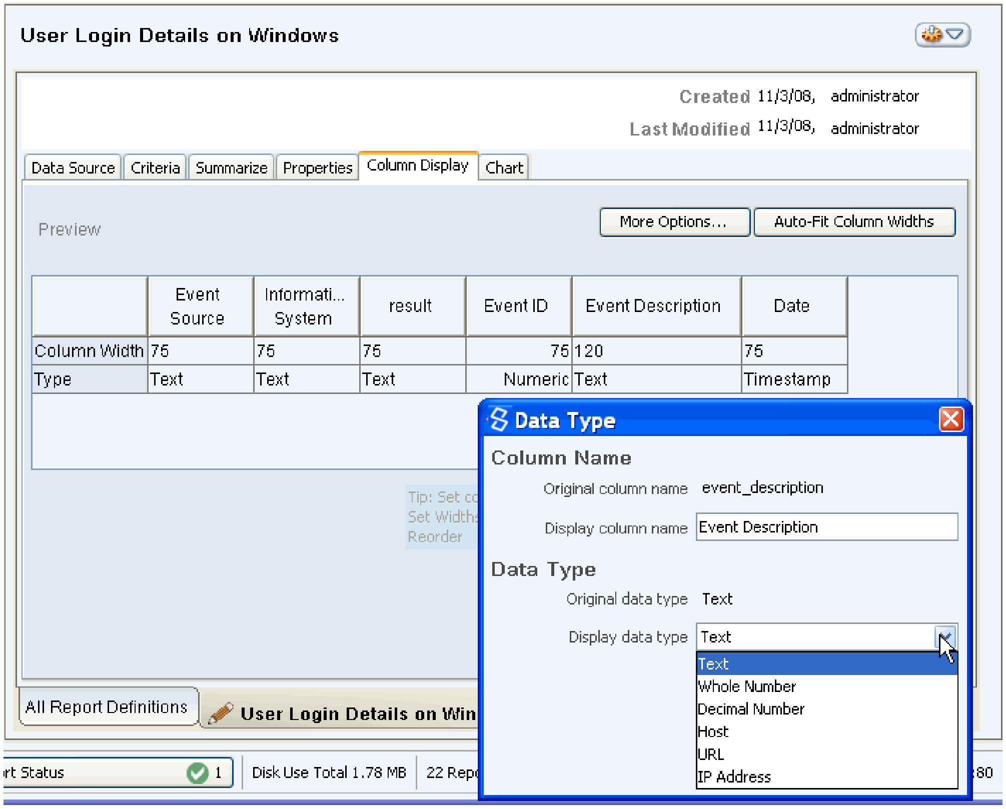
	Event Source	Informati... System	result	Event Description	Event ID	Date
Column Width	75	75	75	120	75	75
Type	Text	Text	Text	Text	Numeric	Timestamp

To change relative display of a column, grab its header and drag it to the desired location.

Changing Column Name and Data Type

Figure 4-41 illustrates the **Data Type** dialog, in which you can change column name and data type. The changes you make here affect display only and not the way this information is stored in the EDW. Click in the Type cell of the desired column to display the **Data Type** dialog.

Figure 4-41: Changing Column Name and Data Type



User Login Details on Windows

Created 11/3/08, administrator
Last Modified 11/3/08, administrator

	Event Source	Informati... System	result	Event ID	Event Description	Date
Column Width	75	75	75	75	120	75
Type	Text	Text	Text	Numeric	Text	Timestamp

Data Type

Column Name

Original column name event_description

Display column name Event Description

Data Type

Original data type Text

Display data type Text

The **Column Name** section of the **Data Type** dialog displays the name of the column as stored in the EDW (**Original column name**) and provides an entry field in which you can rename it for display.

The **Data Type** section of the **Data Type** dialog enables you to change the display data type.

- The display choices in the **Display data type** dropdown depend on the column data type:
  - **Text**—Text, Whole Number, Decimal Number, Host, URL, and IP_Address
    - **Text**—*uses standard text sorting*
    - **Whole Number**—*uses standard numeric sorting*
    - **Decimal Number**—*uses standard numeric sorting*
    - **Host**—*uses standard text sorting.*
    - **URL**–*sorts columns of text as web addresses*
    - **IP Address**—*uses standard text sorting.*

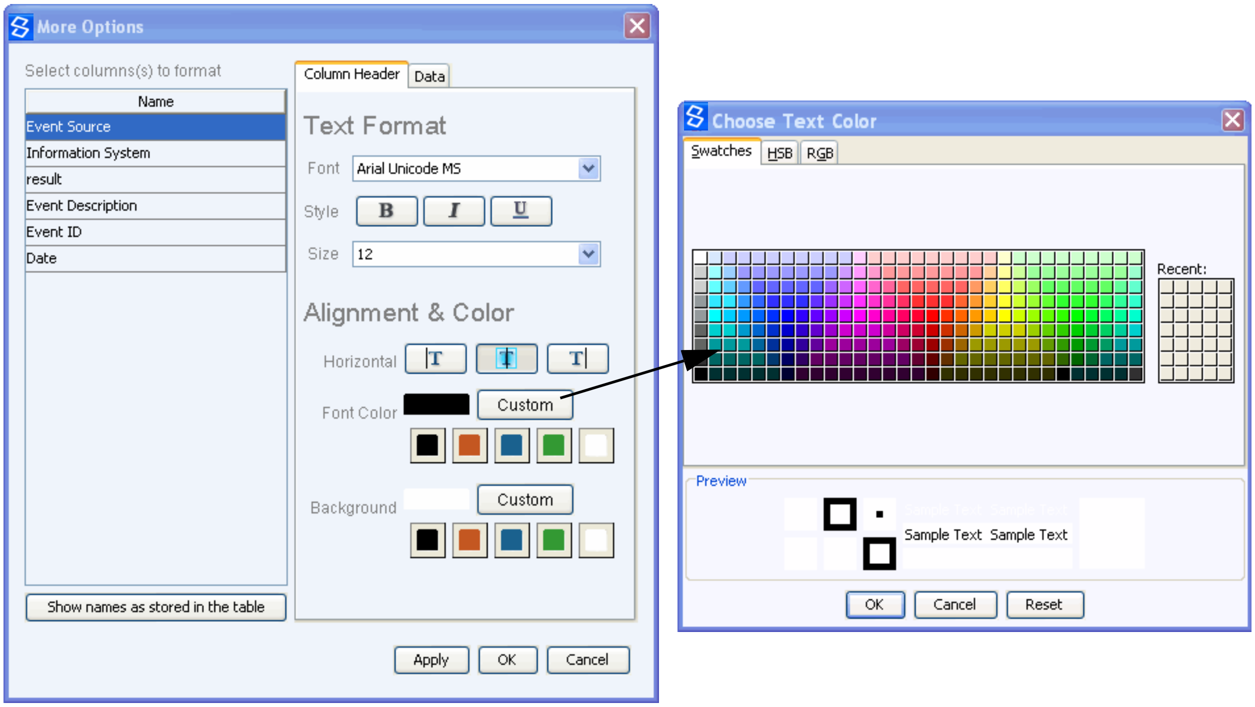
- **Numeric**—Whole Number and Text
- **Timestamp**— Timestamp

Formatting Font, Alignment, and Color

You can separately change the font type, style, and size for each column header and its data values. As illustrated in Figure 4-42, you can also specify horizontal alignment, font color, and background color.

To access these display options, click **More Options....** from the **Column Display** tab.

Figure 4-42: Formatting Column Display



More Options

Select columns(s) to format

Name
Event Source
Information System
result
Event Description
Event ID
Date

Text Format

Font: Arial Unicode MS

Size: 12

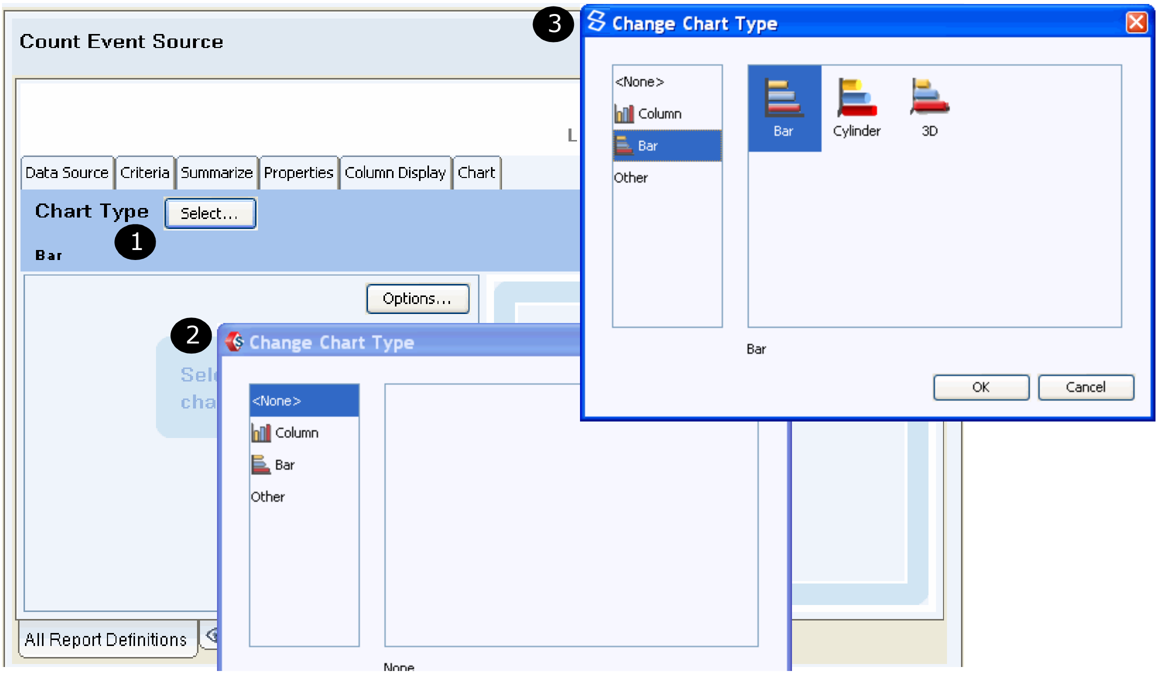
The graphic above illustrates the **More Options** dialog open to the **Column Header** tab. It also illustrates the **Choose Text Color** dialog that displays when you click **Custom** from the **Alignment & Color** pane.

Each column name displays in a list on the left. Currently the user has selected the **Event Source** column and is formatting its header properties. To format its data values, the user would select the **Data** tab. To format a different column, the user would select the column from the list on the left.

Defining Charts

The **Chart** tab enables you to format data as one of several different types of chart. Figure 4-43 illustrates the Chart tab window as it first displays and as the user selects a bar chart style.

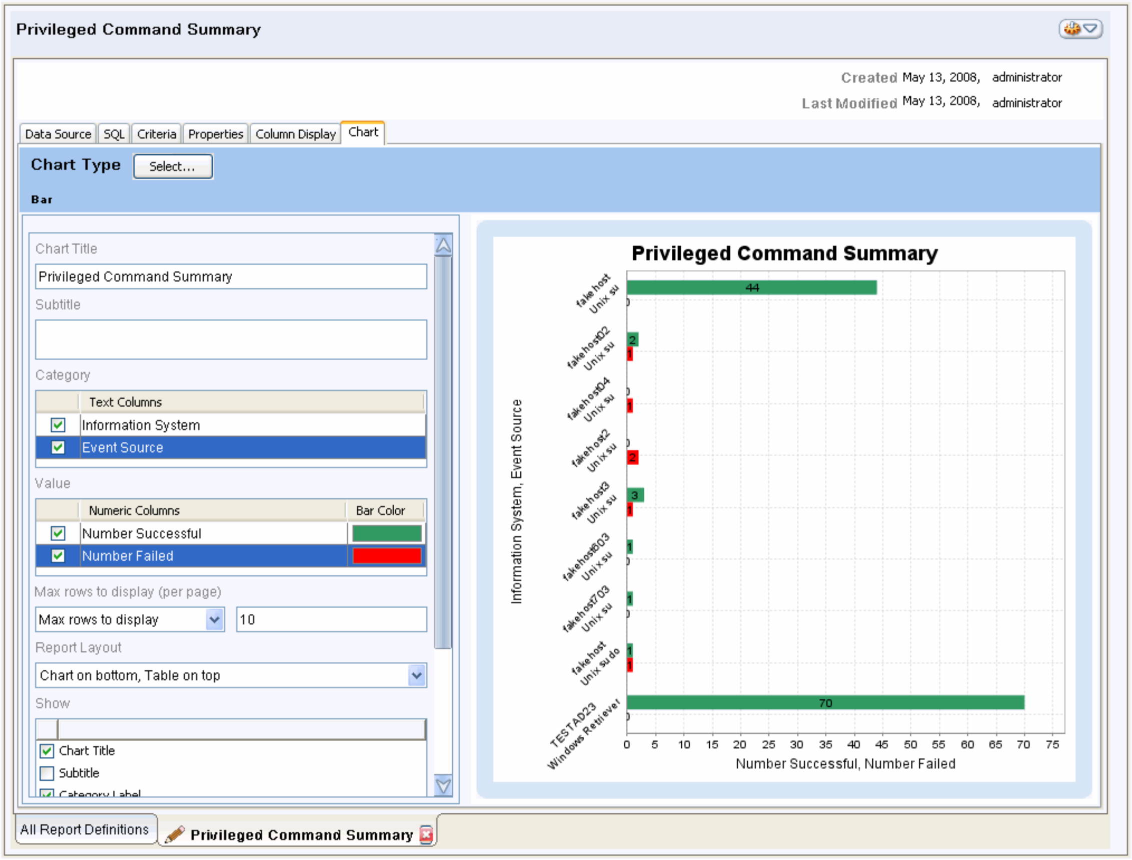
Figure 4-43: Formatting Chart Display



NOTE: Select **Other** to display line chart style.

Figure 4-44 illustrates bar-chart display with one column selected for Y axis and two columns for X axis.

Figure 4-44: Formatting a Report as a Bar Chart



Privileged Command Summary

Created May 13, 2008, administrator
Last Modified May 13, 2008, administrator

Chart Type: Bar

Chart Title: Privileged Command Summary
Subtitle:

Category

	Text Columns
☑	Information System
☑	Event Source

Value

	Numeric Columns	Bar Color
☑	Number Successful	🟩
☑	Number Failed	🟥

Max rows to display (per page): Max rows to display 10

Report Layout: Chart on bottom, Table on top

Show

☑ Chart Title
☐ Subtitle
☑ Category Label

By default only a chart title displays. As illustrated above, the user has also specified a subtitle. In addition to selecting the values to display from the **Category** and **Value** fields, you can specify:

- bar color(s)—for more information, see Figure 4-45
- maximum number of rows to display
- report layout (relative position of chart and table)
- information to display—for more information, see Figure 4-45

You can label the chart data in several ways:

- Category Label — displays the name of the Category column(s)

  The example above displays two Category columns (**Information System** and **Event Source**). These names display vertically along the left side of the example chart.

- Category Title — displays the data in the Category column(s)

  The example above displays data from both Category columns; for example, `fake host` and `Unix su`.

- Value Label — displays the name of the Value column(s)

  The example above displays two Value columns (**Number Successful** and **Number Failed**). These names display horizontally at the bottom of the example chart.

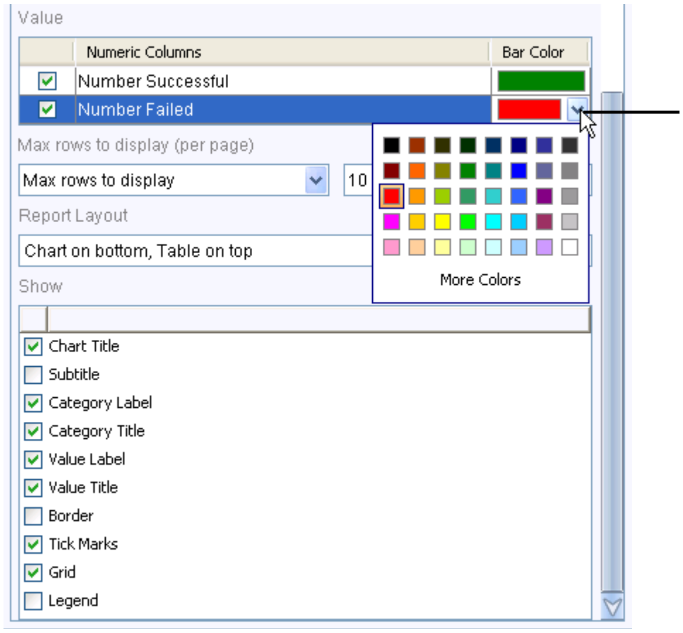
- Value Title — displays the data in the Value column(s)

  The example above displays integers that indicate the value of data in both Value columns.

NOTE: If the text that displays in a Category Label or Value Label exceeds 25 characters, it is truncated to 25 characters followed by three periods (...). If this length is insufficient at your site, your administrator can modify the length in the `chart.max.axis.label.length` property of the `vendor.prop` file. The `vendor.prop` file is documented in HawkEye AP Console Installation and Configuration in Chapter 2, “Configuring HawkEye AP” of the *Installation, Configuration, and Upgrade Guide*.

Figure 4-45 illustrates color selection and the full set of information fields.

Figure 4-45: Selecting Bar Colors and Setting Display Information



Value

	Numeric Columns	Bar Color
☑	Number Successful	🟩
☑	Number Failed	🟥

Max rows to display (per page)

Max rows to display 10

Report Layout

Chart on bottom, Table on top

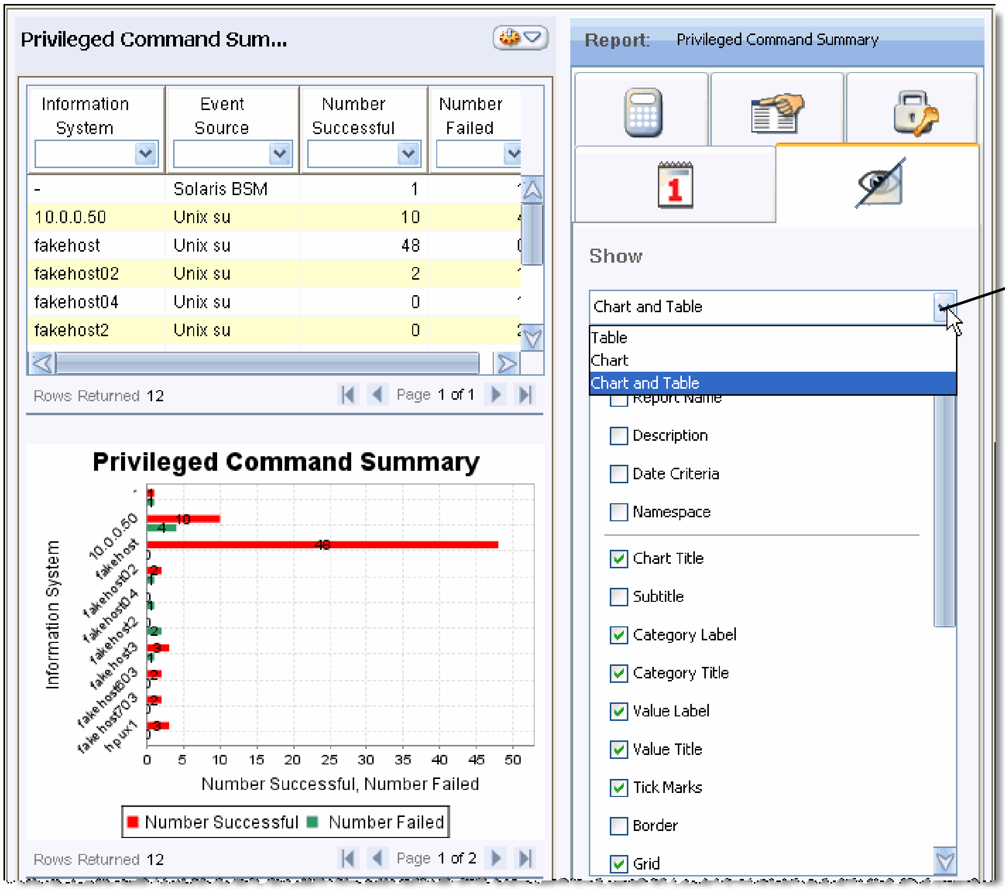
Show

☑ Chart Title
☐ Subtitle
☑ Category Label
☑ Category Title
☑ Value Label
☑ Value Title
☐ Border
☑ Tick Marks
☑ Grid
☐ Legend

Click current color to display color palette

After you configure a report to display as a chart, both the chart and tabular format display when you view the report in Reports mode. You can use the Options Pane to display only chart format or only table format, as illustrated in Figure 4-46.

Figure 4-46: Viewing a Report as Chart and Table in Reports Mode



Privileged Command Sum...

Information System	Event Source	Number Successful	Number Failed
-	Solaris BSM	1	1
10.0.0.50	Unix su	10	4
fakehost	Unix su	48	0
fakehost02	Unix su	2	1
fakehost04	Unix su	0	1
fakehost2	Unix su	0	2

Rows Returned 12 Page 1 of 1

Rows Returned 12 Page 1 of 2

Report: Privileged Command Summary

Show

Chart and Table

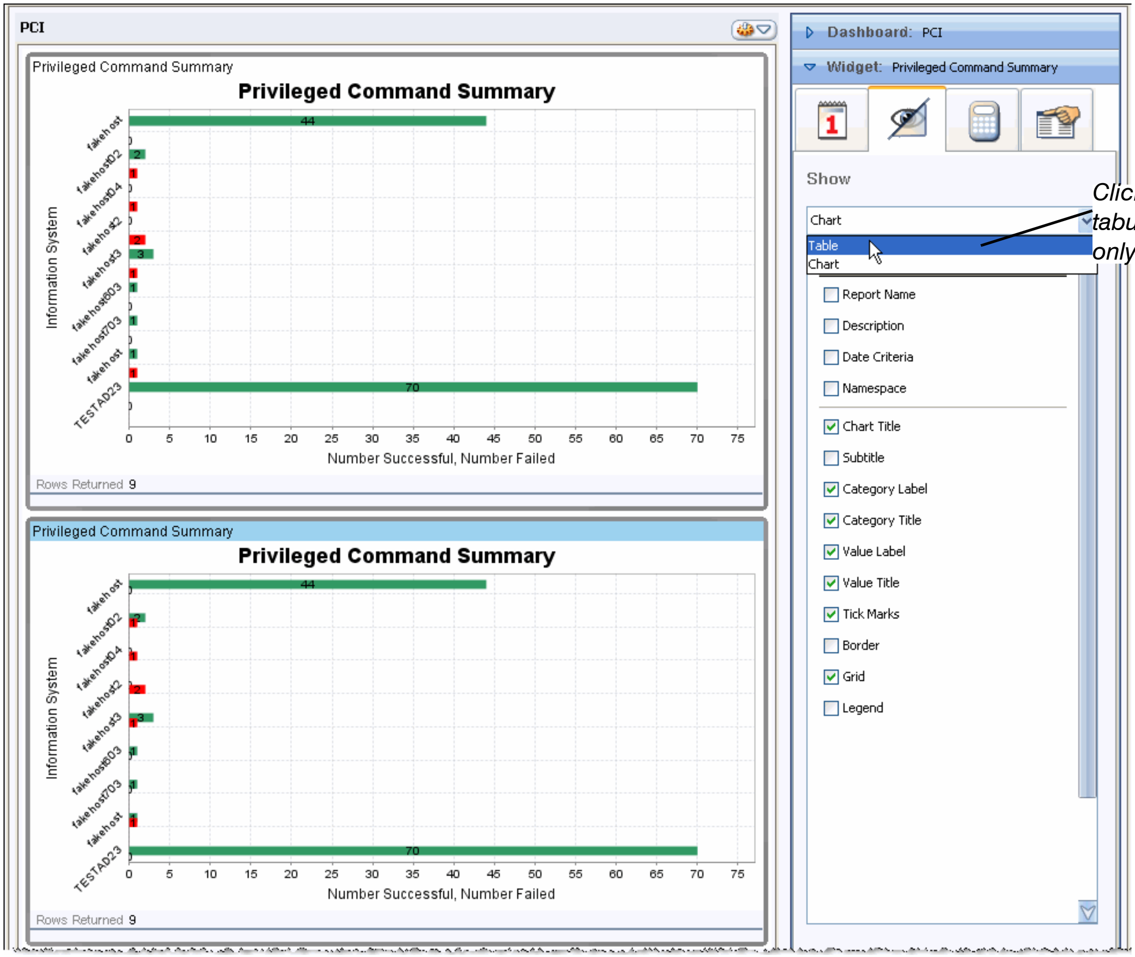
□ Report Name
□ Description
□ Date Criteria
□ Namespace
☑ Chart Title
□ Subtitle
☑ Category Label
☑ Category Title
☑ Value Label
☑ Value Title
☑ Tick Marks
□ Border
☑ Grid

Click to display chart only or table only.

Although both the chart and tabular format display when you view the report in Reports mode, Dashboards mode displays only one format style at a time. By default, when you drag the widget for this report onto a dashboard page, only the chart displays. To display both formats on the same page, drag the same widget twice. Then use the dashboard Options Pane to modify display of one widget to table format. Figure 4-47 illustrates how you would change display of the lower widget from chart to table.

Figure 4-47: Viewing a Report as Chart and Table in Dashboards Mode

1 *Select one report widget.*



Dashboard: PCI

Widget: Privileged Command Summary

Show

Chart

Table
Chart

- □ Report Name
- □ Description
- □ Date Criteria
- □ Namespace
- ☑ Chart Title
- □ Subtitle
- ☑ Category Label
- ☑ Category Title
- ☑ Value Label
- ☑ Value Title
- ☑ Tick Marks
- □ Border
- ☑ Grid
- □ Legend

2 *Click to display tabular format only.*

CHAPTER 5

Creating and Editing SQL Reports

This chapter contains the following sections:

- “Overview of SQL Reports”, next
- “Creating a SQL Report”, on page 177
- “Running and Viewing the Report”, on page 181
- “Editing the SQL Report”, on page 183
- “Viewing and Changing a SQL Report That Has Been Run”, on page 199

OVERVIEW OF SQL REPORTS

To create a SQL report, you enter a SQL query directly into the SQL tab window. A *query* is a Sensage SQL statement that extracts event-log data from the EDW data store. Creating a SQL report requires knowledge of SQL and HawkEye AP extensions to the language.

When you create a SQL report, you can define more advanced queries than you can when you create a Wizard report. For example, a SQL report enables you to include conditional statements, HawkEye AP functions, libraries, and nested queries that use the results of one query as the input to the next.

NOTE: Many Sensage SQL extensions are embedded in the views that HawkEye AP delivers with its Analytics reports. When you base a Wizard report on one of these views, you indirectly take advantage of these powerful extensions. However, if you base your report directly on a table, you must create a SQL report if you want to include conditional statements or libraries or use functions that take advantage of the SQL extensions.

For information on the SQL query language and the HawkEye AP extensions to it, see Chapter 10: Sensage SQL and Chapter 11: SQL Functions.

CREATING A SQL REPORT

There are several steps to creating a SQL report:

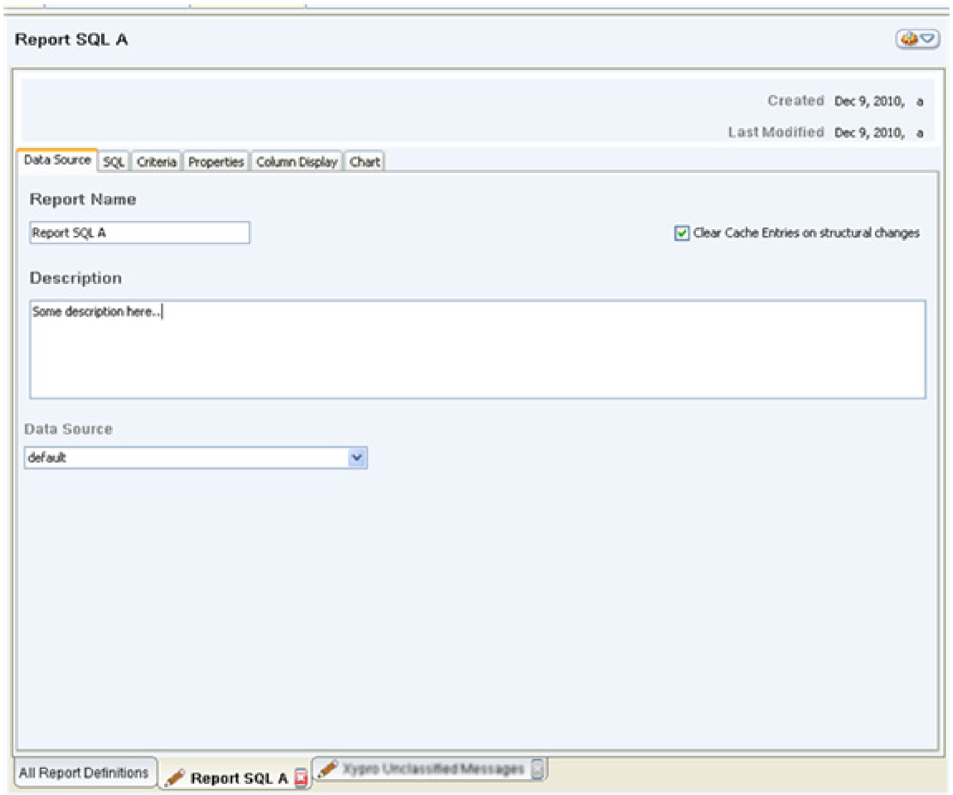
- “Specifying the Report’s Name, Description, and Namespace”, next
- “Entering the Query”, on page 179
- “Specifying a Time Range”, on page 180

Specifying the Report’s Name, Description, and Namespace

To create a SQL report, select **New SQL Report Definition** from any Action Menu or right-click menu in Reports mode.

Figure 5-1 illustrates the first window that displays when you begin defining a new SQL report definition.

Figure 5-1: Specifying the Report's Name, Description, and Namespace



Report SQL A

Created Dec 9, 2010, a

Last Modified Dec 9, 2010, a

Report Name

Report SQL A

☑ Clear Cache Entries on structural changes

Description

Some description here...

Data Source

default

As illustrated above, the first window in SQL-report creation provides the following fields:

- **Report Name**—Every report must have a unique name. Depending on your permissions, you might not see all reports in the All Report Definitions list. However, you cannot name a report identically to one that already exists in your system. The system notifies you if the name you enter is a duplicate.

- **Report Description**—The description, which is optional, displays in the **All Report Definitions** window and can be very helpful to users who have many reports in the All Report Definitions list.

- **Data Source**—If you have access to more than one namespace, the **Data Source** dropdown prompts you to select a namespace. If you have access to ten or more namespaces, the **Select Namespace** dialog displays. This dialog enables you to choose from all available namespaces. If you have access to only one namespace, its name displays in the **Data Source** field. For information about how HawkEye AP groups tables and view in namespaces, see "Namespaces: Using a Single Report or Dashboard to Access Different Data", on page 43.

- **Clear Cache Entries on structural changes**—When this box is checked, all cache entries for this report are deleted any time you make changes to this report definition and save those changes. If you leave this box unchecked and change the report definition, you will not be able to save those changes until you first delete all cache entries for the report. If you try to save changes to a report definition that has existing cache entries, you will see a warning message but at that point, the only way to keep any changes is to copy the SQL and paste it into a new report.

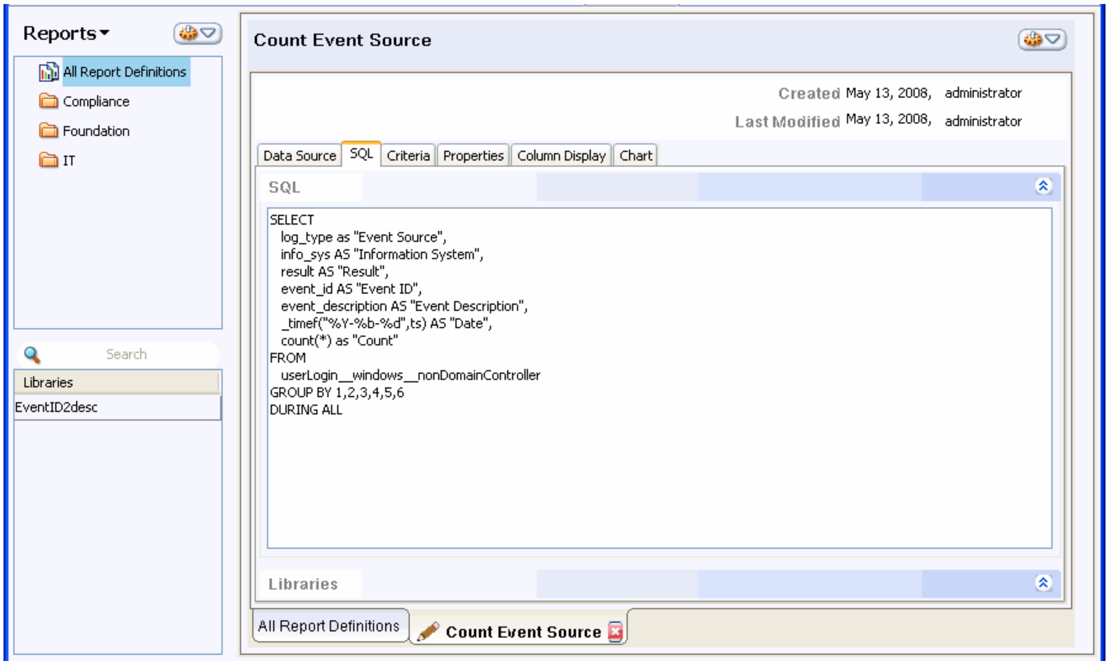
It may be useful to check this box when iteratively developing and debugging a report, so that you do not have to manually delete cache entries after each test run of the report. When the report is ready for use in a production environment, Sensage recommends that you uncheck this box so that cache entries are not unexpectedly deleted.

Entering the Query

Click the **SQL** tab to move to the next window.

Figure 5-2 illustrates the window in which you enter the query that retrieves and formats data for your report.

Figure 5-2: Entering the SQL Statement



Created May 13, 2008, administrator
Last Modified May 13, 2008, administrator

```
SELECT
   log_type as "Event Source",
   info_sys AS "Information System",
   result AS "Result",
   event_id AS "Event ID",
   event_description AS "Event Description",
   _timef("%Y-%b-%d",ts) AS "Date",
   count(*) as "Count"
FROM
   userLogin__windows__nonDomainController
GROUP BY 1,2,3,4,5,6
DURING ALL
```

As illustrated above, you write your query directly in the SQL pane. The query above creates the same report as the one illustrated in “Editing a Wizard Report”, on page 151. Both reports retrieve data from the **`userLogin_windows_nonDomainController`** view.

The final clause of the SQL statement is `DURING ALL`, which represents the time range covered by the result set. Most queries of event-log data involve a specific time frame, whether “yesterday”, “last year,” or “1 hour ago.” To facilitate these types of queries, Sensage SQL supports the proprietary `DURING` clause, which specifies the time range for the query.

The `DURING` clause works with the **ts** timestamp column, which is mandatory in every event-log table. The SQL query engine matches the time range specified in the `DURING` clause with values in the **`ts`** column of the table. Only rows that match the time range are included in the result set.

When a query is run directly against the EDW, the `DURING` clause can contain a specific time range. However, when a report query is run from HawkEye AP Console, the only recommended value for the DURING clause is "DURING ALL". At run time, HawkEye AP replaces all instances of DURING ALL with the time range saved with the report, or specified when the report is run from Reports mode, or by the schedule that runs the report. In other words, when the report query specifies "DURING ALL", the report runs with the date criteria specified in the report definition or at run time. However, if the report query specifies a date period, the report always runs on that period and neither a schedule nor the run-time user can change the date period.

For more information, see:

- "Specifying a Time Range", next
- "Parameterizing Your Query", on page 184

NOTE: If the query specifies a start date and time and end date and time, the report always uses those times when it runs, regardless of what the user enters at run time.

The example query illustrates using the count() aggregation function and the timef() time-formatting function.

For information on these functions, see:

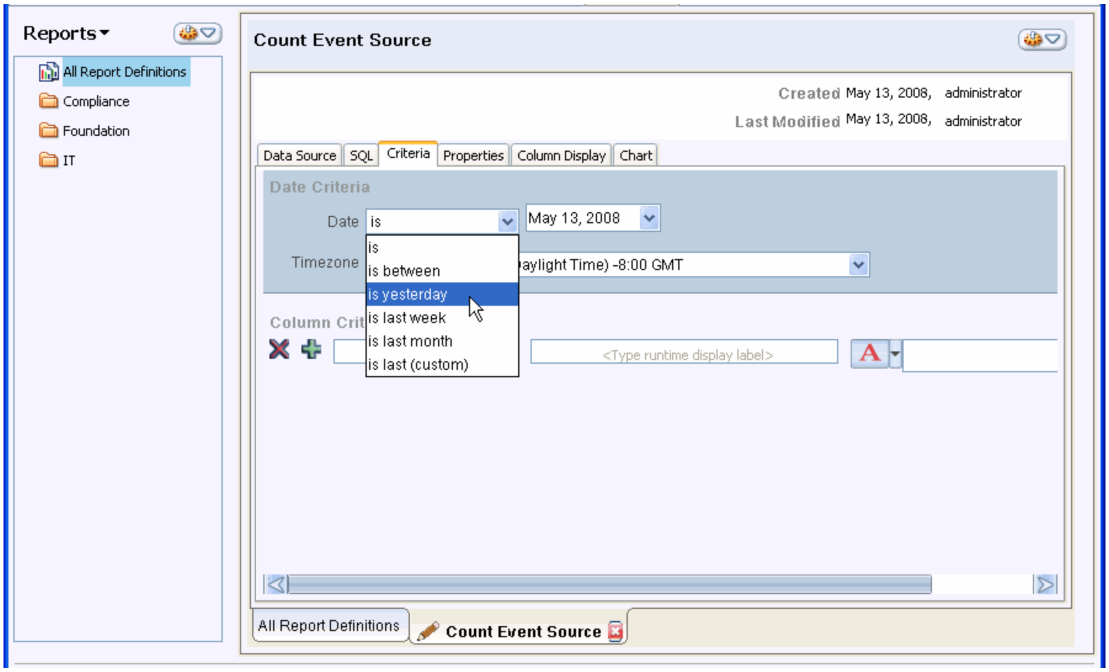
- counting values in a column—"Aggregation Functions", on page 337
- formatting a date value— "_timeformat(), _timef()", on page 380

Specifying a Time Range

Click the **Criteria** tab to move to the next window. Figure 5-3 illustrates the window in which you can specify a default time range and, optionally, column criteria. This topic discusses specifying date criteria. For information on specifying column criteria, see "Parameterizing Your Query", on page 184.

The **Date** dropdown displays all available date options.

Figure 5-3: Specifying Date Criteria



Created May 13, 2008, administrator
Last Modified May 13, 2008, administrator

Date: is — May 13, 2008
Timezone: aylight Time) -8:00 GMT

is
is between
is yesterday
is last week
is last month
is last (custom)

A illustrated above, there are several options for limiting the time range. For more information, see “Date Criteria: Specifying Time Range”, on page 134. Use this window to provide a default time range. Your range can be overridden by a user or schedule that runs the report.

IMPORTANT: If your SQL query selects all columns in a table or view and you specify date criteria that covers a long period, the report results can fill your disk.

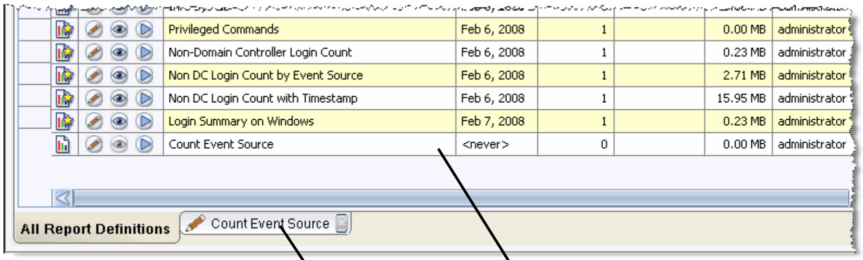
Running and Viewing the Report

After you name your report, identify its namespace, and enter the SQL query, it is a good time to test your report by running it. When you run the report, you can keep your definition window open. To run the report, click the **All Report Definitions** tab that remains open as the leftmost tab while you are in Reports mode.

If the **All Report Definitions** tab remains hidden because multiple tabs display at the bottom of your window, use the page-scroll ◂ ▸ ▾ icon, as documented in “Changing Focus to a Page”, on page 32.

Figure 5-4 shows the new “Count Event Source” report displayed in the All Report Definitions list. The definition for this report remains open and accessible for editing. The **Latest Report Run** column indicates that the report has not yet been run.

Figure 5-4: Running a SQL Report While Its Definition Remains Open for Editing



Privileged Commands	Feb 6, 2008	1		0.00 MB	administrator
Non-Domain Controller Login Count	Feb 6, 2008	1		0.23 MB	administrator
Non DC Login Count by Event Source	Feb 6, 2008	1		2.71 MB	administrator
Non DC Login Count with Timestamp	Feb 6, 2008	1		15.95 MB	administrator
Login Summary on Windows	Feb 7, 2008	1		0.23 MB	administrator
Count Event Source	<never>	0		0.00 MB	administrator

New SQL report definition remains open for editing

New SQL report displays with all other report definitions

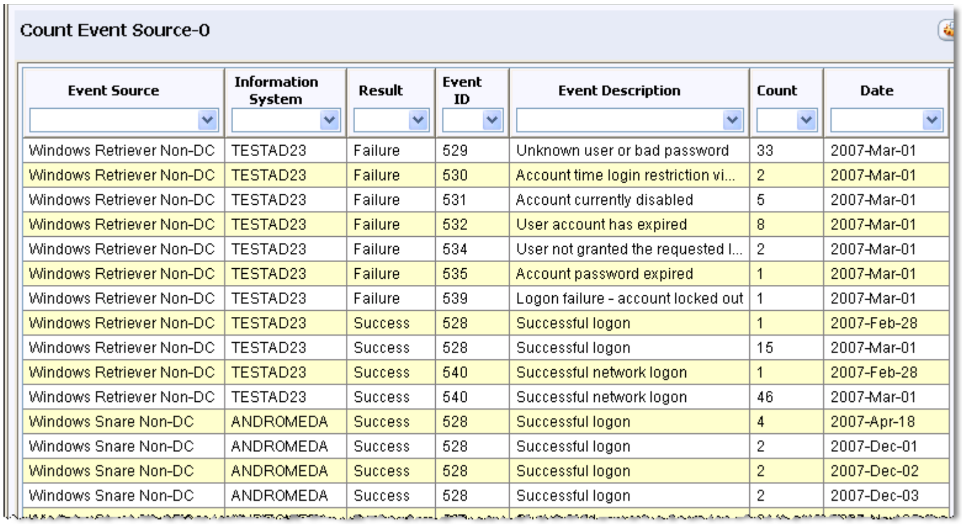
To test the new report, click the ▷ icon or click **Run** from the workspace Action Menu. A dialog displays that enables you to specify the time range and time zone and to change the namespace. After you make your changes in this dialog, click **Run**.

For more information, see “Running a Report”, on page 100.

When processing complete, the **Run Report Status** dialog provides the 👁 icon, which enables you to view the report. This icon is also available from the **Actions** column of the All Report Definitions list. Click this icon to verify that your report displays as desired.

Figure 5-5 illustrates the results of running the “Count Event Source” report.

Figure 5-5: Viewing the Report Results



Count Event Source-0

Event Source	Information System	Result	Event ID	Event Description	Count	Date
Windows Retriever Non-DC	TESTAD23	Failure	529	Unknown user or bad password	33	2007-Mar-01
Windows Retriever Non-DC	TESTAD23	Failure	530	Account time login restriction vi...	2	2007-Mar-01
Windows Retriever Non-DC	TESTAD23	Failure	531	Account currently disabled	5	2007-Mar-01
Windows Retriever Non-DC	TESTAD23	Failure	532	User account has expired	8	2007-Mar-01
Windows Retriever Non-DC	TESTAD23	Failure	534	User not granted the requested l...	2	2007-Mar-01
Windows Retriever Non-DC	TESTAD23	Failure	535	Account password expired	1	2007-Mar-01
Windows Retriever Non-DC	TESTAD23	Failure	539	Logon failure - account locked out	1	2007-Mar-01
Windows Retriever Non-DC	TESTAD23	Success	528	Successful logon	1	2007-Feb-28
Windows Retriever Non-DC	TESTAD23	Success	528	Successful logon	15	2007-Mar-01
Windows Retriever Non-DC	TESTAD23	Success	540	Successful network logon	1	2007-Feb-28
Windows Retriever Non-DC	TESTAD23	Success	540	Successful network logon	46	2007-Mar-01
Windows Snare Non-DC	ANDROMEDA	Success	528	Successful logon	4	2007-Apr-18
Windows Snare Non-DC	ANDROMEDA	Success	528	Successful logon	2	2007-Dec-01
Windows Snare Non-DC	ANDROMEDA	Success	528	Successful logon	2	2007-Dec-02
Windows Snare Non-DC	ANDROMEDA	Success	528	Successful logon	2	2007-Dec-03

NOTE: To delete a SQL report definition, close the definition window and delete the report from the All Report Definition table. For more information, see “Deleting Report Definitions”, on page 118.

EDITING THE SQL REPORT

After you run and test your report, you can refine it further by editing it. HawkEye AP Console provides two ways to edit a report:

- Batch editing

  Use this mode when you want to make the same change to the default time range, time zone, and/or namespace for multiple reports. For more information, see "Batch Editing Multiple Reports for Time Range, Time Zone, and Namespace", on page 151.

- Single editing

  Use this mode to make changes to a report's content or display, or to associate a report to another report or to a security alert. This section documents these options in detail.

As illustrated in Figure 5-2, the window for creating and editing a single SQL report has five tabs:

- **Data Source**—You have already used this tab to name and describe the report and to select its namespace; see "Creating a SQL Report", on page 177.
- **SQL**—You have already used this tab to enter your query. You can also use this tab to:
  - Edit the SQL to provide the run-time user with options to restrict the data returned and a choice of views or tables against which to run the report; for more information, see "Parameterizing Your Query", next.

    NOTE: If you change the query in any way, a dialog displays that informs you that, if you have modified column display or defined a chart for the report, you will lose all your column-display settings and the chart definition. You will be prompted to continue or cancel changing the report structure.
  - Link a library to your query; for more information, see "Adding a Library to Your Query", on page 195.
- **Criteria**—You have already used this tab to specify the default time range. You can also use this tab to create criteria rows that allow the user to select or specify values at runtime. For more information, see "Specifying a Time Range", on page 180 and "Parameterizing Your Query", next.
- **Properties**—This tab provides the same functionality for both Wizard and SQL reports; for information, see "Associating a Report to Another Report", on page 161, "Associating a Report to a Security Alert", on page 164, and "Specifying Print Options and Maximum Rows", on page 166.
- **Column Display**—This tab provides the same functionality for both Wizard and SQL reports; for information, see "Formatting Columns", on page 167.
- **Chart**—This tab provides the same functionality for both Wizard and SQL reports; for information, see "Defining Charts", on page 171.

NOTE: Click **Revert** to undo all changes made since the last time you saved the report. When you revert changes, all changes made in any or all tabs revert to their status as of the last save.

This section contains the following topics:

- "Parameterizing Your Query", next

- “Adding a Library to Your Query”, on page 195
- “Creating a Library”, on page 197

Parameterizing Your Query

“Creating a SQL Report”, on page 177, describes the initial steps to create a report. This section describes how to edit the initial report to provide the user with run-time options.

This section covers the following topics:

- “Background”, next
- “Adding Parameters to the SQL Query”, on page 185
- “Creating Parameter Fields That Display to the User”, on page 187
- “Running and Testing the Parameterized SQL Report”, on page 189
- “Adding a Parameter to the HAVING Clause”, on page 190
- “Adding a Parameter to the FROM Clause”, on page 191
- “Relevant Documentation”, on page 195

Background

A parameterized report enables users to specify filter criteria at runtime that customizes report output. The user does not need to directly edit the underlying SQL. The user also does not need to run a report that returns all values and requires the user to filter them in the HawkEye AP Console (see “Filtering and Sorting Report Data”, on page 46).

You can also create a parameterized report that provides run-time users with a choice of data sources against which to run the report.

Steps to Parameterizing a SQL Report

1 Use the **SQL** tab to add parameter(s) to the query.

See “Adding Parameters to the SQL Query”, on page 185.

2 Use the **Criteria** tab to add criteria row(s) to the Column Criteria pane.

- Each row represents a single parameter.
- Each row identifies the parameter name, specifies a label that clarifies the value choice, and provides the value(s) offered to the user.

  NOTE: The parameter name identified in each row must be identical to the one used in the query, but do not include the dollar sign ($).

See “Creating Parameter Fields That Display to the User”, on page 187.

3 Run and test the report with the parameters.

See “Running and Testing the Parameterized SQL Report”, on page 189.

Adding Parameters to the SQL Query

HawkEye AP Console provides SQL parameters as a simplified way to add macros to your query. As documented in “Macros”, on page 302, a *macro* is a processing directive that declares a constant value for use within a SELECT statement. Macro definitions have the following syntax:

```
WITH $<id> AS <value> [OVERRIDE][, $<id> AS <value> [OVERRIDE] [...]]
```

The SQL query engine replaces each occurrence of `$<id>` after the declaration with the constant value. The values of macros remain constant throughout the execution of SELECT statements. Macros are similar to literal constants, not programming variables. You declare the constant value in one location, and the value is used wherever the macro identifier is encountered in the remainder of the SELECT statement.

When you define a SQL parameter in HawkEye AP Console, you do not use the macro syntax shown above. Instead, you declare the macro as a parameter in the **Criteria** tab. However, you reference the parameter in the query just as you would reference the macro. In other words, you can use a parameter everywhere in a SQL statement that accepts a string *value*, but not everywhere that accepts a string. Typically this means that you include a parameter in a FROM, WHERE, or HAVING clause, but not in other clauses, such as SELECT or GROUP BY. For more information on parameter restrictions, see “Adding a Parameter to the HAVING Clause”, on page 190

Like a macro, you reference a parameter in a query by including it in an expression as a constant value and preceding its name with a dollar sign (`$`). For example, the following WHERE clause references the `EVENT_DESCRIP` parameter, which represents a value in the **Event Description** column:

```
WHERE event_description = $EVENT_DESCRIP
```

The following example query illustrates two parameters in its WHERE clause.

```
SELECT
   _timef("%Y-%b-%d",ts) as "Date",
   log_type AS "Event Source",
   info_sys AS  "Information System",
   result AS "Result",
   event_id AS "Event ID",
   event_description AS "Event Description",
   count (*) AS "Count"
FROM
   userLogin__windows__nonDomainController
WHERE
   log_type LIKE ('%' + $EVENT_SOURCE + '%')
   and
   event_description = $EVENT_DESCRIP
GROUP BY
   1,2,3,4,5,6
DURING ALL
```

The **boldface** text in the query above highlights the statements that include a parameter. The first WHERE clause statement, which uses the LIKE comparison operator, includes the parameter in a string expression.

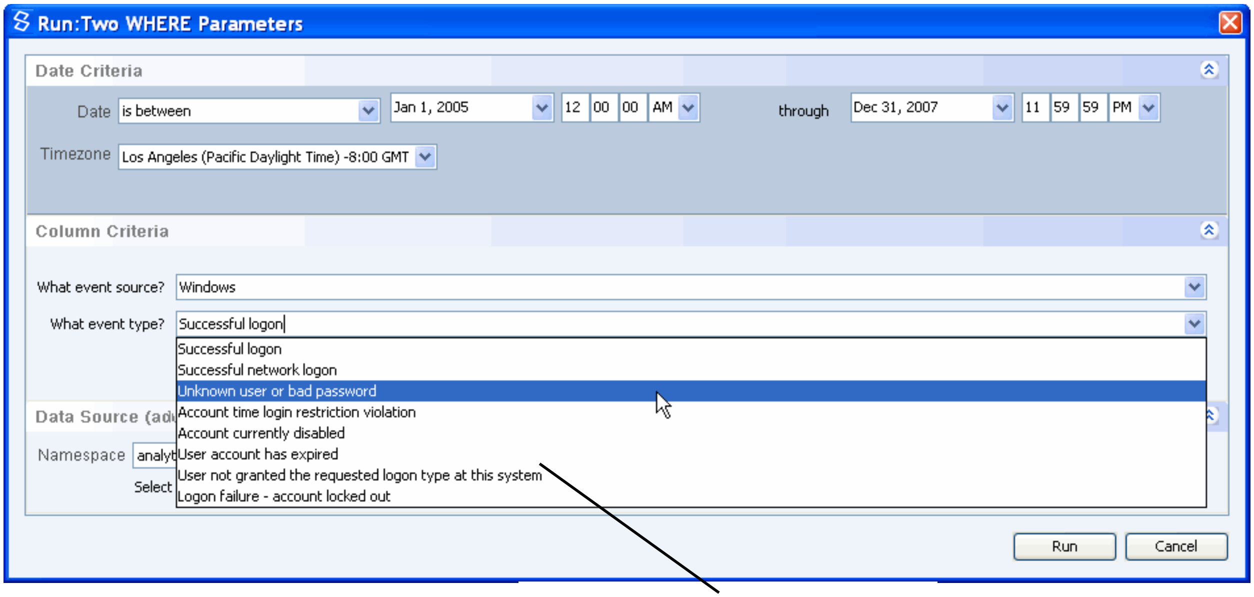
NOTE: To simplify the documentation, the first WHERE clause statement uses the LIKE operator, which is a standard SQL operator. To enhance performance, Sensage recommends that you use its `_strstr()` function instead. The following version of the WHERE clause statement maximizes performance:

```
WHERE
   _strstr(log_type,$EVENT_SOURCE) >-1
   and
   event_description = $EVENT_DESCRIP
```

For more information, see “_strstr()”, on page 358.

Figure 5-6 illustrates how these two parameters display at run time.

Figure 5-6: Displaying Two WHERE Clause Parameters at Run Time



Run:Two WHERE Parameters

Date Criteria

Date: is between | Jan 1, 2005 | 12 | 00 | 00 | AM | through | Dec 31, 2007 | 11 | 59 | 59 | PM

Timezone: Los Angeles (Pacific Daylight Time) -8:00 GMT

Column Criteria

What event source? Windows

What event type? Successful logon

Successful logon
Successful network logon
Unknown user or bad password
Account time login restriction violation
Account currently disabled
User account has expired
User not granted the requested logon type at this system
Logon failure - account locked out
Data Source (ad
Namespace
analyt
Select
Run
Cancel

Select desired value from dropdowns

Unacceptable and Problematic Usage

Unacceptable Usage

You cannot use a parameter to represent a column name or to rename a column:

- Column-name usage differs from table-name usage in that you *cannot* use a parameter to represent a column name. Unlike a table name, a column name must always be specified as an identifier and not as a string expression. Therefore, the following statement is unacceptable:

  ```
  SELECT $COLUMN_ONE, $COLUMN_TWO, $COLUMN_THREE
  ```

- Additionally, although you follow the `AS` keyword in a SELECT clause with a string to rename the result column, you cannot replace this string with a parameter. The following syntax is *not* allowed because the parameter represents a string rather than a string value:

  ```
  SELECT info_sys AS $MEANINGFUL_NAME
  ```

The next topic describes how to create parameters in the **Criteria** tab.

Problematic Usage

If you parameterize a varchar value that contains timestamp data, you must understand the way the data formats the timestamp. For example, assume you parameterize the **WINTS** column of the `microsoft_windows_securityEvent_snare` view. To provide your users with useful values in the parameter-value field, you must know how the date displays. For example, it could display as `2008/05/15` or as `2008-05-15`.

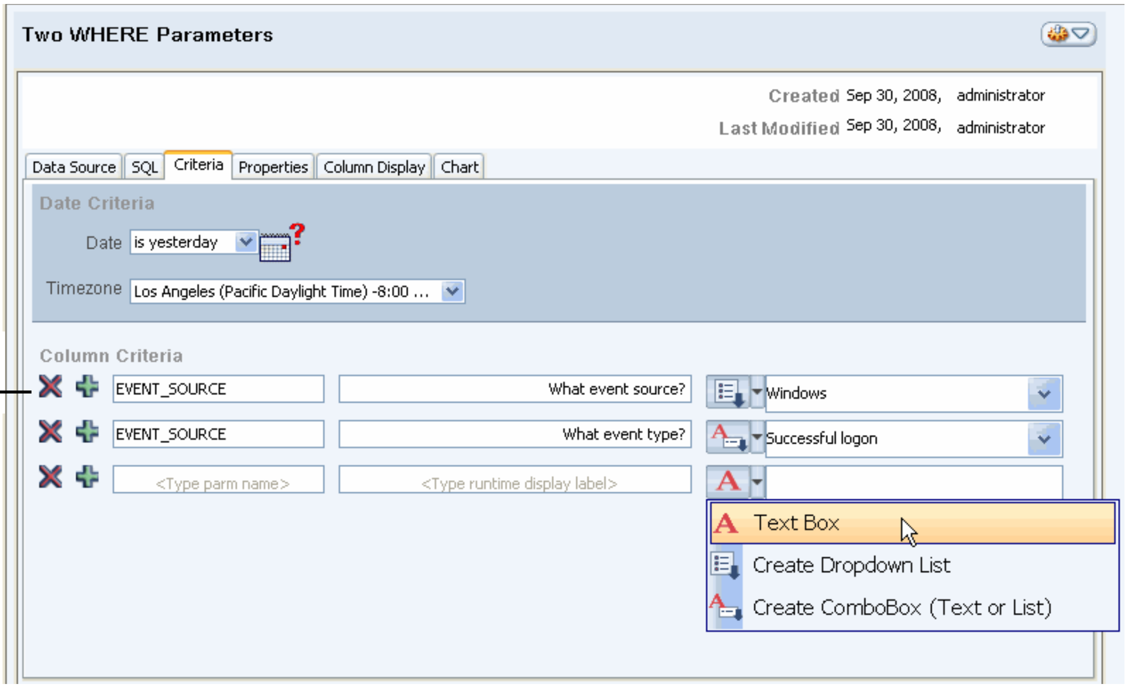
Sensage recommends that you first run the report without the parameter to determine the correct format. Then create parameter values that match the style of the actual date text.

NOTE: This same issue also applies to parameters in Wizard reports.

Creating Parameter Fields That Display to the User

Figure 5-7 illustrates the **Criteria** tab in which you specify date criteria and define report parameters. Specifying date criteria has been described in "Specifying a Time Range", on page 180.

Figure 5-7: Criteria Tab for a SQL Report



Parameter-definition fields

Two WHERE Parameters

Created Sep 30, 2008, administrator
Last Modified Sep 30, 2008, administrator

Data Source | SQL | Criteria | Properties | Column Display | Chart

Date Criteria

Date: is yesterday

Timezone: Los Angeles (Pacific Daylight Time) -8:00 ...

Column Criteria

EVENT_SOURCE	What event source?	Windows
EVENT_SOURCE	What event type?	Successful logon
<Type parm name>	<Type runtime display label>	

The graphic above shows the window in which the two WHERE-clause parameters were defined. It also illustrates the dropdown that displays the three field styles available for parameter values.

Creating a parameter requires you to specify values in the Column Criteria fields illustrated above:

- The text in the left field instructs you to enter the name of the parameter.

  The parameter's name can be a combination of alphanumeric characters and underscores. Spaces are not allowed.

  TIP: When you refer to the parameter within the SQL statement, you must precede its name with a dollar sign ($). However, you should not include the dollar sign when you identify the parameter in its column-criteria field. For more information, see “Adding Parameters to the SQL Query”, on page 185.

- The text in the middle field instructs you to enter a run-time display label.

  Make the label as meaningful as possible so that run-time users will understand the choices they are making. Figure 5-7 illustrates display labels for two fields:

  - `What event source?`
  - `What event type?`

- The field on the right provides three field styles, from which you select one to display column values. The style you select determines your users’ options at runtime:

  - **Text Box** A—Allows run-time users to enter a value

    You enter a single value to display as the default value. The user can overwrite this value at run time.

  - **Dropdown List** ☰—Provides a fixed list of values to run-time users

    You enter multiple values and select one to display as the default value.

  - **ComboBox** A—Provides an editable list of values to run-time users

    You enter multiple values and select one to display as the default value.

    NOTE: This field is like the text box in that the interface allows you to keep the field empty, but doing so causes the same problems described above.

NOTE: The column-values field does not accept a wildcard character at runtime. In other words, a user cannot enter a percent sign (`%`) to represent and retrieve characters that are not displayed. However, you can include the percent sign in your SQL query to provide this flexibility to your users. For example, assume your event source includes `Windows Retriever Non-DC` and `Windows Snare Non-DC`. You can enter the full names of these event sources in a dropdown or you can simply specify `Retriever`, `Snare`, and `Windows`. If you provide the shortened names, you must include the LIKE operator in your query. For an example of such a query, see “Adding Parameters to the SQL Query”, on page 185.

You can specify parameters that represent values in any number of columns in the report. To add more parameters, click the ✚ icon. To delete a parameter, click the ✖ icon.

NOTE: There are syntactic limitations as to which values can be represented by a parameter. For more information, see “Adding Parameters to the SQL Query”, on page 185.

To create a parameter field

1. Enter a parameter name in the left field.

   The name you enter will not appear to the user but will be included in the SQL query.

   Sensage recommends that you enter the name entirely in uppercase letters so that it is clearly identified as a parameter name. Also make the name meaningful so that its usage will be clear in the SQL query.

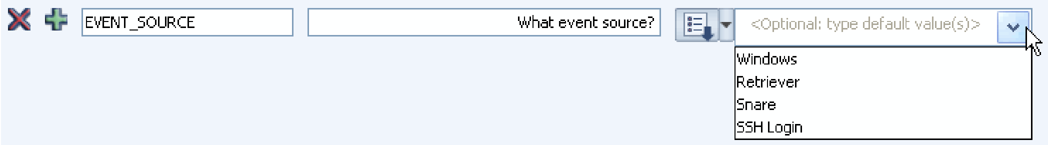
2. Enter a meaningful label in the middle field.

The name you enter does appear to the user and makes meaningful the value selection.

3 Click the field-style icon A ▾ (illustrated in Figure 5-7) to display the three style types, and select the desired style.

4 To create a Dropdown List or ComboBox or Text Box with a default value, enter valid text in the field and press ENTER.

- For a Text Box, your specification for the field is complete.
- For a Dropdown List or ComboBox, the field empties. To view the text, click the dropdown icon, as shown below.



EVENT_SOURCE

What event source?

<Optional: type default value(s)>

Windows
Retriever
Snare
SSH Login

NOTE: You cannot delete a value that you enter accidentally in a Dropdown or ComboBox. You must delete the entire parameter and recreate it. To delete a parameter, click the ✕ icon.

5 To add values to a Dropdown List or ComboBox, enter additional text value(s) into the field, pressing ENTER after each one.

NOTE: Although the interface allows you to keep the Text Box and ComboBox fields empty, doing so does not return all values. Instead, it raises two issues:

- The SQL engine interprets the empty field as an empty string and returns only rows that contain an empty string in the corresponding column. Therefore, the user should always enter or select a value at run time.
- The schedule interface allows you to add the report to a schedule *only* if all parameters have a default value. Because you schedule reports that run independently of user interaction, it is good practice to provide the default value for reports you will schedule.

To create additional parameter fields

1 To add parameters, click the ✚ icon once for each parameter.

2 Repeat the procedure described above for each parameter.

You have created the fields that display to the user. This step is only half of the process. You must also reference each parameter in the SQL query. The order of these two steps is irrelevant. In other words, you can create the query first and then create the fields.

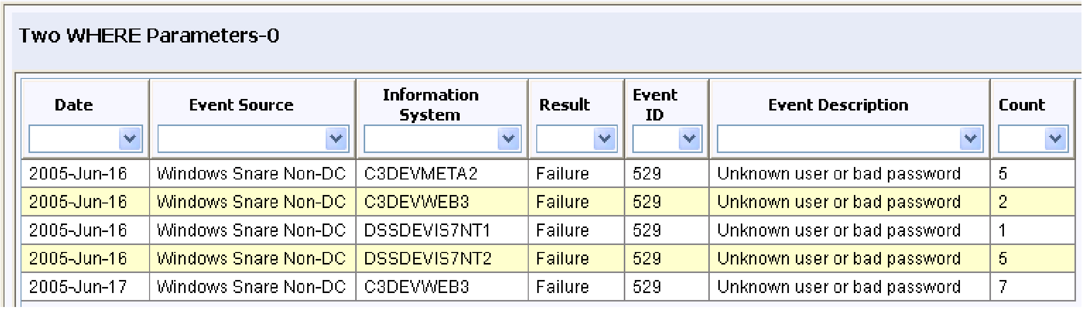
Running and Testing the Parameterized SQL Report

Figure 5-6 illustrates a **Run** window for a SQL report that provides parameterized criteria for the two columns referenced in the example query. At run time, users can specify event source and event type separately from the two dropdowns. Alternately, users can run the report on the default values or enter their own values.

The SQL query engine substitutes the values selected by the user for the two parameters specified in the example WHERE clause.

The report created earlier, displayed in Figure 5-5, returned hundreds of rows that count every type for every event source. The report output displayed in Figure 5-8 returned only five rows, which count only one event type (Unknown user or bad password) for only one event source (Windows Snare Non-DC). This report limited output by offering the user parameterized choices for these two fields. The user's response is evaluated in two different WHERE clause statements.

Figure 5-8: Limited Report Output: WHERE Clauses



Two WHERE Parameters-0

Date	Event Source	Information System	Result	Event ID	Event Description	Count
2005-Jun-16	Windows Snare Non-DC	C3DEVMETA2	Failure	529	Unknown user or bad password	5
2005-Jun-16	Windows Snare Non-DC	C3DEVWEB3	Failure	529	Unknown user or bad password	2
2005-Jun-16	Windows Snare Non-DC	DSSDEVIS7NT1	Failure	529	Unknown user or bad password	1
2005-Jun-16	Windows Snare Non-DC	DSSDEVIS7NT2	Failure	529	Unknown user or bad password	5
2005-Jun-17	Windows Snare Non-DC	C3DEVWEB3	Failure	529	Unknown user or bad password	7

Adding a Parameter to the HAVING Clause

The next example query adds a third parameter, which dynamically determines the content of the query HAVING clause. The HAVING clause uses the value of the **Count** column to limit the groups returned.

```
SELECT
   _timef("%Y-%b-%d",ts) as "Date",
   log_type AS "Event Source",
   info_sys AS  "Information System",
   result AS "Result",
   event_id AS "Event ID",
   event_description AS "Event Description",
   count (*) AS "Count"
FROM
   userLogin__windows__nonDomainController
WHERE
   log_type LIKE ('%' + $EVENT_SOURCE + '%')
   and
   event_description = $EVENT_DESCRIP
GROUP BY
   1,2,3,4,5,6
HAVING
   count(*) > _int32($HAVING_FILTER)
DURING ALL
```

IMPORTANT: There are a few restrictions on where and how you can specify a parameter:

- EDW treats every parameterized value as a string literal.

  In other words, as it interprets each value, the EDW automatically encloses it within quotation marks. Because of this interpretation, you can parameterize only string values or you must use a Sensage SQL conversion function to convert the value to an integer.

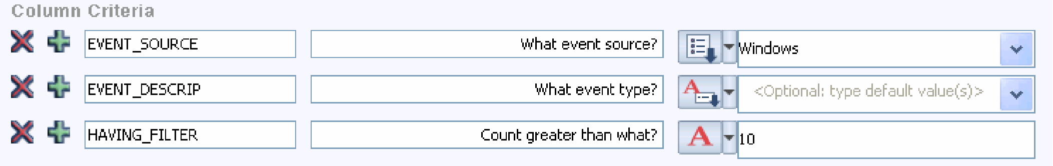
NOTE: Because you can represent a table name as a string expression in a FROM clause, you can use a parameter to specify a table name. For more information see, "Table Specifications in FROM Clauses", on page 273.

- A further restriction is that each parameterized value must represent an actual value. For example, the parameter can represent a data value stored in the EDW or a value interpreted by the WHERE clause. It cannot, however, represent a string such as the one that renames a column in the SELECT clause.

The example HAVING clause above evaluates the value returned by the `count()` function and compares it to a numeric value entered by the user at run time. Because the EDW treats the parameterized value as a string literal, the example clause converts the parameterized value to an 32-bit integer.

Figure 5-9 illustrates specification of a third parameter, a Text Field, that allows the user to specify the minimum value displayed in the **Count** column. The Text Field provides a a single value that users can overwrite at run time.

Figure 5-9: Specifying Parameters for a HAVING Clause

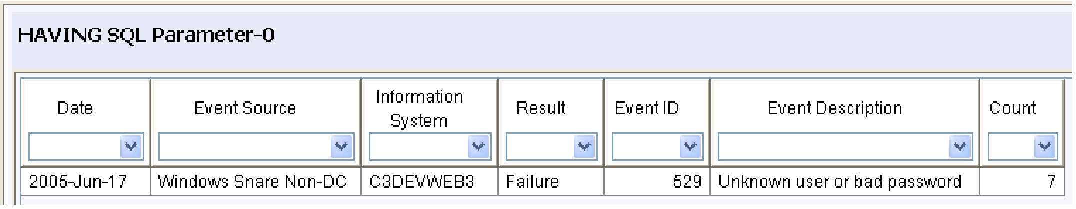


Column Criteria

EVENT_SOURCE	What event source?	Windows
EVENT_DESCRIP	What event type?	<Optional: type default value(s)>
HAVING_FILTER	Count greater than what?	10

As illustrated in Figure 5-10, the example user's input limits the report output to a single row. This report limited output by offering the user a parameterized choice for the **Count** column. The user's response is evaluated in a HAVING clause that examines the count values for each group and eliminates rows that fall below the value specified at runtime by the user.

Figure 5-10: Limited Report Output: HAVING Clause



HAVING SQL Parameter-0

Date	Event Source	Information System	Result	Event ID	Event Description	Count
2005-Jun-17	Windows Snare Non-DC	C3DEVWEB3	Failure	529	Unknown user or bad password	7

Adding a Parameter to the FROM Clause

The last query example adds a fourth parameter, this one to the FROM clause. At run time, the user can dynamically determine the report data source by selecting a value from this parameter.

To provide a choice of data sources requires that each data source to be structured similarly with identical column names and data types. As documented in "Namespaces: Using a Single Report or Dashboard to Access Different Data", on page 43, HawkEye AP provides namespaces, which allow administrators to create identically named tables with identical structures that store different data depending on their location. One advantage this feature offers is the ability to create a single report that retrieves different data from a single table whose data depends on the namespace that contains the table.

A parameterized FROM clause provides similar flexibility *within* a namespace. For a parameterized FROM clause, however, you do not name the tables identically. Instead, you structure them similarly. Every column that the report retrieves must be named identically and have the same data type in all tables that you offer to the run-time user.

The views included with HawkEye AP Analytics provide a good example of this data-source flexibility. Analytics views normalize the data by enforcing consistent use of column names, data types, and formats, and by consistently presenting disparate event data that indicates the same information.

For example, Analytics provides event-type views for login events from Windows systems that use domain controllers and event-type views for login events from *all* Windows systems. These views contain columns that represent data specific to a grouping as well as columns whose data is common to the event type, such as a user account column. For more information, see the *Analytics Guide*.

The example query below adds a parameter to the FROM clause, which allows run-time users to select the desired data source.

```
SELECT
   _timef("%Y-%b-%d",ts) as "Date",
   log_type AS "Event Source",
   info_sys AS  "Information System",
   result AS "Result",
   event_id AS "Event ID",
   event_description AS "Event Description",
   count (*) AS "Count"
FROM
   $TABLE_NAME
WHERE
   log_type LIKE ('%' + $EVENT_SOURCE + '%')
   and
   event_description = $EVENT_DESCRIP
GROUP BY
   1,2,3,4,5,6
HAVING
   count(*) > _int32($HAVING_FILTER)
DURING ALL
```

NOTE: Because you can specify the data source in the FROM clause as an identifier, literal, or expression that evaluates to the name of a table in the EDW, you can also include the FROM clause parameter in a string expression. For example, the following is legal use of a parameter:

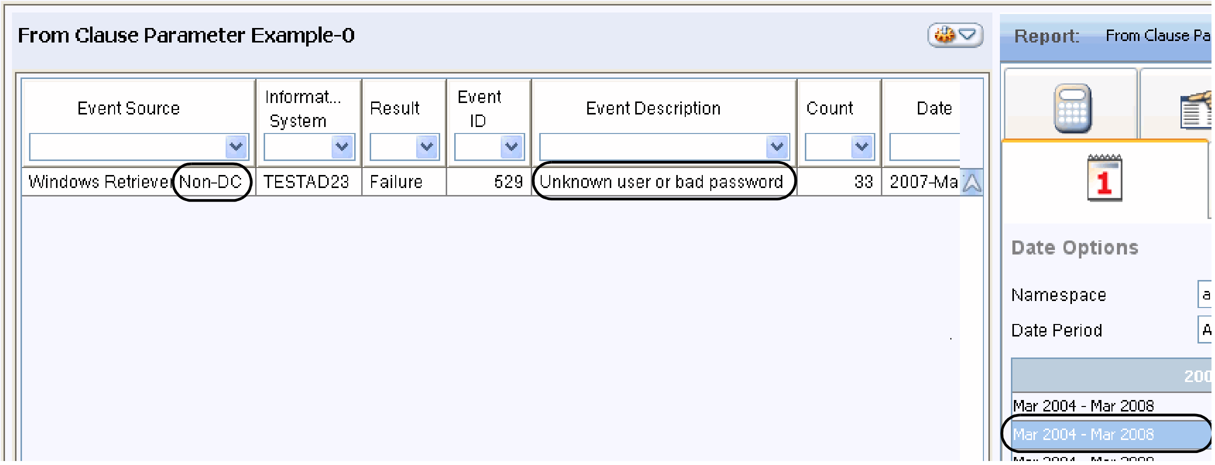
```
FROM "MyNamespace" + "." + $TABLE_NAME
```

For more information, see “Table Specifications in FROM Clauses”, on page 273.

The reports displayed in Figure 5-11 and Figure 5-12 illustrate one report run against two different data sources: `userLogin__windows__nonDomainController` view and `userLogin__windows__DomainController` view. The red ovals in the graphics highlight significant differences in the data.

For example, Figure 5-11 highlights the event source as `Non-DC`. It also highlights the text of the event description and the cache that produced the results shown. This report displays the results in the second cache.

Figure 5-11: Results for userLogin__windows__nonDomainController View

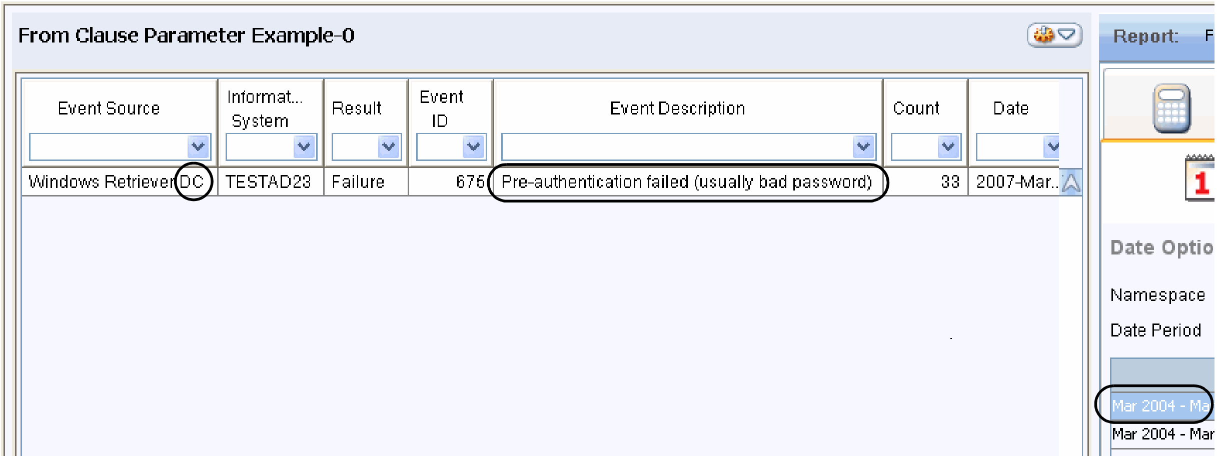


From Clause Parameter Example-0

Event Source	Informat... System	Result	Event ID	Event Description	Count	Date
Windows Retriever Non-DC	TESTAD23	Failure	529	Unknown user or bad password	33	2007-Ma

Figure 5-12 highlights the event source as `DC`. It also highlights the text of a different event description and the cache that produced the results shown. This report displays the results in the first cache.

Figure 5-12: Results for userLogin__windows__DomainController View



From Clause Parameter Example-0

Event Source	Informat... System	Result	Event ID	Event Description	Count	Date
Windows Retriever DC	TESTAD23	Failure	675	Pre-authentication failed (usually bad password)	33	2007-Mar..

NOTE:

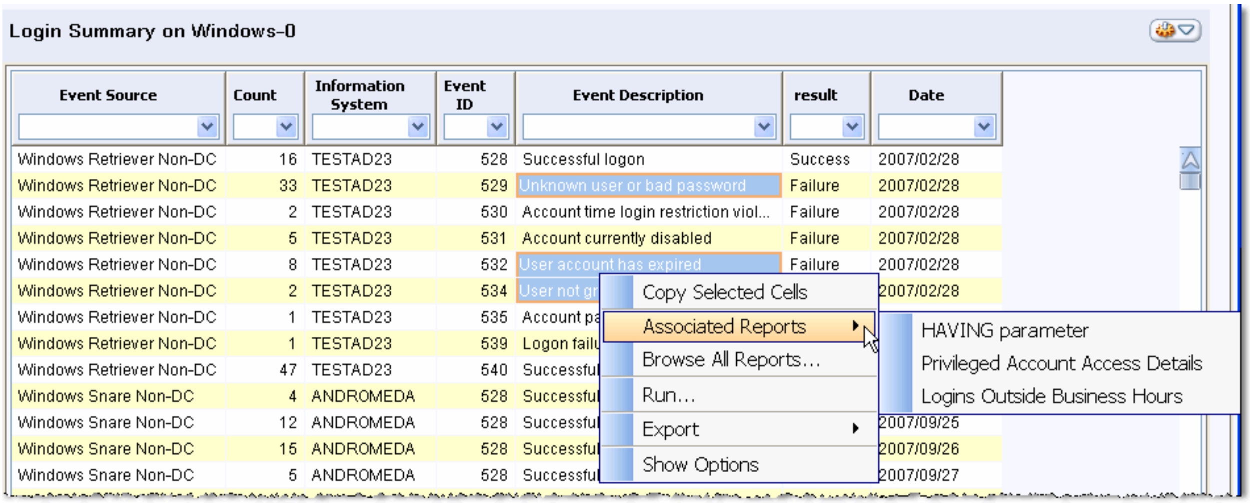
- The event-type and event-description data returned by the two views differs. Therefore, if a report contains a parameterized FROM clause in addition to other parameters, you must ensure that all parameters contain values relevant to all data-source choices.
- When you create the parameter field for the FROM-clause value, provide a default value. If the user runs the report without specifying a value, the report will not run successfully.

Defining Parameters for an Associated Report

As documented in “Associating Reports to a Report or Security Alert”, on page 161, you can associate a report to another report and to a security alert. At run time, the user selects value(s) to examine from one of the columns in a report or security alert and then selects a pre-defined associated report to display one value at a time.

Figure 5-13 illustrates a user selecting three values from the **Event Description** column and right-clicking to display the popup. From the popup, the user chooses to run one of three associated reports. After the user selects the associated report, its **Run** dialog displays

Figure 5-13: Selecting Multiple Column Values for an Associated Report



Login Summary on Windows-0

Event Source	Count	Information System	Event ID	Event Description	result	Date
Windows Retriever Non-DC	16	TESTAD23	528	Successful logon	Success	2007/02/28
Windows Retriever Non-DC	33	TESTAD23	529	Unknown user or bad password	Failure	2007/02/28
Windows Retriever Non-DC	2	TESTAD23	530	Account time login restriction viol...	Failure	2007/02/28
Windows Retriever Non-DC	5	TESTAD23	531	Account currently disabled	Failure	2007/02/28
Windows Retriever Non-DC	8	TESTAD23	532	User account has expired	Failure	2007/02/28
Windows Retriever Non-DC	2	TESTAD23	534	User not gr		2007/02/28
Windows Retriever Non-DC	1	TESTAD23	535	Account pa		
Windows Retriever Non-DC	1	TESTAD23	539	Logon failu		
Windows Retriever Non-DC	47	TESTAD23	540	Successful		
Windows Snare Non-DC	4	ANDROMEDA	528	Successful		
Windows Snare Non-DC	12	ANDROMEDA	528	Successful		2007/09/25
Windows Snare Non-DC	15	ANDROMEDA	528	Successful		2007/09/26
Windows Snare Non-DC	5	ANDROMEDA	528	Successful		2007/09/27

The data that the user selects from the source report or alert populates the column criteria row of the associated report’s **Run** dialog. If the user has selected more than one value to examine, she must select one of them from the appropriate dropdown and click **Run** to run the associated report on that value. The associated report returns only rows that pertain to the selected value for the selected column.

NOTE:

- Because the user selects data to examine from one of the columns in the source report and the selected data populates the parameter of the associated report, every report you design as an associated report should include a parameter to contain the user’s selected value(s).
- A parameterized Wizard report allows users to modify the parameters in the **Run** dialog of the associated report. Users can delete unwanted parameters or select a different column value for the parameter. For more information, see “Running a Report”, on page 100, “Opening an Associated Report”, on page 54, and “Associating a Report to Another Report”, on page 161.
- A parameterized SQL report does *not* allow users to modify the parameters in the **Run** dialog of the associated report. You should consider this limitation when you create a parameterized report as an associated report. You might consider the following options:

You can OR your parameters together so that the query engine interprets only the parameter that contains a value in the **Run** dialog. The query example in “Adding Parameters to the SQL Query”, on page 185, illustrates two WHERE-clause parameters. The query example below has modified that query to use the OR operator rather than the AND operator. This modification allows the run-time user to limit report results by only one column value rather than by two.

```
SELECT
   _timef("%Y-%b-%d",ts) as "Date",
   log_type AS "Event Source",
   info_sys AS  "Information System",
   result AS "Result",
   event_id AS "Event ID",
   event_description AS "Event Description",
   count (*) AS "Count"
FROM
   userLogin__windows__nonDomainController
WHERE
   log_type LIKE ('%' + $EVENT_SOURCE + '%')
   OR
   event_description = $EVENT_DESCRIP
GROUP BY
 1,2,3,4,5,6
DURING ALL
```

- You use the Report Wizard to create flexible associated reports.

Relevant Documentation

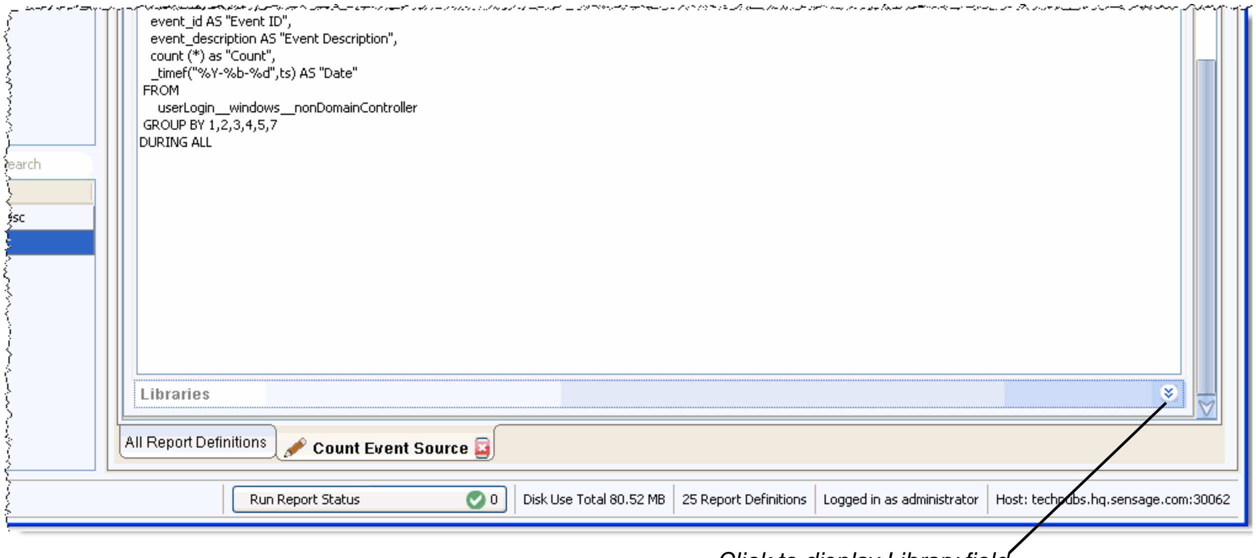
- Processing directives that declare constant values that can be used within a SELECT statement—“Macros”, on page 302.
- String literals—“String Literals”, on page 291
- HawkEye AP operators—“Operators”, on page 293
- HawkEye AP conversion functions that convert a string value to an integer—“Conversion Expressions”, on page 299.

Adding a Library to Your Query

HawkEye AP enables developers to create libraries, which allow common SQL fragments and Perl code to be shared across queries. For example, a library can provide you with easy-to-remember column names. Or, if your reports tend to involve the same subset of columns, a library can provide the specified subset of columns.

Figure 5-2 illustrates the window in which you enter the SQL query. Hidden at the bottom of this window is the Library field. To display this field, click the expand icon, as illustrated in Figure 5-14.

Figure 5-14: Displaying the Libraries Field



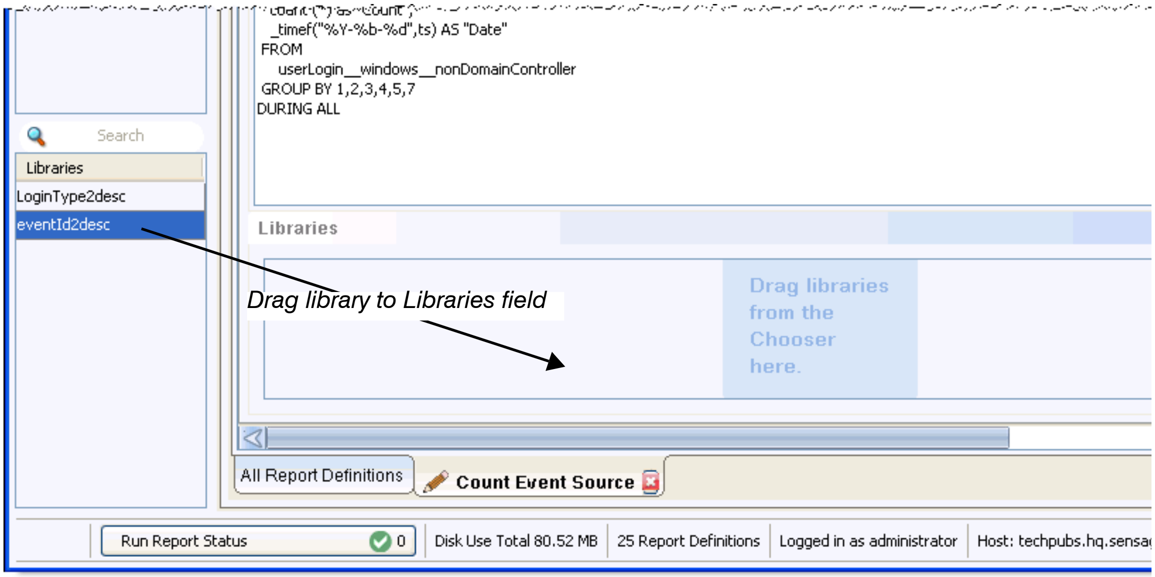
```
  event_id AS "Event ID",
  event_description AS "Event Description",
  count (*) as "Count",
  _timef("%Y-%b-%d",ts) AS "Date"
 FROM
   userLogin__windows__nonDomainController
 GROUP BY 1,2,3,4,5,7
DURING ALL
```

Click to display Library field.

After you expand the Libraries field, you can drag the desired library or libraries to it, as illustrated in Figure 5-15.

Figure 5-15 illustrates the process to add a library to a SQL report.

Figure 5-15: Adding a Library to a SQL Report



```
  count (*) as "Count",
  _timef("%Y-%b-%d",ts) AS "Date"
 FROM
   userLogin__windows__nonDomainController
 GROUP BY 1,2,3,4,5,7
DURING ALL
```

Drag library to Libraries field

NOTE: Before you can add the library, someone must create it in Administration mode. For more information, see "Creating a Library", next.

Creating a Library

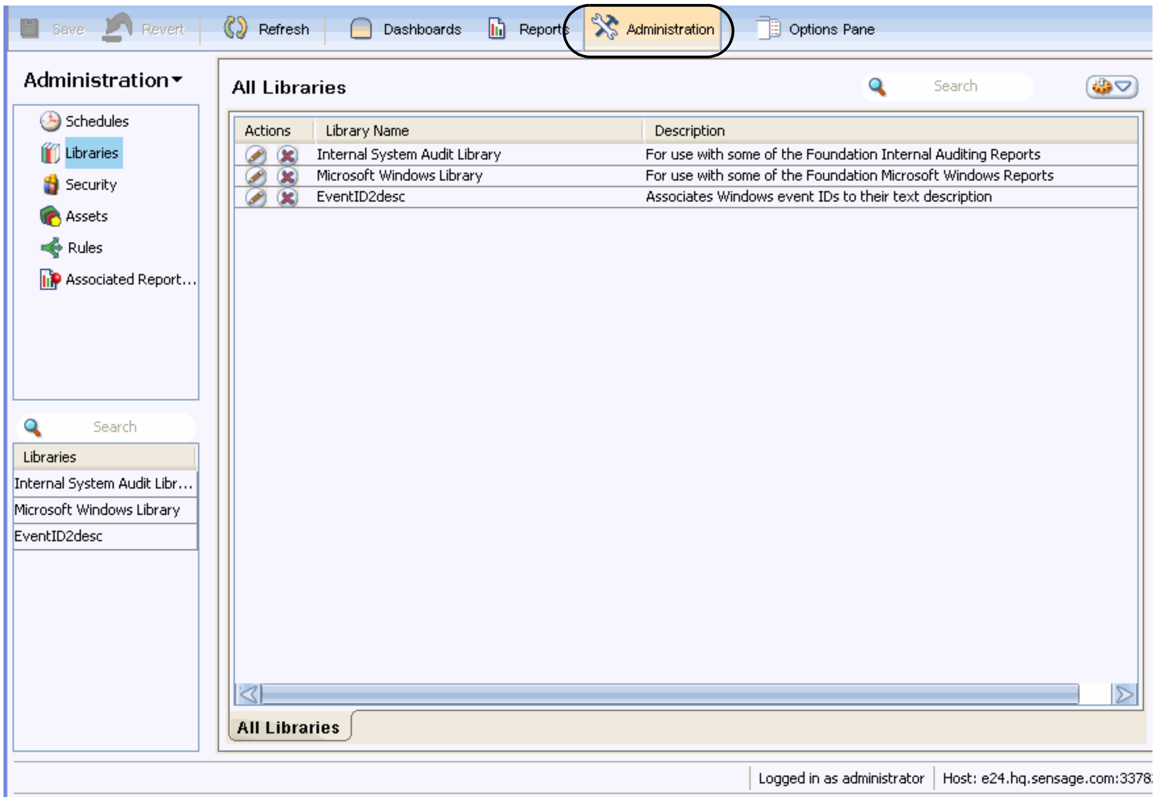
In many cases you will have common SQL fragments that should be identical across many queries. These fragments may include logic to categorize records according to business rules, special business-specific functions such as custom string parsers and decoders, and lists of excluded strings or loads.

To minimize maintenance and ensure consistency, HawkEye AP provides the ability to define SQL syntax common to many queries once and maintain the SQL block centrally as a library. For example, to provide your report developers with easy-to-remember column names, you can create a library that provides each column with an alias by using the "`WITH` *`column_name`* `AS` *`target_name`*" clause. You can also define Perl functions centrally and apply them to queries.

If your reports tend to involve the same subset of columns, consider making a library that contains the specified subset of columns. Such a library enables a report developer to create a query and reference multiple columns at once. This feature is especially useful when the query writer is not familiar with the column names. Additionally, it allows for central control of the target column names and allows an administrator to add columns into existing reports as the schema evolves without having to edit multiple queries.

A user assigned to the **administrator** role creates libraries in **Administration** mode. Figure 5-16 illustrates the All Libraries window in Administration Mode. Currently, three libraries has been created.

Figure 5-16: All Libraries Window

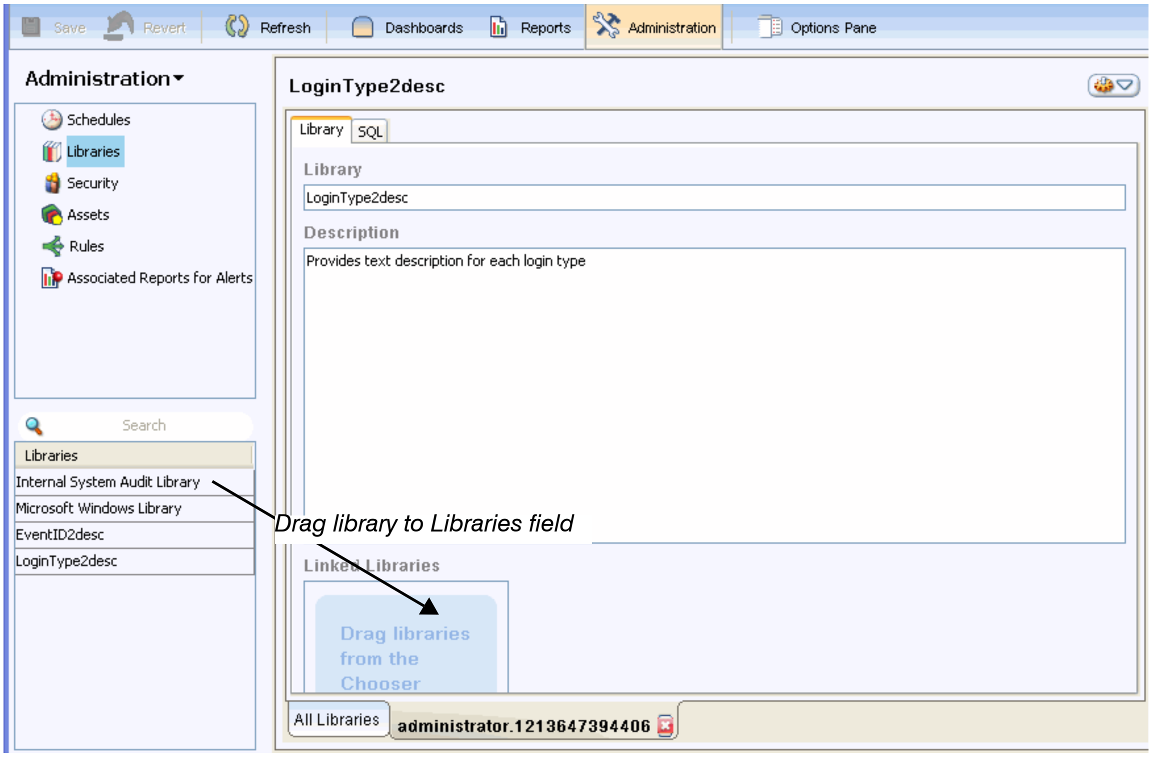


Actions	Library Name	Description
	Internal System Audit Library	For use with some of the Foundation Internal Auditing Reports
	Microsoft Windows Library	For use with some of the Foundation Microsoft Windows Reports
	EventID2desc	Associates Windows event IDs to their text description

Logged in as administrator | Host: e24.hq.sensage.com:3378

To create a library, right click in the workspace or use the workspace action menu to select **New Library**. Figure 5-17 illustrates the window that displays. As illustrated below, use this window to name and describe the new library.

Figure 5-17: Naming, Describing, and Linking A Library



LoginType2desc

Library: LoginType2desc

Description: Provides text description for each login type

Linked Libraries

Libraries

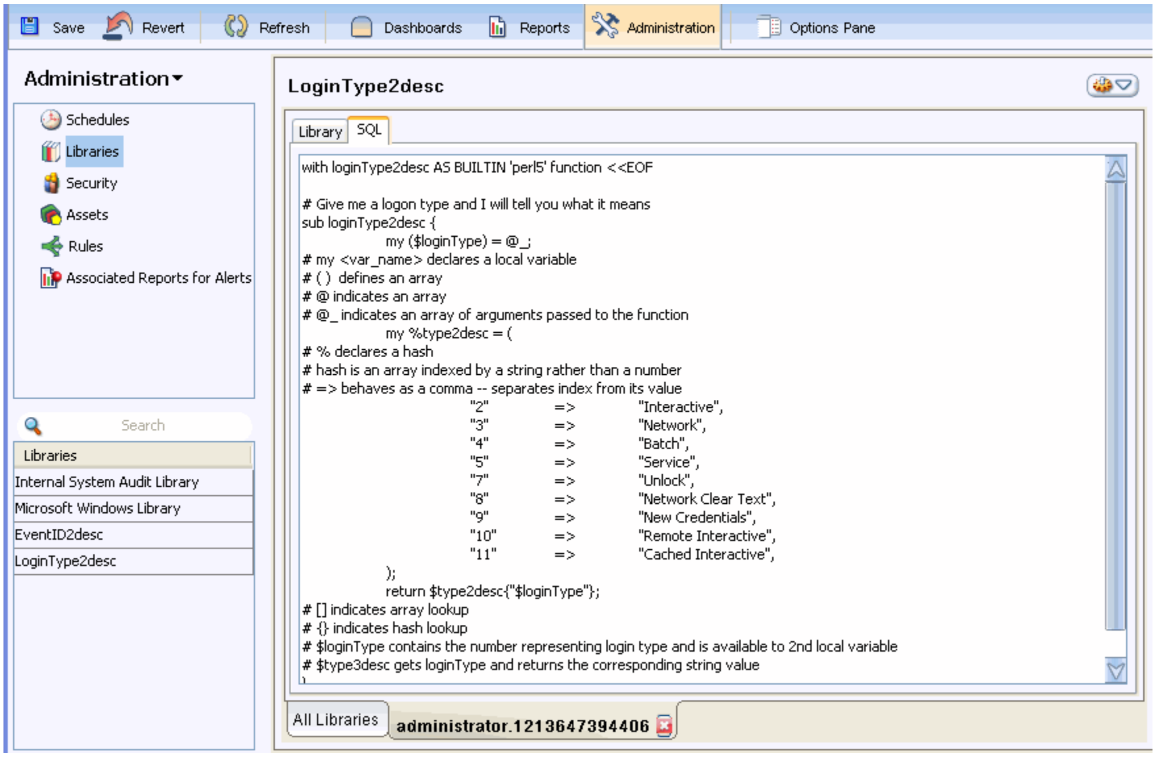
- Internal System Audit Library
- Microsoft Windows Library
- EventID2desc
- LoginType2desc

Drag library to Libraries field

If you want to link an existing library to your new library, you can drag it to the Linked Libraries field, as illustrated above.

To provide the body of the library, click the **SQL** tab. Figure 5-18 illustrates the window in which you enter the library text.

Figure 5-18: Enter the Body of the Library



```
with loginType2desc AS BUILTIN 'perl5' function <<EOF

# Give me a logon type and I will tell you what it means
sub loginType2desc {
                my ($loginType) = @_;
# my <var_name> declares a local variable
# ( )  defines an array
# @ indicates an array
# @_ indicates an array of arguments passed to the function
                my %type2desc = (
# % declares a hash
# hash is an array indexed by a string rather than a number
# => behaves as a comma -- separates index from its value
                                "2"          =>          "Interactive",
                                "3"          =>          "Network",
                                "4"          =>          "Batch",
                                "5"          =>          "Service",
                                "7"          =>          "Unlock",
                                "8"          =>          "Network Clear Text",
                                "9"          =>          "New Credentials",
                                "10"         =>          "Remote Interactive",
                                "11"         =>          "Cached Interactive",
                );
                return $type2desc{"$loginType"};
# [] indicates array lookup
# {} indicates hash lookup
# $loginType contains the number representing login type and is available to 2nd local variable
# $type3desc gets loginType and returns the corresponding string value
}
```

After you click **Save** to save your work, the new library is available to users who create a SQL report.

Viewing and Changing a SQL Report That Has Been Run

After a report has been run, the Options Pane provides the same operations as it does in the dashboard for a report widget. For more information, see the following sections in Chapter 2: Using Dashboards:

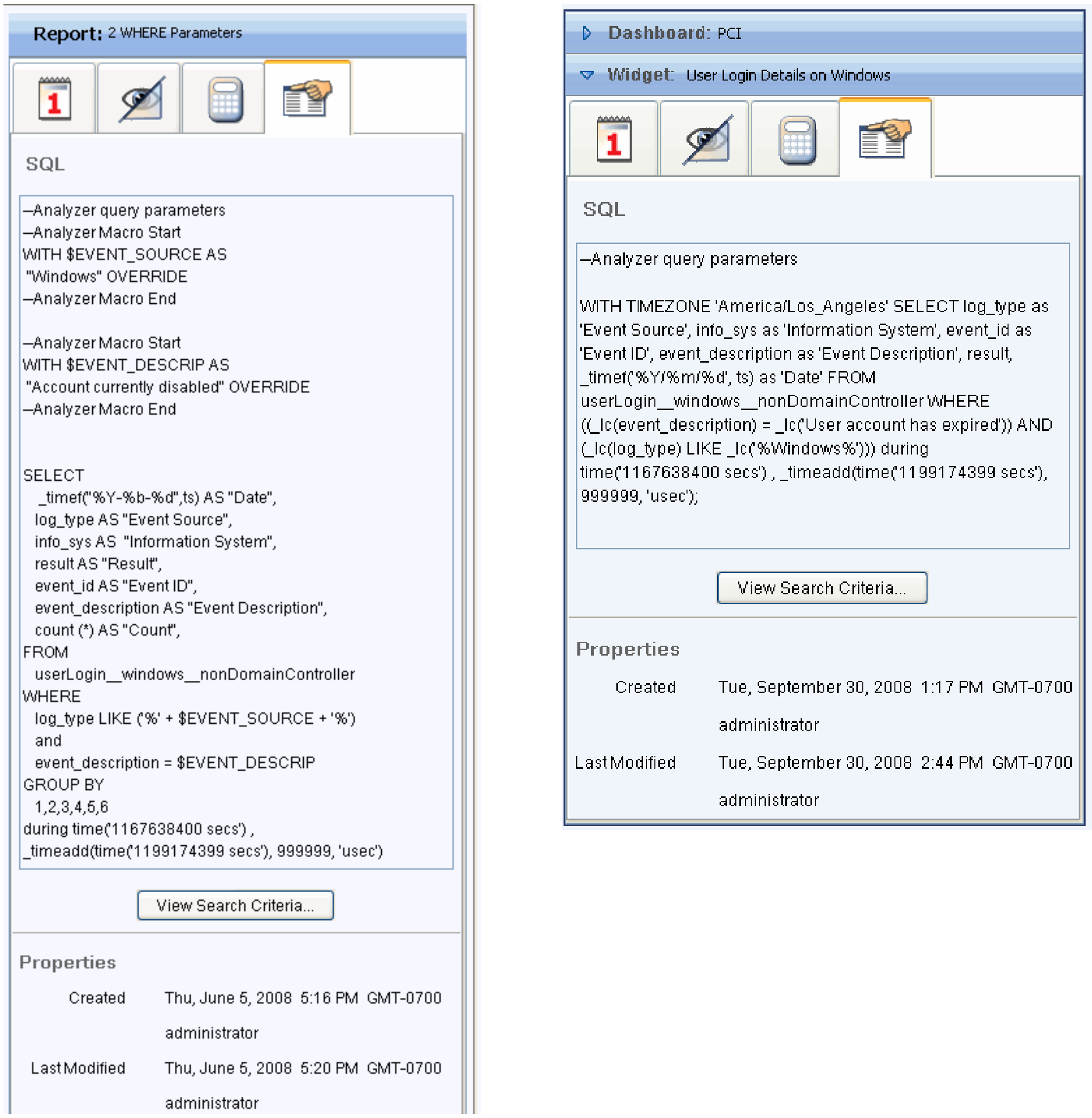
- "Showing and Hiding Report Columns and Metadata", on page 48
- "Viewing and Changing the Time Range and Namespace", on page 51
- "Calculating Report Data", on page 53
- "Changing Between Table and Chart Formats", on page 49
- "Viewing the SQL Query and Other Properties", on page 50

NOTE: Because many dashboard users are not interested in viewing and manipulating the SQL query, the section below provides more detail than the earlier section on viewing the query.

Viewing and Manipulating the SQL Query

Figure 5-19 illustrates the **SQL** tab for two different reports. The tab for the SQL report displays in Reports mode. The tab for the Wizard report displays in a dashboard widget.

Figure 5-19: SQL Tab



SQL Report

```
—Analyzer query parameters
—Analyzer Macro Start
WITH $EVENT_SOURCE AS
 "Windows" OVERRIDE
—Analyzer Macro End

—Analyzer Macro Start
WITH $EVENT_DESCRIP AS
 "Account currently disabled" OVERRIDE
—Analyzer Macro End


SELECT
   _timef("%Y-%b-%d",ts) AS "Date",
  log_type AS "Event Source",
  info_sys AS  "Information System",
  result AS "Result",
  event_id AS "Event ID",
  event_description AS "Event Description",
  count (*) AS "Count",
FROM
  userLogin__windows__nonDomainController
WHERE
  log_type LIKE ('%' + $EVENT_SOURCE + '%')
  and
  event_description = $EVENT_DESCRIP
GROUP BY
  1,2,3,4,5,6
during time('1167638400 secs') ,
_timeadd(time('1199174399 secs'), 999999, 'usec')
```

Created	Thu, June 5, 2008 5:16 PM GMT-0700
	administrator
Last Modified	Thu, June 5, 2008 5:20 PM GMT-0700
	administrator

Wizard Report

```
—Analyzer query parameters

WITH TIMEZONE 'America/Los_Angeles' SELECT log_type as
'Event Source', info_sys as 'Information System', event_id as
'Event ID', event_description as 'Event Description', result,
_timef('%Y/%m/%d', ts) as 'Date' FROM
userLogin__windows__nonDomainController WHERE
((_lc(event_description) = _lc('User account has expired')) AND
(_lc(log_type) LIKE _lc('%Windows%'))) during
time('1167638400 secs') , _timeadd(time('1199174399 secs'),
999999, 'usec');
```

Created	Tue, September 30, 2008 1:17 PM GMT-0700
	administrator
Last Modified	Tue, September 30, 2008 2:44 PM GMT-0700
	administrator

The query displays differently depending on whether the report is a SQL report or a Wizard report:

- If the report is a SQL report, the **SQL** tab displays the SQL statement that a user entered to create the definition.

  If the report definition provides parameters that enable the run-time user to specify value(s), the top of the query displayed in the **SQL** tab shows the user-specified value for each parameter. The value displays as an `OVERRIDE` because the user has overridden the default value for the parameter. For more information, see “Overriding Multiple Macro Declarations”, on page 305.

There is an alternate way to view the parameter value(s) specified by the user at run time. You can display a disabled version of the actual **Run** dialog in which the user specified parameter values; to do so, click **View Search Criteria**.

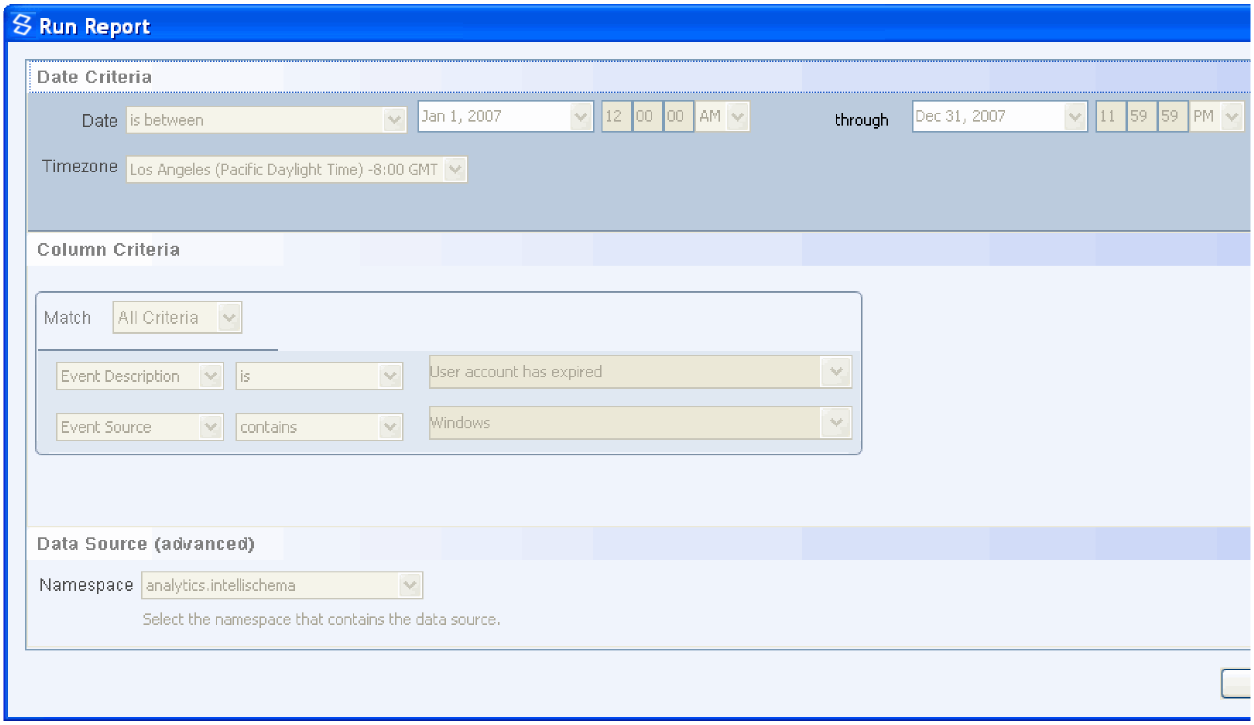
- If the report is a Wizard report, the **SQL** tab displays a generated query.

  NOTE: A user defines the report interactively. The SQL query is generated from the user's definition. You can use this query as the basis of a SQL report. To do so, copy the text from the Options Pane, paste it into the **SQL** tab of a SQL report, and modify it as desired.

  If the report definition provides parameters that enable the run-time user to specify value(s), the query displayed in the **SQL** tab shows the user-specified value for each parameter. Locating the specified value does not require reading the SQL query, however. To display a disabled version of the actual Run dialog in which the user specified parameter values, click **View Search Criteria**.

Figure 5-20 illustrates the window that displays when a user clicks **View Search Criteria** from the Wizard report illustrated above.

Figure 5-20: Viewing Search Criteria



Run Report

Date Criteria

Date: is between · Jan 1, 2007 · 12 : 00 : 00 AM · through · Dec 31, 2007 · 11 : 59 : 59 PM

Timezone: Los Angeles (Pacific Daylight Time) -8:00 GMT

Column Criteria

Match: All Criteria

Event Description · is · User account has expired

Event Source · contains · Windows

Data Source (advanced)

Namespace: analytics.intellischema

Select the namespace that contains the data source.

The graphic above displays a disabled version of the **Run Report** dialog for the example Wizard report. An active version of this dialog displayed to the user at run time. In the run-time dialog, each of the two column-criteria parameters presented a dropdown from which the user selected desired values. The dialog above illustrates the choices the user made at run time; it does not provide the opportunity to make new choices.

From this dialog you can learn the following information about the current cache entry of the example Wizard report:

- Date period and time zone over which it was run
- Column values that the user specified
- Namespace

CHAPTER 6

Creating and Managing Dashboards

This chapter contains the following sections:

- “Overview”, next
- “Creating a Dashboard”, on page 204
- “Managing Dashboards”, on page 216
- “Deploying Dashboards to Your Users”, on page 228

Overview

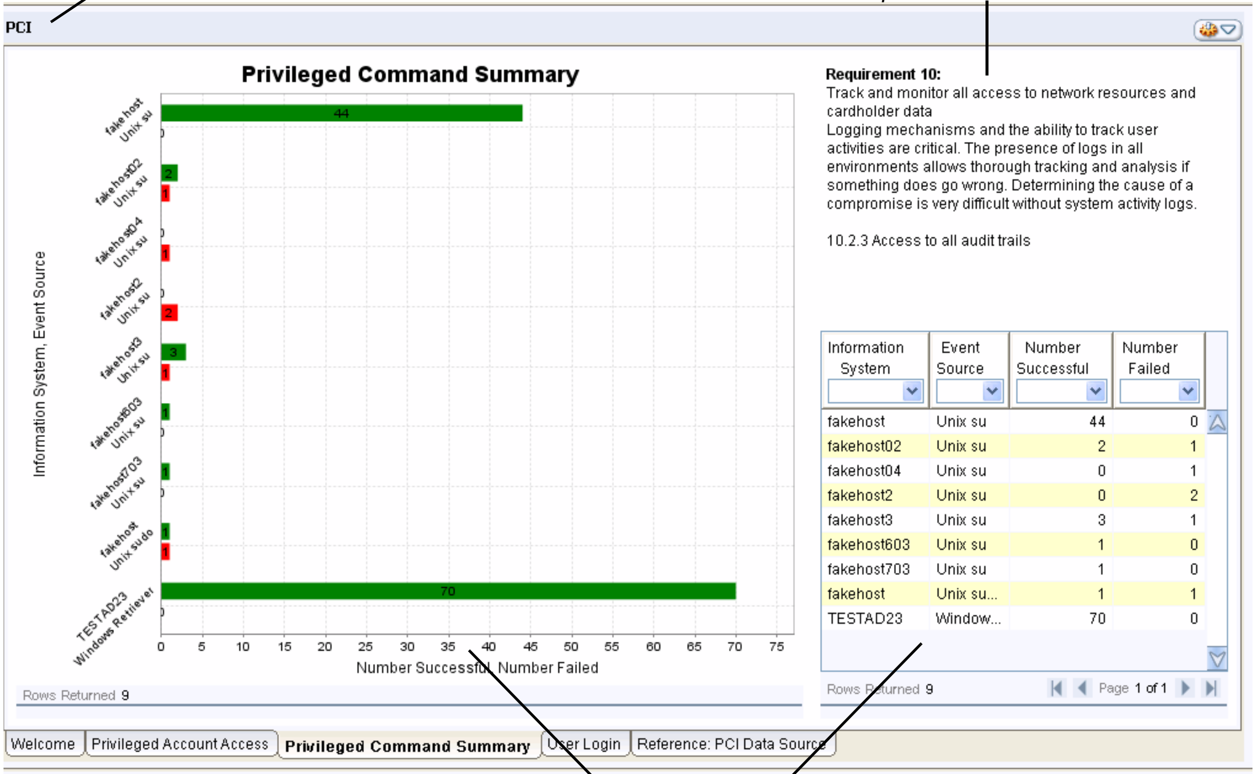
This chapter uses tutorial format to document dashboard creation. The tutorial describes how to create two pages of the example PCI dashboard documented in Chapter 2: Using Dashboards. It also describes how to schedule the dashboard, assign permissions to it, distribute it, and manage it.

Figure 6-1 illustrates one page of the example PCI dashboard: Privileged Command Summary.

Figure 6-1: Example Page with Report and Text Widgets

Dashboard name

Text widget with PCI requirement text



Requirement 10:
Track and monitor all access to network resources and cardholder data
Logging mechanisms and the ability to track user activities are critical. The presence of logs in all environments allows thorough tracking and analysis if something does go wrong. Determining the cause of a compromise is very difficult without system activity logs.

10.2.3 Access to all audit trails

Information System	Event Source	Number Successful	Number Failed
fakehost	Unix su	44	0
fakehost02	Unix su	2	1
fakehost04	Unix su	0	1
fakehost2	Unix su	0	2
fakehost3	Unix su	3	1
fakehost603	Unix su	1	0
fakehost703	Unix su	1	0
fakehost	Unix su...	1	1
TESTAD23	Window...	70	0

Report widgets

This page above displays:

- two versions of the same report widget—one in chart and one in table format

- a text widget that contains the PCI requirement that provides context for the reports

Additionally this tutorial describes how to create the PCI Welcome page, illustrated in Figure 6-13. The Welcome page displays an image widget and two text widgets.

The first part of the tutorial describes how to create a dashboard that contains the Privileged Command Summary and Welcome pages. The rest of the tutorial describes dashboard management and deployment.

CREATING A DASHBOARD

This section describes the following topics:

- "Step 1: Creating the Dashboard", next
- "Step 2: Adding a Report Widget to a Dashboard Page", on page 205
- "Step 3: Switching Between Chart and Table Display", on page 207
- "Step 3: Adding a Second Widget", on page 209
- "Step 5: Working With a Text Widget", on page 211
- "Creating a Page", on page 213
- "Adding and Formatting Image Widgets", on page 214
- "Locking the Dashboard", on page 215

Step 1: Creating the Dashboard

This topic describes the first steps required to create the example dashboard and the page shown in Figure 6-1. The following topics walk you through the steps that populate this page with widgets and format those widgets.

To create the example dashboard

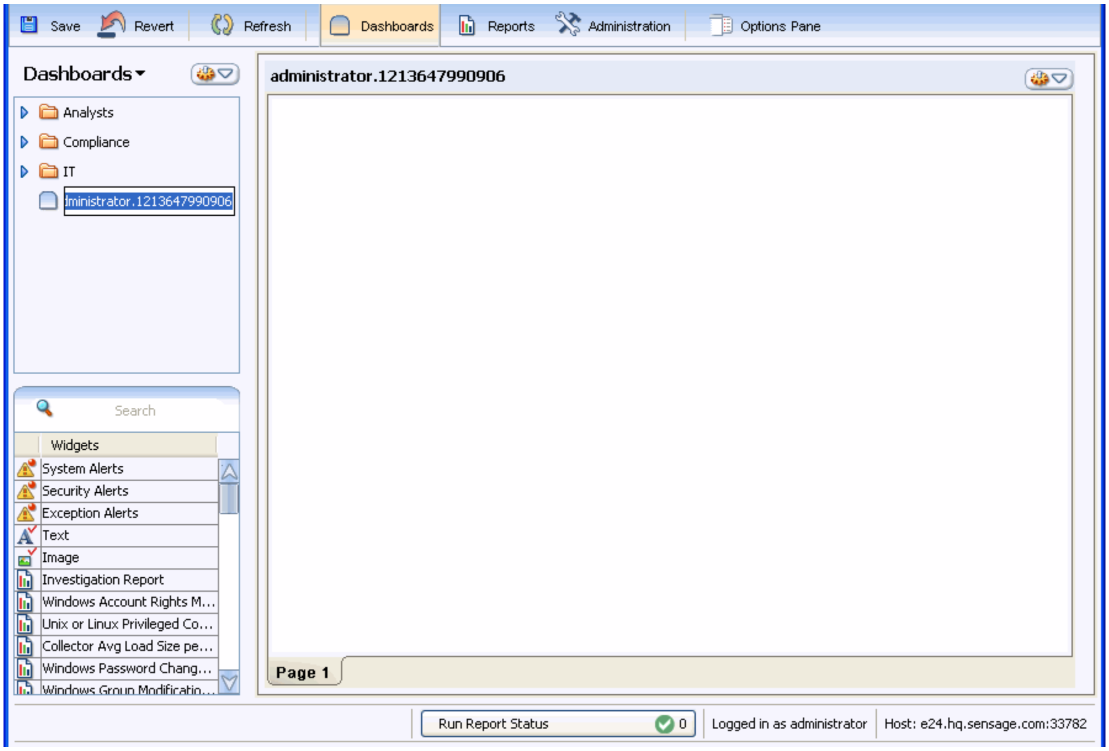
1 Select **New Dashboard** either from an Action Menu or right-click menu.

A highlighted name field displays.

2 Enter "`PCI`" in the highlighted name field and press ENTER.

Figure 6-2 illustrates the first step of dashboard creation. This window opened after the user selected the **New Dashboard** menu. The newly created dashboard provides a field ready for the dashboard name and an empty page named `Page 1`.

Figure 6-2: Creating a Dashboard


Save
Revert
Refresh
Dashboards
Reports
Administration
Options Pane
Dashboards
Analysts
Compliance
IT
lministrator.1213647990906
administrator.1213647990906
Search
Widgets
System Alerts
Security Alerts
Exception Alerts
Text
Image
Investigation Report
Windows Account Rights M...
Unix or Linux Privileged Co...
Collector Avg Load Size pe...
Windows Password Chang...
Windows Group Modificatio...
Page 1
Run Report Status
0
Logged in as administrator
Host: e24.hq.sensage.com:33782

3 Double click the page name tab, change the name of the page to `Privileged Command Summary` and press ENTER.

Figure 6-3 illustrates the new dashboard, named `PCI`. It also shows that the first page has been renamed page to `Privileged Command Summary`.

4 Drag widgets from the Chooser to the page and customize display.

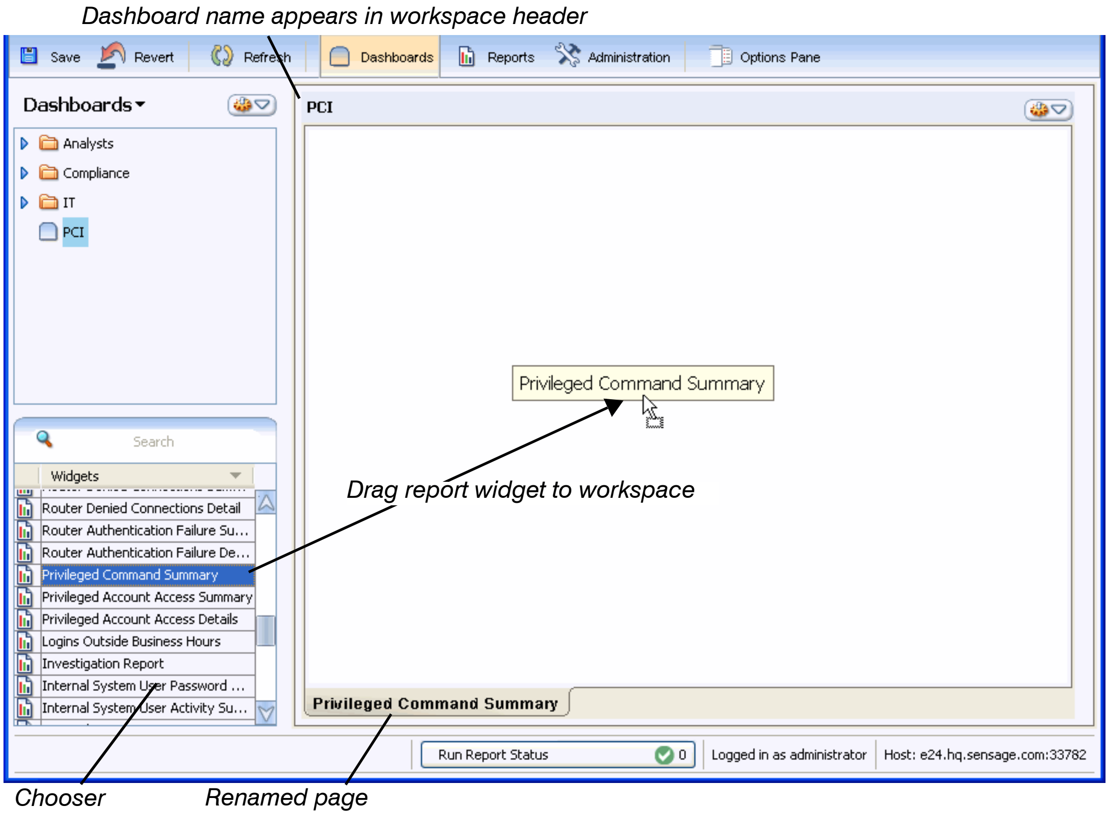
The next sections describe how to work with these widgets.

Step 2: Adding a Report Widget to a Dashboard Page

You populate a page by dragging widgets to it from the Chooser. If your Chooser displays more widgets than you can view at one time, use the **Search** field to limit display by name. For more information on searching for a widget, see “Chooser”, on page 33.

Figure 6-3 illustrates the first widget being dragged to the page from the Chooser. It also illustrates the new dashboard name and page name.

Figure 6-3: Dragging a Widget to the Dashboard



Save
Revert
Refresh
Dashboards
Reports
Administration
Options Pane
Dashboards
Analysts
Compliance
IT
PCI
Search
Widgets
Router Denied Connections Detail
Router Authentication Failure Su...
Router Authentication Failure De...
Privileged Command Summary
Privileged Account Access Summary
Privileged Account Access Details
Logins Outside Business Hours
Investigation Report
Internal System User Password ...
Internal System User Activity Su...
PCI
Privileged Command Summary
Privileged Command Summary
Run Report Status
0
Logged in as administrator
Host: e24.hq.sensage.com:33782

Dashboard name appears in workspace header

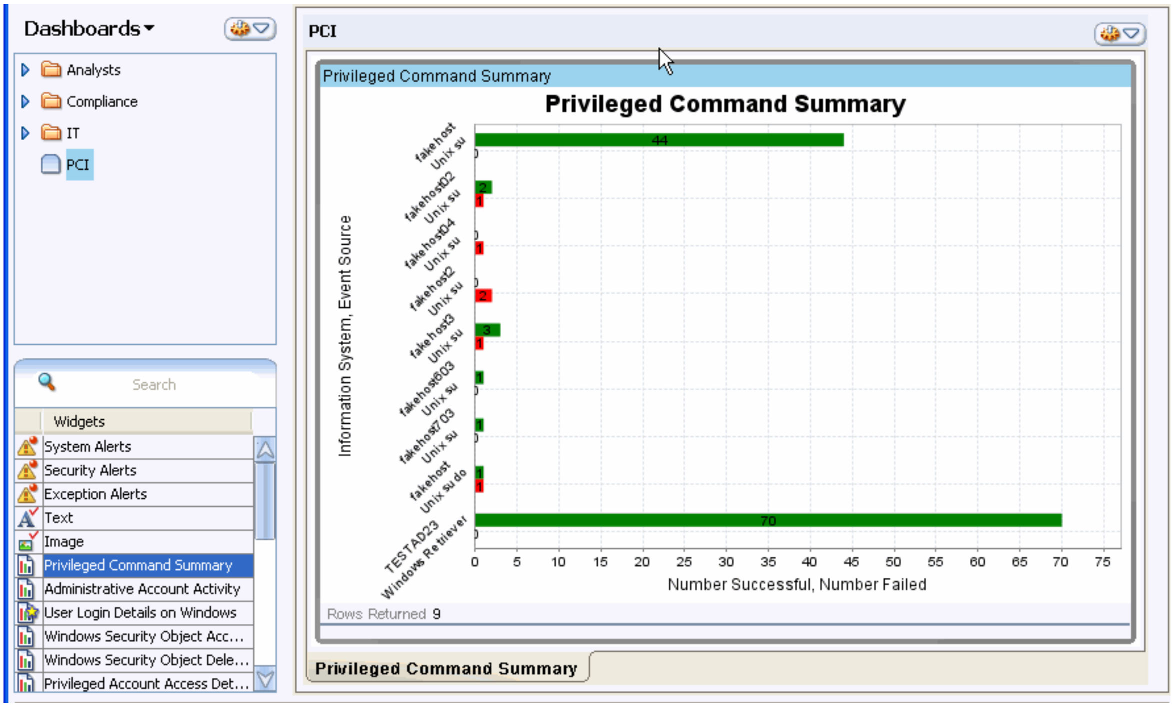
Drag report widget to workspace

Chooser

Renamed page

The first widget fills the page, as illustrated in Figure 6-4.

Figure 6-4: First Widget on a Page


Dashboards
Analysts
Compliance
IT
PCI
Search
Widgets
System Alerts
Security Alerts
Exception Alerts
Text
Image
Privileged Command Summary
Administrative Account Activity
User Login Details on Windows
Windows Security Object Acc...
Windows Security Object Dele...
Privileged Account Access Det...
PCI
Privileged Command Summary
Privileged Command Summary
Information System, Event Source
fakehost Unix su
fakehost02 Unix su
fakehost04 Unix su
fakehost2 Unix su
fakehost3 Unix su
fakehost603 Unix su
fakehost703 Unix su
fakehost Unix sudo
TESTAD23 Windows Retriever
44
0
2
1
0
1
0
2
3
1
1
0
1
0
1
1
70
0
0
5
10
15
20
25
30
35
40
45
50
55
60
65
70
75
Number Successful, Number Failed
Rows Returned 9
Privileged Command Summary

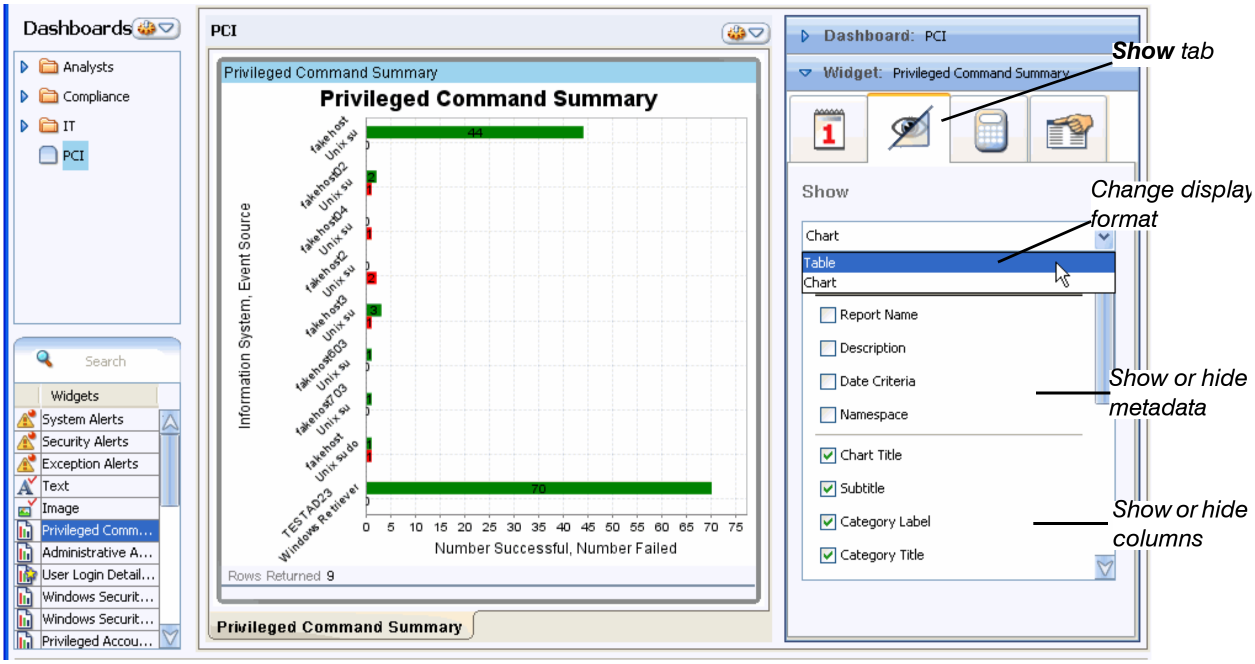
As illustrated above, the report widget displays its data in chart format. A report that has been configured as a chart displays as a chart when you drag its widget to a dashboard. Because the example page displays this report in two widgets, one as a chart and the other as a table, the next step changes the display of this first widget.

Step 3: Switching Between Chart and Table Display

If it is important for your users to view a report in table format or you want to display the same report in both chart and table formats, use the **Show** tab of the Options Pane to change the display of the report widget to table.

Figure 6-5 illustrates how you would change the display of the `Privileged Command Summary` widget just dragged to the example page. This figure also illustrates how you can show or hide metadata and report columns. Metadata information includes the report name, a text description of the report, its namespace, and the date criteria that determines its time period.

Figure 6-5: Changing Report Display



Show: Chart (options: Table, Chart)

- ☐ Report Name
- ☐ Description
- ☐ Date Criteria
- ☐ Namespace
- ☑ Chart Title
- ☑ Subtitle
- ☑ Category Label
- ☑ Category Title

Show tab

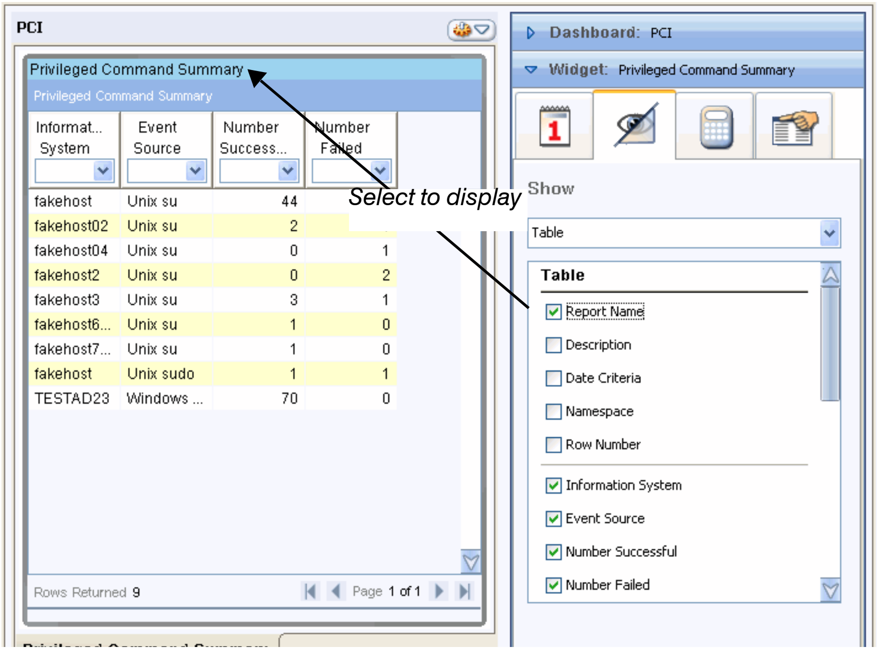
Change display format

Show or hide metadata

Show or hide columns

Figure 6-6 illustrates the results of the operations illustrated in Figure 6-5. The report now displays as a table and its name displays above it.

Figure 6-6: Displaying the First Report as a Table


PCI
Privileged Command Summary
Privileged Command Summary
Rows Returned 9
Page 1 of 1
Dashboard: PCI
Widget: Privileged Command Summary
Show
Table

Informat... System	Event Source	Number Success...	Number Failed
fakehost	Unix su	44	
fakehost02	Unix su	2	
fakehost04	Unix su	0	1
fakehost2	Unix su	0	2
fakehost3	Unix su	3	1
fakehost6...	Unix su	1	0
fakehost7...	Unix su	1	0
fakehost	Unix sudo	1	1
TESTAD23	Windows ...	70	0

Show: Table

Table

- ☑ Report Name
- ☐ Description
- ☐ Date Criteria
- ☐ Namespace
- ☐ Row Number
- ☑ Information System
- ☑ Event Source
- ☑ Number Successful
- ☑ Number Failed

Select to display

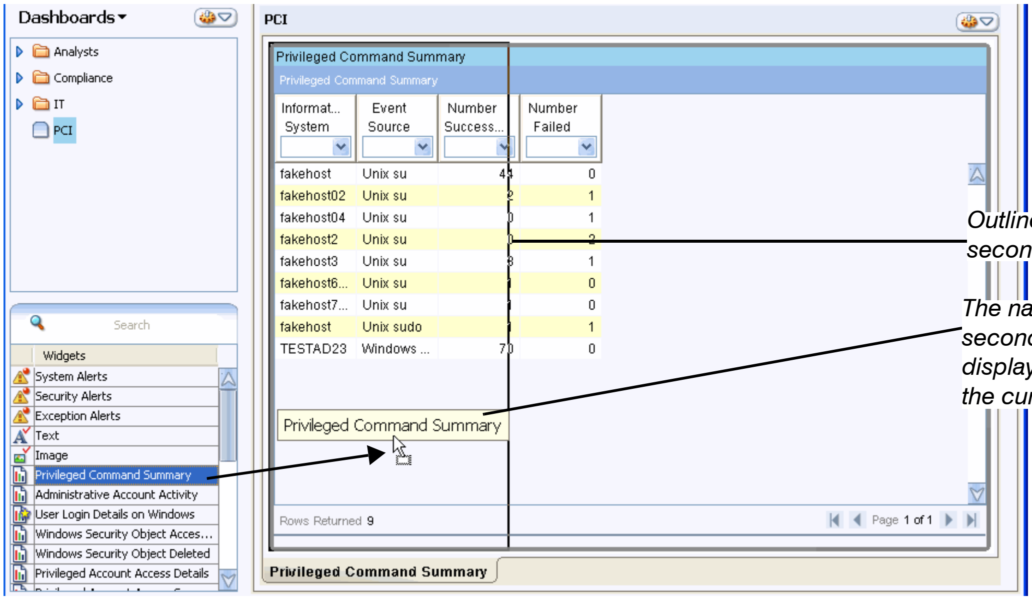
The next step is to drag the same report widget to the page. This widget will display in chart format.

Step 3: Adding a Second Widget

The next step in creating the example dashboard is to drag a second widget onto the page. In this example, you drag the same report widget. You can position the widget as you drag it. You can also rearrange and resize widgets after you have populated the page.

As illustrated in Figure 6-8, as you drag the second widget onto the page, the workspace displays the widget's outline and name. Before you release the mouse to drop the widget, you can move the outline around the page to position it relative to the existing widget. Figure 6-7 illustrates positioning the second widget to the left of the first.

Figure 6-7: Positioning a Second Widget to the Left of the First


Dashboards
Analysts
Compliance
IT
PCI
Search
Widgets
System Alerts
Security Alerts
Exception Alerts
Text
Image
Privileged Command Summary
Administrative Account Activity
User Login Details on Windows
Windows Security Object Acces...
Windows Security Object Deleted
Privileged Account Access Details
PCI
Privileged Command Summary
Privileged Command Summary
Privileged Command Summary
Rows Returned 9
Page 1 of 1
Privileged Command Summary

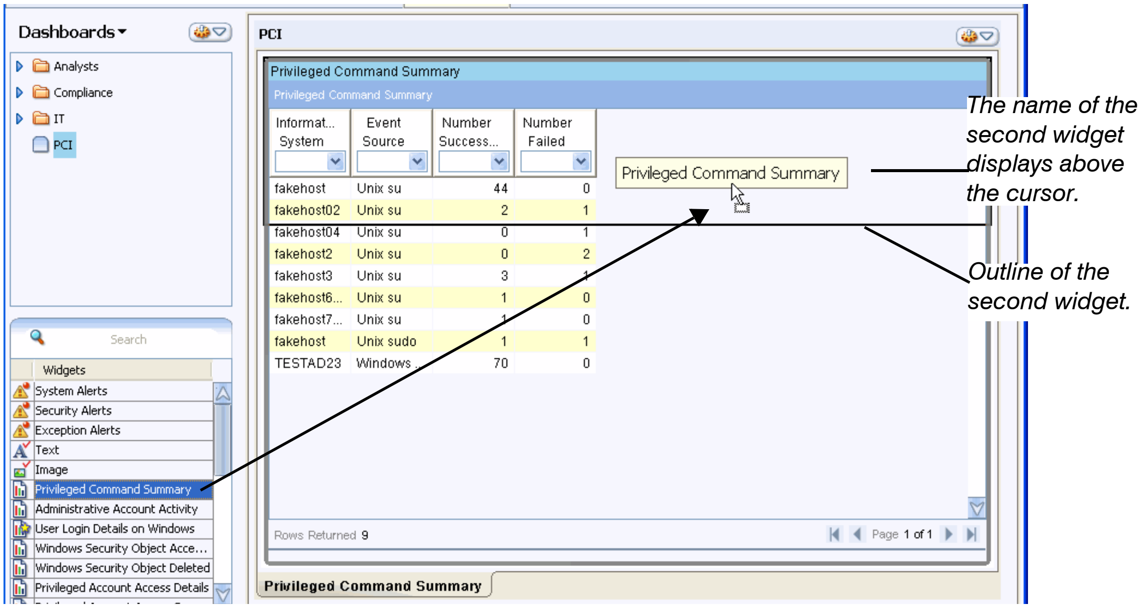
Informat... System	Event Source	Number Success...	Number Failed
fakehost	Unix su	44	0
fakehost02	Unix su	2	1
fakehost04	Unix su	0	1
fakehost2	Unix su	0	2
fakehost3	Unix su	8	1
fakehost6...	Unix su	1	0
fakehost7...	Unix su	1	0
fakehost	Unix sudo	1	1
TESTAD23	Windows ...	70	0

Outline of the second widget.

The name of the second widget displays above the cursor.

Figure 6-8 illustrates positioning the second widget above the first.

Figure 6-8: Positioning a Second Widget Above the First


Dashboards
Analysts
Compliance
IT
PCI
Search
Widgets
System Alerts
Security Alerts
Exception Alerts
Text
Image
Privileged Command Summary
Administrative Account Activity
User Login Details on Windows
Windows Security Object Acce...
Windows Security Object Deleted
Privileged Account Access Details
PCI
Privileged Command Summary
Privileged Command Summary
Privileged Command Summary
Page 1 of 1
Privileged Command Summary

Informat... System	Event Source	Number Success...	Number Failed
fakehost	Unix su	44	0
fakehost02	Unix su	2	1
fakehost04	Unix su	0	1
fakehost2	Unix su	0	2
fakehost3	Unix su	3	1
fakehost6...	Unix su	1	0
fakehost7...	Unix su	1	0
fakehost	Unix sudo	1	1
TESTAD23	Windows ...	70	0

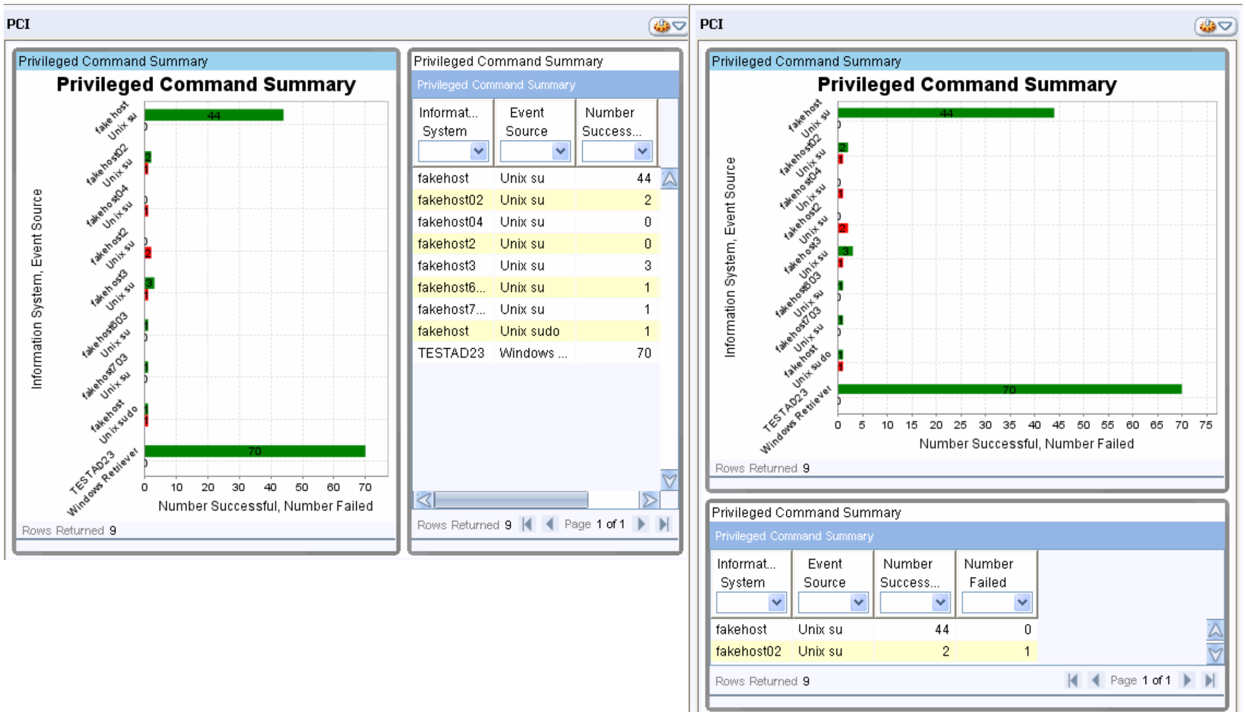
Rows Returned 9

The name of the second widget displays above the cursor.

Outline of the second widget.

After you release the mouse, the second widget remains where you positioned it and the first widget is resized to accommodate the second widget. Figure 6-9 illustrates two versions of the workspace page. These versions display the outcome of the positions illustrated in Figure 6-7 and Figure 6-8.

Figure 6-9: Releasing the Widget to its Location


PCI
Privileged Command Summary
Privileged Command Summary
Information System, Event Source
fakehost Unix su
fakehost02 Unix su
fakehost04 Unix su
fakehost2 Unix su
fakehost3 Unix su
fakehost603 Unix su
fakehost703 Unix su
fakehost Unix sudo
TESTAD23 Windows Retriever
44
70
0 10 20 30 40 50 60 70
Number Successful, Number Failed
Privileged Command Summary
Privileged Command Summary
Page 1 of 1
PCI
Privileged Command Summary
Privileged Command Summary
Information System, Event Source
44
2
3
70
0 5 10 15 20 25 30 35 40 45 50 55 60 65 70 75
Number Successful, Number Failed
Privileged Command Summary
Privileged Command Summary
Page 1 of 1

Rows Returned 9

Informat... System	Event Source	Number Success...
fakehost	Unix su	44
fakehost02	Unix su	2
fakehost04	Unix su	0
fakehost2	Unix su	0
fakehost3	Unix su	3
fakehost6...	Unix su	1
fakehost7...	Unix su	1
fakehost	Unix sudo	1
TESTAD23	Windows ...	70

Rows Returned 9

Rows Returned 9

Informat... System	Event Source	Number Success...	Number Failed
fakehost	Unix su	44	0
fakehost02	Unix su	2	1

Rows Returned 9

After you have released the widgets onto the page, you can drag them to different locations. You can also resize them.

To resize a widget

1. Position the cursor over the border of the widgets whose size you want to change.

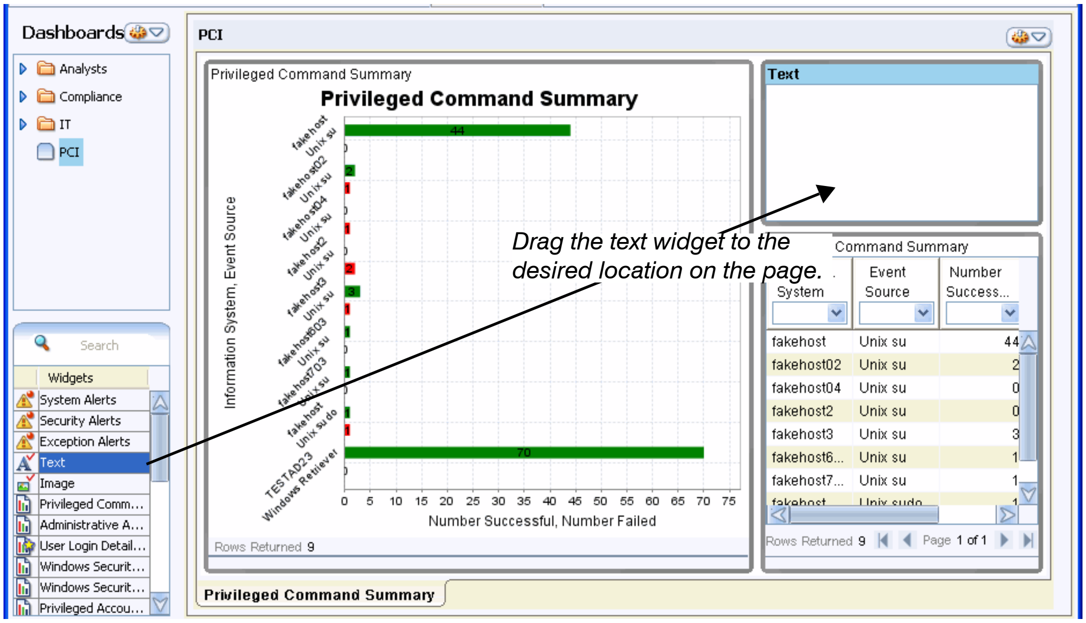
   The mouse pointer changes to the sizing pointer.

2. Click and drag the cursor up and down or right and left as appropriate to change the size of the widget.

Step 5: Working With a Text Widget

The next tasks are to add the text widget, input its text, and format the text. Figure 6-10 illustrates adding this third widget.

Figure 6-10: Adding the Text Widget to the Page

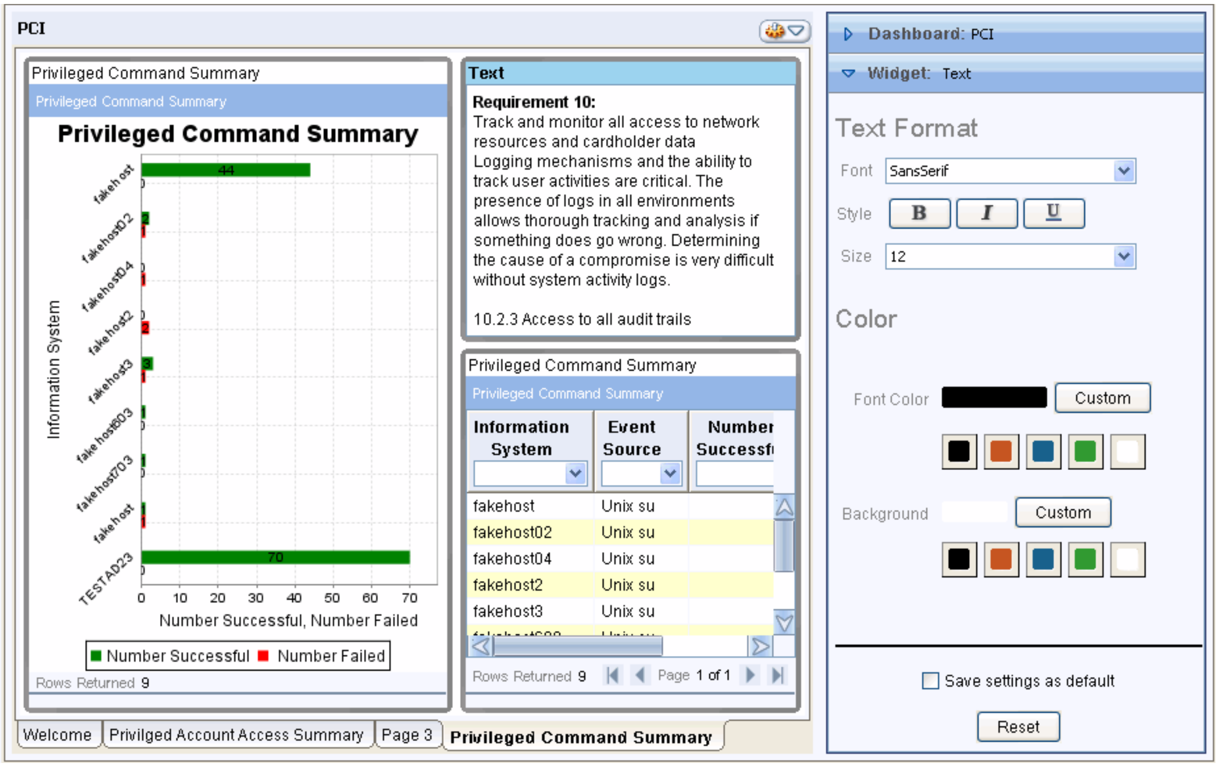


Drag the text widget to the desired location on the page.

System	Event Source	Number Success...
fakehost	Unix su	44
fakehost02	Unix su	2
fakehost04	Unix su	0
fakehost2	Unix su	0
fakehost3	Unix su	3
fakehost6...	Unix su	1
fakehost7...	Unix su	1
fakehost	Unix sudo	1

To add text, click inside the text widget and either type directly into the widget or copy text from an external document and paste it into the widget. The example text widget contains the relevant PCI requirement, as illustrated in Figure 6-11.

Figure 6-11: Formatting Text in a Text Widget



PCI

Privileged Command Summary

Privileged Command Summary

Privileged Command Summary

Rows Returned 9

Text

Requirement 10:
Track and monitor all access to network resources and cardholder data
Logging mechanisms and the ability to track user activities are critical. The presence of logs in all environments allows thorough tracking and analysis if something does go wrong. Determining the cause of a compromise is very difficult without system activity logs.

10.2.3 Access to all audit trails

Privileged Command Summary

Privileged Command Summary

Information System	Event Source	Number Successf
fakehost	Unix su	
fakehost02	Unix su	
fakehost04	Unix su	
fakehost2	Unix su	
fakehost3	Unix su	

Rows Returned 9 Page 1 of 1

Dashboard: PCI

Widget: Text

Text Format

Font: SansSerif

Style

Size: 12

Color

Font Color

Background

□ Save settings as default

You can copy and paste text from an HTML 3.2, MS Word, RTF, or PDF file into a text widget. The text widget preserves all end-of-line tags in text that you paste from these files.

NOTE: After you lock a dashboard:

- Any URL copied from HTML 3.2 into a text widget becomes an active hyperlink.
- HTML formatting elements in a text widget disappear.

Sensage recommends that you save and refresh to verify the changes.

After you enter the text, you can format it. To do so, select the widget and open the Options Pane. Alternately, if the Options Pane is already open, select the desired widget. Figure 6-11 illustrates the Options Pane for the selected text widget.

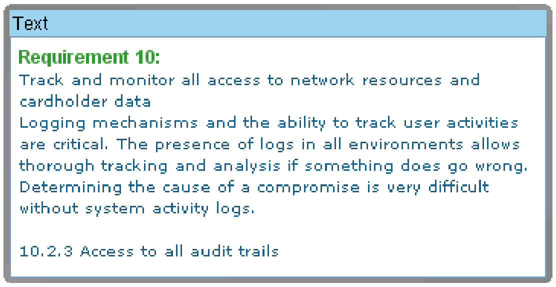
From the Text Format dialog, you can change the type, style, and size of the font that displays the text. You can also change text foreground and background colors. As illustrated in Figure 6-12, you can change heading text separately from body text. Select the desired text first; then modify its format.

In the example below, the heading and body text has been revised as follows:

- Heading text has been enlarged and has had its color changed.

- Body text has a had its font type and color changed.

Figure 6-12: Example of Formatted Text Widget



Requirement 10:
Track and monitor all access to network resources and cardholder data
Logging mechanisms and the ability to track user activities are critical. The presence of logs in all environments allows thorough tracking and analysis if something does go wrong. Determining the cause of a compromise is very difficult without system activity logs.

10.2.3 Access to all audit trails

Before you deliver this page to your users, you will want to lock the page. However, first create the welcome page. For more information, see “Locking the Dashboard”, on page 215.

Creating a Page

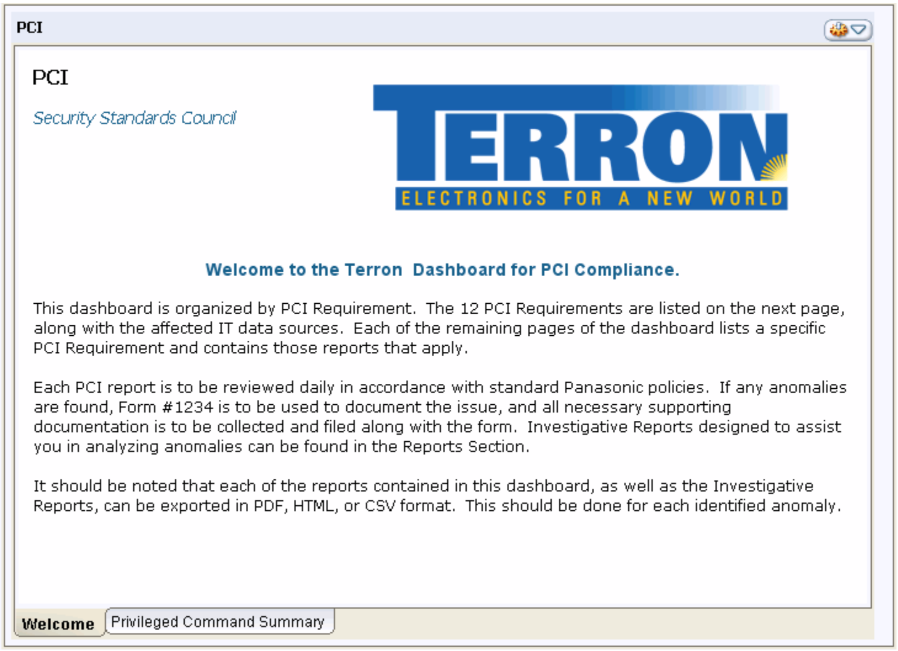
Typically, a dashboard comprises multiple pages. The next step in dashboard creation is to add a page to the dashboard.

To create a page, either select **New** > **Page** from the Workspace Action Menu or right click in the page-tab area and select **New Page** from the popup. While the name is highlighted, enter a meaningful name and press ENTER. Alternately, you can rename it later by double clicking the page tab.

After you create and name a page, you can reorder pages by dragging the page tab to the desired location among other page tabs.

The next topics describe how to create a page like the Welcome page in the example PCI dashboard. Figure 6-13 illustrates the example page.

Figure 6-13: Example Welcome Page with Image and Text Widgets



PCI

Security Standards Council

Welcome to the Terron Dashboard for PCI Compliance.

This dashboard is organized by PCI Requirement. The 12 PCI Requirements are listed on the next page, along with the affected IT data sources. Each of the remaining pages of the dashboard lists a specific PCI Requirement and contains those reports that apply.

Each PCI report is to be reviewed daily in accordance with standard Panasonic policies. If any anomalies are found, Form #1234 is to be used to document the issue, and all necessary supporting documentation is to be collected and filed along with the form. Investigative Reports designed to assist you in analyzing anomalies can be found in the Reports Section.

It should be noted that each of the reports contained in this dashboard, as well as the Investigative Reports, can be exported in PDF, HTML, or CSV format. This should be done for each identified anomaly.

For more information about manipulating a page, see “Changing Focus to a Page”, on page 32 and “Changing the Relative Position of a Page”, on page 32.

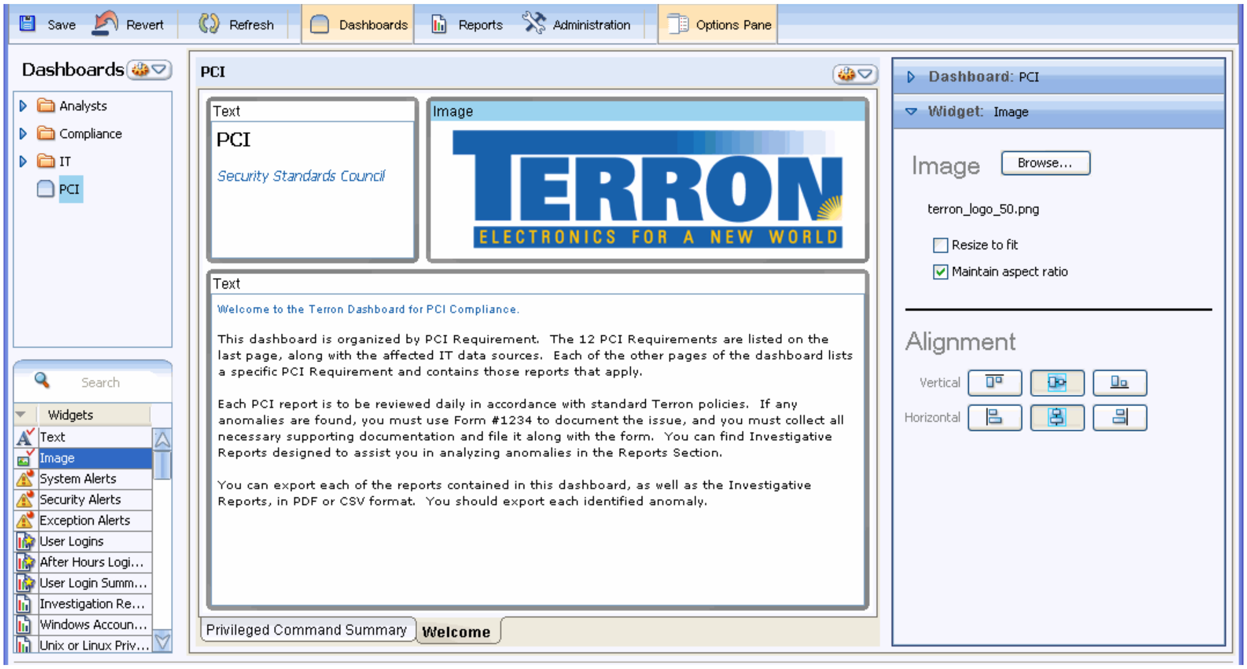
Adding and Formatting Image Widgets

After you create the Welcome page, drag two text widgets to it and populate them with text. Because text wizards were discussed in “Step 5: Working With a Text Widget”, on page 211, this section moves directly to working with Image widgets.

Figure 6-14 illustrates the Welcome page with the two text widgets and the new image widget. The user has entered text directly into one text widget and copied and pasted text into the second widget from a text file. The text widgets have been resized and their text formatted. The user has also dragged an image widget onto the page to the desired location.

As soon as you drop the image widget onto the page, the **Open** dialog displays. This dialog enables you to browse to the directory that contains the desired image file. As shown in Figure 6-14, the user has imported their company logo.

Figure 6-14: Adding and Formatting Image Widget



Text

PCI

Security Standards Council

Text

Welcome to the Terron Dashboard for PCI Compliance.

This dashboard is organized by PCI Requirement. The 12 PCI Requirements are listed on the last page, along with the affected IT data sources. Each of the other pages of the dashboard lists a specific PCI Requirement and contains those reports that apply.

Each PCI report is to be reviewed daily in accordance with standard Terron policies. If any anomalies are found, you must use Form #1234 to document the issue, and you must collect all necessary supporting documentation and file it along with the form. You can find Investigative Reports designed to assist you in analyzing anomalies in the Reports Section.

You can export each of the reports contained in this dashboard, as well as the Investigative Reports, in PDF or CSV format. You should export each identified anomaly.

Image: terron_logo_50.png

☐ Resize to fit

☑ Maintain aspect ratio

The screen above illustrates the Options Pane operations for an image widget. These operations allow you to change the graphic by browsing to a different directory and image file. The pane displays the full path of the currently displayed image.

The formatting options for an image widget also enable you to change its aspect ratio and the vertical and horizontal position of the image within the widget.

At this point you have completed creation of two pages in the example PCI dashboard. You can now move the Welcome page to the left of the Privileged Command Summary page. To do so, grab one of the page tabs and drag it to the desired position.

The final step in dashboard creation is to lock the dashboard and save it. Figure 6-13 illustrates the locked version of the dashboard shown in Figure 6-14.

Locking the Dashboard

Typically you lock a dashboard before you deploy it to your users. When you lock the dashboard:

- The title bar and outline of each displayed widget on all pages disappears and your user cannot reposition the widgets. The appearance of each page is cleaner and crisper.
- All pages in the dashboard are locked.

To lock a dashboard, select **Lock** from the Workspace Action Menu. To unlock, select **Lock** again.

Removing a Widget from a Page

You can delete a widget from a page either by selecting **Delete** > **Widget...** from the Workspace Action Menu or **Delete Widget...** from the popup that displays when you right-click anywhere inside the widget except its header. You will be prompted to confirm the deletion. Click **Yes** in the confirmation dialog.

MANAGING DASHBOARDS

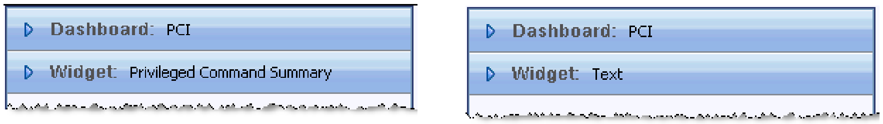
This section contains the following topics:

- “Setting Dashboard Options”, next
- “Scheduling and Modifying Reports from the Dashboard”, on page 222
- “Managing Dashboards in Folders”, on page 225
- “Viewing and Assigning Dashboard and Folder Permissions”, on page 226
- “Running Dashboards”, on page 227
- “Deleting Dashboards”, on page 228
- “Deploying Dashboards to Your Users”, on page 228

Setting Dashboard Options

In Dashboards mode, the Options Pane provides one pane for the dashboard as a whole and another for the widgets on the current page. When the Options Pane is open, the dashboard pane is always available. As you change focus among the widgets on the current page, the title and options of the widget pane change to reflect the selected widget. Figure 6-15 illustrates how the title changes when the user selects two different widgets on the same page.

Figure 6-15: Dynamically Changing Option Pane Display



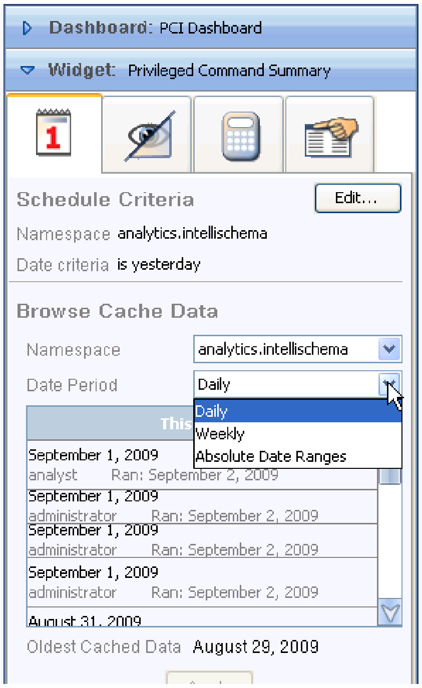
Dashboard options are displayed under three separate tabs. The following sections describe these tabs:

- “Specifying Date Options”, next
- “Setting Permissions”, on page 218
- “Viewing Properties”, on page 220

Specifying Date Options

Figure 6-16 illustrates the **Date Options** tab for the PCI dashboard. This tab enables you to select which report results the dashboard displays. You use this tab to change the displayed cache of all reports on the dashboard.

Figure 6-16: Date Options Tab: Selecting a Schedule and Its Run History



Dashboard: PCI Dashboard

Widget: Privileged Command Summary

Schedule Criteria

Edit...

Namespace analytics.intellischema

Date criteria is yesterday

Browse Cache Data

Namespace analytics.intellischema

Date Period Daily

Daily
Weekly
Absolute Date Ranges

September 1, 2009
analyst Ran: September 2, 2009

September 1, 2009
administrator Ran: September 2, 2009

September 1, 2009
administrator Ran: September 2, 2009

September 1, 2009
administrator Ran: September 2, 2009

August 31, 2009

Oldest Cached Data August 29, 2009

1 *Select desired schedule.*

2 *Select schedule run history.*

3 *Click **Apply**.*

The **Date Options** tab provides two options:

- **Show cache per scheduled run**

  This option causes only scheduled cache entries to display. Every scheduled run of the dashboard adds a cache entry to the display. By default, the latest entry displays.

  If more than one schedule runs the dashboard, the **Schedule** field provides a dropdown of all schedules by name. Select the desired schedule to display its cache entries. If the desired cache entry does not display by default, select it and press **Apply**. Figure 6-16 illustrates this process.

- **Show most recent cache per report**

  This option disables all other options on this pane, including the **Apply** button. Select this option to display unscheduled cache entries. For example, you might select this option for the following reasons:

  - The most recent report cache entries indicate suspicious data.
  - A user has run a report manually from Reports mode and you want to display the latest cache entry.
  - You have created an unscheduled dashboard that you have no need to schedule so you want to run it manually.

To view existing report cache entries that are more recent than the scheduled ones, select this option and click **Refresh**. To create new cache entries for every report on the dashboard, run the dashboard by selecting **Run > Dashboard...** from the Workspace Action Menu. After the run completes, click **Refresh** to display the latest cache entry for each report.

For more information, see “Refreshing Dashboards and Running Items”, on page 85.

Setting Permissions

Users are associated with roles. A user’s membership in one or more role determines the types of actions they can perform on specific items, such as whether they can view a dashboard or modify a report or see a folder.

Administrators give users permission to items by granting their roles permissions to those items. The roles to which a user is assigned as well as the permissions granted to the roles determines whether a user can change a dashboard or only view and run it or only view it or have no access to it.

NOTE:

- Users gain the cumulative set of permissions from all roles to which they are assigned. In other words, if you are assigned to the Human Resources role, which has no permission to access an item, and to the Investigations role, which has permission to run and view a dashboard, you have permission to run and view the dashboard.

  For more information, see “Viewing and Assigning Report, Dashboard, and Folder Permissions”, on page 122 and Authentication in Chapter 8, “Administering Users and Authentication” in the *Administration Guide*.

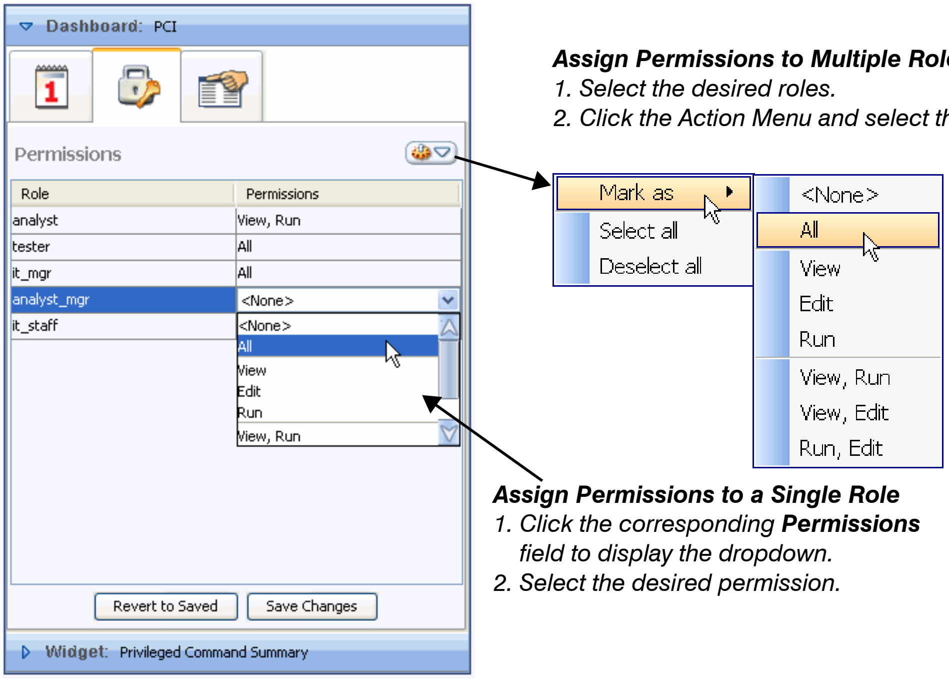
- Users who have Edit permission on an item can assign other users access to it.

Figure 6-17 illustrates the **Permissions** tab for the PCI dashboard. Use this tab to assign dashboard permissions to specific roles and also to view the assigned permissions. For example:

- If you assign the `All` permission, you grant the role the full set of permissions. In other words, you enable all users assigned to the role the ability to view, edit, and run the dashboard.
- If you assign the `View` permission, you limit users assigned to the role to only viewing the dashboard. Such users cannot run or modify the dashboard. They do not even see the items in the Chooser.
- If you assign the `View, Run` permission, you allow users assigned to the role to both view and run the dashboard but not edit it. These users also do not see the items in the Chooser.

There are two ways to assign permissions: to individual roles and to a set of roles. Figure 6-17 illustrates both of these.

Figure 6-17: Permissions Tab: Two Ways to Set Permissions for Each Role



Role	Permissions
analyst	View, Run
tester	All
it_mgr	All
analyst_mgr	<None>
it_staff	

Assign Permissions to Multiple Roles
1. Select the desired roles.
2. Click the Action Menu and select the desired

Assign Permissions to a Single Role
1. Click the corresponding ***Permissions*** *field to display the dropdown.*
2. Select the desired permission.

The **Permissions** tab lists all roles defined for your EDW instance. You assign permissions to specific roles in one of following ways:

To assign permissions to a single role — the Permission dropdown

1 Click the permission field next to the role name.

A dropdown of permissions displays.

2 Select the desired permission from the dropdown.

To assign permissions to multiple roles — the Action Menu

1 Select contiguous or non-contiguous roles and click the Action Menu.

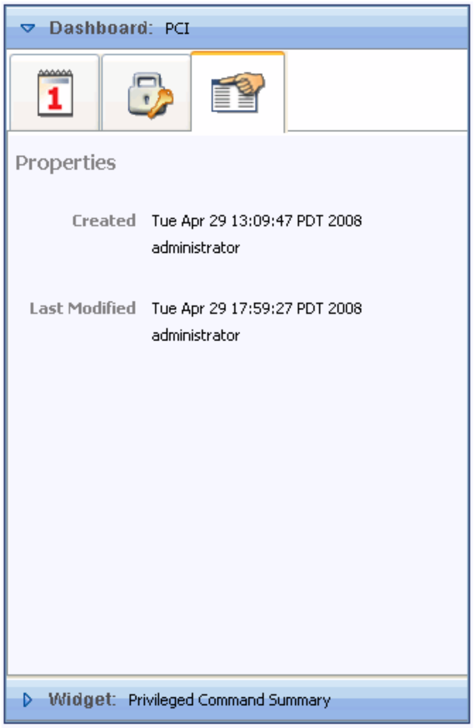
NOTE: Alternately, you can click the Action Menu > **Select all** to grab all roles simultaneously and then use SHIFT-CLICK or CTRL-CLICK to deselect specific roles.

2 Select **Mark as** and then select the desired permission.

Viewing Properties

Figure 6-18 illustrates the **Properties** tab for the PCI dashboard. This tab displays who created the dashboard and when, and who last modified the dashboard and when.

Figure 6-18: Properties Tab



Dashboard: PCI

Properties

Created Tue Apr 29 13:09:47 PDT 2008
administrator

Last Modified Tue Apr 29 17:59:27 PDT 2008
administrator

Widget: Privileged Command Summary

Modifying and Scheduling Report Widgets

This section covers the following topics:

- "Viewing and Changing the Time Range and Namespace", next
- "Scheduling and Modifying Reports from the Dashboard", on page 222
- "Automatically Merging Cache Entries", on page 224
- "Setting Other Widget Options", on page 225

Viewing and Changing the Time Range and Namespace

Every time a dashboard runs, a new cache entry is generated for every report the dashboard displays. When you open a dashboard, the cache entries it displays depends on dashboard configuration. Typically, a dashboard displays the latest cache entry for each report, created when the dashboard ran last.

If a report displays in more than one dashboard, and the different dashboards run at different intervals, the report will have a cache entry for every interval specified for all dashboards that contain it. Additionally, if you or others run the report manually, the report will have a cache entry for each of these manual runs. Each of these manually run cache entries identifies the user who created it and the date created. However, a scheduled dashboard can display unscheduled cache entries only if you select **Show most recent cache per report** for the dashboard. For more information, see “Specifying Date Options”, on page 217.

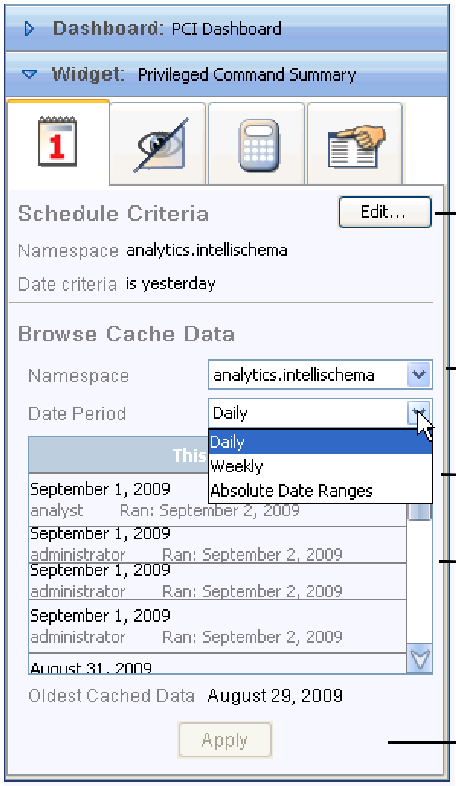
Assume one dashboard runs daily and another weekly, and that they both contain the same report. Daily and weekly cache entries will be available for that report. When you open the daily dashboard, the latest daily cache entry displays. When you open the weekly dashboard, the latest weekly cache entry displays. To switch between these cache entries, or to display an earlier cache for either interval or to merge several cache entries into a single result set, open the **Date Options** tab for the report widget.

As illustrated in Figure 6-19, if a report has been run over more than one date interval, the **Date Period** dropdown displays all cached intervals. If only one interval is available, this field displays the date period as a text label rather than in a dropdown.

NOTE: If a report has been run in more than one namespace, the **Namespace** dropdown contains all of those namespaces. Use this dropdown to switch namespaces.

The field below the **Namespace** and **Date Period** dropdowns organizes cache entries by date and time for the current date period. The most recent entries display at the top.

Figure 6-19: Date Options Tab



Dashboard: PCI Dashboard

Widget: Privileged Command Summary

Schedule Criteria

Namespace analytics.intellischema

Date criteria is yesterday

Browse Cache Data

Namespace analytics.intellischema

Date Period Daily

September 1, 2009
analyst Ran: September 2, 2009

September 1, 2009
administrator Ran: September 2, 2009

September 1, 2009
administrator Ran: September 2, 2009

September 1, 2009
administrator Ran: September 2, 2009

August 31, 2009

Oldest Cached Data August 29, 2009

*Click **Edit...** to:*
- add a namespace to the current list
- specify a new date period

Optionally, select a different namespace from the dropdown.

1 *Select the desired date period from the dropdown.*

2 *Select the desired cache entry or entries from the list.*

3 *Click Apply.*

If a report has multiple cache entries, you can select contiguous entries to display as a single result set. Select one or more cache entries and click **Apply** to display the selected entries. To select multiple entries, select the first entry; then SHIFT-CLICK as you select the last entry.

NOTE:

- Merged results persist only for the current viewing. If you want HawkEye AP Console to automatically merge cache entries that persist, set the auto-merge feature. For more information, see “Automatically Merging Cache Entries”, on page 224.
- The **Date Periods** dropdown displays all cache entries available at the time you opened the Options Pane. To display any cache entry created since you opened the dropdown, you must refresh the dashboard. For more information, see “Refreshing Dashboards and Running Items”, on page 85.
- When you change or combine cache entries, only your view of the data changes. Other users viewing the same dashboard do not see these changes.

If a report has run in more than one namespace, you can display cache entries for another namespace. Select the desired namespace from the **Namespace** dropdown.

If you want to run a report in a namespace that does not display from the dropdown, or none of the cached periods meet your needs, you can specify a different namespace and a new date period from this tab by clicking **Edit...**. For more information, see “Scheduling and Modifying Reports from the Dashboard”, next.

Scheduling and Modifying Reports from the Dashboard

“Step 3: Switching Between Chart and Table Display”, on page 207 describes one reason you might drag the same report widget twice to a dashboard — to display the same report in both chart and table formats. There is another reason you might display the same report widget multiple times in the same dashboard — to run it against different namespaces or over different date periods.

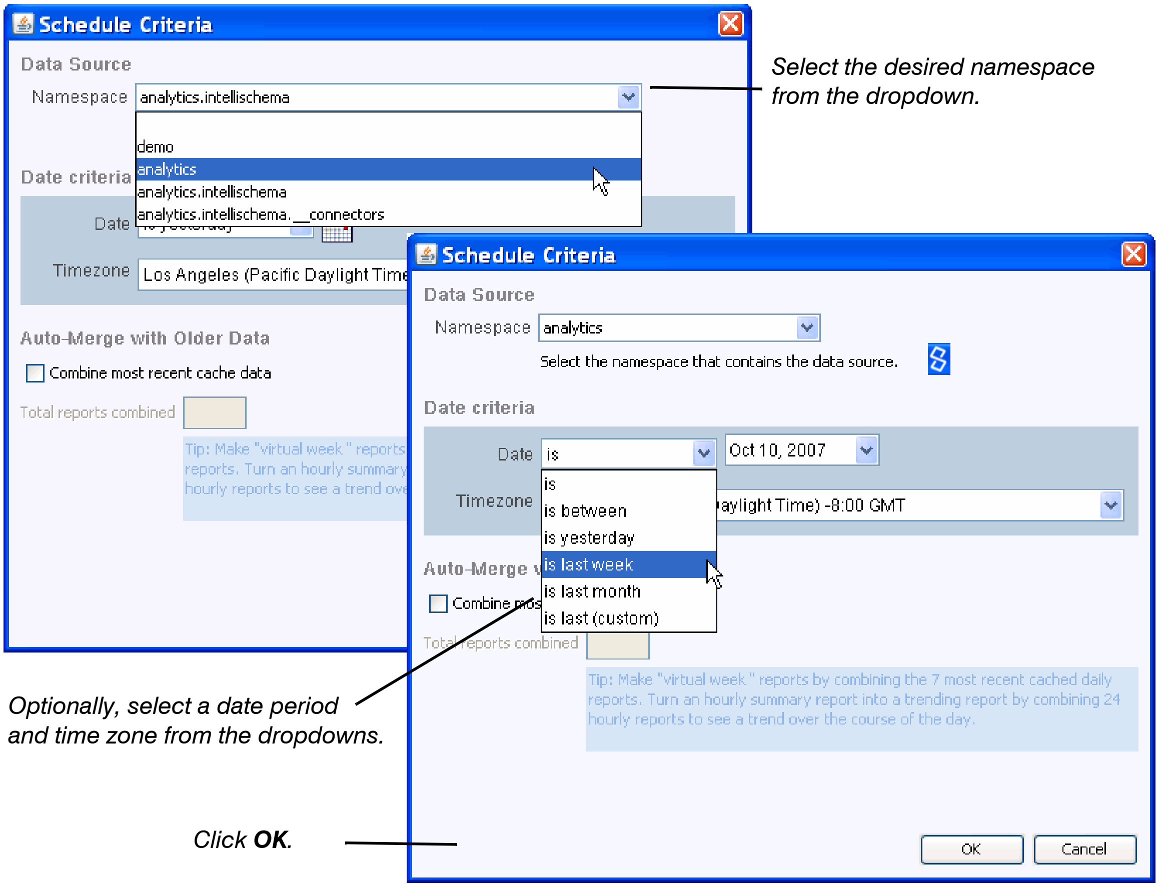
For example, assume you have a login report that is structured to return Windows logins when run in the Windows namespace and Unix logins when run in the Unix namespace. You want to display results from both systems on the same dashboard page. Assume further that the report defaults to running against the Windows namespace. You can use the **Date Options** tab for the report widget to change its namespace and/or its date period. You can also use this tab to automatically merge multiple contiguous report cache entries.

After you click **Edit....** from the **Date Options** tab, the **Schedule Criteria** dialog displays. From this dialog, you can specify new settings for the current report, which the dashboard schedule will use at the next scheduled run. Regardless of whether the dashboard has been scheduled, you can manually run the report with the new settings directly from the dashboard.

After you make and save all desired changes, you can run the widget by selecting it in the dashboard and selecting **Run** > **Widget ...** from the Workspace Action menu. Click **Refresh** to display the latest cache entries.

The report illustrated in Figure 6-20 is a daily report that has been run only in the **analytics** namespace. In this example, that namespace is not available from the **Namespace** dropdown on the **Date Options** tab. To run this report in another namespace to which you have access, click **Edit..**. to display the dialog illustrated in Figure 6-20.

Figure 6-20: Specifying a Different Namespace and Date Criteria



Schedule Criteria

Data Source

Namespace: analytics.intellischema

- demo
- analytics
- analytics.intellischema
- analytics.intellischema.__connectors

Date criteria

Date

Timezone: Los Angeles (Pacific Daylight Time

Auto-Merge with Older Data

□ Combine most recent cache data

Total reports combined

Tip: Make "virtual week" reports reports. Turn an hourly summary hourly reports to see a trend ove

Schedule Criteria

Data Source

Namespace: analytics

Select the namespace that contains the data source.

Date criteria

Date: is | Oct 10, 2007

- is
- is between
- is yesterday
- is last week
- is last month
- is last (custom)

Timezone: aylight Time) -8:00 GMT

Auto-Merge w

□ Combine mos

Total reports combined

Tip: Make "virtual week" reports by combining the 7 most recent cached daily reports. Turn an hourly summary report into a trending report by combining 24 hourly reports to see a trend over the course of the day.

Select the desired namespace from the dropdown.

Optionally, select a date period and time zone from the dropdowns.

*Click **OK**.*

As illustrated above, the **Namespace** field on the **Schedule Criteria** dialog displays all namespaces that you have permission to access. If the current report has been run in only one or two of the available namespaces and you want to run it in a different one, you can select the namespace from this dropdown. The namespace you select must contain the table or view that provides the report data. In the example of the login report that runs in both the Windows and Unix namespaces and is designed to return data from both namespaces, you could select the Unix namespace from this dropdown.

After you select the namespace and run the report, the new namespace displays in the **Namespace** field on the **Date Options** tab.

From this dialog, you can also specify a date period and timezone to create a cache entry that is not yet available from the **Date Options** tab. Additionally, you can use the Auto-Merge pane to combine report caches. For more information on this option, see “Automatically Merging Cache Entries”, next.

After you make all desired changes, click **OK**. The next scheduled run of the dashboard will display the changes you made to the widget. In other words, if you have changed both the namespace and date criteria, the widget will display results for that namespace and date period. Additionally, the **Date Option** tab will include the new namespace and date period in its dropdowns.

Automatically Merging Cache Entries

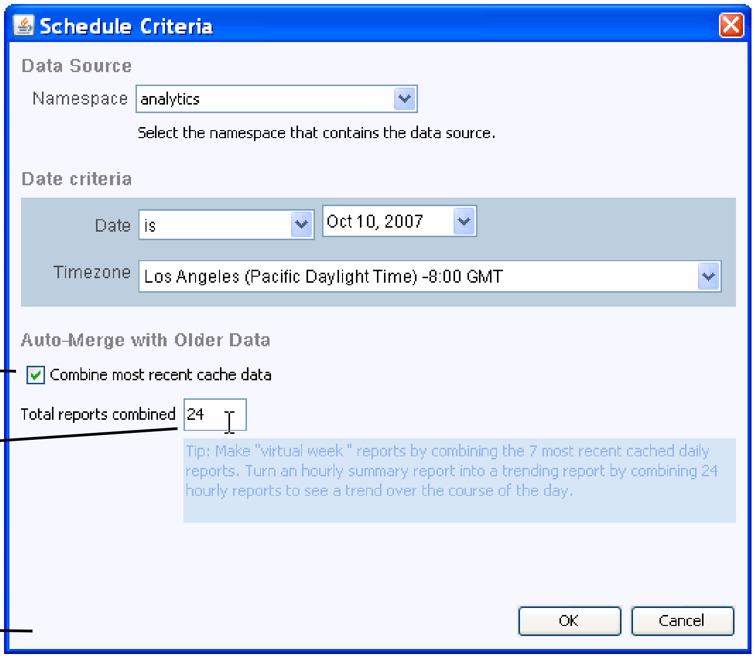
As documented in “Viewing and Changing the Time Range and Namespace”, on page 220, you can create a virtual report that merges multiple existing cache entries. For example, assume a report collects data on an hourly basis. You can use the **Date Period** dropdown from the **Date Options** tab to manually merge 24 hourly reports into a virtual daily report. Such a report enables you to compare the hour values against each other to view how the data changes over the course of the day.

If you use the **Date Period** dropdown from the **Date Options** tab to merge the cache entries, the resulting report displays as soon as you click **Apply** and disappears when you close the dashboard. Such a merged report is also available only to you; other dashboard users cannot see it.

HawkEye AP Console provides another way to merge cache entries, a way that automatically creates the result set every time the schedule runs and saves it even after you close HawkEye AP Console. Such a merged report is available to all dashboard users with permission to view it.

To specify that multiple cache entries be automatically merged and the results saved, use the **Schedule Criteria** dialog illustrated in Figure 6-21.

Figure 6-21: Automatically Merging Report Cache Entries



Schedule Criteria

Data Source

Namespace: analytics

Select the namespace that contains the data source.

Date criteria

Date: is Oct 10, 2007

Timezone: Los Angeles (Pacific Daylight Time) -8:00 GMT

Auto-Merge with Older Data

☑ Combine most recent cache data

Total reports combined: 24

Tip: Make "virtual week " reports by combining the 7 most recent cached daily reports. Turn an hourly summary report into a trending report by combining 24 hourly reports to see a trend over the course of the day.

To activate auto-merge, select ***Combine most recent cache data****.*

Enter the appropriate number; for example, `24` *to generate a daily report from 24 hourly cache entries.*

Click ***OK****.*

The example above assumes the report is run hourly. The user has enabled automatic merging by selecting **Combine most recent cache data**. As the field name indicates, Auto-Merge operates only on the most recent cache entries. The number you specify in **Total reports combined** determines how many cache entries are combined.

For example, if a report has been run daily for seven days over many months, and you specify the value `7` in **Total reports combined**, Auto-Merge combine only the most recent seven cache entries.

After you click **OK**, the settings you specify will be used by the dashboard schedule at its next scheduled run. To force the run, manually run the dashboard. For more information, see "Running Dashboards", on page 227.

Setting Other Widget Options

Widget options depend on widget type. For information on the widget options, see:

- "Step 5: Working With a Text Widget", on page 211
- "Adding and Formatting Image Widgets", on page 214
- "Showing and Hiding Report Columns and Metadata", on page 48
- "Filtering and Sorting Report Data", on page 46
- "Viewing and Changing the Time Range and Namespace", on page 51
- "Calculating Report Data", on page 53
- "Changing Between Table and Chart Formats", on page 49
- "Viewing the SQL Query and Other Properties", on page 50

Managing Dashboards in Folders

Dashboards mode provides folders, which enable you to group dashboards by usage, department, schedule, or other need. Like report folders, which are described in "Managing Report Shortcuts in Folders", on page 110, dashboard folders allow you or an administrator to apply schedules and permissions to a set of dashboards as a group. For example, you can assign schedules to all dashboards in a folder.

NOTE: Setting permissions on a folder does not effect the permissions of the dashboards within the folder or the reports that the dashboards display. Permissions set on a folder are, however, copied to any new dashboards created in (or dragged into) the folder. For more information, see:

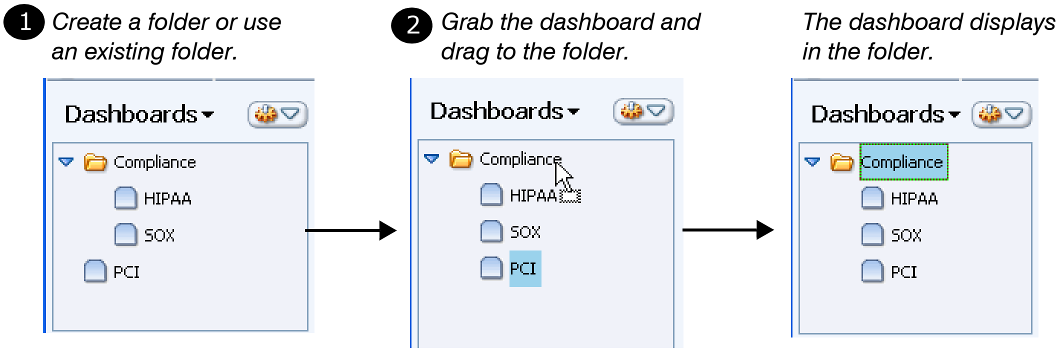
- "Viewing and Assigning Dashboard and Folder Permissions", next
- Managing Access to HawkEye AP Console Reports, Dashboards, and Folders in Chapter 8, "Administering Users and Authentication" of the *Administration Guide*.

You create a dashboard folder in the same way you create a report folder, by selecting **New Folder** from the Navigator Action Menu or from the popup menu that displays when you right click within the Navigator. Change the default name to a meaningful one and, if desired, move it into a folder.

When you move a dashboard into a folder, you drag the dashboard directly from the Navigator to the folder. This effect of this action is different from moving report definitions into a folder. When you move a report definition into a folder, HawkEye AP Console creates a shortcut of the definition in the folder. Shortcuts of the same definition can display in multiple folders. However, unlike report definitions, a dashboard can display in only one folder at a time. Therefore, if you delete the dashboard from the folder, you permanently remove it from your system.

Figure 6-22 illustrates the process to move a dashboard.

Figure 6-22: Moving a Dashboard into a Folder


1 Create a folder or use an existing folder.
2 Grab the dashboard and drag to the folder.
The dashboard displays in the folder.
Dashboards
Compliance
HIPAA
SOX
PCI

Like report folders, you can create a hierarchy of dashboard folders to organize your dashboards. To create hierarchical folders, create and name the desired folders and then drag one folder below another. There is a maximum of ten levels of folders.

Viewing and Assigning Dashboard and Folder Permissions

Users are associated with roles. Their membership in roles determines the types of action users can perform on specific items, such as whether they can view a dashboard or modify a report or open a folder.

Administrators give users permission to reports, dashboards, and folders by granting roles permissions to those items. A user who has Edit permission on a folder, can also give other users permissions on the report. Dashboards and folders do not have owners.

The roles to which a user is assigned as well as the permissions granted to the roles determines whether a user can edit a dashboard or only run and view it or only view it or have no access to it. Users who have no permission to view a dashboard or folder will not see the dashboard or folder listed in the Navigator. They will also not see a dashboard in a folder, even if they have full permissions on the folder but none to the dashboard.

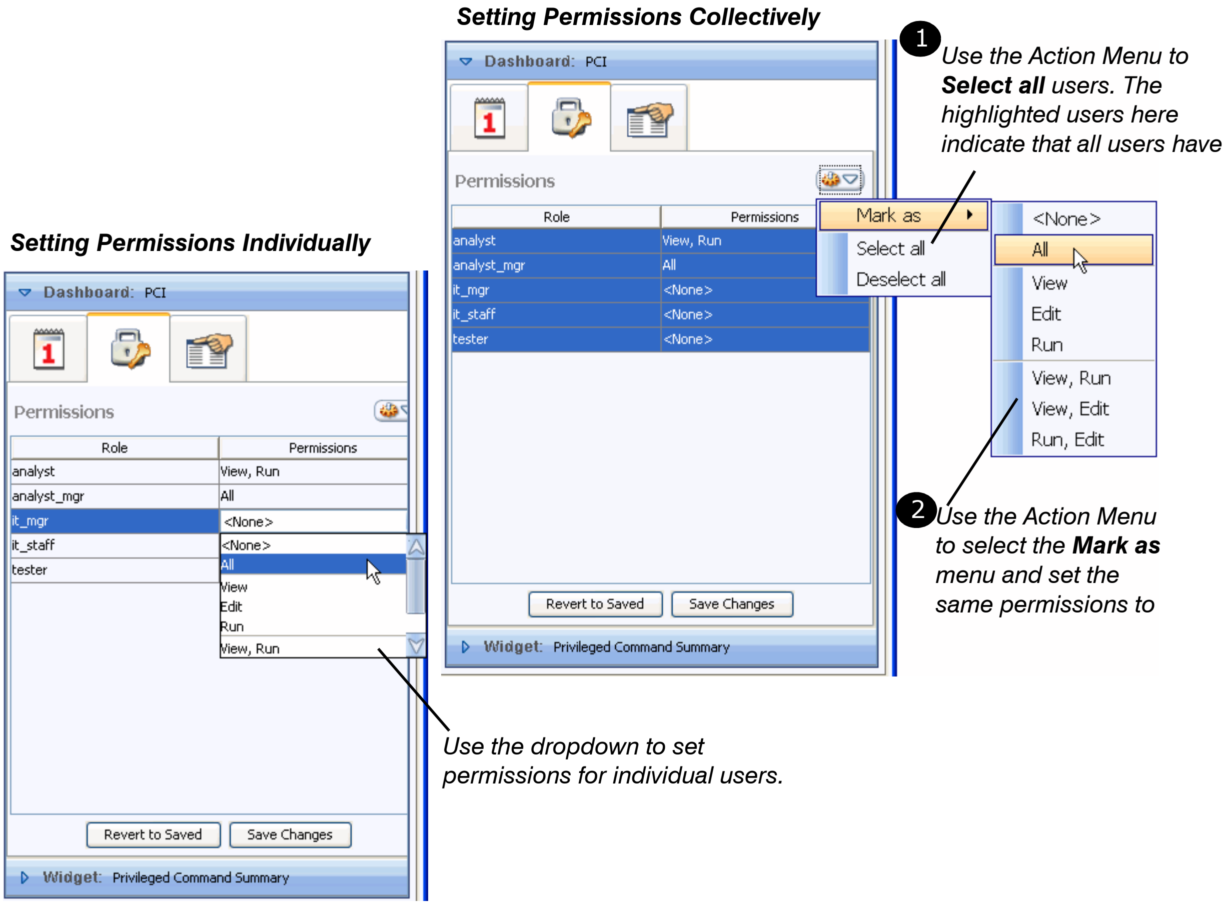
NOTE: Users gain the cumulative set of permissions from all roles to which they are assigned. In other words, if you are assigned to the Human Resources role, which has no permission to access an item, and to the Investigations role, which has permission to run and view a report, you have permission to run and view the report. For more information, see:

- Managing Access to HawkEye AP Console Reports, Dashboards, and Folders in Chapter 8, “Administering Users and Authentication” in the *Administration Guide*
- Managing Access to HawkEye AP Console Reports, Dashboards, and Folders in Chapter 8, “Administering Users and Authentication” in the *Administration Guide*.

Figure 6-23 illustrates the **Permissions** tab for the PCI dashboard. This tab, which functions identically for folder permissions, enables you to set dashboard permissions to specific roles and to view the assigned permissions. A user with administration permission creates and modifies the permissions themselves in Administration mode.

As illustrated below, you can specify permissions separately for each role from the dropdown next to each role. Alternately, you can select all roles and then apply the same permission to all roles.

Figure 6-23: Dashboard Permissions Tab



Setting Permissions Collectively

Setting Permissions Individually

Role	Permissions
analyst	View, Run
analyst_mgr	All
it_mgr	<None>
it_staff	<None>
tester	<None>

Use the Action Menu to ***Select all*** *users. The highlighted users here indicate that all users have*

Use the Action Menu to select the ***Mark as*** *menu and set the same permissions to*

Use the dropdown to set permissions for individual users.

The **Permissions** tab lists all roles defined for your EDW instance.

Running Dashboards

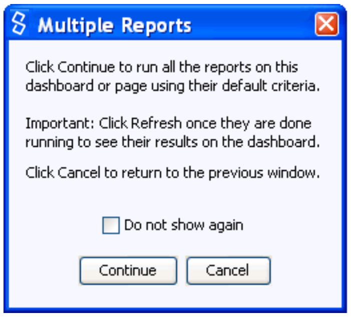
Typically, schedules run the dashboard. However, there are times that you want to run it manually. “Specifying Date Options”, on page 217 provides a few reasons you would run a dashboard manually.

To run a dashboard

1 Select the dashboard in the Navigator.

2 Select **Run** > **Dashboard...** from the Workspace Action Menu.

The Multiple Reports dialog displays, as shown below.



Multiple Reports

Click Continue to run all the reports on this dashboard or page using their default criteria.

Important: Click Refresh once they are done running to see their results on the dashboard.

Click Cancel to return to the previous window.

□ Do not show again

3 Click **Continue** to run all reports on the dashboard.

4 When the run completes, click **Refresh** to display the latest cache entry for each report.

NOTE: You can also run a page or individual widget from the Run menu accessed from the Workspace Action Menu.

Deleting Dashboards

You delete dashboards from the Navigator or folder in which they exist. You can only delete one dashboard at a time. To do so, right-click the dashboard and select the **Delete...** menu option. A dialog displays that prompts you to verify the deletion. Click **Yes** to delete.

Deploying Dashboards to Your Users

After you create a dashboard and assign it permissions, it becomes available to all users that access HawkEye AP Console in the same HawkEye AP installation and who have permission to access the dashboard. Users who do not currently have HawkEye AP Console open will see the dashboard the next time they open the console. Users who do have HawkEye AP Console open, must click **Refresh** to see the dashboard in their console.

CHAPTER 7

Creating and Editing Schedules

This chapter contains the following sections:

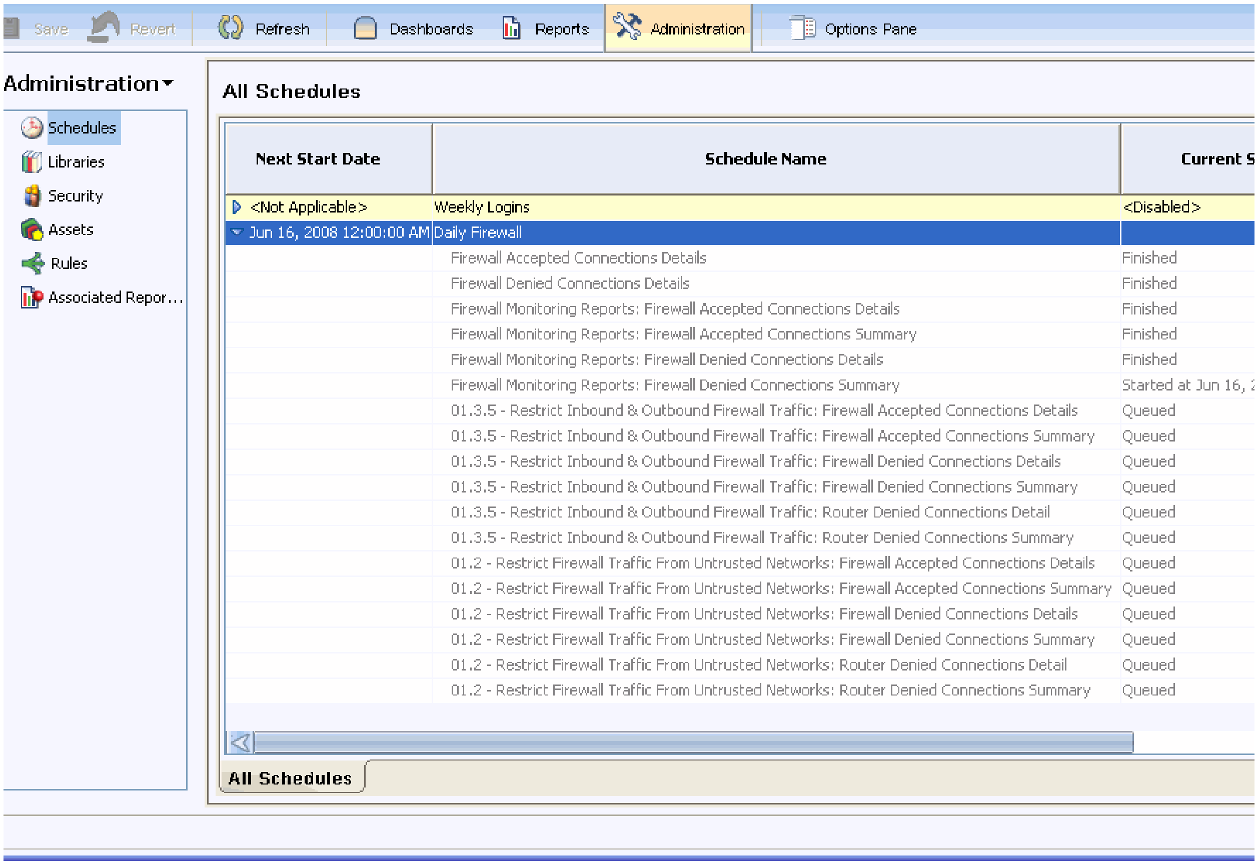
- “Overview”, next
- “Creating Schedules”, on page 232
- “Editing and Deleting Schedules”, on page 244

Overview

This chapter describes how to create and manage a *schedule*, which is a specification that determines when reports and dashboards are run and where and how the results are delivered. In addition to scheduling reports and dashboards, you can also schedule folders of reports and folders of dashboards.

You create and manage schedules from the **Schedules** option in Administration mode. Figure 7-1 illustrates the left side of the window that displays when you access the Schedules module.

Figure 7-1: Displaying the All Schedules Window



Next Start Date	Schedule Name	Current S
<Not Applicable>	Weekly Logins	<Disabled>
Jun 16, 2008 12:00:00 AM	Daily Firewall	
	Firewall Accepted Connections Details	Finished
	Firewall Denied Connections Details	Finished
	Firewall Monitoring Reports: Firewall Accepted Connections Details	Finished
	Firewall Monitoring Reports: Firewall Accepted Connections Summary	Finished
	Firewall Monitoring Reports: Firewall Denied Connections Details	Finished
	Firewall Monitoring Reports: Firewall Denied Connections Summary	Started at Jun 16, 2
	01.3.5 - Restrict Inbound & Outbound Firewall Traffic: Firewall Accepted Connections Details	Queued
	01.3.5 - Restrict Inbound & Outbound Firewall Traffic: Firewall Accepted Connections Summary	Queued
	01.3.5 - Restrict Inbound & Outbound Firewall Traffic: Firewall Denied Connections Details	Queued
	01.3.5 - Restrict Inbound & Outbound Firewall Traffic: Firewall Denied Connections Summary	Queued
	01.3.5 - Restrict Inbound & Outbound Firewall Traffic: Router Denied Connections Detail	Queued
	01.3.5 - Restrict Inbound & Outbound Firewall Traffic: Router Denied Connections Summary	Queued
	01.2 - Restrict Firewall Traffic From Untrusted Networks: Firewall Accepted Connections Details	Queued
	01.2 - Restrict Firewall Traffic From Untrusted Networks: Firewall Accepted Connections Summary	Queued
	01.2 - Restrict Firewall Traffic From Untrusted Networks: Firewall Denied Connections Details	Queued
	01.2 - Restrict Firewall Traffic From Untrusted Networks: Firewall Denied Connections Summary	Queued
	01.2 - Restrict Firewall Traffic From Untrusted Networks: Router Denied Connections Detail	Queued
	01.2 - Restrict Firewall Traffic From Untrusted Networks: Router Denied Connections Summary	Queued

As illustrated above, the **All Schedules** list displays when you select the **Schedules** option from Administration mode.

The **All Schedules** list provides the name and status of each schedule and the number of items assigned to it. For each enabled schedule, the **All Schedules** list also provides the next date and time that the schedule will run as well as the amount of time it took to run each of its items at the last run.

Figure 7-1 illustrates all items assigned to the Daily Firewall schedule. As indicated in the **Current Status** column, five reports have finished running, one has started running, and twelve are queued to run. Figure 7-3 illustrates the **Last Successful Run: Duration (End Time)** column, which indicates the total run time of each report on its last run and the time of the last run. As indicated by the column header, the end time is enclosed within parentheses.

In the illustrated example, three “finished” reports took 15 seconds to run, and two took no seconds. You can use the information in this column to determine when and how to run the schedule. For example, if the scheduled items take longer to run than the schedule period allows, you could expand the schedule period or split the items into different schedules that run at times that do not conflict.

By default, the information for each schedule is collapsed. To display information about each scheduled item Administration Guide for each schedule, click the closed icon ▷ to expand the row.

The **Current Status** column displays the status of the schedule as a whole as well as the status of each item in an expanded schedule:

- A running schedule displays no status but its **Next Start Date** column displays the date and time it will next run. When you expand the schedule, the status of each of its items displays.
- A disabled schedule displays `<Disabled>`. Its **Next Start Date** column displays `<Not Applicable>`.

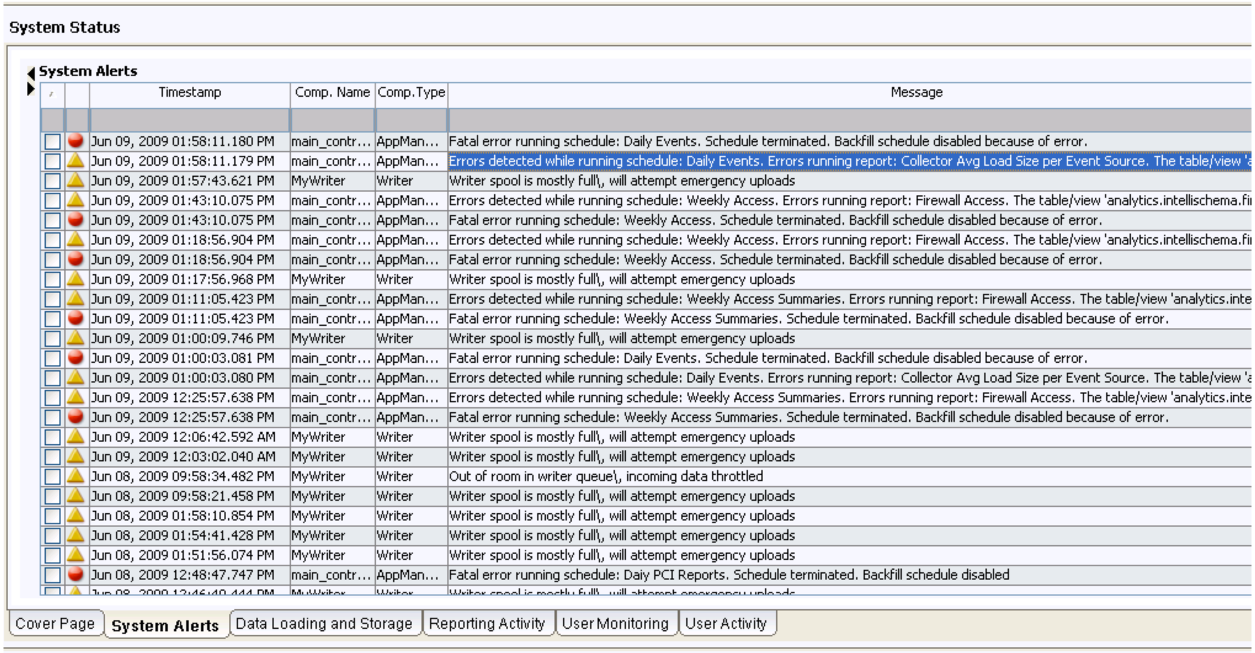
  If a running schedule encounters a problem in any of its items, the status of the schedule changes to `<Disabled>` and the status of the failed item changes to `<Error>`. The `<Error>` status is a link. Click this link to display a dialog box with the error message. After the item runs successfully, the Error link disappears. Other items for the same schedule might display a status of `Finished` or `Queued` or, if currently running, indicate the time the run began.

  NOTE: Many events can cause a schedule to fail. For example, a schedule can be defined that emails PDF files of two different reports to a user who has permission to access only one of the reports. Such a schedule fails before it attempts to send the email. Alternately, an administrator might restart the Real-Time system, which also causes schedule failure. An administrator can disable a schedule when she becomes aware of a potential problem and investigate the problem before she enables the schedule again. For information on investigating schedule failures, see “Configuring Logging for HawkEye AP Console”, on page 184 in the *Administration Guide*.

IMPORTANT: If the HawkEye AP Application Manager component is not running when a schedule runs, the scheduled item runs when the Application Manger is restarted. If this is not desired behavior, disable the schedules before stopping the Application Manager.

NOTE: A user who does not have permission to access the Schedule module but does have permission to access alerts, can locate the error in the System Alerts widget on the dashboard. Figure 7-2 illustrates a schedule error opened in the System Alerts widget.

Figure 7-2: Viewing a Schedule Error in the System Alerts Widget on a Dashboard

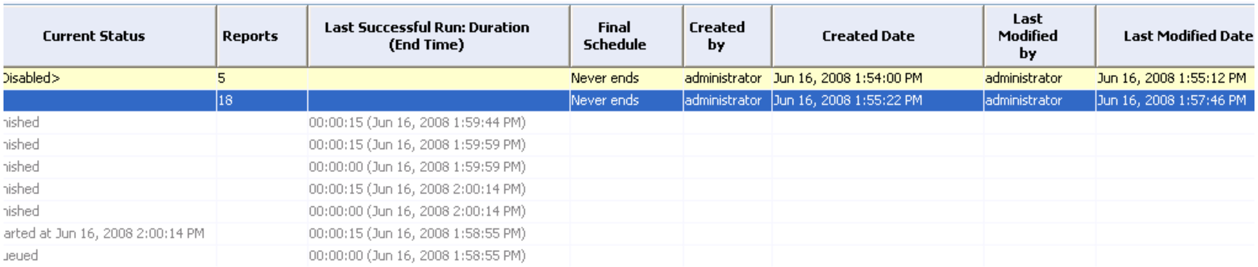


		Timestamp	Comp. Name	Comp.Type	Message
☐	●	Jun 09, 2009 01:58:11.180 PM	main_contr...	AppMan...	Fatal error running schedule: Daily Events. Schedule terminated. Backfill schedule disabled because of error.
☐	⚠	Jun 09, 2009 01:58:11.179 PM	main_contr...	AppMan...	Errors detected while running schedule: Daily Events. Errors running report: Collector Avg Load Size per Event Source. The table/view 'a
☐	⚠	Jun 09, 2009 01:57:43.621 PM	MyWriter	Writer	Writer spool is mostly full\, will attempt emergency uploads
☐	⚠	Jun 09, 2009 01:43:10.075 PM	main_contr...	AppMan...	Errors detected while running schedule: Weekly Access. Errors running report: Firewall Access. The table/view 'analytics.intellischema.fi
☐	●	Jun 09, 2009 01:43:10.075 PM	main_contr...	AppMan...	Fatal error running schedule: Weekly Access. Schedule terminated. Backfill schedule disabled because of error.
☐	⚠	Jun 09, 2009 01:18:56.904 PM	main_contr...	AppMan...	Errors detected while running schedule: Weekly Access. Errors running report: Firewall Access. The table/view 'analytics.intellischema.fi
☐	●	Jun 09, 2009 01:18:56.904 PM	main_contr...	AppMan...	Fatal error running schedule: Weekly Access. Schedule terminated. Backfill schedule disabled because of error.
☐	⚠	Jun 09, 2009 01:17:56.968 PM	MyWriter	Writer	Writer spool is mostly full\, will attempt emergency uploads
☐	⚠	Jun 09, 2009 01:11:05.423 PM	main_contr...	AppMan...	Errors detected while running schedule: Weekly Access Summaries. Errors running report: Firewall Access. The table/view 'analytics.inte
☐	●	Jun 09, 2009 01:11:05.423 PM	main_contr...	AppMan...	Fatal error running schedule: Weekly Access Summaries. Schedule terminated. Backfill schedule disabled because of error.
☐	⚠	Jun 09, 2009 01:00:09.746 PM	MyWriter	Writer	Writer spool is mostly full\, will attempt emergency uploads
☐	●	Jun 09, 2009 01:00:03.081 PM	main_contr...	AppMan...	Fatal error running schedule: Daily Events. Schedule terminated. Backfill schedule disabled because of error.
☐	⚠	Jun 09, 2009 01:00:03.080 PM	main_contr...	AppMan...	Errors detected while running schedule: Daily Events. Errors running report: Collector Avg Load Size per Event Source. The table/view 'a
☐	⚠	Jun 09, 2009 12:25:57.638 PM	main_contr...	AppMan...	Errors detected while running schedule: Weekly Access Summaries. Errors running report: Firewall Access. The table/view 'analytics.inte
☐	●	Jun 09, 2009 12:25:57.638 PM	main_contr...	AppMan...	Fatal error running schedule: Weekly Access Summaries. Schedule terminated. Backfill schedule disabled because of error.
☐	⚠	Jun 09, 2009 12:06:42.592 AM	MyWriter	Writer	Writer spool is mostly full\, will attempt emergency uploads
☐	⚠	Jun 09, 2009 12:03:02.040 AM	MyWriter	Writer	Writer spool is mostly full\, will attempt emergency uploads
☐	⚠	Jun 08, 2009 09:58:34.482 PM	MyWriter	Writer	Out of room in writer queue\, incoming data throttled
☐	⚠	Jun 08, 2009 09:58:21.458 PM	MyWriter	Writer	Writer spool is mostly full\, will attempt emergency uploads
☐	⚠	Jun 08, 2009 01:58:10.854 PM	MyWriter	Writer	Writer spool is mostly full\, will attempt emergency uploads
☐	⚠	Jun 08, 2009 01:54:41.428 PM	MyWriter	Writer	Writer spool is mostly full\, will attempt emergency uploads
☐	⚠	Jun 08, 2009 01:51:56.074 PM	MyWriter	Writer	Writer spool is mostly full\, will attempt emergency uploads
☐	●	Jun 08, 2009 12:48:47.747 PM	main_contr...	AppMan...	Fatal error running schedule: Daiy PCI Reports. Schedule terminated. Backfill schedule disabled
☐	⚠	Jun 08, 2009 12:46:40.444 PM	MyWriter	Writer	Writer spool is mostly full\, will attempt emergency uploads

For more information on monitoring system alerts, see Chapter 10: Administering Assets and Monitoring Alerts in the *Administration Guide*.

The **All Schedules** list also provides metadata that indicates each schedule's end date, creation date, and last modification date, as well as the user that created and last modified the schedule. Figure 7-3 illustrates the metadata columns that are hidden in Figure 7-1.

Figure 7-3: Viewing the Right Side of the All Schedules Window: Displaying Schedule Metadata



Current Status	Reports	Last Successful Run: Duration (End Time)	Final Schedule	Created by	Created Date	Last Modified by	Last Modified Date
)isabled>	5		Never ends	administrator	Jun 16, 2008 1:54:00 PM	administrator	Jun 16, 2008 1:55:12 PM
	18		Never ends	administrator	Jun 16, 2008 1:55:22 PM	administrator	Jun 16, 2008 1:57:46 PM
ıished		00:00:15 (Jun 16, 2008 1:59:44 PM)					
ıished		00:00:15 (Jun 16, 2008 1:59:59 PM)					
ıished		00:00:00 (Jun 16, 2008 1:59:59 PM)					
ıished		00:00:15 (Jun 16, 2008 2:00:14 PM)					
ıished		00:00:00 (Jun 16, 2008 2:00:14 PM)					
arted at Jun 16, 2008 2:00:14 PM		00:00:15 (Jun 16, 2008 1:58:55 PM)					
ɹeued		00:00:00 (Jun 16, 2008 1:58:55 PM)					

Most schedules are defined to run continuously. Their **Final Schedule** column displays `Never ends`. There are times, however, that you run the scheduled items only a limited number of times. For example, assume your analysis of recent reports indicates a potential problem that may have begun in the past. You can create a schedule that runs all relevant reports over the suspicious time period. In this case, you might schedule all items to run once and to delete the schedule after the run. Such a schedule displays the last time the schedule ran in the **Final Schedule** column. For more information, see "Specifying Lifetime", on page 235.

CREATING SCHEDULES

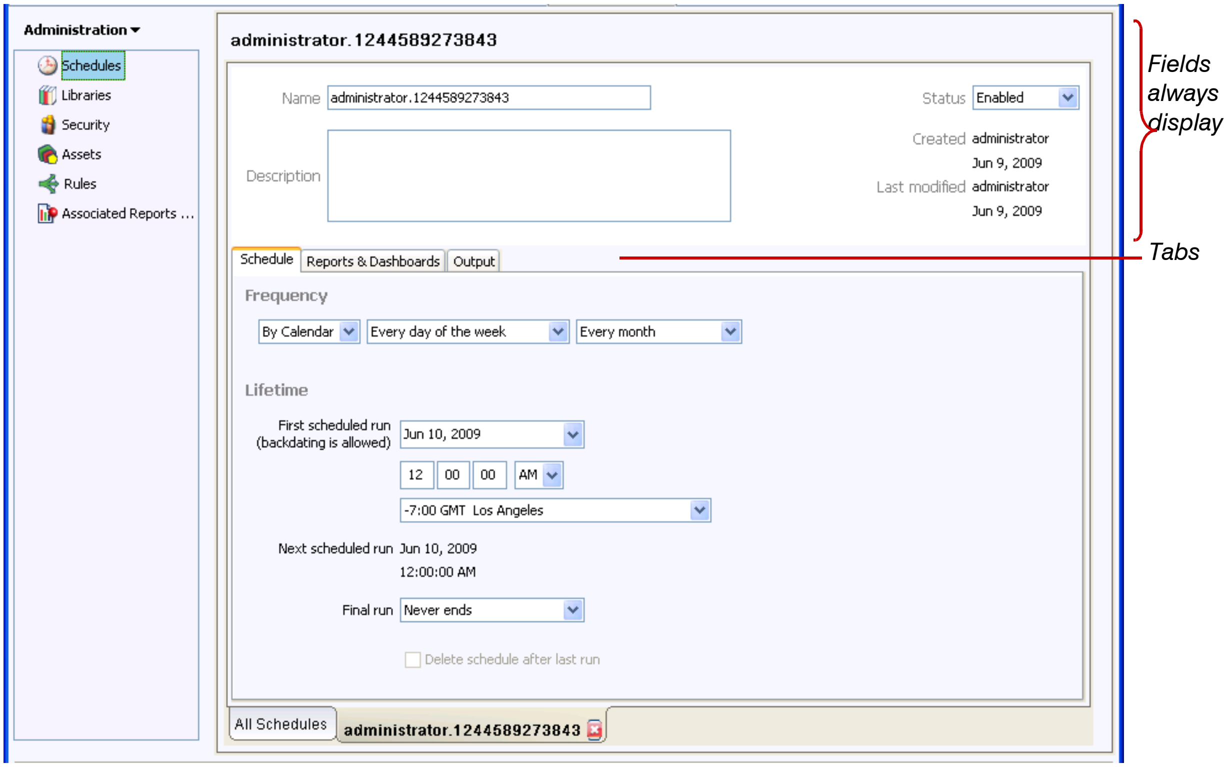
The following topics describe how to create a schedule:

- "Naming and Describing a Schedule", next
- "Schedule Tab: Setting Frequency and Lifetime", on page 233
- "Reports & Dashboards Tab: Selecting and Deleting Items to Schedule", on page 238
- "Output Tab: Specifying Destination", on page 241
- "Enabling and Disabling A Schedule", on page 244

Naming and Describing a Schedule

To create or manage a schedule, select **Schedules** from the Navigator in Administration mode. You create a schedule by selecting **New Schedule** from the Workspace Action Menu or right clicking in the workspace. Figure 7-4 illustrates the schedule-creation window that displays.

Figure 7-4: Specifying a Schedule's Name, Description, Frequency, and Lifetime



administrator.1244589273843

Name: administrator.1244589273843

Description:

Status: Enabled

Created: administrator Jun 9, 2009

Last modified: administrator Jun 9, 2009

Frequency

By Calendar | Every day of the week | Every month

Lifetime

First scheduled run (backdating is allowed): Jun 10, 2009 12 00 00 AM -7:00 GMT Los Angeles

Next scheduled run: Jun 10, 2009 12:00:00 AM

Final run: Never ends

□ Delete schedule after last run

administrator.1244589273843

As illustrated above, the schedule-creation window provides fields at the top that remain displayed throughout the creation process:

- **Name**—Sensage recommends that you enter a name that uniquely identifies the schedule by frequency and items scheduled, for example, `Monthly Firewall` and `Hourly Windows Logins`.

- **Description**—You can enter a short description that clarifies the schedule's purpose.

- **Status**—By default, a new schedule is enabled as soon as you save it. If you do not save final changes, however, the status changes to `Disabled`. You can use this field to manually change the value of a running schedule to disable it.

In addition to the three fields that you can modify, the top of the window displays metadata that provides the name of the users that created and last modified the schedule and the creation and modification dates. These values also display in the **All Schedules** list, as illustrated in Figure 7-3.

Below these fields are three tabs that enable you to specify different schedule features. These tabs, which are illustrated in Figure 7-4, are:

- **Schedule**—enables you to set the frequency and lifetime. For more information, see "Schedule Tab: Setting Frequency and Lifetime", next.
- **Reports & Dashboards**—enables you to associate items for scheduling. For more information, see "Reports & Dashboards Tab: Selecting and Deleting Items to Schedule", on page 238.
- **Output**—enables you to specify the destination of the scheduled items, such as emailed output or dashboard alert. For more information, see "Output Tab: Specifying Destination", on page 241.

Schedule Tab: Setting Frequency and Lifetime

As illustrated in Figure 7-4, the **Schedule** tab provides the following fields:

- **Frequency**—Use this field to specify how often the schedule runs.
- **Lifetime**—Use this field to specify when scheduling begins and when, if ever, it ends. For more information, see "Specifying Lifetime", on page 235.

Specifying Frequency

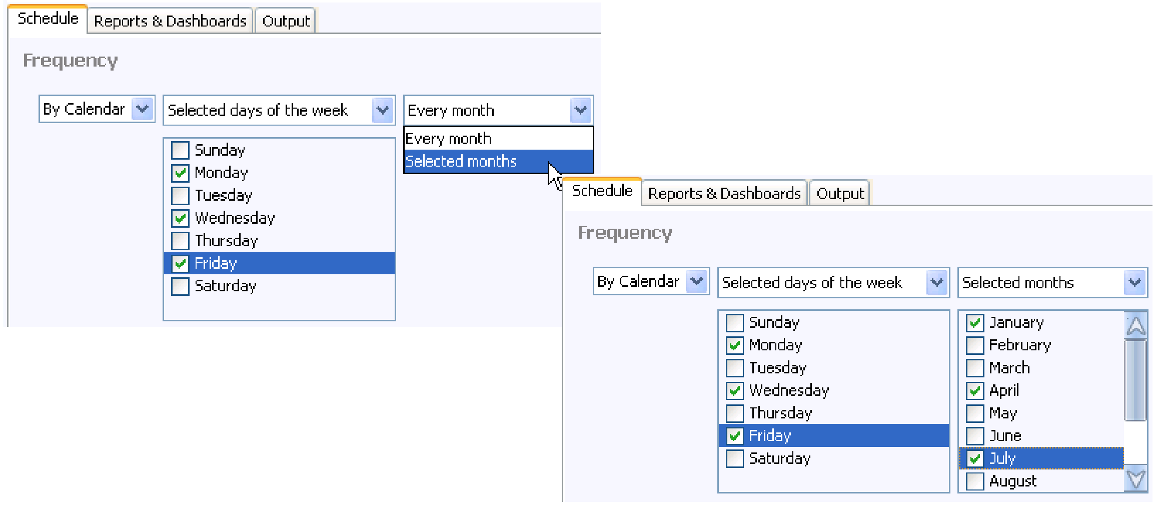
This field provide two options that offer very different granularity. You can schedule:

- **By Calendar**—This options enables you to set the frequency to every day of the week, selected days of the week, or selected days of the month. You can also specify whether the schedule runs every month or only selected months.
- **Run Every**—This option enables you to set the frequency to a specified number of hours or minutes.

By Calendar

Figure 7-4 illustrates the default calendar setting: every day of the week every month. Figure 7-5 illustrates a different calendar option: selected days of the week for selected months.

Figure 7-5: Specifying Days of the Week



By Calendar | Selected days of the week | Every month

Every month / Selected months

☐ Sunday
☑ Monday
☐ Tuesday
☑ Wednesday
☐ Thursday
☑ Friday
☐ Saturday

By Calendar | Selected days of the week | Selected months

☐ Sunday
☑ Monday
☐ Tuesday
☑ Wednesday
☐ Thursday
☑ Friday
☐ Saturday

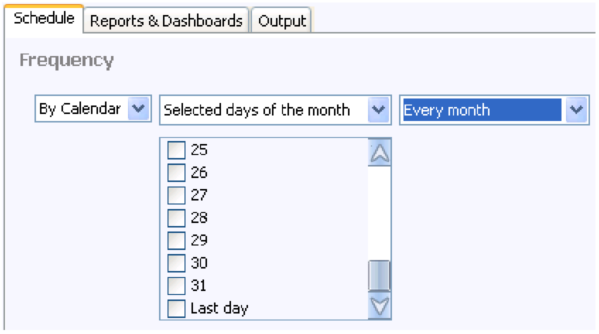
☑ January
☐ February
☐ March
☑ April
☐ May
☐ June
☑ July
☐ August

As illustrated above, you can select specific days to run the schedule. You can also specify whether the schedule runs on those days every month or only selected months.

Figure 7-6 illustrates how you can specify selected days of the month. This option provides a selection field for every possible day of the month, including the relative value “`Last day`”. The `Last day` value allows you run a report at the very end of every month regardless of the number of days the months contains.

Like the option to specify days of the week, you can set the schedule to run on the specified days every month or only specified months.

Figure 7-6: Specifying Days of the Month



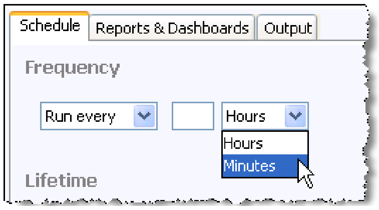
By Calendar | Selected days of the month | Every month

☐ 25
☐ 26
☐ 27
☐ 28
☐ 29
☐ 30
☐ 31
☐ Last day

Run Every

Figure 7-7 illustrates how to set the schedule to run by hours or minutes rather than by days, weeks, months, or years.

Figure 7-7: Specifying Days of the Week



Run every
Hours

As illustrated above, you can set the schedule to run at a specified time interval. Enter the desired number in the empty entry field.

NOTE:

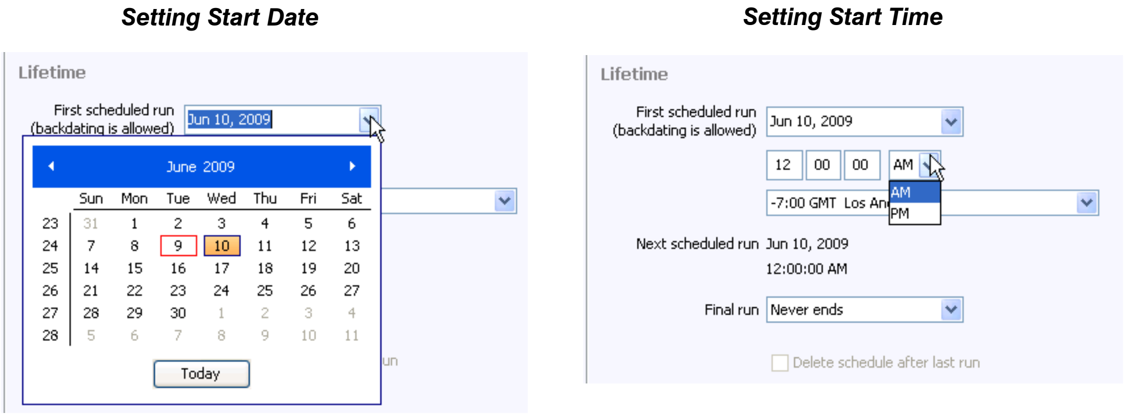
- When you schedule a report to run every few hours or a specified number of minutes, the report definition should also be set to run over such a short duration.
- In Dashboards mode, you can specify that a specific number of the most recent report cache entries be automatically merged into a single result set. In other words, you can schedule a virtual report that merges multiple existing cache entries. The merged results persist only for the current viewing. For example, assume a report collects data on an hourly basis. You can specify that the most recent 24 hourly report cache entries be automatically merged every day into a virtual daily report. For more information, see "Automatically Merging Cache Entries", on page 224.

Specifying Lifetime

Specifying Start Date and Time

By default, each schedule begins running when the current day ends; that is, at midnight today. You can change the date and time by editing the **First scheduled run** field. Alternately, you can change the date by selecting the desired date from the preview calendar. As illustrated in Figure 7-8, click the dropdown icon to display the calendar. Also illustrated in this figure is the preview calendar, which encloses the current date in a box and changes the color of the default start date.

Figure 7-8: Specifying Start Date and Time



Setting Start Date

Setting Start Time

First scheduled run (backdating is allowed) Jun 10, 2009

First scheduled run (backdating is allowed) Jun 10, 2009

12 00 00 AM

-7:00 GMT Los An

Next scheduled run Jun 10, 2009 12:00:00 AM

Final run Never ends

□ Delete schedule after last run

Typically, you will change the default start time. For example, if the Collector finishes collecting and loading data at 4:00 am, you will want your schedule to start running after that time. In fact, you will want to allow a safety period to ensure that all data is loaded before you begin reporting on it. In this case, you might set the start time for 5:00 am.

The label for the **First scheduled run** field includes a note in parentheses that backdating is allowed. *Backdating* sets the schedule to begin earlier than the current day. Backdating occurs when you select a date or time earlier than the current date and time. When you select this option, a dialog advises that setting the start date earlier than the current date causes the scheduled reports to reflect past data and that creating backdated reports can take a long time. If you select this option, set the report to run at a time that will least interfere with other scheduled runs. You set the time in the field below **First scheduled run**.

After you select the start date, you can change the default time and time zone. These fields are illustrated in Figure 7-8. After you set the start date and time, the selected values display in the **Next scheduled run** field.

The time zone dropdown allows you to set a time zone that is different from your local one. For example, assume a New York user has defined a report whose date criteria specifies data collection from Monday through Friday in the Tokyo time zone. This user wants to schedule the report to run on Saturday in the Tokyo time zone to ensure that all data is collected before the report runs. When this user creates the schedule, he sets the time zone to Tokyo time and sets the report to run on Saturdays.

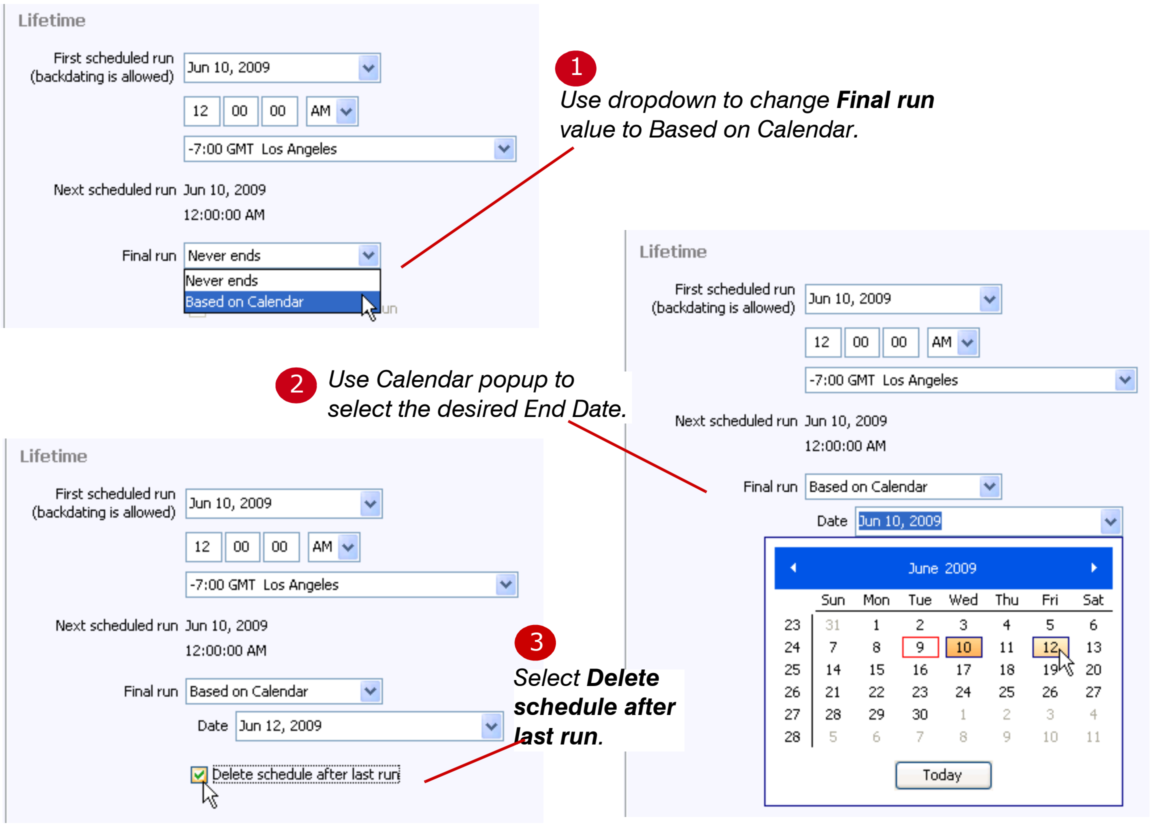
After the schedule runs and the user views the list of all schedules, he sees that all dates and times reflect his local New York time zone. Displaying the dates and times within a single time zone provides a consistent view of scheduled runs. This consistency enables the user to understand which schedules are currently running and which will run next. This information is critical to understanding whether the schedules are properly load balanced.

SPECIFYING END DATE

By default, each schedule begins running at midnight and has no termination date. If you create a schedule to investigate suspicious events, you might run the schedule only once or multiple times within a short period of time. In this case, you would specify the end date and set the schedule to delete after its last run. You might also want to run a report only once if the report is huge and you don't want to impact EDW performance during work hours. In this case, you would schedule this report to run once at night and would also set the schedule to delete after its last run.

Figure 7-9 illustrates the process for scheduling a report that is not ongoing.

Figure 7-9: Specifying End Date



1 *Use dropdown to change **Final run** value to Based on Calendar.*

2 *Use Calendar popup to select the desired End Date.*

3 *Select **Delete schedule after last run**.*

Lifetime

First scheduled run (backdating is allowed): Jun 10, 2009

12 : 00 : 00 AM

-7:00 GMT Los Angeles

Next scheduled run: Jun 10, 2009 12:00:00 AM

Final run: Never ends

Final run: Based on Calendar

Date: Jun 10, 2009

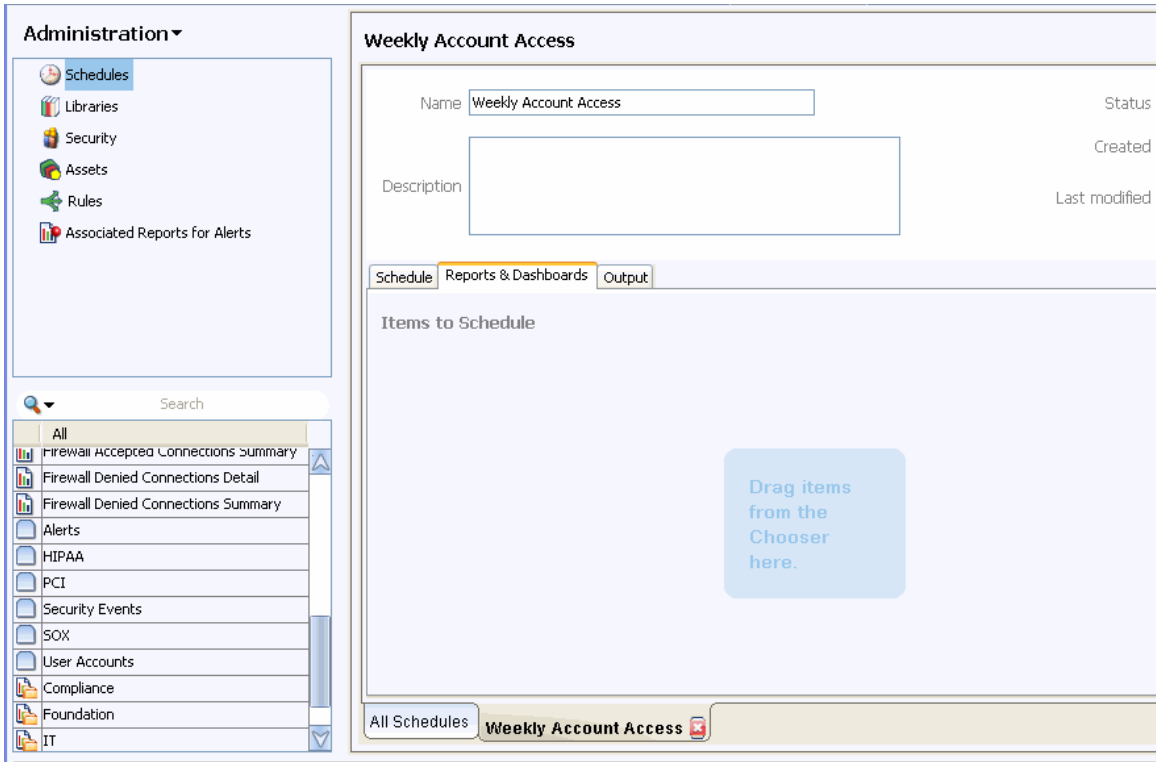
Date: Jun 12, 2009

☑ Delete schedule after last run

Reports & Dashboards Tab: Selecting and Deleting Items to Schedule

Figure 7-10 illustrates the tab in which you assign items to a schedule by dragging the item(s) from the Chooser to the **Reports & Dashboards** pane.

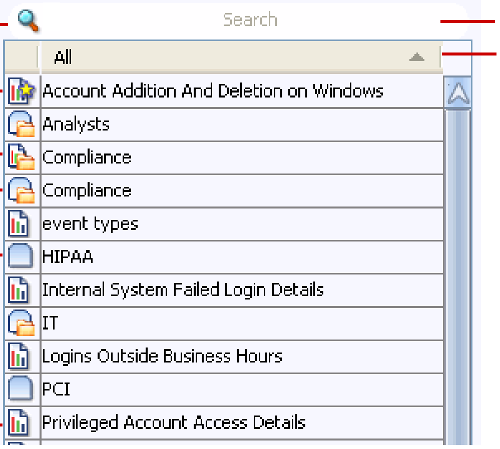
Figure 7-10: Assigning Items to the Schedule



Name: Weekly Account Access

As illustrated in Figure 7-11, you can drag many types of items to a schedule.

Figure 7-11: Example of Item Types in Schedule Assignment



Search
All
Account Addition And Deletion on Windows
Analysts
Compliance
Compliance
event types
HIPAA
Internal System Failed Login Details
IT
Logins Outside Business Hours
PCI
Privileged Account Access Details

Search field

Wizard Report

Folder of Reports

Folder of Dashboards

Dashboard

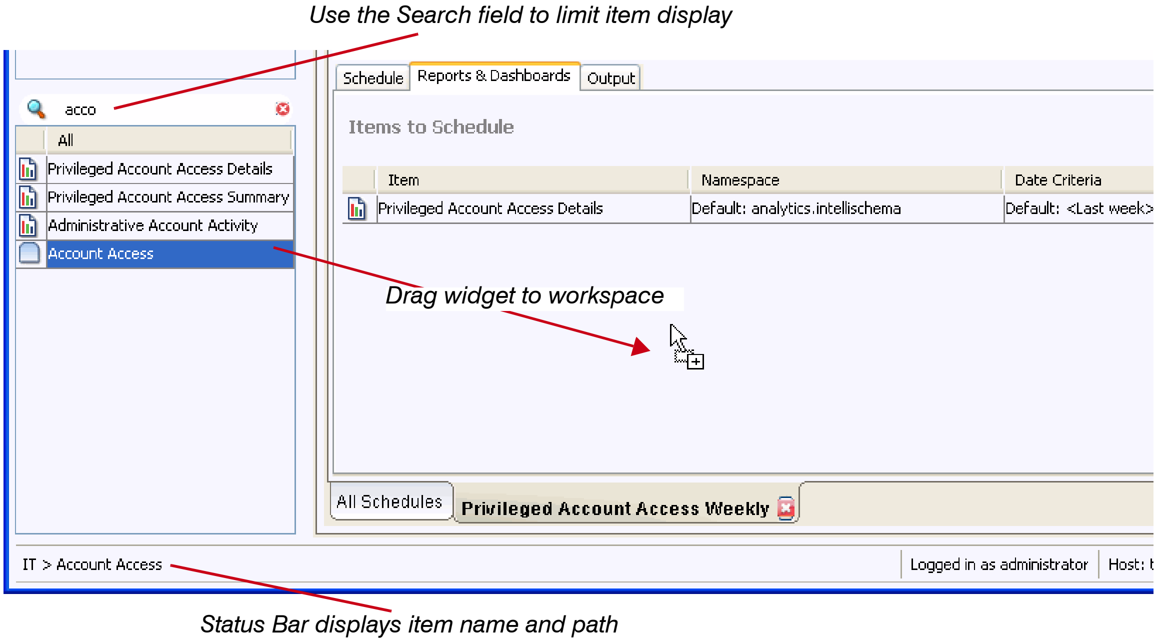
SQL Report

Search by item name

Click to sort

If the Chooser lists more items than can display at once, you can limit the displayed items by entering meaningful text in the **search** field or expand the size of the Chooser. After you limit the items, drag desired items to the workspace, as illustrated in Figure 7-12.

Figure 7-12: Limiting Displayed Items and Assigning Items

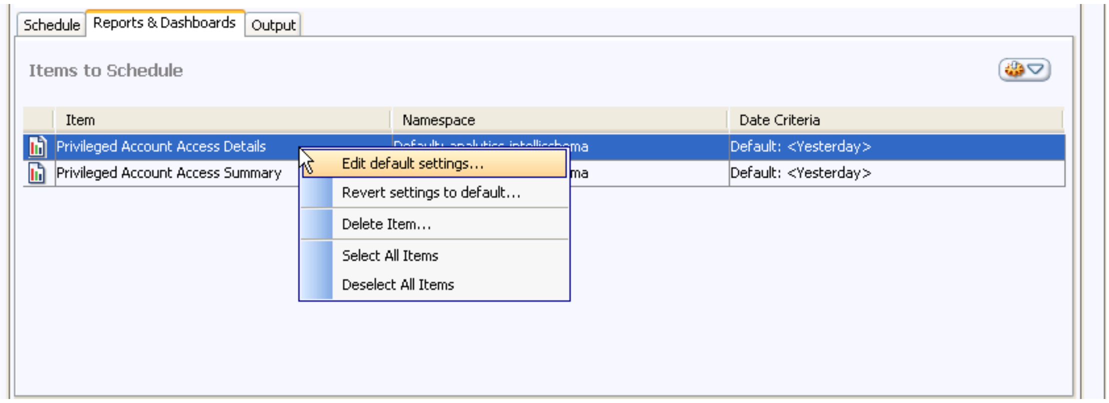

Use the Search field to limit item display
acco
All
Privileged Account Access Details
Privileged Account Access Summary
Administrative Account Activity
Account Access
Schedule
Reports & Dashboards
Output
Items to Schedule
Item
Namespace
Date Criteria
Privileged Account Access Details
Default: analytics.intellischema
Default: <Last week>
Drag widget to workspace
All Schedules
Privileged Account Access Weekly
IT > Account Access
Logged in as administrator
Host: t
Status Bar displays item name and path

The operation above illustrates how to assign a dashboard to the schedule. As the user drags the dashboard, its name displays in the status bar, below the Chooser. If the item is in a folder, the full path to the item displays. For example, because the Account Access dashboard is in the IT folder, the status bar displays `IT > Account Access`.

After you select desired items for scheduling, you can change each item's default namespace and date criteria. For example, assume you have created a single login report that runs in both the Unix and Windows namespaces. Its report definition defaults the report to run in the Windows namespace. You can create a single schedule that runs the report in both namespaces. To do so, drag the report item twice from the Chooser to the **Items to Schedule** pane. Keep the default namespace for one of the items and change the namespace of the other. If desired, you can also change the date criteria for one of these reports.

First select the report whose settings you want to change. Then select **Edit default settings ...** from the Action menu. Figure 7-13 illustrates these steps.

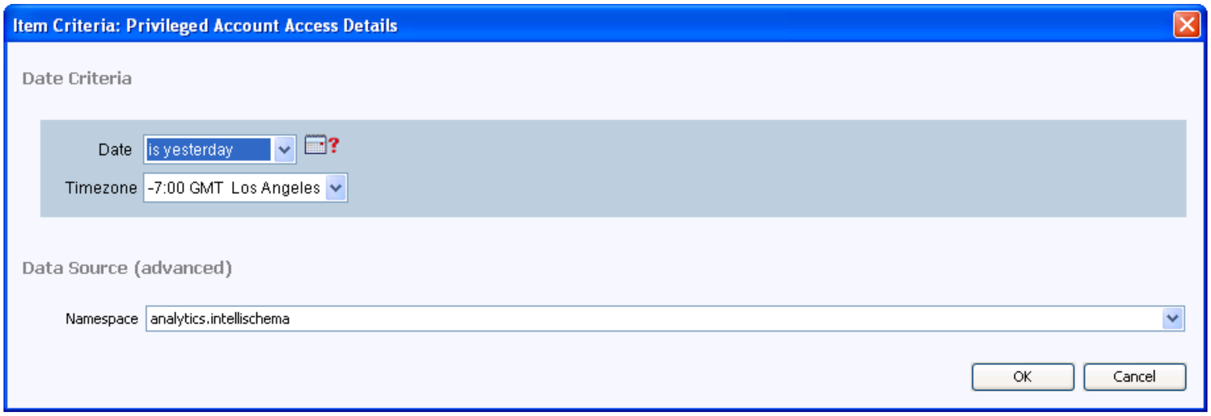
Figure 7-13: Changing An Item's Namespace or Date Criteria



Item	Namespace	Date Criteria
Privileged Account Access Details	Default: analytics.intellischema	Default: <Yesterday>
Privileged Account Access Summary	ma	Default: <Yesterday>

In the **Item Criteria** dialog for the selected item, make your desired changes. Figure 7-14 illustrates this dialog.

Figure 7-14: Dialog for Changing Criteria



Date: is yesterday
Timezone: -7:00 GMT Los Angeles
Namespace: analytics.intellischema

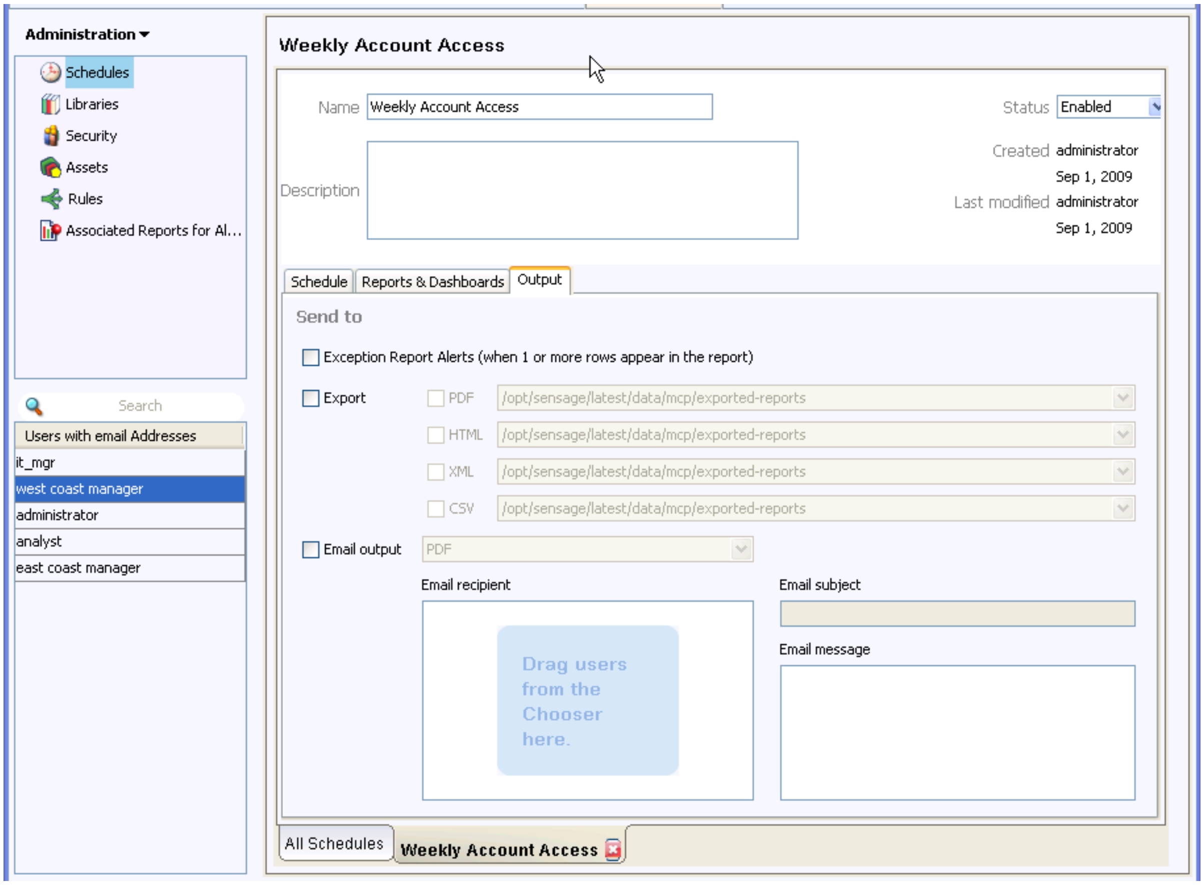
To change the criteria for all items at the same time, select **Select All Items** from the Action menu before you select **Edit default settings ...** . As illustrated in Figure 7-13, you can also use the Action menu to delete a selected item and to revert to default settings.

The last step in schedule creation is specifying who or what receives the scheduled items and the format of the items.

Output Tab: Specifying Destination

Figure 7-15 illustrates the tab in which you specify the format and destination of the scheduled items.

Figure 7-15: Specifying Destination



Weekly Account Access

Name: Weekly Account Access

Status: Enabled

Description:

Created: administrator Sep 1, 2009

Last modified: administrator Sep 1, 2009

Send to

□ Exception Report Alerts (when 1 or more rows appear in the report)

□ Export

□ PDF /opt/sensage/latest/data/mcp/exported-reports

□ HTML /opt/sensage/latest/data/mcp/exported-reports

□ XML /opt/sensage/latest/data/mcp/exported-reports

□ CSV /opt/sensage/latest/data/mcp/exported-reports

□ Email output PDF

Email recipient

Drag users from the Chooser here.

Email subject

Email message

As illustrated in Figure 7-15, the **Output** tab provides the following fields:

- **Exception Report Alerts**—Select this field to trigger a row to display in the Exception Alerts widget on a dashboard.

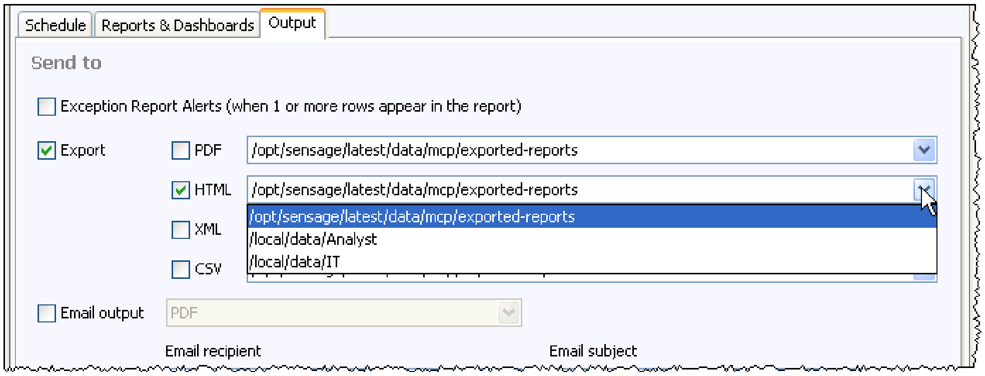
  One row displays in the Exception Alerts widget every time the scheduled report contains one or more rows. For example, you might schedule a report that lists after-hours logins. In this case, the scheduled report triggers a row in the Exception Alerts widget when a user logs into your system over the weekend or after close of business. The alert row in the widget lists the name of the report and the number of rows returned. You can open the report to view the results by right clicking the alert row in the Exception Alerts widget.

  NOTE: Because this feature applies to all reports in the schedule, schedule only reports that return values when a problem occurs.

  For more information, see "Viewing Exception Alerts", on page 84 and Monitoring Exception Alerts in Chapter 10, "Administering Assets and Monitoring Alerts" of the *Administration Guide*.

- **Export**—Use this field to specify the destination and format of the exported report (PDF, HTML, XML, or CSV).

  You can select all or a subset of formats. After you select a format, select its file-system destination from the associated dropdown, as illustrated below.

  Send to

  □ Exception Report Alerts (when 1 or more rows appear in the report)

  ☑ Export
  □ PDF /opt/sensage/latest/data/mcp/exported-reports
  ☑ HTML /opt/sensage/latest/data/mcp/exported-reports
  /opt/sensage/latest/data/mcp/exported-reports
  /local/data/Analyst
  /local/data/IT
  □ XML
  □ CSV

  □ Email output PDF

  Email recipient Email subject

  NOTE: Directories display in the dropdown only after an administrator has configured them as destination directories. For more information, see Specifying EDW Hosts in Chapter 2, “Configuring HawkEye AP” in the *Installation, Configuration, and Upgrade Guide*.

- **Email output**—Use this field to email the scheduled items to specified users; you can email in one of the following formats: PDF, HTML, XML, CSV, link. For more information, see “Emailing Scheduled Items”, next.

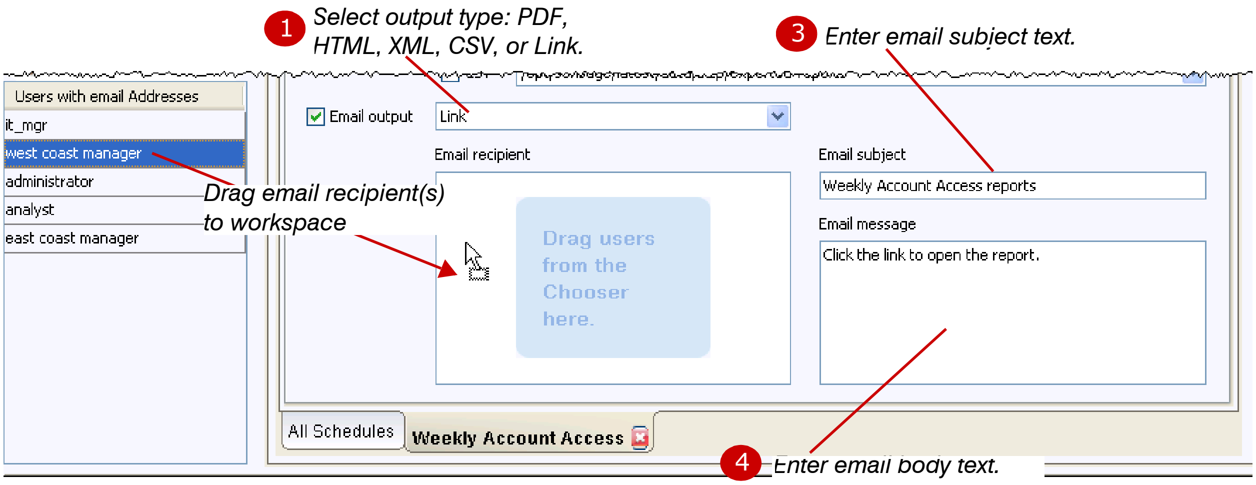
  Sometimes scheduled reports produce zero rows. You can configure HawkEye AP to suppress the sending of reports with zero rows. For more information, see Suppressing the Sending of Reports with Zero Rows in Chapter 2, “Configuring HawkEye AP” of the *Installation, Configuration, and Upgrade Guide*.

NOTE: HTML, XML, and CSV export format are available only for reports and report folders. These options are disabled when the schedule contains only dashboards or dashboard folders.

Emailing Scheduled Items

Figure 7-16 illustrates the process for emailing scheduled items.

Figure 7-16: Emailing Scheduled Items



1 Select output type: PDF, HTML, XML, CSV, or Link.

3 Enter email subject text.

Drag email recipient(s) to workspace

4 Enter email body text.

Users with email Addresses: it_mgr, west coast manager, administrator, analyst, east coast manager

☑ Email output: Link

Email recipient: Drag users from the Chooser here.

Email subject: Weekly Account Access reports

Email message: Click the link to open the report.

NOTE:

- If you specify **`Link`** as the email type, the email contains a link for each scheduled item.

  When you click each emailed link, HawkEye AP Console opens to the appropriate mode and displays the linked item. You will be prompted to log into HawkEye AP Console to enable it to open.

- The Chooser displays every user created for the current EDW instance for whom an email address has been specified.

  An administrator can create users in one of two ways:

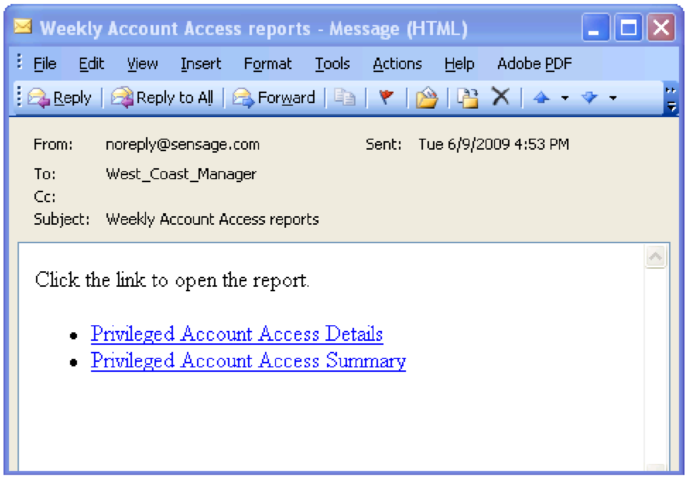
  - HawkEye AP Console Security option—accessed from Administration mode
  - Command line—using the `atmanage` utility.

    NOTE: Although an administrator can create users in either of the above ways, only the HawkEye AP Console Security module in Administration mode enables the administrator to specify an email address for a user. The administrator can add the email address in HawkEye AP Console after creating the user in the command line or HawkEye AP Console. However, it may take an hour before the user created in the command line displays in the console. For more information, see Chapter 8: Administering Users and Authentication in the *Administration Guide*.

- Although the email subject and body text are not required, entering this text is helpful to your recipients.

Figure 7-17 illustrates example email that contains links to two different reports.

Figure 7-17: Example Email With Report Links


Weekly Account Access reports - Message (HTML)
File Edit View Insert Format Tools Actions Help Adobe PDF
Reply | Reply to All | Forward

From: noreply@sensage.com
Sent: Tue 6/9/2009 4:53 PM
To: West_Coast_Manager
Cc:
Subject: Weekly Account Access reports

Click the link to open the report.

- Privileged Account Access Details
- Privileged Account Access Summary

Enabling and Disabling A Schedule

After you specify frequency and lifetime, assign the items to run, and set the desired destinations, you must save your changes and close the edit window to start the schedule. The schedule will begin running at its first scheduled time.

To disable a running schedule, you can do either of the following:

- Select it in the **All Schedules** list, right click, and select **Disabled** from the popup menu or the Action Menu.
- Open it for editing and select `Disabled` from the **Status** dropdown.

Editing and Deleting Schedules

To edit a schedule, select it in the **All Schedules** list, right click, and select **Edit** from the popup menu or the Action Menu.

- If the schedule is enabled but not running, it is automatically disabled when you can open its definition for editing. Although the **Status** field indicates that the schedule is `Enabled`, it is disabled for the entire time you have it open to edit. After you save your changes, you must close the schedule definition to enable it again.
- If the schedule is currently running, a dialog displays that prompts whether you want to disable the schedule. After you disable it, you must wait until the run completes before you can edit it. After you save your changes, you must close the schedule definition to enable it again.
- If a schedule has already run, the Edit window displays the first three fields in the Lifetime pane as disabled. You can no longer change the day and time of the first scheduled run. Nor can you change the schedule time zone. You can, however, still change the time zone of its scheduled objects, as described in "Reports & Dashboards Tab: Selecting and Deleting Items to Schedule", on page 238.

To delete a schedule, select it in the **All Schedules** list, right click, and select **Delete...** from the popup menu or the Action Menu. You will be prompted to confirm the delete.

CHAPTER 8

Creating Alerting Rules from Templates

The chapter contains the following sections:

For more information on the HawkEye AP Real-Time system, see “Processing Real-Time Streaming Events”, on page 18 in the *Event Collection Guide*.

OVERVIEW

HawkEye AP uses parsing rules to analyze and transform streaming log data from multiple systems in different formats into consistent data that it stores and makes available for analysis. Its Real-Time system uses alerting rules to examine the transformed data and raise alerts when it detects specified conditions in the event data. HawkEye AP Console displays the alerts in the Security Alerts widget on a dashboard, as described in “Viewing Security Alerts”, on page 64.

HawkEye AP provides multiple parser and alerting rules, which display in the HawkEye AP Console Rules module in Administration mode. The parsing and alerting rules are written in the HawkEye Event Processing Language (HEPL). HEPL is shipped with the HawkEye AP product to enable developers to write their own parsing and alerting rules. In addition to the parsing rules and alerting rules delivered with the product, HawkEye AP provides alerting rule templates that enable analysts to create their own alerting rules. Analysts can easily configure these to their specific needs.

HawkEye AP Console displays the rules in its Rules module. This chapter describes how the templates work and how to use HawkEye AP Console to create alerting rules from the templates. It also describes how to activate, deactivate, and delete HawkEye AP rules.

For information on the HEPL, see the *HawkEye Event Processing Language Developers Guide*.

INTRODUCTION TO ALERTING RULE TEMPLATES

HawkEye AP provides templates that enable and simplify the process for creating alerting rules. The template names identify the source of their data and their function. For example:

- `Unix login Substring Match` — matches any substring within a Unix login message
- `Windows Security (Windows Retriever) EVENTID Match` — matches any event ID across all Windows hosts
- `Syslog Host and Substring Match`— matches any substring on specified hosts for all Syslog events

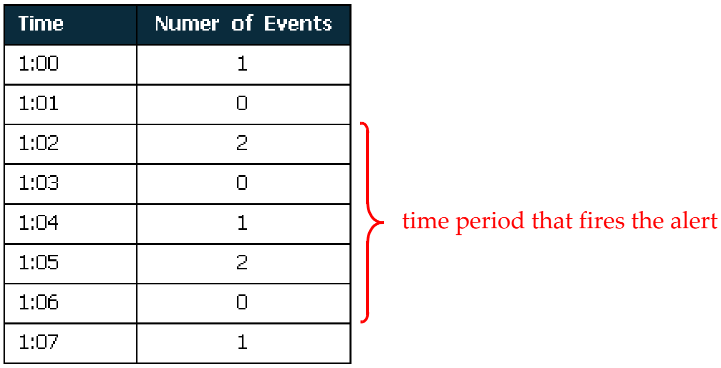
- `McAfee DLP Alert Threshold` — triggers a specified number of alerts when it detects a specified number of Data Loss Prevention (DLP) events from a single source host within a specified time period
- `Cisco PIX Substring Match`— matches any substring within Cisco PIX events

Each template provides criteria fields that enable analysts to customize the alert to their needs. These fields typically include the following:

- **Event Threshold**— A numeric field into which you specify the number of events before the rule fires an alert.
- **Event Window**— A numeric field into which you specify the time period the alert allocates to track the number of events.

  For example, if you set the **Event Threshold** to `5` and the **Event Window** to `300` seconds, the rule triggers an alert when 5 events occur within 5 minutes.

  NOTE: The alerting rules that HawkEye AP provides use a sliding time window. For example, assume you have set the **Event Threshold** to `5` and the **Event Window** to `300` seconds. The graphic below illustrates how the 5-minute window slides to ensure that the rule captures all events within the specified event window. Within the first 5-minute period beginning at 1:00, too few events occur to fire an alert. However, because the 5-minute period begins again at 1:01 and 1:02, enough events occur during the period beginning at 1:02 to fire the alert.



Time	Numer of Events
1:00	1
1:01	0
1:02	2
1:03	0
1:04	1
1:05	2
1:06	0
1:07	1

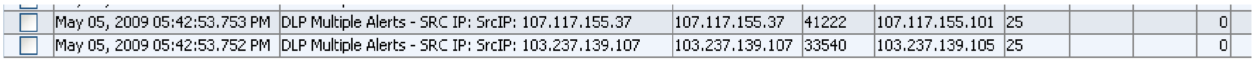
time period that fires the alert

- **Alert Threshold**— A numeric field into which you specify the maximum number of alerts triggered before the rule suppresses triggering additional alerts.

  IMPORTANT: Limiting the number of alerts fired protects your system from being flooded by redundant alerts. For more information, see "Understanding Alert Thresholds and Alert Windows", on page 250.

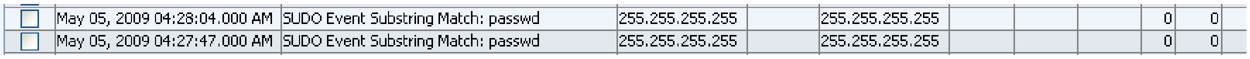
- **Alert Window**— A numeric field into which you specify the time period the alert allocates to track the number of alerts.
- **Alert Name**— A text field into which you uniquely identify the alert in the Security Alerts widget's **Name** field.

  Some alerts append the match string to the end of the alert name. For these alerts, it's helpful to include in the **Alert Name** the meaning of the string. For example, the `McAfee DLP Alert Threshold` rule displays the source IP that triggered the alert. Therefore, you might name your rule: "`DLP Multiple Alerts -- SRC IP:` ". The graphic below illustrates two rows in the Security Alert widget for an alert with this name.



□	May 05, 2009 05:42:53.753 PM	DLP Multiple Alerts - SRC IP: SrcIP: 107.117.155.37	107.117.155.37	41222	107.117.155.101	25			0
□	May 05, 2009 05:42:53.752 PM	DLP Multiple Alerts - SRC IP: SrcIP: 103.237.139.107	103.237.139.107	33540	103.237.139.105	25			0

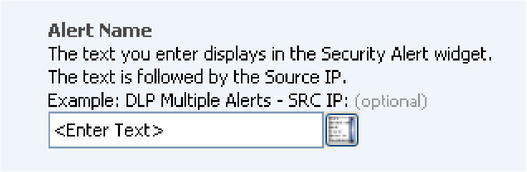
Most alerts do not append the match string to the end of the alert name. For these alerts, it’s helpful to include the string in the name. For example, the `Unix sudo Substring Match` rule matches a substring within the Unix sudo message but does not display the string in the alert name. Therefore, you might identify the match string in the rule’s name; for example: “`Unix sudo Substring Match -- password`“. The graphic below illustrates two rows in the Security Alert widget for an alert with this name.



□	May 05, 2009 04:28:04.000 AM	SUDO Event Substring Match: passwd	255.255.255.255		255.255.255.255				0	0
□	May 05, 2009 04:27:47.000 AM	SUDO Event Substring Match: passwd	255.255.255.255		255.255.255.255				0	0

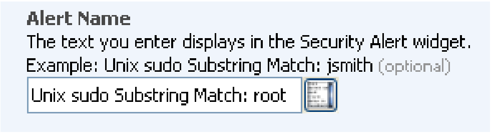
NOTE: Each alerting rule template that HawkEye AP provides indicates whether the rule appends the search text to the rule name. The graphic below provides an example from two rule templates, one of which appends the search value and one of which does not.

Rule template explains that the search value is appended to the rule name and provides an example name



Alert Name
The text you enter displays in the Security Alert widget.
The text is followed by the Source IP.
Example: DLP Multiple Alerts - SRC IP: (optional)
<Enter Text>

Rule template does not mention the search value and provides an example name



Alert Name
The text you enter displays in the Security Alert widget.
Example: Unix sudo Substring Match: jsmith (optional)
Unix sudo Substring Match: root

In addition to the criteria fields that are common to most templates, several templates include one or more of the following criteria fields.

- **Substring Match** — A text field into which you specify text to be found in the message of the event that triggers the alert.

  For example, for a login alert, this text might be a user name or login status (such as success or failure). For an alert that fires whenever a specified process begins, this text might be a comma-separated list of processes.

  The template labels describe the text that the alert expects.

- **Event ID**— A text field into which you enter the ID of the event that will trigger the alert.

  For example, for a Windows alert, this text might be “`529`” (failed login).

- **Host Match List** — A text field into which you enter one or more host names that will trigger the alert; you must use commas to separate multiple host names.

- **User Match List** — A text field into which you enter one or more user names that will trigger the alert; you must use commas to separate multiple user names.

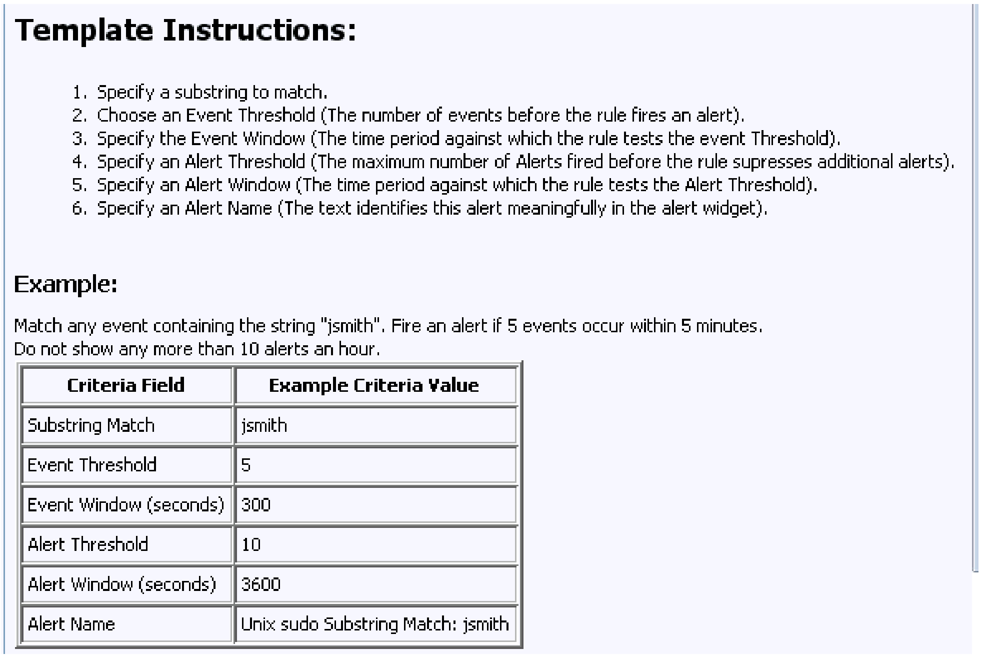
Other templates may contain other criteria fields.

UNDERSTANDING ALERT THRESHOLDS AND ALERT WINDOWS

The **Alert Threshold** and **Alert Window** criteria fields prevent your HawkEye AP system from being flooded with alerts. When you create a rule from a template, you can configure how many alerts should display over a specified time period. The HawkEye AP-provided alerting rules already specify safe values for these two criteria.

Figure 8-1 illustrates a portion of the configuration pane that displays when you create or edit an alert from a template.

Figure 8-1: Alert Threshold and Alert Window Explained



Template Instructions:

1. Specify a substring to match.
2. Choose an Event Threshold (The number of events before the rule fires an alert).
3. Specify the Event Window (The time period against which the rule tests the event Threshold).
4. Specify an Alert Threshold (The maximum number of Alerts fired before the rule supresses additional alerts).
5. Specify an Alert Window (The time period against which the rule tests the Alert Threshold).
6. Specify an Alert Name (The text identifies this alert meaningfully in the alert widget).

Example:

Match any event containing the string "jsmith". Fire an alert if 5 events occur within 5 minutes.
Do not show any more than 10 alerts an hour.

Criteria Field	Example Criteria Value
Substring Match	jsmith
Event Threshold	5
Event Window (seconds)	300
Alert Threshold	10
Alert Window (seconds)	3600
Alert Name	Unix sudo Substring Match: jsmith

The example configuration panes illustrate the **Event Threshold** and **Event Window** as well as for the **Alert Threshold** and **Alert Window** criteria fields. The text above the table of example values explains how these fields work together. The example explains that the rule fires an alert if 5 events (**Event Threshold**) occur within 5 minutes (**Event Window** = 300 seconds) and that the rule does not fire more than 10 alerts (**Alert Threshold**) each hour (**Alert Window** = 3600 seconds).

Sensage recommends that you consider your system carefully when you set these values. For example, the HawkEye AP-provided alerting rules `windowsAccountLockoutGlobal`, `windowsAccountLockoutDomain, and windowsAccountLockoutUser` set these values differently because they each represent a significantly different scope:

- `windowsAccountLockoutGlobal` — limits 10 account lockouts to occur within 1 minute
- `windowsAccountLockoutDomain` — limits 10 account lockouts to occur within 2 minutes
- `windowsAccountLockoutUser` — limits 3 account lockouts to occur within 5 minutes

NOTE: The **Alert Threshold** and **Alert Window** criteria fields limit the number of alerts that display in the alerts widget by reducing the number of alerts actually fired.

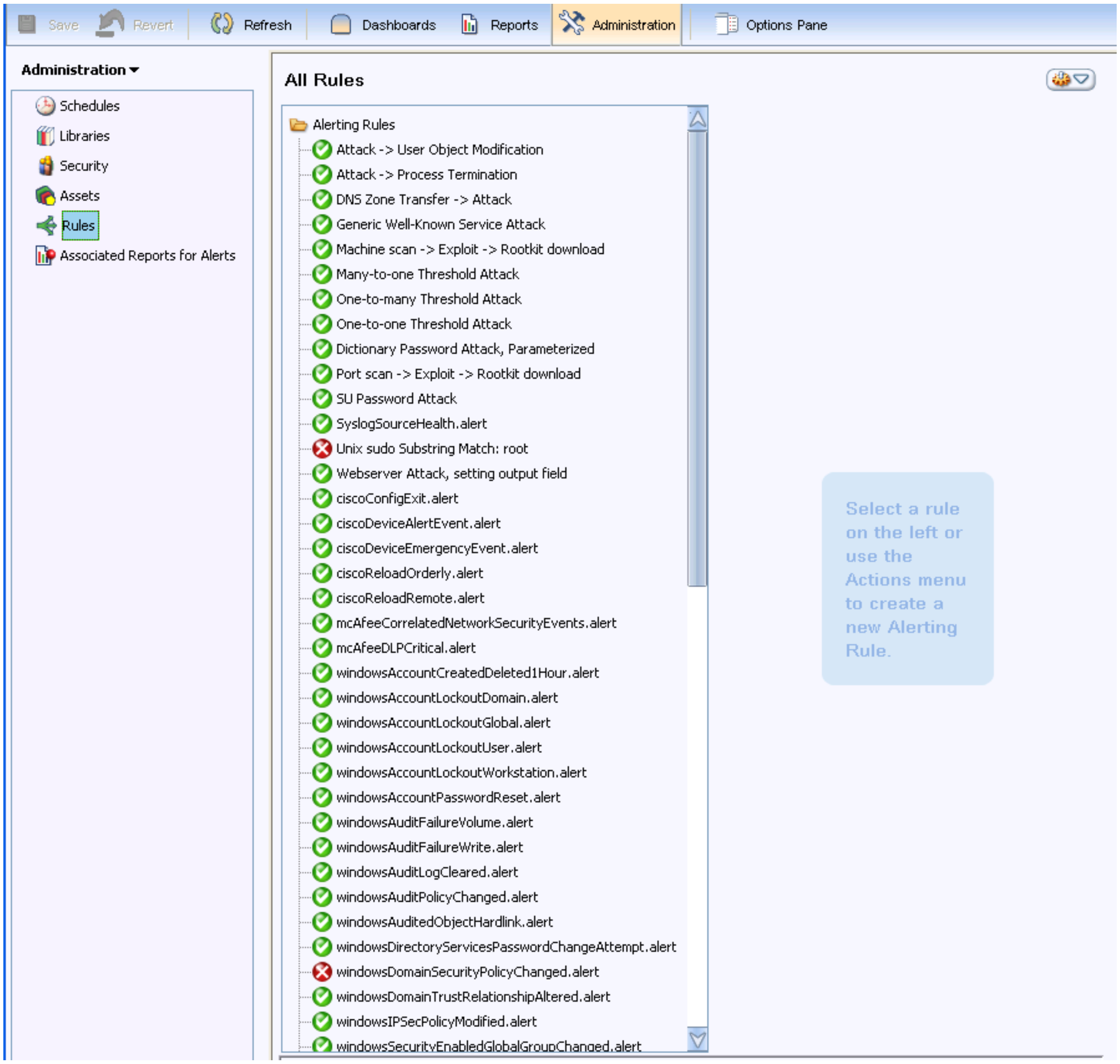
Creating and Modifying Alerting Rules from Templates

Use the HawkEye AP Console Rules module to:

- view all available rules
- activate, deactivate, and delete rules
- create new alerting rules from rule templates
- modify existing alerting rules

Figure 8-2 illustrates the rules that display at the top of the Rules module.

Figure 8-2: The Rules Module: Top of Displayed Rules (Alerting Rules)



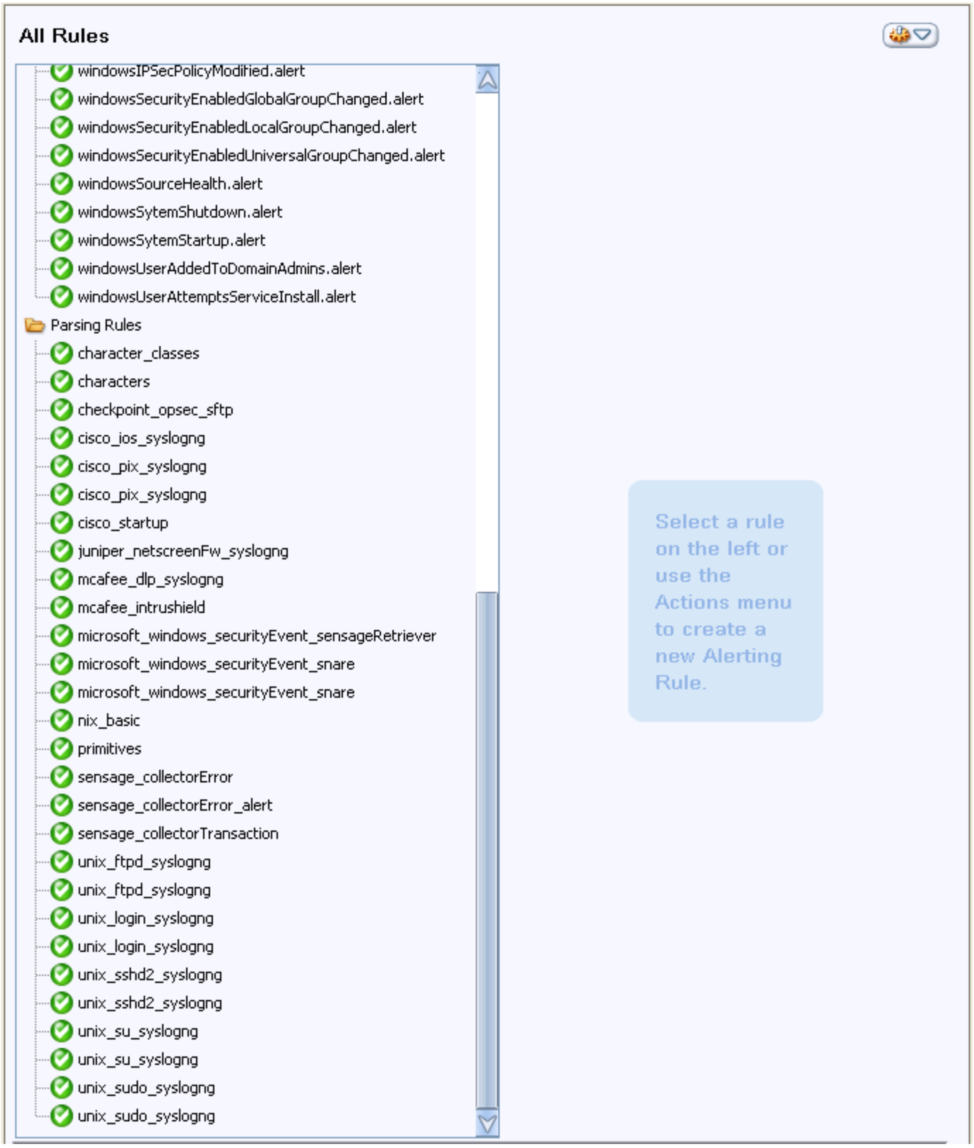
All Rules

Alerting Rules
- ✓ Attack -> User Object Modification
- ✓ Attack -> Process Termination
- ✓ DNS Zone Transfer -> Attack
- ✓ Generic Well-Known Service Attack
- ✓ Machine scan -> Exploit -> Rootkit download
- ✓ Many-to-one Threshold Attack
- ✓ One-to-many Threshold Attack
- ✓ One-to-one Threshold Attack
- ✓ Dictionary Password Attack, Parameterized
- ✓ Port scan -> Exploit -> Rootkit download
- ✓ SU Password Attack
- ✓ SyslogSourceHealth.alert
- ✗ Unix sudo Substring Match: root
- ✓ Webserver Attack, setting output field
- ✓ ciscoConfigExit.alert
- ✓ ciscoDeviceAlertEvent.alert
- ✓ ciscoDeviceEmergencyEvent.alert
- ✓ ciscoReloadOrderly.alert
- ✓ ciscoReloadRemote.alert
- ✓ mcAfeeCorrelatedNetworkSecurityEvents.alert
- ✓ mcAfeeDLPCritical.alert
- ✓ windowsAccountCreatedDeleted1Hour.alert
- ✓ windowsAccountLockoutDomain.alert
- ✓ windowsAccountLockoutGlobal.alert
- ✓ windowsAccountLockoutUser.alert
- ✓ windowsAccountLockoutWorkstation.alert
- ✓ windowsAccountPasswordReset.alert
- ✓ windowsAuditFailureVolume.alert
- ✓ windowsAuditFailureWrite.alert
- ✓ windowsAuditLogCleared.alert
- ✓ windowsAuditPolicyChanged.alert
- ✓ windowsAuditedObjectHardlink.alert
- ✓ windowsDirectoryServicesPasswordChangeAttempt.alert
- ✗ windowsDomainSecurityPolicyChanged.alert
- ✓ windowsDomainTrustRelationshipAltered.alert
- ✓ windowsIPSecPolicyModified.alert
- ✓ windowsSecurityEnabledGlobalGroupChanged.alert

Select a rule on the left or use the Actions menu to create a new Alerting Rule.

Figure 8-3 illustrates the rules that display at the bottom of the Rules module.

Figure 8-3: The Rules Module: Bottom of Displayed Rules (Parsing Rules)



All Rules

- ✓ windowsIPSecPolicyModified.alert
- ✓ windowsSecurityEnabledGlobalGroupChanged.alert
- ✓ windowsSecurityEnabledLocalGroupChanged.alert
- ✓ windowsSecurityEnabledUniversalGroupChanged.alert
- ✓ windowsSourceHealth.alert
- ✓ windowsSytemShutdown.alert
- ✓ windowsSytemStartup.alert
- ✓ windowsUserAddedToDomainAdmins.alert
- ✓ windowsUserAttemptsServiceInstall.alert

Parsing Rules

- ✓ character_classes
- ✓ characters
- ✓ checkpoint_opsec_sftp
- ✓ cisco_ios_syslogng
- ✓ cisco_pix_syslogng
- ✓ cisco_pix_syslogng
- ✓ cisco_startup
- ✓ juniper_netscreenFw_syslogng
- ✓ mcafee_dlp_syslogng
- ✓ mcafee_intrushield
- ✓ microsoft_windows_securityEvent_sensageRetriever
- ✓ microsoft_windows_securityEvent_snare
- ✓ microsoft_windows_securityEvent_snare
- ✓ nix_basic
- ✓ primitives
- ✓ sensage_collectorError
- ✓ sensage_collectorError_alert
- ✓ sensage_collectorTransaction
- ✓ unix_ftpd_syslogng
- ✓ unix_ftpd_syslogng
- ✓ unix_login_syslogng
- ✓ unix_login_syslogng
- ✓ unix_sshd2_syslogng
- ✓ unix_sshd2_syslogng
- ✓ unix_su_syslogng
- ✓ unix_su_syslogng
- ✓ unix_sudo_syslogng
- ✓ unix_sudo_syslogng

Select a rule on the left or use the Actions menu to create a new Alerting Rule.

As illustrated in the two figures above, the All Rules window displays two folders of items:

- **Alerting Rules**

  Three types of alerting rules display in this folder:

  - User-defined rules that were created from alerting rule templates.

    These rules were written in the HawkEye Event Processing Language (HEPL).

  - HawkEye AP-provided alerting rules that are delivered with the HawkEye AP product software.

    These rules were written in HEPL.

  - Legacy rules.

These rules were written in an earlier HawkEye AP correlation-rule language.

- **Parsing Rules**

  Two types of parsing rules display in this folder:

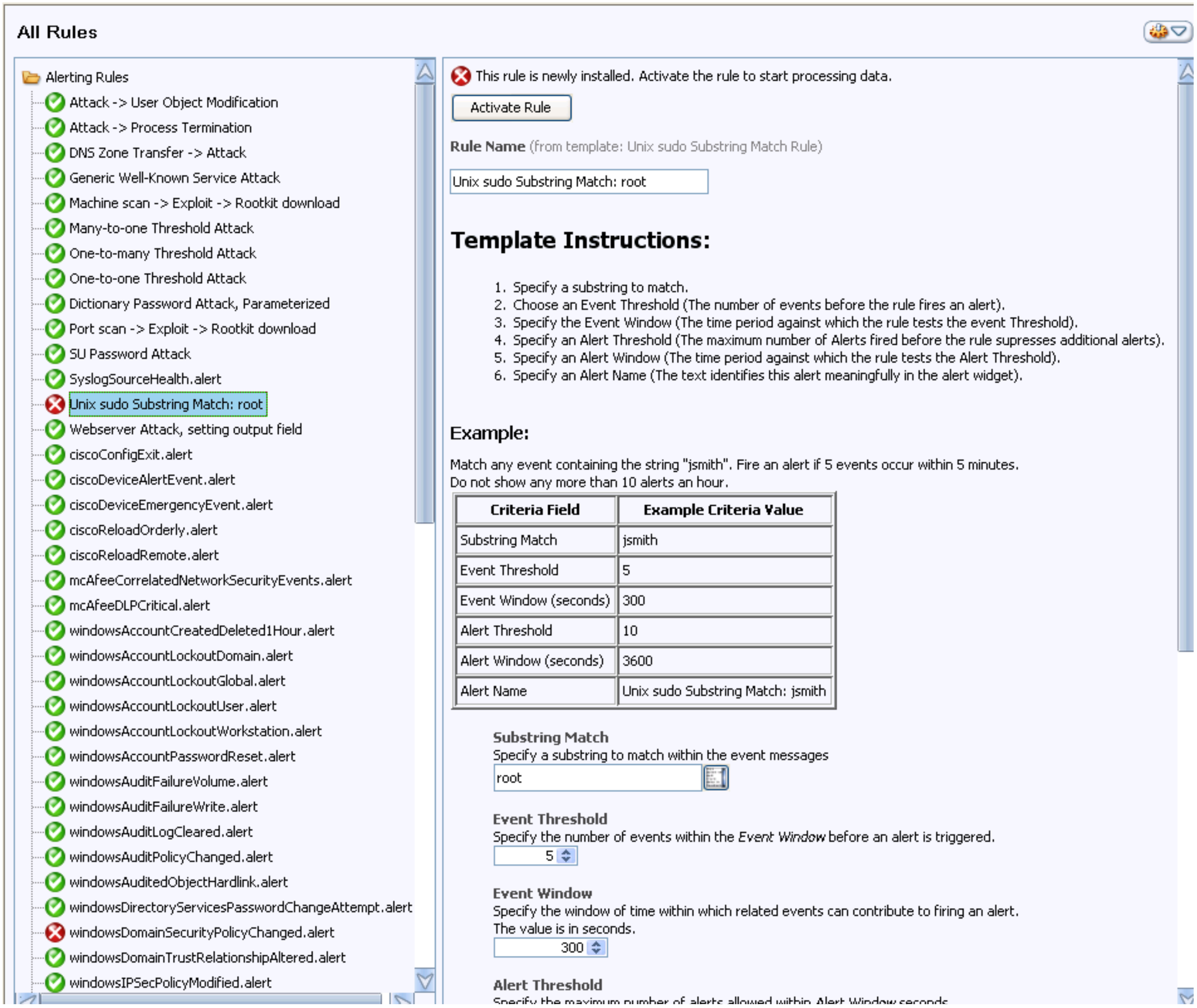
  - Most of these rules were written in the HEPL language.
  - There are a few legacy parsing rules that were written in an earlier HawkEye AP parsing language.

For information on the HEPL language, see the *HawkEye Event Processing Language Developers Guide*. For information on the earlier languages used to create the parsing and alerting rules, see the *Event Collection Guide*.

Figure Figure 8-4 illustrates a set of activated alerting rules. Only two of the rules have not been activated. Rules that have been activated display a green circle with a check mark. Rules that have not been activated display a red circle with an X character.

When you select an alerting rule on the left, its properties display in the pane on the right, as illustrated in Figure 8-4.

Figure 8-4: Properties for the Selected Rule



All Rules

- Alerting Rules
  - ✓ Attack -> User Object Modification
  - ✓ Attack -> Process Termination
  - ✓ DNS Zone Transfer -> Attack
  - ✓ Generic Well-Known Service Attack
  - ✓ Machine scan -> Exploit -> Rootkit download
  - ✓ Many-to-one Threshold Attack
  - ✓ One-to-many Threshold Attack
  - ✓ One-to-one Threshold Attack
  - ✓ Dictionary Password Attack, Parameterized
  - ✓ Port scan -> Exploit -> Rootkit download
  - ✓ SU Password Attack
  - ✓ SyslogSourceHealth.alert
  - ✗ Unix sudo Substring Match: root
  - ✓ Webserver Attack, setting output field
  - ✓ ciscoConfigExit.alert
  - ✓ ciscoDeviceAlertEvent.alert
  - ✓ ciscoDeviceEmergencyEvent.alert
  - ✓ ciscoReloadOrderly.alert
  - ✓ ciscoReloadRemote.alert
  - ✓ mcAfeeCorrelatedNetworkSecurityEvents.alert
  - ✓ mcAfeeDLPCritical.alert
  - ✓ windowsAccountCreatedDeleted1Hour.alert
  - ✓ windowsAccountLockoutDomain.alert
  - ✓ windowsAccountLockoutGlobal.alert
  - ✓ windowsAccountLockoutUser.alert
  - ✓ windowsAccountLockoutWorkstation.alert
  - ✓ windowsAccountPasswordReset.alert
  - ✓ windowsAuditFailureVolume.alert
  - ✓ windowsAuditFailureWrite.alert
  - ✓ windowsAuditLogCleared.alert
  - ✓ windowsAuditPolicyChanged.alert
  - ✓ windowsAuditedObjectHardlink.alert
  - ✓ windowsDirectoryServicesPasswordChangeAttempt.alert
  - ✗ windowsDomainSecurityPolicyChanged.alert
  - ✓ windowsDomainTrustRelationshipAltered.alert
  - ✓ windowsIPSecPolicyModified.alert

✗ This rule is newly installed. Activate the rule to start processing data.

Activate Rule

Rule Name (from template: Unix sudo Substring Match Rule)

Unix sudo Substring Match: root

Template Instructions:

1. Specify a substring to match.
2. Choose an Event Threshold (The number of events before the rule fires an alert).
3. Specify the Event Window (The time period against which the rule tests the event Threshold).
4. Specify an Alert Threshold (The maximum number of Alerts fired before the rule supresses additional alerts).
5. Specify an Alert Window (The time period against which the rule tests the Alert Threshold).
6. Specify an Alert Name (The text identifies this alert meaningfully in the alert widget).

Example:

Match any event containing the string "jsmith". Fire an alert if 5 events occur within 5 minutes.
Do not show any more than 10 alerts an hour.

Criteria Field	Example Criteria Value
Substring Match	jsmith
Event Threshold	5
Event Window (seconds)	300
Alert Threshold	10
Alert Window (seconds)	3600
Alert Name	Unix sudo Substring Match: jsmith

Substring Match
Specify a substring to match within the event messages
root

Event Threshold
Specify the number of events within the *Event Window* before an alert is triggered.
5

Event Window
Specify the window of time within which related events can contribute to firing an alert.
The value is in seconds.
300

Alert Threshold
Specify the maximum number of alerts allowed within Alert Window seconds

In Figure 8-4, the **Unix sudo Substring Match Rule** rule has been selected, as indicated by its highlighted name in the Alerting Rules pane. This is a user-defined rule that was created from an alerting rule template. Its properties display in the right pane.

At the top of the properties pane, the rule status indicates whether the rule has been activated or deactivated. If an error occurred during activation, this field displays the error text.

Below the status field and within parentheses, the **Rule Name** field identifies the template on which the rule is based. If the selected rule had been one of the HawkEye AP-provided alerting rules, the template would display as `none`. If the selected rule had been one of the legacy alerting rules, the pane to the right displays different information that does not include template information.

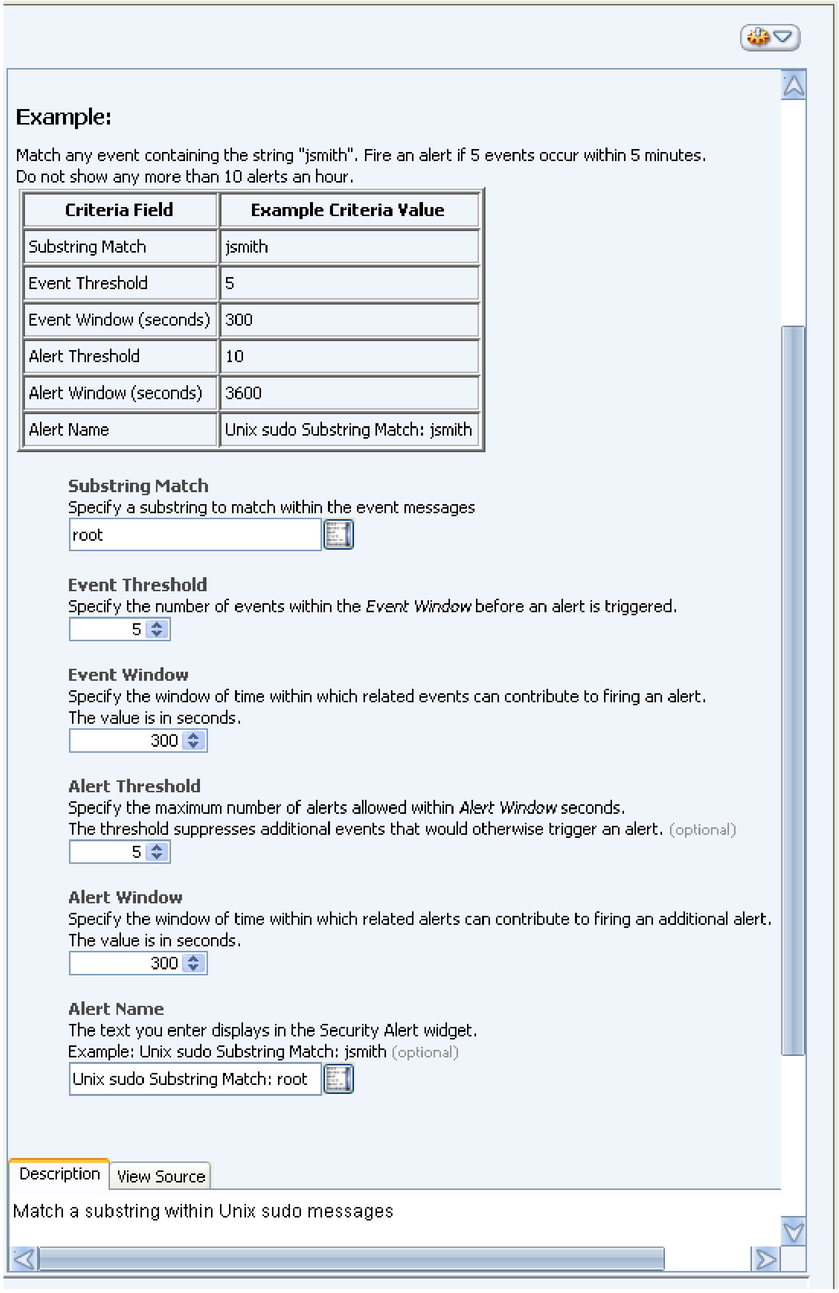
In Figure 8-4, below the template name is an entry field that provides a default name for the rule. This name displays in the Alerting Rules folder to the left. You can change this value to reflect the way you configure the rule. For example, if you configure this rule to search for “`root`” as its substring, you might rename the rule to “`Unix sudo Substring: root`”. If you configure this rule to search for “`passwd`” as its substring, you might rename the rule to “`Unix sudo Substring: passwd`”.

If you create more than one alerting rule from the same template, and each rule searches for a different substring, the name you provide here differentiates each rule variant in the left pane.

Because the rule illustrated in Figure 8-4 is a user-defined one, the top of the properties pane also displays **Template Instructions**. These instructions are followed by an example table that explains each criteria field and illustrates how these example values effect rule processing.

Only three criteria fields are shown above in Figure 8-4. Figure 8-5 illustrates all the criteria.

Figure 8-5: Example of Alerting Rule Properties



Example:

Match any event containing the string "jsmith". Fire an alert if 5 events occur within 5 minutes.
Do not show any more than 10 alerts an hour.

Criteria Field	Example Criteria Value
Substring Match	jsmith
Event Threshold	5
Event Window (seconds)	300
Alert Threshold	10
Alert Window (seconds)	3600
Alert Name	Unix sudo Substring Match: jsmith

Substring Match
Specify a substring to match within the event messages
root

Event Threshold
Specify the number of events within the *Event Window* before an alert is triggered.
5

Event Window
Specify the window of time within which related events can contribute to firing an alert.
The value is in seconds.
300

Alert Threshold
Specify the maximum number of alerts allowed within *Alert Window* seconds.
The threshold suppresses additional events that would otherwise trigger an alert. (optional)
5

Alert Window
Specify the window of time within which related alerts can contribute to firing an additional alert.
The value is in seconds.
300

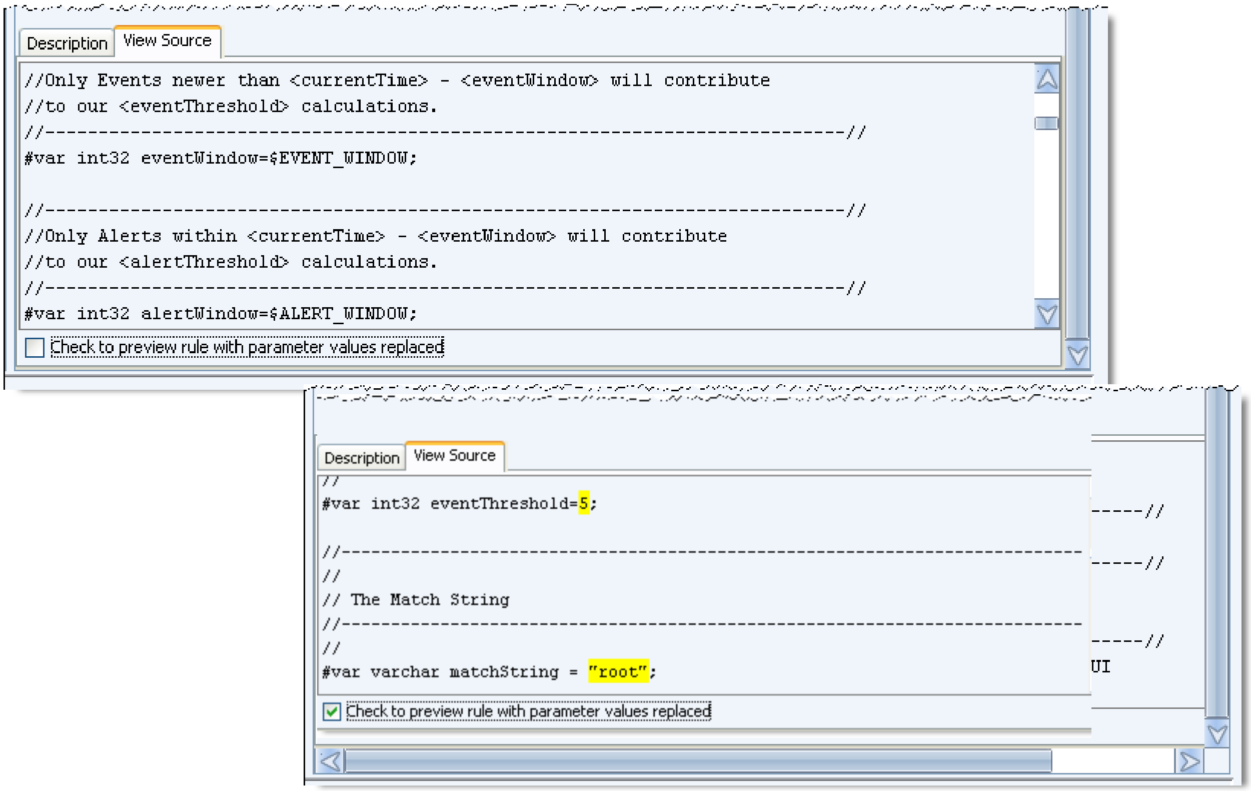
Alert Name
The text you enter displays in the Security Alert widget.
Example: Unix sudo Substring Match: jsmith (optional)
Unix sudo Substring Match: root

Match a substring within Unix sudo messages

The graphic above illustrates all the criteria fields available in one template that HawkEye AP provides. For information about these properties, see “Introduction to Alerting Rule Templates”, on page 247.

Below all the configuration fields, the properties pane displays a description of what the rule does. The description field has two tabs. The **Description** tab displays by default. The **View Source** tab displays the rule's code, sections of which are illustrated in Figure 8-6.

Figure 8-6: Viewing the Rule's Code



```
//Only Events newer than <currentTime> - <eventWindow> will contribute
//to our <eventThreshold> calculations.
//------------------------------------------------------------------------//
#var int32 eventWindow=$EVENT_WINDOW;

//------------------------------------------------------------------------//
//Only Alerts within <currentTime> - <eventWindow> will contribute
//to our <alertThreshold> calculations.
//------------------------------------------------------------------------//
#var int32 alertWindow=$ALERT_WINDOW;
```

□ Check to preview rule with parameter values replaced

```
//
#var int32 eventThreshold=5;

//---------------------------------------------------------------------------
//
// The Match String
//---------------------------------------------------------------------------
//
#var varchar matchString = "root";
```

☑ Check to preview rule with parameter values replaced

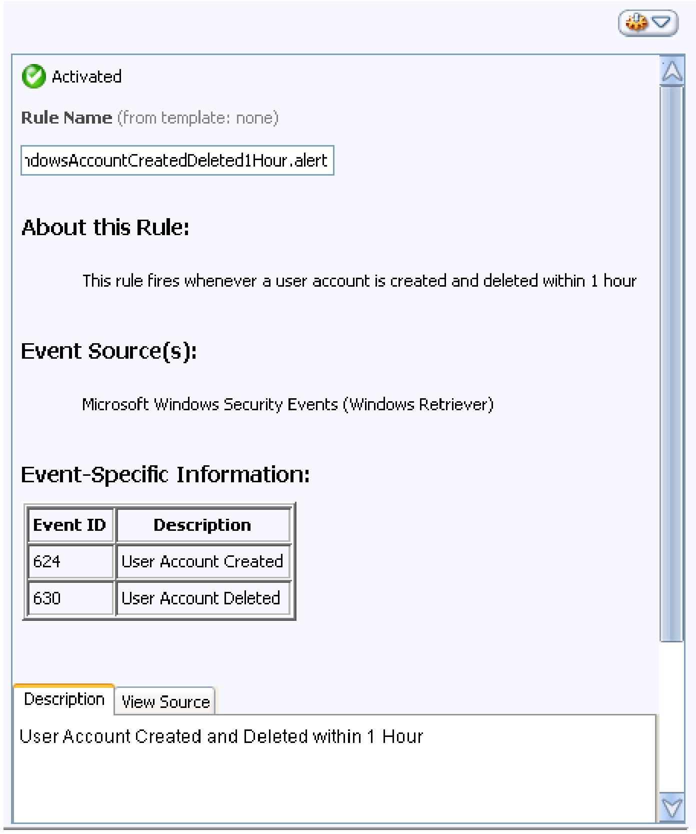
As shown above, when you select the **Check to preview rule with parameter values replaced** checkbox, your criteria field values are highlighted in the rule's code.

NOTE:

- If you modify the value in a criteria field, you must save the change to see it reflected in the View Source window.
- The properties of a HawkEye AP-provided alerting rule and a legacy alerting rule display differently. The properties pane for these rules also include the status, rule name, and description, but there are significant differences.

Figure 8-8 illustrates the properties of a HawkEye AP-provided alerting rule.

Figure 8-7: Example of HawkEye AP-Provided Alerting Rule Properties



Activated

Rule Name (from template: none)

ndowsAccountCreatedDeleted1Hour.alert

About this Rule:

This rule fires whenever a user account is created and deleted within 1 hour

Event Source(s):

Microsoft Windows Security Events (Windows Retriever)

Event-Specific Information:

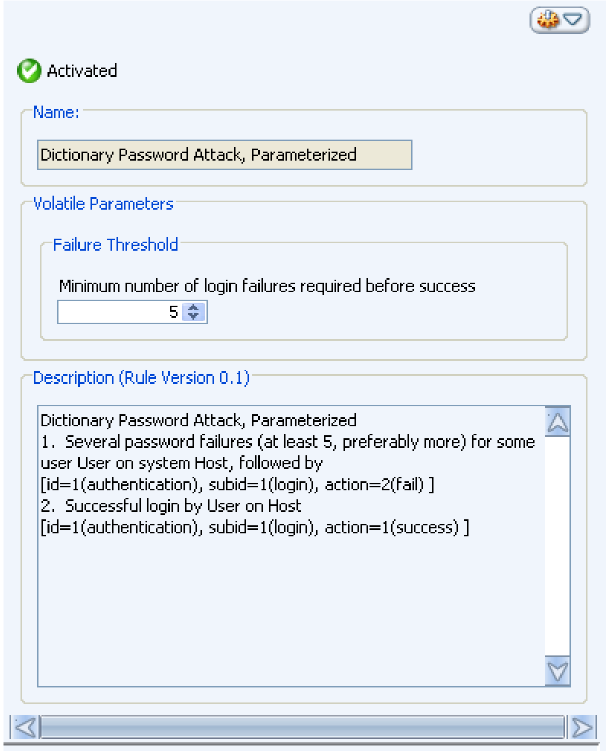
Event ID	Description
624	User Account Created
630	User Account Deleted

User Account Created and Deleted within 1 Hour

This rule is not configurable. The information pane is provided to explain the functioning of the rule.

Figure 8-8 illustrates the properties of a legacy alerting rule that provides configurable conditions.

Figure 8-8: Example of Legacy Alerting Rule Properties



Activated

Name:

Dictionary Password Attack, Parameterized

Volatile Parameters

Failure Threshold

Minimum number of login failures required before success

5

Description (Rule Version 0.1)

Dictionary Password Attack, Parameterized
1. Several password failures (at least 5, preferably more) for some user User on system Host, followed by
[id=1(authentication), subid=1(login), action=2(fail)]
2. Successful login by User on Host
[id=1(authentication), subid=1(login), action=1(success)]

The graphic above illustrates the properties of the **Dictionary Password Attack, Parameterized** legacy alerting rule. If a specified number of login failures are followed by a successful login on the same host, the rule triggers a security alert. By default, this rule tracks five failed logins prior to a successful one. However, you can configure the rule to track a different number of failures.

Because this rule was not created from a template, it mentions no template. Moreover, its **Description** field does not include a tab for viewing the code.

NOTE: Not all legacy rules are configurable. The information pane for a non-configurable legacy rule does not include the Parameters field.

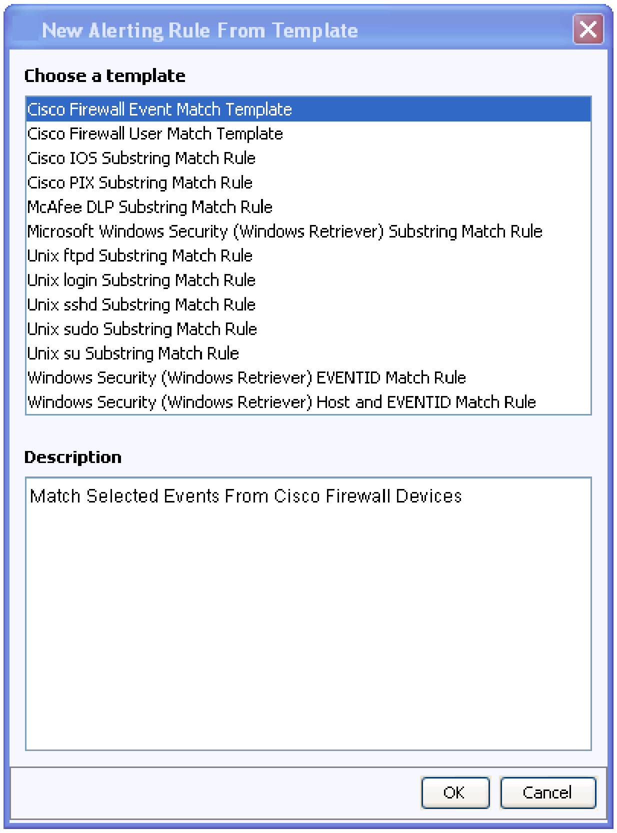
Creating a Rule from a Template

This section describes how to create an alerting rule from a template.

To create an alerting rule from a template

1 Select **New Alerting Rule from Template...** from the Action Menu, the right-click popup menu, or CTRL+N.

The New Alerting Rule From Template dialog displays, as illustrated below.



New Alerting Rule From Template

Choose a template

Cisco Firewall Event Match Template
Cisco Firewall User Match Template
Cisco IOS Substring Match Rule
Cisco PIX Substring Match Rule
McAfee DLP Substring Match Rule
Microsoft Windows Security (Windows Retriever) Substring Match Rule
Unix ftpd Substring Match Rule
Unix login Substring Match Rule
Unix sshd Substring Match Rule
Unix sudo Substring Match Rule
Unix su Substring Match Rule
Windows Security (Windows Retriever) EVENTID Match Rule
Windows Security (Windows Retriever) Host and EVENTID Match Rule

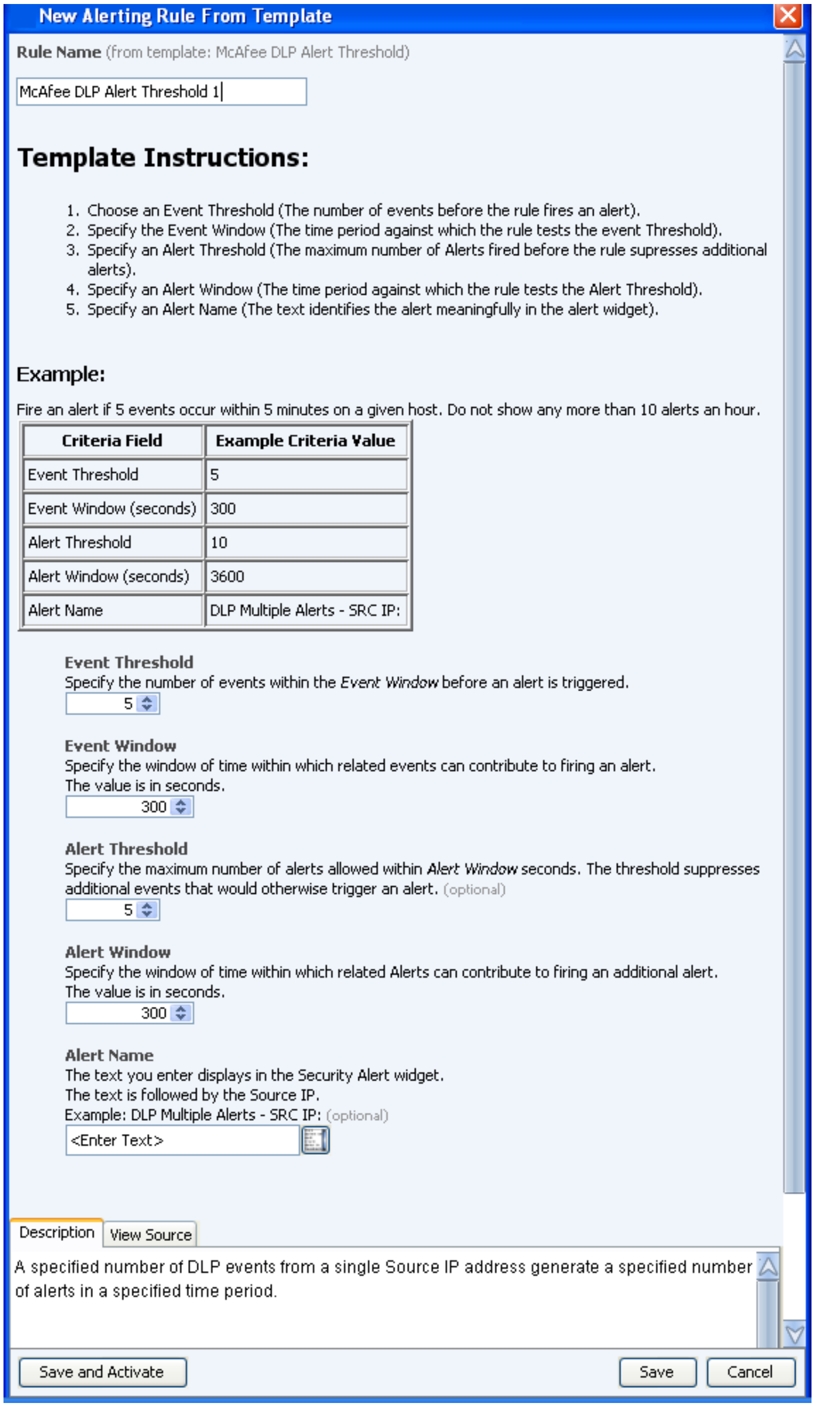
Description

Match Selected Events From Cisco Firewall Devices

2 Select the desired template from the list of available templates, and click **OK**.

NOTE: As you select a rule in the dialog, a description of the rule displays at the bottom.

The New Alerting Rule From Template dialog changes to display the conditions available for the selected rule, as illustrated below.



New Alerting Rule From Template

Rule Name (from template: McAfee DLP Alert Threshold)

McAfee DLP Alert Threshold 1

Template Instructions:

1. Choose an Event Threshold (The number of events before the rule fires an alert).
2. Specify the Event Window (The time period against which the rule tests the event Threshold).
3. Specify an Alert Threshold (The maximum number of Alerts fired before the rule supresses additional alerts).
4. Specify an Alert Window (The time period against which the rule tests the Alert Threshold).
5. Specify an Alert Name (The text identifies the alert meaningfully in the alert widget).

Example:

Fire an alert if 5 events occur within 5 minutes on a given host. Do not show any more than 10 alerts an hour.

Criteria Field	Example Criteria Value
Event Threshold	5
Event Window (seconds)	300
Alert Threshold	10
Alert Window (seconds)	3600
Alert Name	DLP Multiple Alerts - SRC IP:

Event Threshold
Specify the number of events within the *Event Window* before an alert is triggered.
5

Event Window
Specify the window of time within which related events can contribute to firing an alert.
The value is in seconds.
300

Alert Threshold
Specify the maximum number of alerts allowed within *Alert Window* seconds. The threshold suppresses additional events that would otherwise trigger an alert. (optional)
5

Alert Window
Specify the window of time within which related Alerts can contribute to firing an additional alert.
The value is in seconds.
300

Alert Name
The text you enter displays in the Security Alert widget.
The text is followed by the Source IP.
Example: DLP Multiple Alerts - SRC IP: (optional)
<Enter Text>

A specified number of DLP events from a single Source IP address generate a specified number of alerts in a specified time period.

3 Specify a meaningful name for the rule and set desired values for the criteria.

NOTE:

- The rule name can be a maximum of 59 characters and must be unique to your HawkEye AP deployment. After you specify the name, HawkEye AP appends `.rule` to the end of the name.
- If the text you enter in a criteria field exceeds the size of the text field, click the Edit button (▤) icon. This icon is not available for the rule name field because the name cannot exceed the size of the entry field.

4 When you are satisfied with your settings, you can click either **Save** or **Save and Activate**. Both buttons save the rule. The **Save and Activate** button saves the step of activating separately.

Your new rule displays in the Alerting Rules folder.

For more information, see “Activating, Deactivating, and Deleting Rules”, next.

Activating, Deactivating, and Deleting Rules

Activating Rules

If you do not activate an alerting rule when you save it, you can select **Activate** from the Action Menu or the right-click menu.

Parsing rules and legacy alerting rules do not have an edit pane with a **Save and Activate** button. Therefore, to activate these, select the rule and then select **Activate** it from the Action Menu or the right-click menu.

Deactivating Rules

To deactivate an alerting rule or a parsing rule, select the rule and then select **Deactivate** from the Action Menu or the right-click menu. You can reactivate the rule at any time by following the instructions above.

Deleting Rules

To delete an alerting rule or a parsing rule, select the rule and then select **Delete** from the Action Menu or the right-click menu.

NOTE: A deleted rule is not recoverable. You must reinstall and reactivate the rule to enable it again.

CHAPTER 9

Report Libraries Reference

Report libraries provide functions for common look-up and conversion operations you may wish to perform in your SQL reports. For more information on using libraries with SQL reports, see "Adding a Library to Your Query", on page 195.

The following libraries are included with your HawkEye AP distribution:

- "Geo IP Utility", on page 263
- "IP Conversion Utility", on page 264
- "Internal System Audit Library", on page 265
- "Microsoft Windows Library", on page 265

Geo IP Utility

The Geo IP Utility library contains functions to look up domain and country of origin from a URL.

domain()

Returns the domain name portion of the URL.

Synopsis

```
domain (<URL>)
```

Arguments

Argument	Description
<URL>	Universal Resource Locator (URL)

Example

```
SELECT
  URL as "URL",
  domain(URL) as "Domain"
FROM
  myTable
DURING ALL
```

Return Value

The domain name portion of the URL. For example, the domain name of the URL "`www.foo.com`" is "`foo.com`"

get_country_from_domain()

Returns the country where a domain is located.

Synopsis

```
get_country_from_domain(<domain_name>)
```

Arguments

Argument	Description
<domain_name>	Domain name

Example

```
SELECT
  URL as "URL",
  domain(URL) as "Domain"
  get_country_from_domain(domain(URL)) as "Country of Origin"
FROM
  myTable
DURING ALL
```

Return Value

The Country where the domain is located, displayed as a two-character code using ISO 3166-1-alpha-2 code elements. See English country names and code elements for a complete list.

IP CONVERSION UTILITY

The IP Conversion Utility library provides the `hex_to_dotted_quad()` function.

hex_to_dotted_quad(IP address in Hexadecimal format)

Converts an IP address stored in hexadecimal format to the standard dotted quad format.

Synopsis

```
hex_to_dotted_quad(<Hex_IP>)
```

Arguments

Argument	Description
<Hex_IP>	IP address in hexadecimal format

Return Value

IP address in dotted quad format. For example, the following IP address is displayed in dotted quad format:

```
127.0.0.1.
```

Example

```
SELECT
  hex_to_dotted_quad(SOURCEIPADDR) as "IP Address"
FROM
  myTable
DURING ALL
```

INTERNAL SYSTEM AUDIT LIBRARY

The Internal System Audit library provides a function for use with creating reports on HawkEye AP internal auditing tables.

service2Description()

Translates an internal service name to a textual description.

Synopsis

```
service2Description(<service_name>)
```

Arguments

Argument	Description
<service_name>	Internal HawkEye AP service name

Return Value

A textual description of the service name.

Example

```
service2Description(ReportService.createReportDefinition)

SELECT
  SERVICE_NAME as "Service Name"
  service2Description(SERVICE_NAME) as "Service Description"
FROM
  myTable
DURING ALL
```

MICROSOFT WINDOWS LIBRARY

The Microsoft Windows library provides the following functions for use in processing Windows event data:

- “loginType2desc()”, on page 266
- “eventId2desc()”, on page 266

- “rights2desc()”, on page 267
- “k5code2desc()”, on page 267

loginType2desc()

Translates numeric login types to a textual description.

Synopsis

```
loginType2desc (<login_type>)
```

Arguments

Argument	Description
<login_type>	Windows login type code

Return Value

A textual description of the Windows login type code.

Example

```
SELECT
  LOGIN_TYPE as "Login Type Code",
  loginType2desc(LOGIN_TYPE)as "Login Description"
FROM
  myTable
DURING ALL
```

eventId2desc()

Translates numeric event ID codes into a textual description.

Synopsis

```
eventID2desc(<Event_ID_code>)
```

Arguments

Argument	Description
<Event_ID_code>	Windows event ID code

Return Value

A textual description of the Windows event ID code.

Example

```
SELECT
  EVENT_ID as "Event ID Code",
  eventID2desc(EVENT_ID) as "Event Description"
FROM
```

```
  myTable
DURING ALL
```

rights2desc()

Translates Windows user rights strings to a textual description.

Synopsis

```
rights2desc(<Windows_user_right>)
```

Arguments

Argument	Description
<Windows_user_right>	A string representing user rights. For example: `SeBackupPrivilege`.

Return Value

A textual description of the user right.

Example

```
SELECT
  USER_RIGHT as "User Right",
  rights2desc(USER_RIGHT) as "User Right Description"
FROM
  myTable
DURING ALL
```

k5code2desc()

Translates a Kerberos hexadeicmal error code into a textual description.

Synopsis

```
k5code2desc(<hex_code>)
```

Arguments

Argument	Description
<hex_code>	Kerberos error code in hexadecimal format

Return Value

Textual description of the Kerberos error code.

Example

```
SELECT
  KERBEROS_ERROR as "Kerberos Error Code",
  k5code2desc(KERBEROS_ERROR) as "Kerberos Error Description"
FROM
  myTable
DURING ALL
```

CHAPTER 10

Sensage SQL

Sensage SQL is a special implementation of standard SQL that is optimized for event-log analysis. This chapter describes the SELECT statement. Other statements found in standard SQL, such as UPDATE or DELETE, are not supported in Sensage SQL to avoid the risk of evidence tampering.

This chapter includes the following sections:

- "Overview of Sensage SQL SELECT Statements", next
- "Target Clauses", on page 271
- "FROM Clauses", on page 273
- "DURING Clauses", on page 275
- "WHERE Clauses", on page 277
- "GROUP BY Clauses and Aggregation Queries", on page 278
- "SLICE BY Clauses", on page 282
- "HAVING Clauses", on page 283
- "ORDER BY Clauses", on page 284
- "UNION ALL Clauses", on page 285
- "Data Types", on page 286
- "Data Source Expressions", on page 289
- "Data Processing Expressions", on page 292
- "Processing Directives", on page 301
- "Working with Lists", on page 313

Overview of Sensage SQL SELECT Statements

This section describes these topics:

- "Basic SELECT Syntax", next
- "Keywords and Clauses Required in SELECT statements", on page 270
- "About Tables and Namespaces", on page 270

Basic SELECT Syntax

In Sensage SQL, a SELECT statement has the following, basic syntax:

```
   SELECT [ALL|DISTINCT] [{FIRST|TOP|LAST|BOTTOM} n ]
          <expression>[AS <target>][, <expression>[AS <target>][...]]
     FROM <table_specification>
   [WHERE <conditional_expression>]
[GROUP BY [<expression>|<target>[, <expression>|<target>[...]]] [IN n PASSES]
[SLICE BY <conditional_expression>]
  [HAVING <conditional_expression>]
[ORDER BY [<expression>|<target>[, <expression>|<target> [...]]]
  [DURING {ALL|<start_time>, <end_time>[ OR <start_time>, <end_time>[ ...]]}]
         ;
```

Significant Terms

- Capitalized words along the left margin are called *keywords*. They have special meaning and introduce *clauses* within a SELECT statement.
- Clauses can include expressions, modifier keywords, and identifiers.
- An *identifier* is a reference to the name of such objects as a table, view, column, macro, or a target in a SELECT clause; typically the name is restricted to an alphanumeric character and an underscore.
- Every table column is defined with a data type, which include varchar, int32, and boolean. The term for a data type and its value is *literal*. Examples of literals are:
  - **string literal**—"This text is a string literal."
  - **numeric literals**—`27` and `4.50` and `-9`
  - **boolean literals**—`T`, `F`, `TRUE`, `FALSE`, `1, 0`

NOTE: Clauses must appear in the order shown above. SELECT statements end with semicolons (;).

Keywords and Clauses Required in SELECT statements

The following are required in SELECT statements:

- The `SELECT` keyword, which begins the statement
- The target clause, which lists the columns in result sets returned by the statement.
- The `FROM` keyword, followed by a specification of the table to query.
- The `DURING` keyword, followed by the time frames to include in result sets returned by the statement.
- A terminating semi-colon (`;`).

The following example shows a SELECT statement with only the required parts:

```
SELECT *
  FROM mytable
  DURING ALL
;
```

About Tables and Namespaces

The EDW organizes tables into a hierarchy of *namespaces*. The namespace hierarchy is similar to a hierarchical file system, with namespaces corresponding to directories or folders, and tables corresponding to files or documents. Like directories in file systems, one namespace can contain another. Two namespaces are separated by a period For example, `ns1.ns2.ns3.mytable` references a different table than `ns1.ns2.mytable`.

The default namespace is `default`. To specify a table in a namespace within the default, you must explicitly specify the full namespace. The following example specifies a table whose namespace is within the default namespace:

```
default.ns1.ns2.ns3.mytable
```

In contrast, the following example specifies a table in a namespace that descends directly from the root of the hierarchy instead of from the default namespace.

```
ns1.ns2.ns3.mytable
```

Use namespaces to organize your tables in a meaningful way. For example you can organize by company.division.department, or by user, or application, or other structure.

Target Clauses

SELECT statements begin with a *target* clause, which lists the columns returned in the result sets of queries. Target clauses follow immediately after the SELECT keyword.

This section describes these topics:

- "Computed expressions in Target Clauses", next
- "Named Targets", on page 271
- "Invisible targets", on page 272
- "DISTINCT Modifier Keyword", on page 272
- "FIRST and LAST Modifiers", on page 272

Computed expressions in Target Clauses

Target clauses include expressions that may or may not mention columns in the table specified in the FROM clause. Target expressions can include math, string, comparison, and logic operators, as well as functions.

For example, the following SELECT statement returns a result set with a single column that contains values from the ts column formatted with the `_timef()` function.

```
SELECT _timef("%m/%d/%Y", ts)
  FROM example_webserv_100
  DURING ALL
;
```

Named Targets

The SQL query engine chooses default names for target columns in result sets, based on their expressions. You can specify the names for target columns with the AS modifier keyword.

For example, the following SELECT statement returns a result set with one column that has "Date" as its name.

```
SELECT _timef("%m/%d/%Y", ts) AS Date
  FROM example_webserv_100
  DURING ALL;
```

Invisible targets

Sometimes, it's helpful to include a target column for use in query processing that you do not want included as a column in the result set. Beyond convenience, this also saves client-server bandwidth for large result sets. Remove such target columns with the AS modifier keyword and specify a name with two underscores (__) as a prefix.

For example, the following SELECT returns a result set with one column of client-machine DNS names, sorted by timestamp, even though timestamp is not included in the result set.

```
SELECT ClientDNS, ts AS __hidden
  FROM example_webserv_100
  ORDER BY 2
  DURING ALL;
```

DISTINCT Modifier Keyword

Sometimes, a query produces lots of identical records in the result set. If you would like to keep only the unique records, include the DISTINCT modifier keyword after the SELECT keyword that begins the statement or subquery.

For example, the following SELECT statement returns a result set with every unique DNS address seen in a given web log.

```
SELECT DISTINCT ClientDNS
  FROM example_webserv_100
  DURING ALL;
```

IMPORTANT: The DISTINCT keyword causes the SQL query engine to consume a lot of memory. Be careful about using DISTINCT in SELECT statements on tables with lots of rows. To enhance performance, Sensage recommends that you use GROUP BY instead of DISTINCT. For example, the following query returns the same results as the example above, but in less time:

```
SELECT ClientDNS
  FROM example_webserv_100
  GROUP BY 1
  DURING ALL;
```

FIRST and LAST Modifiers

Often, only the first or last few rows of a query output are desired in the results set. To limit the output of a query to the rows at the top or bottom of the results, include FIRST or LAST and the number of rows, between the SELECT keyword and the target clause.

As examples:

```
SELECT FIRST 10 ClientDNS, RespSize
  FROM example_webserv_100
  DURING ALL;

SELECT LAST 10 ClientDNS, RespSize
  FROM example_webserv_100
  DURING ALL;
```

When you combine DISTINCT with FIRST or LAST, the DISTINCT modifier keyword is processed to derive an intermediate result set. The SQL query engine then applies FIRST or LAST to derive the final result set.

NOTE: The TOP and BOTTOM modifier keywords are synonyms for FIRST and LAST.

FROM Clauses

The FROM clause in a SELECT statement specifies the table to be queried. Unlike standard SQL, in which FROM clauses can include a list of tables, HawkEye AP SQL SELECT statements generally only permit one table to be specified. Joins are not supported by HawkEye AP SQL.

This section describes these topics:

- "Syntax of FROM Clauses", next
- "Table Specifications in FROM Clauses", on page 273
- "Example FROM Clauses", on page 274
- "Including Rows from Bad Loads", on page 275

Syntax of FROM Clauses

The FROM clause has the following syntax:

```
FROM <table_specification> [<alias_name>] [{INCLUDE_BAD_LOADS}]
```

The `<table_specification>` is an expression that evaluates to the name of a table in the EDW. For more information, see "Table Specifications in FROM Clauses", next.

As an option, the FROM clause lets you use `<alias_name>` to specify an abbreviated name for the table. Table names can be quite long when namespaces are included. Use the alias as a table identifier in later lines of the SELECT statement.

As an option, the `{INCLUDE_BAD_UPLOADS}` modifier keyword instructs the SQL query engine to include rows that were loaded by failed load operations. For more information, see "Including Rows from Bad Loads", on page 275.

NOTE: The curly braces (`{}`) are part of the `{INCLUDE_BAD_UPLOADS}` keyword.

Table Specifications in FROM Clauses

Table specifications can be identifiers, literals, or expressions that evaluate to the name of a table in the EDW. For example, the specification can have one of these forms:

- The name of a table, including the namespace if there is one:

  ```
  FROM namespace1.mytable
  ```

- A string expression that evaluates to the name of a table:

  ```
  WITH $namespace = "myNamespace"
  ...
  FROM $namespace + "." + "example_webserv_100"
  ```

```
...
```

TIP: To learn how to include data from several tables in a single result set, see "UNION ALL Clauses", on page 285 and Defining Views with Sensage SQL in Chapter 4, "Administering an EDW Instance" of the *Administration Guide*.

Example FROM Clauses

The examples that follow show different ways to specify the table FROM clauses:

- Identifiers as the Table Name
- String Expressions as the Table Name
- Expression Macros as the Table Name

The queries in the examples below all return the distinct IP addresses from the `example_webserv_100` table*.*

Identifiers as the Table Name

The simplest table specification in a FROM clause is the table name itself, used as an identifier without enclosing quotation marks.

```
SELECT distinct ClientIP
  FROM example_webserv_100        -- note the table name is an identifier
  DURING all;
```

String Expressions as the Table Name

The table specification in a FROM clause can be a string concatenation expression that resolves to the name of a table. The following example uses a concatenation expression.

```
SELECT distinct ClientIP
  FROM 'example' + '_webserv_100'
  DURING all
```

When the query engine executes the statement, it queries the `example_webserv_100` table. Any expression or function that returns a string value can be used as the table expression.

Expression Macros as the Table Name

Often there are many tables that share a common schema. Use an expression macro as the table expression so that you can write a common SELECT statement for all tables that share the common schema. The expression macro provides a default table name, which can be overridden when the query engine executes the statement.

```
WITH $source as 'example_webserv_100'

SELECT distinct ClientIP
  FROM $source
  DURING all;
```

For more information, see "Expression Macros", on page 302.

Including Rows from Bad Loads

The EDW treats each attempt to load log-entries into a table as a single unit of work. If a load operation fails, any rows loaded into the table before the failure occurs are considered suspect. When you query a table, rows from failed loads are excluded by default. If you want rows from bad loads included in query processing, use the {INCLUDE_BAD_UPLOADS} modifier keyword at the end of the FROM clause.

For example:

```
SELECT distinct ClientIP
  FROM example_webserv_100 {INCLUDE_BAD_UPLOADS}
  DURING all;
```

NOTE:

- Include the curly braces (`{}`).
- A successful load does not necessarily insert every row from the source log file. Typically log files contain data that cannot be parsed and loaded; however, bad data does not prevent these files from loading successfully. A failed load is caused by a protocol error, such as not receiving all the expected data or a load cancellation.
- If a load and a query operation are running against the same table at the same time and the load completes before the query, the query may return none, some, or all of the data from the load (depending on when the query is executed).

For related information, see:

- Tracking Uploads in the EDW in the Managing the EDW Data Store chapter of the *Administration Guide*
- system.upload_info in Chapter 4, “Administering an EDW Instance” and system.raw_upload_info in Chapter 4, “Administering an EDW Instance” in the *Administration Guide*
- Monitoring for Inconsistent Loads in Chapter 12, “Troubleshooting” in the *Administration Guide*

DURING CLAUSES

Most queries of event-log data involve a specific time frame, whether it's “yesterday”, “last year,” or “1 hour ago.” To facilitate these types of queries, Sensage SQL supports the proprietary DURING clause, which allows you to specify the time frame for the query.

The DURING clause works with the timestamp column named `ts`, which is mandatory in every event-log table. The SQL query engine matches the time frame you specify in the DURING clause with values in the `ts` column of the table. Only rows that match the time frame are included in the results set.

Alternative Formats for DURING Clauses

The DURING clause has several forms:

- **DURING ALL**

The simplest form of the DURING clause uses the ALL modifier keyword to specify the time frame. It specifies the entire time frame represented in the table. In other words, the SQL query engine ignores values in the ts column when it determines which rows to include in the result set.

- **DURING <*lower_bound*>, <*upper_bound*>**

  The DURING clause takes two expressions, separated by a comma, that evaluate to timestamps. Together, the two expressions restrict query results to a single time frame. Only rows with values in the ts column that fall between the bounding timestamps are included in the results set. Specify the <*upper_bound*> and <*lower_bound*> with expressions that evaluate to timestamps. For example, you can use functions like _time(), _strptime(), and so on.

- **DURING [<*lower_bound*>, <*upper_bound*>] OR DURING [<*lower_bound*>, <*upper_bound*>] OR DURING ...**

  The DURING clause lets you specify a series of time frames. Enclose each time frame within square brackets, and introduce each additional time frame with OR DURING.

For example, the following statement specifies a single time frame. The SELECT statement returns the values of the ClientDNS and RespSize columns from rows in the example_webserv_100 table where the ts column is later than or equal to midnight, February 1, 2002, and earlier than or equal to the last second of March 31, 2002.

```
SELECT ClientDNS, RespSize
  FROM example_webserv_100
  DURING time('Feb 01 00:00:00 2002'), time('Apr 01 00:00:00 2002');
```

NOTE: The end time above includes the first microsecond of April 1.

Timestamp Precision in DURING Clauses

The DURING clause compares specified timestamps with the timestamps in the ts. The precision of ts columns is microseconds, provided the source log entries carry timestamps with that precision.

You can query time frames with microsecond precision, depending on the format of the varchar value you provide to the time() function. The examples in this chapter use the Unix date format, which specifies timestamps with a precision of seconds. Use the ISO 8601 format to specify timestamps with a precision of microseconds.

For more information, see Chapter 11: SQL Functions.

Subqueries and Views and the DURING Clause

A view or subquery whose FROM clause specifies table(s) instead of views or subqueries must include a DURING clause. A view or subquery whose FROM clause specifies another view or subquery instead of a table, does not require a DURING clause.

NOTE: Ensure that the DURING clause behaves as you expect. A badly formed DURING clause can cause the query to return incorrect results. The sections below describe considerations when defining the DURING clause.

Specifying DURING ALL in the View—HawkEye AP Console Usage

If a view has specified `DURING ALL` as its time range and you query against the view in HawkEye AP Console, always include the `DURING ALL` clause at the end of the main select.

- If you include the `DURING ALL` clause in the main select, the run-time user can limit the time range through the Date Criteria field.
- If you do not include the `DURING ALL` clause in the main select, the EDW will scan all data in the tables from which the view selects its data.

Specifying Actual Time Range in the View—Overlapping Time Ranges

If the `DURING` clause in the view specifies an actual time range, and the statement that selects from the view specifies a different but overlapping time range, the query engine resolves the time period to the intersection of the two time ranges.

In other words, if the view specifies a time range between January 1 and June 30 of last year and the `SELECT` statement specifies a time range between April 1 and August 30 of the same year, the query engine resolves the time range to April 1 through June 30.

NOTE: The results are the same when the `SELECT` statement is run from HawkEye AP Console. If the time range specified in the Date Criteria field overlaps the time range specified in the view, the query engine resolves the time range to the intersection of the two time ranges.

Specifying Actual Time Range in the View—Non-Overlapping Time Ranges

If the `DURING` clause in the view specifies an actual time range, and the statement that selects from the view specifies a different and *non*-overlapping time range so that the time ranges do *not* intersect, the query returns no rows. This behavior is identical to a query whose `WHERE` clause evaluates to false.

In other words, if the view specifies a time range between January 1 and June 30 of last year and the `SELECT` statement specifies a time range between July 1 and December 30 of the same year, the query returns no rows.

NOTE: The results are the same when the `SELECT` statement is run from HawkEye AP Console. If the time range specified in the Date Criteria field does *not* overlap the time range specified in the view, the query returns no rows.

WHERE Clauses

Use the WHERE clause to describe the characteristics of rows to be included in the result set. The WHERE clause takes a *conditional expression* that specifies which rows to include. Conditional expressions use the Boolean logic operators AND, OR, and NOT, and comparison operators, such as BETWEEN, IN, LIKE, <, <=, =, >, and >=.

NOTE: Column filters enhance performance only for queries that use the equals (= or `==`) or not equals (<> or `!=`) operators. A column filter does not enhance performance with other operators and functions, such as less than or greater than (<, >, >=, <=), `_min()`, `_max()`, `_upper()`, `_lower()`, or `IN`. For more information, see Defining Column Filters with Sensage SQL in Chapter 4, “Administering an EDW Instance” in the *Administration Guide*.

For more details, see “Operators”, on page 293.

For example, the following SELECT statement uses a WHERE clause to ensure that only rows with `RespSize` greater than zero are included in the result set.

```
SELECT ClientDNS, RespSize
  FROM example_webserv_100
  WHERE RespSize > 0
  DURING ALL;
```

You can also use a subquery with the `IN` operator inside of a `WHERE` clause. See “Using Subqueries with the IN Operator”, on page 295.

GROUP BY Clauses and Aggregation Queries

Aggregate queries use a GROUP BY clause, a HAVING clause, or contain an aggregate function. The semantics of aggregate queries differ from ordinary queries in an important way.

Generally, result sets contain one row for each row in the queried table that meets the selection criteria of the WHERE clause and which falls within the specified time frame of the DURING clause. In aggregation queries, rows are first collected into groups and the result will contain one row for each group of rows.

The SQL query engine performs aggregation before arranging the aggregated result rows in the sequence specified in the ORDER BY clause. When aggregation queries contains the FIRST or LAST keyword, only the specified first or last aggregation rows are returned in the final result set.

This section describes these topics:

- “Aggregation Partitioning”, next
- “Aggregation Functions”, on page 279
- “Multi-column GROUP BY”, on page 281
- “IN n PASSES”, on page 281
- “SLICE BY Clauses”, on page 282

Aggregation Partitioning

Usually, the GROUP BY option is used to specify how columns are partitioned. In the following example, the GROUP BY 1 option instructs the SQL query engine to collect all the rows from the specified time frame, which are separated into groups, one for each unique value of the first expression in the target specification.

```
SELECT ClientDNS, max(RespSize)
  FROM example_webserv_100
  GROUP BY 1
  DURING time('Feb 01 00:00:00 2002'), time('Mar 31 23:59:59 2002')
```

The final result will contain `ColumnA` and the largest value of `RespSize` in each group

```
+-------------------------------------------------+
| Results for SQL file >example-howto-sql-11.sql< |
+---------------+-------+-------------------------+
|   ClientDNS   |  max  |
|   (varchar)   |(int32)|
+---------------+-------*
```

```
output is post-sorted
+---------------+-------*
|163.191.183.106|   7121|
|194.114.63.8   |  17252|
|199.166.228.8  |   7121|
|203.164.82.149 |  17252|
|208.147.25.154 |  37361|
|209.102.202.154|  37361|
|212.242.116.219|      0|
|212.9.190.79   |  17252|
|216.239.46.200 |  12203|
|216.239.46.58  |      0|
|216.239.46.79  |      0|
|65.184.59.157  |    216|
|65.194.51.154  |  37361|
|66.127.84.10   |  12203|
+---------------+-------*
```

GROUP BY arguments can also be expressions. The above example could have been written as `GROUP BY ClientDNS`.

Occasionally, it is useful to compute an aggregate expression over a single group representing all the values, in which case no GROUP BY is needed. For example, the following query will return at most one row containing the total number of records in the time range in which `RespSize` is positive.

```
SELECT count(*)
  FROM example_webserv_100
  WHERE RespSize > 0
  DURING time('Feb 01 00:00:00 2002'), time('Mar 31 23:59:59 2002')

+-----------------------------------------------+
| Results for SQL file >example-howto-sql-12.sql< |
+-------+---------------------------------------+
| count |
|(int64)|
+-------*
output is post-sorted
+-------*
|     96|
+-------*
```

Aggregation Functions

The previous examples demonstrate the use of aggregate functions. Unlike ordinary functions that return one value for each row, aggregate functions return one value for each group of rows. These are the aggregate functions you can use in target clauses.

Function	Returns
`min(`*x*`)`	Smallest value of *x* in the group
`max(`*x*`)`	Largest value of *x* in the group
`count(*)`	Number of rows in the group
`sum(`*x*`)`	Sum of the values of *x* in the group

Function	Returns
`avg(x)`	Average of the values of *x* in the group
`median(x)`	Median of the values of x in the group
`_first(x)`	First of the values of *x* in the group seen by the system
`_last(x)`	Last of the values of *x* in the group seen by the system

NOTE: For more details about each of these aggregation functions, see “Aggregation Functions”, on page 337.

The following restrictions apply to aggregate functions:

- They may not appear in the GROUP BY column expressions.
- They may not appear in the arguments to other aggregates (for example, no `min(sum(x))`).
- Not all aggregates support all data types (for example, no `sum(timestamp)`).
- When aggregates are used in a query, then all columns outside the GROUP BY expressions must be contained within aggregates.

The last restriction requires further explanation. Consider the invalid query:

```
-- example of an invalid query
SELECT count(*), ClientDNS
  FROM example_webserv_100
  WHERE RespSize > 0
  DURING time('Feb 01 00:00:00 2002'), time('Mar 31 23:59:59 2002')
```

Because `count(*)` appears in the expression and no GROUP BY is present, this query will return at most one row for the group consisting of all records. However the second expression in the select semantically represents the multiple values of `ClientDNS`. Note the SQL query engine has no meaningful way of returning two streams of columns with differing numbers of values in each.

Often when a query such as this is encountered, what is intended is usually something like:

```
SELECT count(*), _first( ClientDNS )
  FROM example_webserv_100
  WHERE RespSize > 0
  DURING time('Feb 01 00:00:00 2002'), time('Mar 31 23:59:59 2002')

+--------------------------------------------------+
| Results for SQL file >example-howto-sql-14.sql< |
+--------+-------------+--------------------------+
| count |    first    |
|(int64)|  (varchar)  |
+-------+-------------*
output is post-sorted
+-------+-------------*
|     96|65.194.51.154|
+-------+-------------*
```

Multi-column GROUP BY

The GROUP BY option also allows grouping on a combination of columns. For example, the following query returns the number of records in the specified time frame for each unique combination of the first and second column in the select (in this case `ClientDNS` and `RespSize`).

```
SELECT ClientDNS, RespSize, count(*)
  FROM example_webserv_100
  GROUP BY 1, 2
  DURING time('Feb 01 00:00:00 2002'), time('Mar 31 23:59:59 2002')
```

```
+-------------------------------------------------+
| Results for SQL file >example-howto-sql-15.sql< |
+---------------+--------+-------+----------------+
|   ClientDNS   |RespSize| count |
|   (varchar)   |(int32) |(int64)|
+---------------+--------+-------*
output is post-sorted
+---------------+--------+-------*
|163.191.183.106|     204|      1|
|163.191.183.106|    7121|      1|
|194.114.63.8   |   17252|      1|
|199.166.228.8  |    7121|     10|
|203.164.82.149 |   17252|      1|
|208.147.25.154 |   37361|      6|
|209.102.202.154|   37361|      1|
|212.242.116.219|       0|      1|
      ...
|212.9.190.79   |   12935|      1|
|212.9.190.79   |   17252|      1|
|216.239.46.200 |       0|      1|
...
|66.127.84.10   |    7121|      2|
|66.127.84.10   |   12203|      1|
+---------------+--------+-------*
```

IN *n* PASSES

The GROUP BY clause can cause the EDW to consume a lot of memory on each host in the EDW instance. To reduce this memory burden, include the proprietary IN *n* PASSES subclause at the end of the GROUP BY clause, where *n* is an integer in the range of 2-10. Each pass then uses *1/n* as much memory as a complete pass but requires *n* times as many scans through the data. For this reason, Sensage discourages the use of this feature unless absolutely necessary for getting queries to complete -- for queries that don't consume lots of memory, it will dramatically reduce performance. Here's an example:

```
SELECT ClientDNS, RespSize, count(*)
  FROM example_webserv_100
  GROUP BY 1, 2 IN 3 PASSES
  DURING time('Feb 01 00:00:00 2002'), time('Mar 31 23:59:59 2002')
```

```
+-------------------------------------------------+
| Results for SQL file >example-howto-sql-16.sql< |
+---------------+--------+-------+----------------+
|   ClientDNS   |RespSize| count |
|   (varchar)   |(int32) |(int64)|
+---------------+--------+-------*
output is post-sorted
+---------------+--------+-------*
|163.191.183.106|     204|      1|
|163.191.183.106|    7121|      1|
|194.114.63.8   |   17252|      1|
|199.166.228.8  |    7121|     10|
|203.164.82.149 |   17252|      1|
|208.147.25.154 |   37361|      6|
|209.102.202.154|   37361|      1|
|212.242.116.219|       0|      1|
...
|212.9.190.79   |   12935|      1|
|212.9.190.79   |   17252|      1|
|216.239.46.200 |       0|      1|
...
|66.127.84.10   |    7121|      2|
|66.127.84.10   |   12203|      1|
+---------------+--------+-------*
```

SLICE BY CLAUSES

The SLICE BY clause is used with the GROUP BY clause to separate records that would normally be aggregated in a single group into multiple groups. For example, the following SELECT statement collects rows into groups according to the value of `ClientDNS` and creates a distinct group whenever a record with `Url = '/robots.txt'` is seen.

```
SELECT ClientDNS, count(*)
  FROM example_webserv_100
  GROUP BY 1
  SLICE BY Url = '/robots.txt'
  DURING time('Feb 01 00:00:00 2002'), time('Mar 31 23:59:59 2002')
```

```
+-------------------------------------------------+
| Results for SQL file >example-howto-sql-17.sql< |
+---------------+-------+-------------------------+
|   ClientDNS   | count |
|   (varchar)   |(int64)|
+---------------+-------*
output is post-sorted
+---------------+-------*
|163.191.183.106|      2|
|194.114.63.8   |      1|
|199.166.228.8  |     10|
|203.164.82.149 |      1|
|208.147.25.154 |      6|
|209.102.202.154|      1|
|212.242.116.219|      1|
|212.9.190.79   |     37|
|216.239.46.200 |      3|
|216.239.46.58  |      1|
|216.239.46.79  |      1|
|65.184.59.157  |      1|
|65.194.51.154  |      6|
|66.127.84.10   |     29|
+---------------+-------*
```

The SLICE BY option is commonly used with the `_fifo()` function to split a group of records based on the previous value of an expression. The `_fifo()` function implements a simple first-in-first-out queue and for a given value of `<key>`, `_fifo(<key>,<a>,<b>)` always returns b on the first call and on each subsequent call the previous value of a is returned. This means the following query will collect the rows into groups according to the value of `ClientDNS` but will create a distinct group whenever `_int32(ts)` increases by more than 10 (that is, more than 10 seconds have elapsed between records).

```
SELECT ClientDNS, count(*)
  FROM example_webserv_100
  GROUP BY 1
  SLICE BY _int32(ts) - _fifo(ClientDNS, _int32(ts), _int32(ts)) > 10
  DURING time('Feb 01 00:00:00 2002'), time('Mar 31 23:59:59 2002');
```

```
+-------------------------------------------------+
| Results for SQL file >example-howto-sql-18.sql< |
+---------------+-------+-------------------------+
|   ClientDNS   | count |
|   (varchar)   |(int64)|
+---------------+-------*
output is post-sorted
+---------------+-------*
|163.191.183.106|      2|
|194.114.63.8   |      1|
|199.166.228.8  |      1|
      ...
|199.166.228.8  |      1|
|203.164.82.149 |      1|
|208.147.25.154 |      1|
|208.147.25.154 |      1|
      ...
|65.194.51.154  |      1|
|66.127.84.10   |      1|
|66.127.84.10   |      1|
|66.127.84.10   |     27|
+---------------+-------*
```

HAVING CLAUSES

A SELECT statement uses the HAVING clause to eliminate groups from the result. For example, the following SELECT statement:

- collects rows into groups according to the value of the `ClientDNS` column
- counts the records in each group
- eliminates groups in which the sum of the values in the `RespSize` column is greater than 100

```
SELECT ClientDNS, count(*)
  FROM example_webserv_100
  GROUP BY 1
  HAVING sum(RespSize) > 100
  DURING time('Feb 01 00:00:00 2002'), time('Mar 31 23:59:59 2002')
```

```
+-------------------------------------------------+
| Results for SQL file >example-howto-sql-19.sql< |
+---------------+-------+------------------------+
|   ClientDNS   | count |
|   (varchar)   |(int64)|
+---------------+-------*
output is post-sorted
+---------------+-------*
|163.191.183.106|      2|
|194.114.63.8   |      1|
|199.166.228.8  |     10|
|203.164.82.149 |      1|
|208.147.25.154 |      6|
|209.102.202.154|      1|
|212.9.190.79   |     37|
|216.239.46.200 |      3|
|65.184.59.157  |      1|
|65.194.51.154  |      6|
|66.127.84.10   |     29|
+---------------+-------*
```

NOTE:

- The column referenced in the HAVING clause does not need to be included in the SELECT clause.
- Although the query must include at least one aggregated value, it does not require a GROUP BY clause.

The HAVING clause is similar to the WHERE clause. Both clauses restrict the values in the final result. The main difference is that the SQL query engine uses the WHERE clause to eliminate rows before it aggregates them, and it uses the HAVING clause to eliminate groups after it aggregates them.

ORDER BY CLAUSES

When executing a SELECT statement without an ORDER BY clause, the SQL query engine returns rows in whatever order they arrive from the storage system. If you include an ORDER BY clause, the engine sorts rows in an intermediate result set before producing the final result set. You provide a list of target columns as part of the ORDER BY clause to instruct the SQL query engine how to sort the results. Target columns can be specified by their ordinal number within the target clause.

For example, the following SELECT statement specifies that the result set should be sorted in ascending order of values in the target `ClientDNS` column, which has an ordinal number of 1.

```
SELECT ClientDNS
  FROM example_webserv_100
  ORDER BY 1
  DURING ALL;
```

ORDER BY clauses can have multiple arguments, in which case it sorts the results by the first argument (the major sort field), using the second argument as the tie-breaker, and so on. The arguments in ORDER BY clauses can be expressions as well as ordinal numbers. For example:

```
SELECT ClientDNS, ts
  FROM example_webserv_100
  ORDER BY ClientDNS, 2
  DURING ALL;
```

Control the sort order of each argument with the ASC or DESC modifier keywords. For example:

```
SELECT ClientDNS, RespSize
  FROM example_webserv_100
  ORDER BY _strlowercase(UserAgent) DESC, 1 ASC
  DURING ALL;
```

ORDER BY clauses sort result rows in ascending order by default.

UNION ALL Clauses

Generally, a single SELECT statement queries one table and generates a results set. Sometimes you want a results set with information queried from several tables, where the tables have the same schema but contain different sets of data.

For example, you may have several different tables that hold weblog entries from different web servers. The UNION ALL clause lets you combine queries of tables for each region into a SELECT statement that produces a unified result.

```
SELECT ClientDNS, RespSize
  FROM example_webserv_100
  DURING ALL

UNION ALL

SELECT ClientDNS, RespSize
  FROM example_webserv2_100
  DURING ALL
;
```

NOTE: HawkEye AP SQL does not support UNION or UNION DISTINCT. You can eliminate duplicate rows in each subsidiary SELECT statement with the DISTINCT keyword, but you cannot eliminate rows that are duplicated across the subsidiary results. For information on an alternative solution, see “Subqueries and UNION ALL Clauses”, on page 308.

Restrictions on UNION ALL

The SQL query engine places the following restrictions on the UNION ALL clause:

- All the subselects within the SELECT statement must return the same number of target columns.
- Each target column must have the same data type in each subselect, respectively.

Alternative to UNION ALL

Sensage SQL supports *list expressions* as an alternative way of producing a unified result set from two or more similar tables. For more information, see “List Expressions and FROM Clauses”, on page 315.

Data Types

This section describes the data types that HawkEye AP SQL supports:

- "bool", next
- "float", on page 286
- "int32", on page 287
- "int64", on page 287
- "timestamp", on page 287
- "varchar", on page 288

bool

The bool data type represents the logical Boolean values "true" and "false."

The result of `CONVERT(<target_data_type>, <bool_value>)` is as follows:

Target Data Type	Conversion Result
int32	1 if the Boolean value is logically "true", or 0 if the Boolean value is logically "false"
int64	1 if the Boolean value is logically "true", or 0 if the Boolean value is logically "false"
float	1 if the Boolean value is logically "true", or 0 if the Boolean value is logically "false"
timestamp	Not allowed; an exception is raised on attempts to convert bool values to timestamp values
varchar	"true" or "false", depending on the logical Boolean value

float

The float data type represents fractional quantities and very larger or small numbers. The internal representation is s 64-bit floating-point value in the range of -1.797693e+308 to 1.797693e+308.

The result of `CONVERT(<target_data_type>, <float_value>)` is as follows:

Target Data Type	Conversion Result
bool	Logical "true" if the value is non-zero; logical "false" if the value is 0
int32	The floating-point value converted to the closest integer, without rounding
int64	The floating-point value converted to the closest integer, without rounding
timestamp	The floating-point value, which represents the number of seconds since 1/1/1970 GMT, converted to the number of microseconds since 1/1/1970.
varchar	Formatted floating-point number. If the number of fractional digits exceeds 6, the number is formatted in scientific notation; otherwise, regular numeric formatting is used.

int32

The int32 data type represents signed 32-bit integer values in the range of -2,147,483,648 to +2,147,483,647. The int32 data type is the default for integer quantities. Use the int64 data type for larger integer values.

The result of `CONVERT(<target_data_type>, <int32_value>)` is as follows:

Target Data Type	Conversion Result
bool	Logical “true” if the value is non-zero; logical “false” if the value is 0
int64	The equivalent 64-bit integer value
float	The equivalent floating-point value
timestamp	Positive integer values, which represent the number of seconds since 1/1/1970 GMT, are converted to the number of microseconds since 1/1/1970. If the integer value is too large, the timestamp is set to the maximum value. For negative integer values, the timestamp is set to the minimum value.
varchar	Unformatted integer value

int64

The int64 data type represents large integer quantities. The internal representation is a signed 64-bit integer value in the range of -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807.

The result of `CONVERT(<target_data_type>, <int64_value>)` is as follows:

Target Data Type	Conversion Result
bool	Logical “true” if the value is non-zero; logical “false” if the value is 0
int32	The equivalent 32-bit integer value
float	The equivalent floating-point value
timestamp	Positive integer values represent the number of microseconds since 1/1/1970 GMT and are stored in the timestamp value without conversion. For negative integer values, the timestamp is set to the minimum value.
varchar	Unformatted integer value

timestamp

The timestamp data type represents date-and-time-of-day values, expressed as the number of microseconds since January 1, 1970, GMT. The latest timestamp that can be represented falls in the year 2038.

The result of `CONVERT(<target_data_type>, <timestamp_value>)` is as follows:

Target Data Type	Conversion Result
bool	Not allowed; an exception is raised on attempts to convert timestamp values to bool values
int32	The number of seconds since January 1, 1970, GMT, if the value is greater than 0; otherwise 0.
int64	The number of microseconds since January 1, 1970, GMT, if the value is greater than 0; otherwise 0.
float	The number of seconds since January 1, 1970, GMT, if the value is greater than 0; otherwise 0.
varchar	An ISO 8601 formatted timestamp. For more information, see "Converting varchar Values to timestamp Values", on page 289.

varchar

The varchar data type represents character data stored with UTF-8 character encoding. This encoding enables the representation of international characters from the Unicode character set.

The maximum length of a varchar value is 2147483646 bytes, which is two bytes fewer than two gigabytes.

The result of `CONVERT(<target_data_type>, <varchar_value>)` is as follows:

Target Data Type	Conversion Result
bool	Logical "true" if the value is "true" or "max"; otherwise, logical "false"
int32	The equivalent 32-bit integer value, if the complete varchar value contains only leading spaces, followed by an optional plus or minus sign, followed by a sequence of decimal numerals with no thousands or decimal separators; otherwise, 0. If the varchar value contains only "min" the int32 value is -2,147,473,647; if the varchar value contains only "max" the int32 value is 2,147,473,647.
int64	The equivalent 64-bit integer value, if the complete varchar value contains only leading spaces, followed by an optional plus or minus sign, followed by a sequence of decimal numerals with no thousands or decimal separators; otherwise, 0. If the varchar value contains only "min" the int64 value is -9,223,372,036,854,775,807; if the varchar value contains only "max" the int64 value is 9,223,372,036,854,775,807.
float	The equivalent floating-point value, if the complete value contains: 1) only leading spaces, followed by an optional plus (+) or minus (-) sign, followed by a sequence of decimal numerals with no thousands and at most one decimal separator; 2) only leading spaces followed by a number expressed in valid scientific notation; otherwise, 0. If the varchar value contains only "min" the float value is -1.797693e+308; if the varchar value contains only "max" the float value is 1.797693e+308.
timestamp	A string containing the ISO 8601 (Time & Date) representation of the value. For example, the following varchar value represents a timestamp for October 10th, 1997, at 6:09 PM, UTC: `'1997-10-10T17:18:09.000000Z'` For more details, see "Converting varchar Values to timestamp Values", next.

Converting varchar Values to timestamp Values

The rules for converting a varchar value to a timestamp value are as follows:

- If the varchar value is 'min', the timestamp is assumed to be the smallest possible time supported by the system (the first microsecond of 1/1/1970).
- If the varchar value is 'max', the timestamp is assumed to be the greatest possible time supported by the system: 2,147,483,647,000,000 microseconds since 1/1/1970 (which is sometime in 2038-01-19).
- If the varchar value is a legal ISO 8601 timestamp (for example, `1997-10-10T17:18:09.000000Z`), the timestamp is assumed to be the corresponding microsecond.
- If the varchar value has the form 'Mon Day Hour:Min:Sec Year' (for example, `Oct 10 17:18:09 1997`), the timestamp is assumed to be the first microsecond of the corresponding second.
- If the varchar value has the form "MM/DD/YY" (for example, `10/10/97`), the timestamp is assumed to be the first microsecond of the corresponding day.
- If the varchar value has the form "MM/DD/YY HH:MM:SS" (for example, '10/10/97 17:18:09') then the timestamp is assumed to be the first microsecond of the corresponding second.
- If the varchar value is a relative time, the timestamp is assumed to be that number of seconds since 1/1/1970.

Varchar Representations of Durations

A duration is a value representing an amount of time. The internal representation is a structure with a string containing a number for the amount of time, followed by a string with one of the following codes that indicates the unit of time:

Unit Code	Unit of Time
`usec`	Microseconds. For example, `5usec`.
`msec`	Milliseconds. For example,`5msec`.
`sec`	Seconds. For example,`5sec`.
`min`	Minutes. For example,`5min`.
`hr` or `hour`	Hours. For example,`5hr` or `5hour`.
`day`	Days. For example,`5day`.
`mon`	Months. For example,`5mon`.
`yr` or `year`	Years. For example,`5yr` or `5year`.

DATA SOURCE EXPRESSIONS

Data source expressions define the input data used in a SELECT statement.

This section describes the two basic sources of data:

- "Column Expressions", next
- "Literal constants", on page 290

Column Expressions

A *column expression* represents a column of data from a table or subquery. In the following SELECT statement for example, `ClientDNS` is a column expression representing the `ClientDNS` column in the `example_websrv_100` table.

```
SELECT ClientDNS
  FROM example_webserv_100
  DURING ALL;
```

Column expressions consist of a letter followed by a series of letters and digits. Generally the expression must match the name of a column in the table to be queried. Any expression that resolves to the name of a column can be used as a column expression.

Column expressions may include the table name as prefix, separated with a period (`.`) from the column name. For example, the previous statement is equivalent to the following:

```
SELECT example_webserv_100.ClientDNS
  FROM example_webserv_100
  DURING ALL;
```

The data type of a column expression must match the type of the underlying column in the table or subquery.

Optional column expressions

An *optional* column expression specifies a list of columns. The SQL query processor searches the list until it finds a column that is present in the table specified in the FROM clause. If none of the candidates are present, the value of the final (default) expression is returned. The data types of the columns in the list must be the same.

For example, based on the following SELECT statement, the query processor attempts to return values from the `HostName` if it exists in the table to be queried:

```
SELECT _optional_column(HostName, ClientDNS, '-')
  FROM example_webserv_100
  DURING ALL;
```

If the table does not contain a column named `HostName`, the query processor attempts to return values from `ClientDNS`. If neither column is found, the literal constant '-' is returned.

TIP: Optional columns are particularly helpful when used with list expressions in the FROM clause. For more information, see “List Expressions and FROM Clauses”, on page 315.

Literal constants

A *literal constant* is simply a value. For example, the value `100` in the following SELECT statement is a literal constant:

```
SELECT ClientDNS
  FROM example_webserv_100
  WHERE RespSize > 100
  DURING ALL;
```

There are four kinds of literals:

- Integer Literals
- Floating-Point Literals
- String Literals
- Typecast Literals

Integer Literals

An integer literal is a series of base 10 digits (i.e. '0', '1', '2', '3', '4', '5', '6', '7', '8' and '9'). In the query above, 100 is an integer literal. Integer literals have an implicit data type of int32.

Floating-Point Literals

A floating-point literal is a series of base 10 digits followed by a decimal point, some more digits, and an optional exponent. For example, the following values are floating-point literals:

```
1000.0 -- one thousand
3.1416 -- pi to 4 decimal places
1.0e6 -- one million
```

Floating-point literals have an implicit data type of `float`.

String Literals

A string literal is a sequence of characters enclosed between single or double quotes. For example, the values `'Feb 01 00:00:00 2002'` and `'Mar 31 23:59:59 2002'` are string literals. The following values are also string literals:

```
'' -- two single quotes make an empty string
"" -- as do two double quotes
'http://www.acme.com' -- the literal value is everything between the quotes
```

Quotes themselves are not part of literal values; `'a'` is a string literal containing the single character `a`.

String literals may include new-line characters. For example, the following text is a single string literal that begins and ends with a single quote and contains newline codes and double quotes.

```
'sub Link {
    return "<a href=\"$_[0]\">$_[0]</a>";
}
'
```

A string literal may use the standard SQL convention to include embedded quotes by repeating the quote. The following two literals have the same value:

- `'adam''s new shoes'` -- if you use two single quotes (`''`) in a literal, it means include a single quote
- `"adam's new shoes"` -- or use double quotes around the value to include a single quote within it

String literals have an implicit data type of `varchar`.

Syntax for Defining Long String Literals

It can be inconvenient to deal with quoting issues with apostrophes, backslashes, and so on -- particularly in complex and multi-line character strings. For example, you cannot simply cut-and-paste Perl code when declaring Perl functions, because the code may contain special characters and generally spans multiple lines.

To make it easier to use long varchar literals, the SQL parser supports *here document* syntax. Here document syntax alerts the SQL query engine that a long varchar value with special characters and multiple lines has been defined. The syntax begins with double chevrons (<<) followed by a user-defined *limit string*. From that point forward, all characters, including new lines and other special characters, are treated as a single varchar value. The value ends with the next occurrence of *`limit-string`*, which must be at the beginning of its own line.

For example:

```
WITH $bigstring AS <<EOF
    This is a very long string with all sorts of funny characters
    '~!@#$%&*()_=-<>?,./[]\{}|;':"
    and it doesn't matter
EOF

SELECT top 1 $bigstring
  FROM example_webserv_100
  DURING ALL;
```

Note that the new-line character immediately after `<<EOF` and the new-line character before the final `EOF` are not part of the `varchar` value.

TIP: Use here document syntax when you declare user-defined functions and aggregates. For more information, see "User-Defined Subroutines", on page 306.

Typecast Literals

A typecast literal is a string literal preceded by a type name. For example, the following values are typecast literals:

- `int32 '10'` -- The integer 10 as a 32 bit integer
- `int64 '10'` -- The integer 10 as a 64 bit integer
- `bool '0'` -- The Boolean value 0 (false)
- `float '10.0'` -- The floating point value 10.0
- `timestamp 'Feb 05 00:00:00 2001'` -- The timestamp value for February 5th, 2001
- `varchar '10'` -- Same as '10'

Typecast literals specify explicitly the data type of a value. For example, if you specify the integer literal `10`, it will be an int32 value; specify `int64 '10'` instead if you need 10 to be an int64 value.

Data Processing Expressions

Data processing expressions do the computational work in a SELECT statement.

This section describes these topics:

- “Operators”, next
- “Functions”, on page 297
- “Conversion Expressions”, on page 299
- “CASE Expressions”, on page 301

Operators

An *operator* computes the result of a specific operation on two operands. In the following SELECT statement, the expression `ColumnB > 100` uses a comparison operator to include only rows where the value of `ReapSize` is greater than 100. The greater-than symbol (>) is the operator; `ReapSize` and `100` are the operands.

```
SELECT ClientDNS
  FROM example_webserv_100
  WHERE RespSize > 100
  DURING ALL;
```

This section describes these topics:

- “Math Operators”, next
- “The String Concatenation Operator”, on page 294
- “Comparison Operators”, on page 294
- “Logical Operators”, on page 296
- “Operator Precedence”, on page 297

Math Operators

Math operators perform simple arithmetic computations. These are the math operators supported by Sensage SQL.

Symbol	Arithmetic Operation	Example
+	Addition	`3 + 2` yields `5`
-	Subtraction	`3 - 2` yields `1`
-	Unary minus	-3 yields the negative of 3; it operates as if the expression were `0 - 3`. When you apply the unary minus operator to a negative value, the result is its absolute value.
*	Multiplication	`3 * 2` yields `6`
/	Division	`3 / 2` yields `1.5`
%	Modulo	3 % 2 yields 1

For example, the following SELECT statement uses the multiplication operator to compute the product of RespSize and `2`.

```
SELECT ClientDNS
  FROM example_webserv_100
  WHERE RespSize * 2 > 100
  DURING ALL;
```

Math operations can encounter results that require truncation:

- For numeric data types (`int32`, `int64`, `float`), the semantics of the underlying 'C' data type govern the result if an overflow or underflow occurs.
- If the result of a `timestamp` expression exceeds 2,147,483,648,000,000 microseconds, it is set to 2,147,483,648,000,000, which falls sometime in the year 2038.

The result of a math operation has `a` data type that depends on the data types of the operands. Generally, the result has the data type that is suitable for the result, which may differ from the data types of the operands.

The String Concatenation Operator

You can use the plus symbol (+) as a *string concatenation* operator when the operands are `varchar` values. For example, the following expression yields the varchar value "`myTableSpace.myTable`".

```
'myTableSpace' + '.' + 'myTable'
```

If the result of a string concatenation operation exceeds 32,767 characters, the concatenated value is truncated.

The result of a string concatenation operation has `varchar` as its data type.

Comparison Operators

Comparison operators perform simple comparisons between one value and another. The two values and the comparison operator resolve to a `bool` value of "true" or "false", depending on the outcome of the comparison. Use comparison operators to construct simple conditional expressions for WHERE, SLICE BY, and HAVING clauses.

Basic Comparison Operators

These are the basic comparison operators supported by HawkEye AP SQL.

Symbol	Comparison Operation	Example Expression	Result
<	Less than	`3 < 2`	`false`
<=	Less than or equal to	`3 <= 3`	`true`
	Greater than	`3 > 2`	`true`
>=	Greater than or equal to	`2 >= 2`	`true`

=
==
	Equal to	`3 = 2`
		`3 == 2`
			`false`

<>
!=
	Not equal to	`3 <> 2`
		`3 != 2`
			`true`

For example, the following SELECT statement returns only `ClientDNS` values from rows in the `example_webserv_100` table where the corresponding value of `RespSize` is greater than 100.

```
SELECT ClientDNS
  FROM example_webserv_100
  WHERE RespSize > 100
  DURING ALL;
```

The result of a basic comparison operation has `bool` as its data type.

ADVANCED COMPARISON OPERATORS

These are the advanced comparison operators supported by HawkEye AP SQL.

Operator	Set Operation	Example
`IN`	Is the operand included in a set of enumerated values?	`3 IN (2, 7, 5, 15, 3)` yields `true`
`BETWEEN`	Is the operand included in the set of values defined by a beginning and ending range of consecutive values?	`3 BETWEEN 2, 15` yields `true`
`LIKE`	Does the varchar operand match a particular pattern of characters?	`’abc’ LIKE ’a__’` yields `true`

NOTE: HawkEye AP SQL does not support the IS NULL and IS NOT NULL comparison operators, because the EDW does not support columns with NULL values.

The IN Operator

The IN comparison operator yields true if the value of an operand exists within a list of possible values. For example, the following SELECT statement returns `ClientDNS` values from rows where the value of `HttpVers` is `'HTTP/1.0'` or `'HTTP/1.1'`.

```
SELECT ClientDNS
  FROM example_webserv_100
  WHERE HttpVers IN ('HTTP/1.0', 'HTTP/1.1')
  DURING ALL;
```

Use the corresponding NOT IN operator to determine if an operand is excluded from a list of values.

The result of an IN comparison has `bool` as its data type.

TIP: You can use a list variable as the right-hand operand of the IN operator. For more information on list variables, see “Working with Lists”, on page 313.

Using Subqueries with the IN Operator

Instead of using the `IN` operator to specify an explicit list of values, you can use a subquery to derive the list of values. The subquery can reference the same table, view, or subquery as that used for the outer query, or the subquery can reference a different view, table, or subquery. The user must have the required permissions to access the view or table. Correlated subqueries that reference expressions from the outer query are not allowed. The subquery can be any general query and can include `UNION` clauses, `WITH` clauses, and subqueries. The subquery can have only one expression in the target list of its `SELECT` clause.

For example, the following SQL statement uses the `WHERE` clause to limit the results to those users in the engineering department:

```
SELECT username, hostname, result
  FROM logins
  WHERE username IN
  (SELECT username FROM staff_list WHERE department="engineering")
  DURING ALL;
```

The BETWEEN Operator

The BETWEEN comparison operator yields true if the value of an operand falls within a range of possible values. Use the corresponding NOT BETWEEN operator to determine if an operand is excluded from a range values.

The result of a BETWEEN comparison has `bool` as its data type.

The LIKE Operator

The LIKE comparison operator yields true if the value of a varchar operand matches a particular pattern of characters. The matching pattern can contain explicit characters and special wildcard characters. The percent sign (%) is a wildcard that matches zero or more characters of any kind; an underscore matches any single character in a particular position within the pattern.

For example, the following SELECT statement returns `ClientDNS` values from rows where the value of `HttpVers` begins with the characters `'HTTP/1'`.

```
SELECT ClientDNS
  FROM example_webserv_100
  WHERE HttpVers LIKE 'HTTP/1%'
  DURING ALL;
```

Because the matching pattern includes the percent wildcard (`%`), rows with `’HTTP/1.0’` and `’HTTP/1.1’` are returned in the result set.

NOTE: Sensage SQL supports neither asterisks (`*`) nor questions marks (`?`) as wildcard characters, nor does it support the optional ESCAPE subclause in LIKE comparisons.

Use the corresponding NOT LIKE operator to determine if a varchar operand does not match a pattern of characters.

The result of a LIKE comparison has `bool` as its data type.

Logical Operators

Logical operators perform simple “true” or “false” comparisons between expressions that themselves resolve to `bool` values. Use logical operators to construct compound conditional expressions for WHERE, SLICE BY, and HAVING clauses.

These are the logical operators supported by HawkEye AP SQL.

Operator	Logic Operation	Examples
`AND`	Logical AND; true only if both operands are true.	• `true AND true` yields `true` • `true AND false` yields `false` • `false AND false` yields `false`
`OR`	Logical OR; true if either operand is true.	• `true OR true` yields `true` • `true OR false` yields `true` • `false OR false` yields `false`
`NOT`	Logical negation	• `NOT true` yields `false` • `NOT false` yields `true`

In the following SELECT statement, the WHERE clause has a compound conditional expression that compares two simple conditional expressions with the AND logical operator.

```
SELECT ClientDNS
  FROM example_webserv_100
  WHERE (RespSize > 100) AND (HttpVers = 'HTTP/1.1')
  DURING ALL;
```

The statement returns values in the `ClientDNS` column only for rows where both comparisons are true; the value of `RespSize` must be greater than 100, and the value of `HttpVers` must be the string `'HTTP/1.1'`.

The result of a logic operation has `bool` as its data type.

Operator Precedence

When several operators appear in a complex expression, they take precedence in the following order.

Symbol	Operation
`-`	Unary minus
`* / %`	Multiplication, division, and modulo
`+ -`	Addition, subtraction, and string concatenation
`< <= > >= = <> IN BETWEEN LIKE`	Comparison
`NOT`	Logical negation
`AND`	Logical AND
`OR`	Logical OR

Functions

Function expressions perform specialized calculations that return a single result. For example, the following SELECT statement uses two functions to calculate timestamp values for the DURING clause:

```
SELECT ClientDNS
  FROM example_webserv_100
  DURING time('Feb 01 00:00:00 2002'),
         _timeadd( time('Feb 01 00:00:00 2002'), 1, 'day')
;
```

The first function, `time()`, calculates the timestamp value that corresponds to the literal constant `'Feb 01 00:00:00 2002'`. The function takes one *argument*, a varchar that contains a date. It yields a timestamp value as its *return value*.

The second function, `_timeadd()`, also calculates a timestamp value. It takes three arguments instead of one, and it uses a different algorithm to compute its return value. The first argument is a timestamp, to which a specified amount and unit of time are added. This function interprets the second two arguments and adds them to the first. In the example above, the `_timeadd()` function returns a date that is one day later than the specified timestamp. In other words, this function returns "`Feb 02 00:00:00 2002`". Notice that this function also contains the `time()` as an argument.

Function Syntax

In general, a function expression comprises the name of the function, followed by a comma-separated list of arguments enclosed in parentheses. For example, the following are function expressions:

- `_now()` -- a function that takes no arguments; it returns a timestamp for the current system time.
- `time('02/01/01')` -- a function that takes a varchar argument; in this case the argument is a literal constant, but any expression that yields a varchar representation of a date and time-of-day could be used.
- `_strcat(ColA, ColB, ColC)` -- a function that takes three varchar arguments and returns a varchar value that concatenates them; the arguments to this function are column expressions. The data types of the columns `ColA`, `ColB`, and `ColC` must be varchar.

The result of a function has the data type that the function declaration defines.

RELATED TOPICS

- To learn about built-in Sensage SQL functions, including their arguments and their return types, see Chapter 11: SQL Functions.
- To learn how to declare user-defined functions, see “User-Defined Subroutines”, on page 306, and Chapter 12: Perl Subroutines.

Asterisks as Column-Expression Arguments

An asterisk (`*`**)** can be used as a special column-expression argument. The SQL query engine replaces the asterisk with a comma-separated list of the columns in the table identified in the FROM clause.

For example, the following SELECT statement:

```
SELECT _strcat(*)
  FROM example_webserv_100
  DURING ALL;
```

is equivalent to:

```
SELECT _strcat(ts, ClientIP, ClientDNS, Method, Url,
               HttpVers, RespCode, RespSize, Referrer,
               UserAgent, RespTime)
  FROM example_webserv_100
  DURING ALL;
```

COUNT(*)

When an asterisk is used as an argument, the `count()` aggregate function operates as a special case. Instead of replacing the asterisk with a comma-separated list of the columns in the table, the SQL query engine treats `count(*)` as if it were `count(1)`. The the special nature of the `count(*)` expression is supported for compatibility with standard SQL.

The SORT BY Modifier Keyword

When an aggregate function is called, it is passed arrays of column values in the order by which the underlying column expression generates them. However, if a function expression that takes column expressions as arguments is followed by the SORT BY modifier keyword, the SQL query engine first sorts the column values on the specified function argument before invoking the function.

Consider the following SELECT statement:

```
SELECT ClientDNS,
        _strsum( Url, ts ) SORT BY 2
  FROM example_webserv_100
  GROUP BY 1
  DURING ALL ;
```

The `_strsum()` aggregate function ignores the second argument as a target column. However, the `Url` values will be sorted according to the value of `ts` before `_strsum()` is invoked. The `_strsum()` function receives the `ClientDNS` values in the order of the `ts` values. This is often useful when using `_strsum()` to concatenate string values.

NOTE: Unless SORT BY is specified, the order of the records passed into an aggregate is non-deterministic. Two invocations can yield different sort orders. The EDW runs queries in parallel across a cluster of machines and cannot guarantee which machines will produce records the fastest.

The DISTINCT Modifier Keyword

The DISTINCT modifier keyword ensures that only the unique sets of values are passed into aggregate functions.

```
SELECT count( DISTINCT ClientDNS )
  FROM example_webserv_100]
  DURING ALL;
```

If an aggregate function takes multiple arguments, the DISTINCT modifier keyword makes the entire set unique. If any argument differs, the set is considered unique and the aggregate is called. The DISTINCT modifier keyword is useful with Perl aggregates.

NOTE: The DISTINCT and SORT BY modifiers can work together; both operations happen at once.

IMPORTANT: The SQL query engine can use a lot of memory to process function expressions that use the DISTINCT modifier keyword.

Conversion Expressions

A *conversion* expression performs a simple conversion of a data value from its current data type to another. Use conversion expressions to put data values into the form expected by subsequent operations.

There are three kinds of conversion expressions:

- CONVERT Expressions
- Function-style Conversion Expressions

- Typecast Literal Conversion Expressions
- Typecast Literal Conversion Expressions

CONVERT Expressions

A *CONVERT* expression has the form:

```
CONVERT(<target_data_type>, <value_to_convert>)
```

Use one of these allowed target data types.

Target Data Types
`bool`
`int32`
`int64`
`float`
`timestamp`
`varchar`

For example, the following SELECT statement uses a CONVERT expression to convert values in the `ts` column from timestamp values to human-readable, character-based values:

```
SELECT CONVERT(varchar,ts)
  FROM example_webserv_100
  DURING ALL;
```

Function-style Conversion Expressions

A *function-style* conversion expression has one of the following forms, depending on the target data type:

Conversion Function	Conversion Result
`_bool(x)`	Equivalent to `CONVERT(bool,x)`
`_int32(x)`	Equivalent to `CONVERT(int32,x)`
`_int64(x)`	Equivalent to `CONVERT(int64,x)`
`_float(x)`	Equivalent to `CONVERT(float,x)`
`_timestamp(x)`	Equivalent to `CONVERT(timestamp,x)`
`_varchar(x)`	Equivalent to `CONVERT(varchar,x)`

The example below illustrates how you can use the `_varchar()` function as an alternate to the example SELECT statement for CONVERT expressions:

```
SELECT _varchar(ts)
  FROM example_webserv_100
  DURING ALL;
```

Typecast Literal Conversion Expressions

A *typecast literal* conversion expression has the form:

```
<typecast_literal> '<literal_value>'
```

Use one of these typecast literals, depending on the target data type.

Typecast Literal
bool
int32
int64
float
timestamp
varchar

This means the example SELECT statement for CONVERT expressions could also be written as:

```
SELECT varchar ts
  FROM example_webserv_100
  DURING ALL;
```

CASE Expressions

A *CASE* expression supports conditional target specifications. CASE expressions have the following syntax:

```
CASE WHEN <conditional-expression> THEN <value>
     WHEN <conditional-expression> THEN <value>
     ...
     ELSE <value>
END
```

For example, the following SELECT statement returns result rows with VARCHAR values of `<1k`, `1k`, or `>1k,` depending on whether the value of `RespSize` is less than, equal to, or greater than 1024.

```
SELECT CASE WHEN RespSize < 1024 THEN "<1k"
            WHEN RespSize = 1024 THEN "1k"
            ELSE ">1k"
       END
  FROM example_webserv_100
  DURING ALL;
```

Processing Directives

The proprietary WITH keyword introduces clauses that direct how the SQL query engine processes SELECT statements. WITH clauses let you declare symbols, such as named constants, user-defined functions, and subqueries, that you can use within SELECT statements. WITH clauses also let you configure the execution environment in which queries run. The syntax of WITH clauses depends on the kind of directive the WITH clause declares.

This section describes these topics

- "Macros", next
- "User-Defined Subroutines", on page 306
- "Subqueries", on page 307
- "Table-Name Substitutes", on page 309
- "WHERE Clause Filters", on page 310
- "Settings", on page 311
- "The Scope of Processing Directives", next

Macros

This section describes these topics:

- "About Macros", next
- "Expression Macros", on page 302
- "Star Macros", on page 303
- "Multiple Declarations of a Given Macro", on page 305
- "Resolving Macro Identifiers", on page 305
- "Overriding Multiple Macro Declarations", on page 305

About Macros

Macros are processing directives that declare constant values that can be used within a SELECT statement. Macro definitions have the following syntax:

```
WITH $<id> AS <value> [OVERRIDE][, $<id> AS <value> [OVERRIDE] [...]]
```

For each occurrence of `$<id>` after the declaration, the query engine replaces it with the constant value. The values of macros remain constant throughout the execution of SELECT statements. Macros are similar to literal constants, not programming variables.

Their benefits of macro directives over literal constants are:

- You declare the constant value in one location, and the value is used wherever the macro identifier is encountered in the remainder of the SELECT statement. To change the value of a literal constant repeated throughout a statement, you must find and change every occurrence.

- Macro identifiers can be much shorter than the values they define. Using the shorter, macro identifier in place of the longer literal expression makes your SELECT statements easier to read and comprehend.

- You can use macros in Perl subroutines so that your subroutines and your SELECT statement can use the same values during execution. For more information, see "Using Macros in Perl Subroutines", on page 410.

Expression Macros

An *expression* macro defines a named expression. The expression can be a column expression, a literal constant, or a complex expression that involves comparisons, functions, and logic operations.

For example, the following SELECT statement returns the number of log entries between 2/1/01 and 2/2/01:

```
WITH $start as time('Feb 01 00:00:00 2002'), $end as _timeadd($start, 1, 'day')

SELECT count(*)
  FROM example_webserv_100
  DURING $start, $end;
```

The preceding and following statements are equivalent, but the statement that uses expressions macros is easier to read.

```
SELECT count(*)
  FROM example_webserv_100
  DURING time('Feb 01 00:00:00 2002'),
     _timeadd( time('Feb 01 00:00:00 2002'), 1, 'day');
```

Expression macros are useful to break up complex expressions into more understandable forms and to enable parameterized queries.

For more information on how to use expression macros as query parameters, see:

- "Parameterizing Your Query", on page 184
- Querying Data in Chapter 3, "Loading, Querying, and Managing the EDW"in the *Administration Guide*.

Automatic Expression Macros

When you use the `atload` command to load source log entries into the EDW, you write a Sensage SQL SELECT statement that describes the columns in the target table that will store the source log data. The command declares some expression macros automatically for use in you SELECT statement.

For more information, see Automatic Expression Macros in Chapter 3, "Loading, Querying, and Managing the EDW" in the *Administration Guide*.

Star Macros

WITH can be used to declare macros that can be used as shortcuts for specifying multiple expressions in the same way that asterisks (`*`) can be used as a shortcut for specifying the all the columns in a table. For example, the following query returns the minimum and maximum values for the timestamp column:

```
WITH $exprs as '*'(min(ts) AS 'min time', max(ts) as 'max time')

SELECT $exprs, count(*)
  FROM example_webserv_100
  DURING ALL;
```

The above query is equivalent to:

```
SELECT min(ts) AS 'min time', max(ts) as 'max time', count(*)
  FROM example_webserv_100
  DURING ALL;
```

Star macros may be used in the target clause, the GROUP BY clause, and the ORDER BY clause. For example:

```
-- isolate the actual group and order criteria from the rest of the query
WITH $groupkey AS '*'(ClientDNS as "hostname" ASC, ClientIP as "ipaddress" DESC)

SELECT $groupkey, count(*)
  FROM example_webserv_100
  GROUP BY $groupkey
  ORDER BY $groupkey
  DURING ALL;
```

During the ordinary macro expansion process, the SQL compiler sees the `$groupkey` macro in the target list, finds the corresponding WITH statement, and replaces the `$groupkey` macro in the target list with the specified targets.

Here are some additional examples:

Example 1

The following query:

```
WITH $columns AS '*'(ts AS 'Time', Host, ClientIP)

SELECT $columns
  FROM test
  DURING ALL;
```

returns the same result as:

```
SELECT ts AS 'Time', Host, ClientIP
  FROM test
  DURING ALL;
```

Example 2

A star macro can contain a general list of expressions and can be used with functions in the obvious way:

```
WITH $args1 AS '*'("%m/%d/%Y %H:", ts)
WITH $args2 AS '*'("%02d", 30*((_int32(ts)%3600)/1800))

SELECT _timef($args1) + _sprintf($args2) AS 'time'
  FROM test
  DURING ALL;
```

Example 3

A star macro may contain other star macros, as long as they are not recursive. Example 1 above could be written as:

```
WITH $host AS Host
WITH $columns1 AS '*'($host, ClientIP)
WITH $columns AS '*'(ts AS 'TIME', $columns1)

SELECT $cols
  FROM test
  DURING ALL;
```

Multiple Declarations of a Given Macro

In general, only one declaration for any given macro should be present in a SELECT statement. Including two declarations of the same macro typically results in a SQL syntax error.

The exceptions to this general rule are as follows:

Exception	Description
Multiple Scopes	Local definitions of the same macro can be made as long as each such definition is in a different scope.
OVERRIDE	For any given macro, there can be up to one definition that includes the OVERRIDE keyword. This definition will override any subsequent definitions for that macro in the scope in which it is declared, and any nested scopes thereof. Note that it is legal to have more than one OVERRIDE definition of the same macro; however, the first OVERRIDE definition takes precedence.

Resolving Macro Identifiers

When the SQL query engine encounters a macro identifier in a SELECT statement, it searches back through the statement for the macro declaration. Conceptually, it is looking for the “closest” declaration of the macro.

Overriding Multiple Macro Declarations

The OVERRIDE keyword is helpful when a macro identifier has multiple declarations within the statement. When the SQL processor searches for the closest declaration, it continues searching if the closest one does not include the OVERRIDE keyword. It continues searching further back for a declaration with the OVERRIDE keyword. If it finds an overriding declaration, the processor uses it. If no overriding declaration is found further back, it uses the closest non-overriding definition.

For example:

```
SET $definition TO NULL
SET $frame TO $current-frame

WHILE $frame NOT NULL DO
  SET $this-def TO FIND($macro IN $frame)
  IF $this-def NOT NULL DO
    IF IS_OVERRIDE($this-def) DO
      SET $definition TO $this-def
      EXIT-WHILE
    ENDIF
    IF $definition NOT NULL DO
      SET $definition TO $this-def
    ENDIF
  SET $frame TO PARENT_OF($frame)
ENDWHILE

IF $definition IS NULL DO
  ERROR( "Undefined Symbol: %s", $macro)
ENDIF
```

By default, the SQL processor checks for redundant WITH statements of the same type (for example, two Perl functions with the same name) and throws an error. This behavior is especially important when building libraries of WITH clauses, to avoid accidentally using the same name twice.

However, it is also helpful to be able to override the default WITH declarations inside of these libraries, subqueries, and PTL files. Thus, any WITH statement may be appended with the word OVERRIDE, and it will override any future definitions with the same name (and type).

Putting it together:

```
include howto-sql-30.inc
WITH $count AS 52

SELECT $count
  FROM example_webserv_100
  DURING ALL;
```

If `howto-sql-30.inc` contained `WITH $count AS count(*)`, then you would get an error, but if it contained `WITH $count AS count(*) OVERRIDE`, the query would return the number of rows in the table.

User-Defined Subroutines

User-defined subroutines are processing directives that let you declare custom Perl functions or aggregates. Once declared, you can use these subroutines within SQL SELECT statements. The declarations of user-defined subroutines have the following syntax:

```
WITH <id> AS [BUILTIN] '<language>' [<return_type>] {FUNCTION|AGGREGATE} <<EOF
  <declaration>
EOF [OVERRIDE]
```

The optional BUILTIN keyword causes the declaration to be treated as if the function is built into the SQL query engine. This allows you to invoke the function directly by name. If you omit the BUILTIN keyword, you must invoke the function indirectly with the `_perl()` or `_perlagg()` SQL functions. For more information on these built-in SQL functions for calling Perl subroutines, see Chapter 12: Perl Subroutines.

The value for `'<language>'` should always be `'perl15'`.

The value of `<return_type>` is any of the supported HawkEye AP SQL data types: `bool`, `float`, `int32`, `int64`, and `varchar`. The data type of the return value is `varchar` by default.

The keywords FUNCTION and AGGREGATE indicate whether `<declaration>` is a Perl function or a Perl aggregate.

The value of `<declaration>` is a Perl script declaration. Generally, the `<declaration>` is enclosed within here document syntax, because declarations span multiple lines and use special characters, including semi-colons. For more information, see “Syntax for Defining Long String Literals”, on page 292.

TIP: You can use macros in Perl subroutines. For more information, see “Using Macros in Perl Subroutines”, on page 410.

Declaring Perl Functions

The following example declares a Perl function called `Link`, which returns an HTML link based on its argument.

```
WITH Link AS BUILTIN 'perl5' FUNCTION <<EOF
  sub Link {
    return "<a href=\"$_[0]\">$_[0]</a>";
  }
EOF

SELECT Link('http://' + ClientDNS)
  FROM example_webserv_100
  DURING ALL;
```

If the value of `ClientDNS` for a row in the table contains the text `’www.acme.com’`, the result set contains a corresponding row with the value:

```
’<a href="http://www.acme.com">www.acme.com</a>’
```

For a more detailed explanation, see “Declaring Perl Functions”, on page 406.

Declaring Perl Aggregates

The following example declares a Perl aggregate called `last()`. Because the declaration does not include the BUILTIN keyword, the SELECT statement invokes `last()` with the built-in `_peragg()` SQL function.

```
WITH last AS 'perl5' AGGREGATE <<EOF
  my %state;
  sub last { $state{$_[1]} = $_[2]; }
  sub last_final { return $state{$_[1]}; }
EOF

SELECT ClientDNS, _perlagg('last', ClientDNS, RespSize)
  FROM example_webserv_100
  GROUP BY 1
  DURING ALL;
```

Within the declaration of Perl aggregates, you define two Perl subroutines and any global variables that the two subroutines can access. The first subroutine has the same name as the Perl aggregate declaration. The second subroutine has the same name with a suffix of `_final`, and it must have `return` statement.

For a more detailed explanation, see “Declaring Perl Aggregates”, on page 407.

Subqueries

Subqueries are processing directives that declare named SELECT phrases for use in your main SELECT statement. The result sets of subqueries can be used as if they were tables in the EDW. Declarations of subqueries have the following syntax:

```
WITH <id> AS (<subquery> [OVERRIDE][, <id> AS (<subquery>) [OVERRIDE][...]]
```

For example, the following SELECT statement queries the result set from the subquery named `timestamps`:

```
WITH timestamps AS (SELECT DISTINCT ts FROM example_webserv_100 DURING ALL)

SELECT *
  FROM timestamps;
```

You can nest subquery declarations, as the next example shows:

```
WITH subq1 AS (WITH subq2 AS (SELECT ts FROM foo DURING ALL) SELECT * FROM subq2)

SELECT *
  FROM subq1;
```

The main SELECT statement queries for its results from `subq1`. The results of `subq1` are selected from another subquery, `subq2`, which is declared within the first subquery.

Subqueries are useful when multiple levels of aggregation are required. For example, the following query returns the number of distinct values of `ClientDNS`:

```
WITH subquery AS ( SELECT ClientDNS
                     FROM example_webserv_100
                     GROUP BY 1
                     DURING time('Feb 01 00:00:00 2002'),
                            time('Mar 31 23:59:59 2002')
                 )

SELECT count(*)
  FROM subquery;
```

IMPORTANT: For performance reasons, avoid subqueries that generate too many rows in their result sets; that is, 100,000 rows returned by a subquery is reasonable, but millions of rows may result in performance problems.

Subqueries with DURING Clauses

The DURING clause is required only in subqueries that select data directly from tables. Subqueries and SELECT statements that specify subqueries in the FROM clause instead of tables do not require DURING clauses unless they are run from HawkEye AP Console.

IMPORTANT: When you create a subquery that selects data directly from tables, ensure that the DURING clause behaves as you expect. A badly formed DURING clause can cause the query to return incorrect or no results. For more information, see “Subqueries and Views and the DURING Clause”, on page 276

Subqueries and UNION ALL Clauses

Subqueries can include UNION ALL clauses. For example:

```
WITH subq AS ( ... UNION ALL ... )

SELECT *
  FROM subq;
```

When the query engine executes the preceding SELECT statement, it executes the nested subqueries and produces a union of their result sets. The query engine then executes the main SELECT statement against the intermediate result set produced by the subquery `subq`.

Creating Unions of Subqueries

You can create a union of subqueries in the main SELECT statement. For example:

```
WITH subq1 AS ( ... )
WITH subq2 AS ( ... )

SELECT *
  FROM subq1

UNION ALL

SELECT *
  FROM subq2;
```

Arbitrary nesting of subqueries and/or UNION ALLs is allowed.

Achieving Distinct Union Results

With subqueries, you can overcome the limitation that UNION DISTINCT is not supported. Perform the UNION ALL operations in a subquery declaration. Then, use DISTINCT in the main SELECT statement. For example:

```
WITH subquery AS (SELECT ClientDNS, RespSize
                    FROM example_webserv_100
                    DURIING ALL

                  UNION ALL

                  SELECT DISTINCT ClientDNS, RespSize
                    FROM example_webserv2_100
                    DURIING ALL
                )

SELECT DISTINCT * FROM subquery;
```

Subqueries and the WHERE clause

You can use the `IN` or `NOT IN` operators inside of a `WHERE` clause to limit results to an explict list of values. These operators can also use subqueries to create the list of results. See “Using Subqueries with the IN Operator”, on page 295.

Table-Name Substitutes

Table-name substitutes are processing directives that declare table names which substitute for table identifiers in subqueries and in SELECT statements. Declarations of table-name substitutes have the following syntax:

```
WITH <id> AS TABLE <table> [OVERRIDE][, <id> AS TABLE <table> [OVERRIDE][...]]
```

The value of *<id>* is a table identifier that occurs elsewhere within subqueries or the SELECT statement. The value of *<table>* is the table name that should substitute for *<id>* when the query engine executes the compiled subqueries and SELECT statement.

For example:

```
WITH websrv AS TABLE example_webserv_100

SELECT count(*)
  FROM websrv
  DURING ALL;
```

NOTE: You can achieve the same effect with the `--tableswap` option of the `atload` command. For more information, see the topic Querying Data in Chapter 3, “Loading, Querying, and Managing the EDW” in the *Administration Guide*.

WHERE Clause Filters

WHERE clause filters are processing directives that declare a conditional expression to be added to WHERE clauses for a specified table. Declarations of WHERE clause filters have the following syntax:

```
WITH WHERE <table_id> AS (<conditional-expression>)
```

The value of *<table_id>* is a table identifier that occurs in FROM clauses within subqueries or the SELECT statement. The value of *<conditional-expression>* is a conditional expression that references columns in the identified table. When the query engine executes the SQL statement, it combines the WHERE clause filter and the explicit conditional expression of the WHERE clause. It uses the AND logical operator as the conjunction. If there is no explicit WHERE clause, the WHERE clause filter alone is applied.

For example, the following SELECT statement declares a WHERE clause filter to ensure that only rows from the `example_webserv_100` table that have `’HTTP/1.0’` in their `HttpVers` columns are included in the results set, regardless of the conditions in the explicit WHERE clause.

```
WITH WHERE example_webserv_100 AS (HttpVers = 'HTTP/1.0')

SELECT count(*)
  FROM example_webserv_100
  WHERE ReapSize > 100
  DURING ALL;
```

The preceding example returns the same results as if the WHERE clause had been written as follows:

```
WHERE (HttpVers = 'HTTP/1.0') AND (ReapSize > 100)
```

You can include multiple WITH WHERE declarations for the same table, and the effect is cumulative. Each *<conditional-expression>* is combined the others using the AND logical operator as the conjunction.

TIP: If the *<conditional-expression>* is long and needs to span multiple lines, use here document syntax. For more information, see “Syntax for Defining Long String Literals”, on page 292.

Settings

Settings are processing directives that control the SQL query engine and configure the context in which SQL statements execute. Declarations of settings have the following syntax:

```
WITH <setting> <value> [OVERRIDE]
```

The following settings can be controlled through WITH clauses.

Setting	Value	Meaning
TIMEZONE	GMT	Sets a default time zone for manipulating timestamps
NOWTIMESTAMP	The current time	The timestamp returned by _time('now')
MINTIMESTAMP	1970-01-01T00:00:00	The timestamp returned by _time(<time_specification>)
MAXTIMESTAMP	2038-01-19T03:14:07	The timestamp returned by _time(<time_specification>)
GROUPCACHEPARTITIONS		Controls behavior of the SQL Aggregation engine
GROUPCACHEPARTITIONING		Controls behavior of the SQL Aggregation engine
AGGCACHEPARTITIONS		Controls behavior of the SQL Aggregation engine
AGGCACHEPARTITIONING		Controls behavior of the SQL Aggregation engine
FIFOCACHEPARTITIONS		Controls behavior of the SQL Aggregation engine
FIFOCACHEPARTITIONING		Controls behavior of the SQL Aggregation engine
DIVIDEBYZERO		Controls behavior when a divide-by-zero condition occurs
MODULOBYZERO		Controls behavior when a modulo-by-zero condition occurs
SUPRESSECEPTIONS		Controls whether error conditions stop query execution
ALLOWEXITPERL	false	Controls whether Perl subroutines can exit
DGBVARS		Supports debugging
NOPROGRESS	false	Suppresses the generation of progress indicators

You can declare the same setting multiple times. The earliest declaration takes effect and overrides any later declarations of the same setting.

The TIMEZONE Setting

One of the most common settings is TIMEZONE. It changes how the SQL query engine interprets timestamps when the time zone is unspecified. Many source log entries do not include time-zone indicators, and some formats for timestamps passed to the `time()` function cannot specify time zones. The setting has the following format:

```
WITH TIMEZONE '<time_zone>'
```

The allowed values for <*time_zone*> are listed in "Appendix B: Time Zones" in the *Administration Guide*.

The following SELECT statement declares the TIMEZONE setting to be Pacific Time. The varchar arguments to the two `time()` expressions in the DURING clause do not specify their time zones. The TIMEZONE setting tells the query engine to interpret their time zones as Pacific Time. Without the setting, the query engine assumes their time zones are GMT.

```
WITH TIMEZONE "PST8PDT"

SELECT count(*)
  FROM example_webserv_100
  DURING time('Feb 01 00:00:00 2002'), time('Mar 31 23:59:59 2002')
```

Specifying time zones is important to ensure that queries return the intended results, especially when the source of the data generates times in a different time zone than yours.

The Scope of Processing Directives

Processing directives are classified by one of two types of scope:

- Global State Modifiers
- Local Definitions

Global State Modifiers

Directives that are *global state modifiers* affect the entire SELECT statement, including subqueries. You can declare them only in the top level of a SELECT statement. Syntax errors occur when you include global state modifiers as parts of subquery declarations.

These directives have the scope of global state modifiers.

Directive Syntax	For more information, see...

WITH <*id*> AS [BUILTIN] '<*language*>'
  [<*return_type*>] {FUNCTION|AGGREGATE} <<EOF
  <*declaration*>
EOF [OVERRIDE]
	"User-Defined Subroutines", on page 306

WITH <*id*> AS TABLE <*table*> [OVERRIDE]	"Table-Name Substitutes", on page 309
WITH WHERE <*table_id*> AS (<*conditional-expression*>	"WHERE Clause Filters", on page 310
WITH <*setting*> <*value*> [OVERRIDE]	"Settings", on page 311

Local Definitions

Directives that are *local definitions* affect only the parts of SELECT statement in which they are declared. You can declare them anywhere within a SELECT statement, including subqueries.

These directives have the scope of local definitions.

Directive Syntax	For more information, see...
`WITH $<id> AS <value> [OVERRIDE]`	"Macros", on page 302
`WITH <id> AS (<subquery>){OVERRIDE}`	"Subqueries", on page 307

WORKING WITH LISTS

Event-log data often has fields that contain lists of data. HawkEye AP SQL provides facilities for working with lists.

This section describes these topics:

- "Multiple Values as Lists", next
- "Functions that Return Lists", on page 313
- "List Acceptors", on page 314
- "INTO Keyword", on page 315
- "List Expressions and FROM Clauses", on page 315
- "EXPLODE Keyword", on page 316
- "EXPLODE BY Keyword Phrase", on page 316
- "Some Helpful List Examples", on page 317

Multiple Values as Lists

Expressions and functions normally evaluate to a single value during query processing. However, there are times when you want them to evaluate to a list of values. When an expression or function evaluates to a list, you indicate whether you want to access the list or the single return value.

The syntax options for indicating your intent are:

- **<*expression*>**

  Access the single return value and discard the list.

- **<*expression*> INTO <*varname*>**

  Access the single return value and save the list in `<varname>` for later use

- **@<*expression*>**

  Access the list and discard the single return value

- **@<*expression*> INTO <*varname*>**

  Access the list and saves save the list in `<varname>` for later use. This form is rarely used, because `<varname>[0]` contains the normal single value; that is, it is not lost.

Functions that Return Lists

These functions can return lists instead of single return values when you precede them with the list symbol (`@`). Without the symbol, these functions return a single value.

- `_strmatchlist()`

  If your regular expression contains matching sets of parentheses, the matching text between them becomes a return value. The normal return value is the number of elements matched.

- `_strsplit()` and `_strsplitxsv()`

  When parsing strings into multiple elements, the results are returned as a list. The normal return value is the number of elements parsed.

- `_perl()`

  Perl functions can call `Addamark::setInto()` to return multiple values. See the "List Support and Perl Functions", on page 409, for more information.

- `_into()`

  The `_into()` function takes its arguments and stores them in a list. Specifically, `_into(*)` can be used to convert the list of table-columns (or subquery-targets) into a list.

- `_lookup()`

  The multiple columns of each row are returned as a list. The first column is also copied into the normal result value, which makes `_lookup()` easier to use when there is only one result column.

List Acceptors

The following types of expressions can consume lists. The @ symbol forces the SQL query engine to access the list, rather than the regular return value from the expression.

- **<*listgen*> INTO <*varname*>**

  This clause can be appended after a list generator expression to give the list a name, such that the list elements can be used in other expressions. For more details, "INTO Keyword", next.

- **EXPLODE @<*listgen*>**

  Transposes the list into multiple records. For more details, "EXPLODE Keyword", on page 316, and "EXPLODE BY Keyword Phrase", on page 316.

- **<*expression*> IN @<*listgen*>**

  Matches the `<expression>` against the elements of `<listgen>` and returns Boolean "true" if there is a match.

- **_perl()**

  Perl functions can be called with lists as their arguments, that is, `_perl("myfun", arg1, arg2, @<varname>, arg3)`. Perl programmers will recognize similarities with lists in Perl. For more information, see Chapter 12: Perl Subroutines.

- **_strcat() and _strjoin()**

  You can concatenate the values of a list together.

INTO Keyword

The INTO keyword accepts multiple return values and puts them in a named a list variable. Use of the INTO keyword has the following syntax:

```
<expression> INTO <listname>
```

The `<listname>` element of the syntax is called the *INTO* variable. You can use INTO variables elsewhere in the SQL statement. Use array index notation to access individual items in the list. Actual list items begin at `<listname>[1]`. For convenience, the item at `<listname>[0]` has a copy of the normal expression result.

The following example uses INTO to capture multiple matches from `_strmatchlist()`:

```
SELECT match[1], match[2]
  FROM example_webserv_100
  WHERE _strmatchlist(UserAgent, '([^(]*)\\(([^)]+)') INTO match
  DURING ALL;
```

In this query, the expressions `match[1]` and `match[2]` represent the additional values returned by the query.

Generally, INTO variables can be used anywhere in the SQL statement like any other expression or column identifier. For example, INTO variables can be passed into functions and used in mathematical or string expressions. However, causality must be preserved and loops are not allowed. For example:

```
-- this is illegal because it creates an INTO-loop
SELECT _strmatchlist(foo, 'x=(..),y=(..)') INTO bar,
       _strmatchlist(bar, 'x=(..),y=(..)') INTO foo,
  FROM Table
  DURING ALL;
```

The INTO keyword is optional. Use it only when you want to capture the list that an expression or function can return. You cannot use the INTO directive with an expression that produces only single return values. For example the following is invalid:

```
4+5 INTO sum -- incorrect use of INTO with a single-value expression
```

The expression `4+5` only evaluates to a single value, never to a list.

List Expressions and FROM Clauses

List expressions may be used in the FROM clause to specify an implicit union of the specified tables. For example, the following query returns the distinct `ClientIP` values from both the `example_webserv_100` and `example_webserv2_100` tables:

```
 SELECT DISTINCT ClientIP
   FROM @_list('example_webserv_100', 'example_webserv2_100')
   DURING all;
```

Whenever the SQL query engine sees a FROM clause with a list expression, the engine verifies that all the tables in the list contain all the columns selected, and that all the types of all the columns are the same.

IMPORTANT: The DISTINCT keyword causes each query to produce unique intermediate results, but the final result set does not eliminate duplicates found in the separate tables.

EXPLODE Keyword

The EXPLODE keyword lets you transpose a list of values into a target column. For example, you may need to pass list elements into aggregates. The keyword has the following syntax:

```
EXPLODE <list_expression> [AS <target>]
```

The SQL query engine iterates through *<list_expression>*, replicating the input row in the output result set for each list element. The current list element is substituted in each iteration.

For example:

```
-- returns three records: "fred 12000", "mary 12000" and "bob 12000"
WITH subquery as (
    SELECT TOP 1 ts
      FROM example_webserv_100
      DURING ALL
)

SELECT EXPLODE @_strsplit(",", "fred,mary,bob") AS name, 12000 AS salary
  FROM subquery -- only one record
;
```

The expression that follows the EXPLODE keyword must evaluate to a list variable.

IMPORTANT: You may use only one EXPLODE keyword per SELECT phrase. A statement can have two or more EXPLODE keyword only if each one exists in a separate subquery.

EXPLODE BY Keyword Phrase

The EXPLODE BY keyword phrase lets you transpose a list embedded in a `varchar` expression. The phrase has the following syntax:

```
EXPLODE <varchar_expression> BY '<separator>'
```

The value of *<varchar_expression>* must contain a list of items separated by the special character specified by *<separator>*. The expression to explode can be a single-byte or multi-byte string

For example, the expression in the statement below uses a comma (`,`) to separate the values in list:

```
WITH subquery as (
    SELECT TOP 1 ts
      FROM example_webserv_100
      DURING ALL
)

SELECT EXPLODE "fred,mary,bob" BY ',' ,12000 AS salary
  FROM subq -- only one record
```

```
;
```

A common use of EXPLODE BY is to compute histograms, as the next example shows:

```
WITH subquery as (
    SELECT EXPLODE Url BY '/' AS component
      FROM example_webserv_100
      DURING ALL
)

SELECT component, count(*)
  FROM subquery
  GROUP BY 1
  ORDER BY 2 DESC
;
```

Some Helpful List Examples

The following examples are often helpful in understanding lists.

```
_strjoin(",", @_strsplit(",", "a,b,c")) == "a,b,c"
_strjoin(",", _strsplit(",", "a,b,c")) == _strsplit(",", "a,b,c") == "3" (number
of elements parsed)
_if(_strsplit(",", "a,b,c") INTO v > 0, v[2], "") == "b"

-- should return 12, which is four records each exploded three times
WITH subq as (select top 4 ts from example_webserv_100 during all)
SELECT COUNT(EXPLODE @_strsplit(",", "a,b,c"))
  FROM subq
```

CHAPTER 11

SQL Functions

Sensage SQL provides these functions for use in expressions in Sensage SQL Select statements:

- “Conditional Evaluation Functions”, next
- “List Functions”, on page 321
- “Lookup Functions”, on page 323
- “Aggregation Functions”, on page 337
- “Statistical Aggregate Functions”, on page 344
- “Logarithmic and Exponential Functions”, on page 348
- “Numeric Rounding Functions”, on page 351
- “String Functions”, on page 353
- “Time Functions”, on page 374
- “Network Address Functions”, on page 387
- “Miscellaneous Functions”, on page 394

NOTE: For type-conversion functions, see “Conversion Expressions”, on page 299.

CONDITIONAL EVALUATION FUNCTIONS

This section describes these functions that perform conditional evaluations.

Func- tion	Purpose	Page
_if()	Evaluate a Boolean expression and return different values for true and false	page 31 9
_iftable()	Return different expressions depending on the table being queried	page 32 0

_if()

The _if() function evaluates a Boolean expression and returns different values for true or false.

Synopsis

```
_if( <condition>, <expr1>, <expr2> )
```

Description

If *<condition>* evaluates to `true`, the `_if()` function returns *<expr1>*. Otherwise, it returns *<expr2>*. Both expressions must evaluate to the same data type. You can use `_if()` recursively, as either *<expr1>* or *<expr2>*.

The following example uses `_if()` recursively as the second expression argument:

```
_if( <condition1>, <expr1>, _if( <condition2>, <expr2>, <expr3> ) )
```

The function evaluates *<condition1>* and returns *<expr1>* if it is `true`. Otherwise, the inner `_if()` function is invoked, and the outer `_if()` returns the result of evaluating *<condition2>*. If *<condition2>* is `true`, the outer `_if()` returns *<expr2>*; otherwise, it returns *<expr3>*. You can achieve further recursion if you specify `_if()` for *<expr2>* or *<expr3>*.

Expression arguments are evaluated only when their corresponding conditions are found to hold. For example, in simple, non-recursive uses of the function, *<expr1>* is evaluated only when *<condition>* is `true`. This is called *short-circuit boolean evaluation*. Short-circuit evaluation avoids unnecessary processing and the unwanted side effects of expression evaluation, such as evaluating Perl functions that modify global Perl variables.

Arguments

Argument	Description
<condition>	A `bool`, `int32`, or `float` expression
<expr1>	The value returned when *<condition>* evaluates to false or 0.
<expr2>	The value returned when *<condition>* evaluates to true or a non-zero value.

Return Value

The return data type of the `_if()` function is the data type of the last argument.

Exceptions

The `_if()` function raises SQL processing exceptions under any of these conditions:

- The condition arguments do not evaluate to a data type of `bool`, `int32`, or `float`.
- The data types of expression arguments do not match the data type of the last argument.

Example

The following query returns the IP addresses for clients which visited the `'/robots.txt'` URL more than three times.

```
SELECT ClientDNS
  FROM example_webserv_100
  GROUP BY 1
  HAVING sum( _if( Url='/robots.txt',1,0 ) ) > 3
  DURING ALL;
```

_iftable()

The `_iftable()` functions returns different expressions depending on the table being queried.

Synopsis

```
_iftable( <type>, <name1>, <expr1>, [<name2>, <expr2>[...], <else_expr> )
```

Description

If the table being scanned is *<name1>*, the `_iftable()` function returns *<expr1>;* otherwise, it tries each successive name. When a match is found, the function returns the corresponding expression. If no match is found, the function returns *<else_expr>*.

Expression arguments to `_iftable()` are evaluated only when the corresponding condition is true. This is called short-circuit Boolean evaluation. Short-circuit evaluation avoids unnecessary work and provides correct behavior for expressions that have side effects, such as Perl functions that modify global Perl variables.

Arguments

Argument	Description
<type>	The data type of the return value
<name>	Expressions that evaluate to table names
<expr>	Expressions returned when the table being scanned matches a specified name
<else_expr>	The expression returned when the table being scanned matches none of the names

Return values

The return type of the `_iftable()` expression is the data type specified by the first argument.

Exceptions

`The _iftable()` raises an SQL processing exception under any of these conditions:

- The data types of *<name>* arguments are not varchar.
- The data types of *<expr>* arguments do not match the data type specified in the *<type>* argument.

List Functions

This section describes functions that perform operations on lists of expressions or values.

Function	Purpose	Page
_list()	Create a list variable from a list of values	page 321
_nth()	Return a value from a list variable	page 322

_list()

The `_list()` function creates a list variable from a list of values.

Synopsis

```
[@]_list( <expression>[, <expression>[...]] )
```

Description

Evaluates each *<expression>* and makes it available as the corresponding element of the list variable. The list variable returned by the function is used with the `EXPLODE` keyword and the `IN` comparison operator.

For for more information on the use of the list variables the _list() function returns, see “EXPLODE Keyword”, on page 316 and “Advanced Comparison Operators”, on page 295.

Arguments

Argument	Description
<expression>	Expressions that evaluate to scalar values

Return Value

`_list()` returns the first element in the list.

`@_list()` returns the list variable.

List variables are sometimes called INTO variables, because the INTO keyword also creates list variables.

For more information, “INTO Keyword”, on page 315.

Example

The following query filters on a comma-separated list of items.

```
SELECT ClientDNS
  FROM example_webserv_100
  WHERE _strmatch(UserAgent,'([^ ]+)','') IN
              @_list("Mozilla/4.0", "Mozilla/4.08", "Mozilla/4.5")
  DURING ALL;
```

_nth()

The `_nth()` function returns a value from a list variable.

Synopsis

```
_nth( <type>, <index>, <list_expr> [, <expr>[...]] )
```

Description

Returns the value of the specified item in a list of expressions or values, converting it to the specified *<type>*, if necessary.

For related information, see:

- “INTO Keyword”, on page 315
- “_list()”, on page 321
- “_strmatch()”, on page 358
- “_strsplit()”, on page 360
- “_strcat()”, on page 371

Arguments

Argument	Description
<type>	The data type of the return value
<index>	An `int32` expression that evaluates to an index within *<list_expr>*
<list_expr>	A list expression

Return Value

The `_nth()` function returns the value from *<list_expr>* that corresponds to the specified *<index>*. If necessary, the value is converted to the data type specified `by` *<type>*. The first value in *<list_expr>* has 0 as its index. If the value of *<index>* is negative, it is considered to be 0. If the value of *<index>* exceeds the number of values in the list, the last expression is returned. If any @ list expressions are present in the list of expressions, their list values will be expanded prior to selection.

Exceptions

The `_nth()` function raises a SQL processing exception if the value selected by the index cannot be converted to the specified data type.

Example

The following query splits the values into a list and returns the value of the fifth item in the list. If `_nth()` has fewer than five items defined as values, the last item will be used.

```
SELECT _nth(int32, 4, @_strsplit("/", Referrer))
  FROM example_webserv_100
  DURING ALL;
```

Lookup Functions

This section describes these functions that look up and return values based on values passed as arguments.

Function	Purpose	Page
_lookup()	Look up values stored in external data sources	page 324
_rev_dns()	Looks up the DNS host name for a specified IP address	page 334
_tablematch()	Returns a list of table and view names that match a pattern	page 334

_lookup()

The `_lookup()` function uses values stored in an EDW table to retrieve and return values stored in flat files. This function enables you to provide consistent and meaningful values for data that is stored inconsistently. You can also use this function to correlate data between tables.

NOTE: The character encoding of the lookup file must be UTF-8.

Synopsis

```
_lookup( <filename>, <key>, <default_key_value_pair>[, <option_string> )
```

Description

The *<key>* argument identifies a column in an EDW table or view. The `_lookup()` function uses the value of *<key>* in each table row to search for a corresponding value in the lookup file identified by *<filename>*. If the function finds the *<key>*, it returns the corresponding value from the field in the lookup file. If it finds no match, the function returns *<default_key_value_pair>*, which provides a default value for a specific key.

Use *<option_string>* to identify the formatting of the lookup file. Also use *<option_string>* to specify which field(s) in the lookup file contain values to return to your query. These arguments are optional only because the function provides default values if you do not specify a value. For information about the default values, see "Basic Lookup Options", on page 327.

You can also use *<option_string>* to tailor the operation of the `_lookup()` function. For example, you can specify how to parse the lookup value associated with *<key>* to extract values that are returned in a list instead of as a single value. For an example of such a query, see "Example #3: Accepting Multiple Return Values", on page 330.

The data in the lookup file is stored in character-separated fields. The number of fields in a row can vary from row to row.

Arguments

Argument	Description
<filename>	The fully qualified filename of the lookup file. NOTE: • If the value of *<filename>* begins without a slash (/), the lookup file and its path are relative to the `dsroot` directory. • The lookup file must be located in the same path on all EDW hosts. For more information, see "Working with Lookup Files", on page 331.
<key>	The column in the EDW table whose value is used to look up a value in the lookup file.

Argument	Description
<default_key_value_pair>	If the lookup fails, the function uses the default key and textual equivalent to form the return value. This argument must: • Evaluate to a constant; that is, it cannot vary by row. • Provide a default key value and a corresponding text value for each field represented in the option string. For example, if the option string returns values for columns 2 and 4, *<default_key_value_pair>* must provide a default value the key as well as for four fields. The function parses the four fields to return the default value for columns 2 and 4. For an example, see "Example #3: Accepting Multiple Return Values", on page 330.
<option_string>	Describes the format of the lookup file, identifies the field in the lookup file to use as the key, and specifies which field(s) to return. NOTE: • The *<option_string>* argument must evaluate to a constant; that is, it cannot vary by row. • This argument is optional because default values are provided for its two critical options. For more information, "Basic Lookup Options", on page 327. For more information, see "Specifying Options in the Option-String Argument", next.

Specifying Options in the Option-String Argument

The value of the *<option_string>* is a list of options separated by semi-colons (`;`). The general syntax is:

```
<option_name>=<setting>[;<option_name>=<setting>[...]]
```

The options are classified as follows:

- "Basic Parsing Options", next
- "Basic Lookup Options", on page 327

TIP: Some `_lookup()` function options are required or very useful for basic queries. Others are needed only to perform advanced lookups. To facilitate usage of this function, this section provides tips that separate the basic from the advanced options.

- Basic options include:
  - parsing options—the field-separator character in the lookup file. If the file uses whitespace as the separator, you do not need to specify this basic parsing option.
  - lookup options—the field in the lookup file that serves as the key and its data type, and the fields that contain the values to return and their data types
- Advanced options include:
  - parsing options—the quotation-mark character in the lookup file; the function ignores separator characters within the quoted string and strips the quotation marks. You can also specify a non-default escape character that keeps the function from escaping special characters not enclosed within the specified quoting character.
  - lookup options—the character in the lookup file that identifies a comment, which causes the function to ignore text that follows the comment character.

Basic Parsing Options

The basic options to control parsing of the strings in the lookup file are `fieldsep` and `escape`.

- **`fieldsep`**

  `fieldsep=<`*`style`*`>[,<`*`style`*`>[...]]`

  For the style, specify a comma-separated list of field separators used in the lookup file. The default style is `"whitespace"`, which allows multiple whitespace characters between fields. You can specify `<`*`style`*`>` in any of the following ways:

  - `{comma, csv, tab, tsv, pipe, psv, colon, semi, semicolon}`—interprets the separator as any single instance of any of the included characters; this style is the equivalent of `str:<`*`char`*`>`.
  - `str:<`*`string`*`>`—sets a literal string as the separator; for example, `fieldsep=str:foo` uses `foo` as the separator.
  - `char:<`*`ascii_num`*`>`—specifies the ASCII number of the character to use as the separator; for example, `fieldsep=char:13` specifies a new-line character as the separator.
  - `{ws|whitespace}`—sets any number of contiguous whitespace characters as the separator; in other words, allows multiple whitespace characters between fields.

- **`escape`**

  `escape={backslash|none}`

  - `backslash`—treats `\<`*`char`*`>` as `<`*`char`*`>`

    This setting specifies treatment of the backslash character and field separators as well as ordinary characters. For example, if your escape character is a backslash and your data includes two consecutive backslashes (`\\`), a backslash is treated as part of the data. The escape character defaults to `backslash`

    TIP: Basic usage of the `lookup()` function does not require changing the default value for the **`escape`** option. If you change the value to `none`, you must also set the **`quoting`** option.

  - `none`—special characters cannot be escaped to be included in data values unless the data values are enclosed with the quoting character.

Advanced Parsing Option

The function also provides the `quoting` parsing options, which allow you to specify the style of the quotation mark character. Its syntax is:

`quoting=<`*`style`*`>[,<`*`style`*`>[...]]`

Comma-separated list of quoting options allowed, defaults to “`singleordouble`”.

- `{single|single-quote}`—if single-quotes are found around a string field, then ignore separator characters in that string and also strip the quotes.
- `{double|double-quote}`—same as single-quote, but for double-quotes
- `char:<`*`ascii_num`*`>`—specifies the ASCII number of the character to use as the quotation symbol; for example, `quoting=char:37` sets the percent (`%`) character as the quoting character

Advanced Usage: Rule of Thumb When Specifying Parsing

If the data in your lookup file contains field separators, use quoting options to isolate the data in specific fields. If the data in your lookup file contains quotation marks, use escape characters to isolate the data in specific fields.

For example, assume your escape character is `backslash` and your field separator is a comma. Assume also that the lookup data associated with a <*key*> has the following fields, one of which uses a comma (`,`) to separate an apartment number (`#416`) into a different field from street address (`9617 Delaware St.`):

```
234, Vijay Kumar, 9617 Delaware St., #416, Berkeley, CA, 94705
567, Jose Sanchez, 123 Main St., Oakland, CA, 94612
```

To force the apartment number into the same field as street address, you can use `escape=backslash` and format the data row as follows:

```
234, Vijay Kumar, 9617 Delaware St.\, #416, Berkeley, CA, 94705
```

Alternately, you could use `quoting=single` and format the data row as follows:

```
234, Vijay Kumar, '9617 Delaware St., #416', Berkeley, CA, 94705
```

NOTE: If commas, single quotes, and double quotes are part of your data, you can use `fieldsep=char:`<*n*>, where <*n*> is the ASCII number of a character that is not in your data. For example, use `fieldsep=char:037` to specify a percent symbol (`%`) as the separator and format the data row as follows:

```
234% Vijay Kumar% 9617 Delaware St., #416% Berkeley% CA% 94705
```

NOTE: The `_strsplitxsv()` function uses these same parsing options. To learn more about the function, see "_strsplitxsv()", on page 361.

Basic Lookup Options

Use the following options to control how to perform the lookup.

- **`keycol`**

  `keycol=<`*`n`*`>[:<`*`type`*`>]`

  This option identifies the field in the lookup file to use as the key when performing the lookup. Indexing begins at 1. You can append the field number with an optional type name, which forces the EDW engine to parse this column as the specified data type. The following example specifies that the key is the third field in the lookup file and the value of this field should be treated as a 32-bit integer:

  ```
  keycol=3:int32
  ```

  Allowable types are `int32`, `int64`, `float`, `timestamp`, and `varchar`. The default for the `keycol` option is `1:varchar`. The default for <*type*> is `varchar`.

- **`valcols`**

  `valcols=<`*`column_spec`*`>[,<`*`column_spec`*`>[...]]`

This option specifies which fields to return when performing lookups. The format for `<column_spec>` is:

```
<n>[:<type>]
```

Indexing begins at 1. You can append each column number `<n>` with an optional specification of its data `type`. The default for the `valcols` option is `2:varchar`. The default for `<type>` is `varchar`.

Advanced Lookup Option

The function also provides the `commentprefix` lookup option, which allows you to configure the function to ignore comments in the lookup file. Its syntax is:

```
commentprefix=<style>
```

Typically, the value of `<style>` is the string that begins a comment line. Examples include `#` (shell-style or perl-style), `--` (SQL-style), or `//` (C++-style).

If the value of `<style>` starts with the string `inline:`, the function locates comments on lines with actual values. For example, if you specify `inline:#`, a lookup file can contain lines like the following:

```
# normal comment
# another normal comment -- whitespace is allowed.
123,col2val,col3val    # this is an inline comment (including the whitespace)
124,col2val # this is a comment, col3val #<-- oops, col3val is ignored
```

NOTE:

- In the example above, the value in the third column (**col3val**) is ignored because it is included in the inline comment.
- The default for `<style>` is `""`; that is, no comments are allowed in the lookup file.

Examples

This section provides three examples of `_lookup()` function usage:

- "Example #1: Providing Meaningful and Consistent Values", next
- "Example #2: Correlating Data Between Tables", on page 330
- "Example #3: Accepting Multiple Return Values", on page 330

Example #1: Providing Meaningful and Consistent Values

A common usage for the `_lookup()` function is to provide consistent and meaningful values for data that is stored inconsistently. For example, data collected from multiple sources often represents "success" and "failure" in different ways. Typically these values are stored as integers to reduce disk space. You can create a lookup file that stores corresponding text values, and use `_lookup()` to convert integer values stored in the EDW to text values that are meaningful to your users.

HawkEye AP IntelliSchema presents examples of such lookup-file usage. IntelliSchema is a pre-defined data structure that uses SQL views to present unified access to data from disparate systems. Analytics reports generally query IntelliSchema views to display normalized event data for common event types from multiple information systems.

For example, one set of IntelliSchema views returns user-login information. Each view creates a **result** column to store integer values that represent the success or failure of the login. Each view uses the `userLogin.lookup` file to provide meaningful textual equivalents to the stored integer values. The lookup file contains all possible login values and their textual equivalent. Three lines are:

```
-1,Unknown
0,Failure
1,Success
```

To return meaningful values about the success of the login, the user-login IntelliSchema views include the following statement in the target list of the SELECT statement:

```
_lookup("<path>/userLogin.lookup",result,"-
1,Unknown","fieldsep=comma;keycol=1:int32;valcols=2:varchar") AS result,
```

Given that the general syntax is:

```
_lookup( <filename>, <key>, <default_key_value_pair>[, <option_string> )
```

The statement above uses the `_lookup()` function to:

- Locate the lookup file: `"<path>/userLogin.lookup"`
- Specify which column in the table represents the key: `result`
- Specify the default key/value pair: `"-1,Unknown"`
- Specify which field in the lookup file contains a value that should match a value in the key column: `keycol=1:int32`
- Specify which field in the lookup file to return as the value of the key column: `valcols=2:varchar`

When this statement is run against an EDW table, the EDW engine steps through the table and retrieves the value of the **result** column for each row. It compares the value to one of the values in the first field of the lookup file. If it finds a match, it displays the value of the second field as the output of the **result** column.

Because the default value is `"-1,Unknown"`, if no value in the **result** column matches a value in the first field of the lookup file or the value of the **result** column evaluates to `-1`, the query output displays `Unknown` as the value of the **result** column.

NOTE: The default value is specified as two fields to match the two fields specified in the option string:

- the value of the key column as stored in the EDW table
- the corresponding lookup-file value that should be returned to represent the value of the key column.

The IntelliSchema login views contain the **privilege** column as well as the **result** column. The **privilege** column stores integer values that represent the login type: either standard or privileged. Because the values stored in this column are different from those in the **result** column, the `userLogin.lookup` file contains two additional lines. The full lookup file contains:

```
-1,Unknown
```

```
0,Failure
1,Success
20,Privileged
21,Standard
```

As illustrated above, the same lookup file enables lookup for two different columns. The user-login IntelliSchema views include both of the following statements in the target list of the SELECT statement:

```
_lookup("<path>/userLogin.lookup",result,"-
1,Unknown","fieldsep=comma;keycol=1:int32;valcols=2:varchar") AS result,
_lookup("<path>/userLogin.lookup",privileged,"-
1,Unknown","fieldsep=comma;keycol=1:int32;valcols=2:varchar") AS privileged
```

The second statement behaves like the first but, when this statement is run against an EDW table, the EDW engine steps through each row of the table and retrieves the value of the **privilege** column.

Example #2: Correlating Data Between Tables

Data often derives from disparate sources and applications. You can use key-field information to link the data across EDW tables and lookup files.

For example, assume your EDW instance uses separate tables to store data about sales staff and customers. The **sales** table stores the sales person's name, territory, and phone number. The **customer** tables stores company name, street address, city, state, and country. Because territories change frequently, assume also that your site uses an application to dump territory information regularly into a flat file. The flat file contains each territory and the states they contain.

To report all customers assigned to each salesperson, you would create a query that:

- Retrieves each customer's state from the **customer** table
- Matches each state to a territory by running `_lookup()` against the territory file
- For each territory, locates the salesperson from the **sales** table

Example #3: Accepting Multiple Return Values

Whereas Example #1: Providing Meaningful and Consistent Values above illustrates usage of the `_lookup()` function in the `SELECT` statement, the example below illustrates usage of the function in the `WHERE` clause. It also illustrates use of the `INTO` keyword to accept multiple return values and to put them in a named list variable.

The lookup data for this example is similar to that used in "Advanced Usage: Rule of Thumb When Specifying Parsing", on page 327:

```
234, Vijay Kumar, 9617 Delaware St.\, #416, Berkeley, CA, 94705
567, Jose Sanchez, 123 Main St., Oakland, CA, 94612, 415-644-9753
```

NOTE:

- The first row above uses the backslash (`\`) character to escape the comma (`,`) . If the comma were not escaped, it would separate an apartment number (`#416`) into a different field from street address.

To keep the apartment number in the same field as the street address, you must specify the backslash as the escape character in your `_lookup()` call.

- The second row above contains a phone number in addition to the standard information.

  The `_lookup()` function does not require every row in the lookup file to contain the same number of fields. However the fields you return must be in the same position in every row of the file.

The query below specifies settings for the `fieldsep`, `escape`, `keycol`, and `valcols` options:

```
SELECT
   UserId,
   userinfo[1] as UserName,
   userinfo[2] as State
FROM
   sometable
WHERE
   _lookup("/tmp/users", UserId, "-1,anonymous,-,-,-",
   "fieldsep=comma;keycol=1:int32;valcols=2,5;escape=backslash")
   INTO userinfo != -1
DURING ALL;
```

The query above uses the value of the **UserId** column to locate each user's name and state from the lookup file and loads the values into the `userinfo` list. This query differs from the basic example, which used a value in the lookup file (`Unknown`) as its default value. If no value in the **UserId** column matches a value in the first field of the lookup file or the value of **UserId** evaluates to `-1`, the query output displays `anonymous` as the value of the **UserName** column. This query also specifies a hyphen (-) as the default value for the remaining columns. In other words, if no value in the **UserId** column matches a value in the first field of the lookup file or it evaluates to `-1`, the query output displays a hyphen as the default value of column 5 (`State`) as well as the default value for columns 3 and 4.

For information on the `INTO` keyword, see "INTO Keyword", on page 315.

Exceptions

The `_lookup()` function raises SQL processing exceptions under any of the following conditions:

- The *<key>* or the fields in *<default_key_value_pair>* cannot be parsed as the data type specified in *<options_string>*.
- The file specified by *<filename>* cannot be opened or read.
- Memory to hold the cached lookup file is exhausted.

Working with Lookup Files

Typically, the lookup file is either a static file that rarely changes, or the result of a "dump" from an application or corporate database, such as the list of active users.

IMPORTANT: To use the `_lookup()` function, the lookup file has to be propagated to every host in the EDW instance so that the EDW can read it. Specifically, this file must have the same path on every host and be readable by the EDW server running on each system (for example, readable by the "lms" user). One implementation, for example, is to mount the directory containing the lookup file onto each system. For an alternative configuration, see "Centralizing the Lookup File", on page 333.

DEBUGGING AND SCALE-UP

NOTE: The `_lookup()` function works by loading the entire file into memory. If a lookup file is too big, the EDW starts paging to disk, and eventually ends your query. For this reason, Hexis Cyber Solutions recommends testing on subsets of large lookup files first, then increasing the subset until the whole file has been loaded.

Caching the Lookup Data: Requirements and Limitations

To enhance performance, the lookup file is cached in memory. The first time it is called, the `_lookup()` function does the following:

1 Loads a lookup table from the lookup file into main memory.

2 Performs subsequent calls to this cached main memory representation.

IMPORTANT: Because the lookup data is cached:

- The administrator must manually distribute the lookup file to every host in the EDW instance and ensure that each copy is identical. For more information, see "Working with Lookup Files", on page 331. For information on how to distribute the file, see Copying Files and Directories to Each Host (clsync) in Chapter 2, "Configuring and Managing HawkEye AP" in the *Administration Guide*
- The lookup data must fit into memory. For more information, see "Calling the _lookup() Function While Loading or Querying", next.

NOTE: The following is also true of the lookup data:

- The administrator must manually extract the data into the lookup file.

  For example, `_lookup()` cannot automatically download a webpage (URL) that contains lookup data.
- The lookup key values must be unique.

  The `_lookup()` function assumes that each key in the lookup file matches at most one record. This function cannot perform searches that scan or return multiple records.
- There is no support for XML-based lookup files.

  Binary data is supported, but it cannot contain NULL characters (ASCII 0) or new-line characters(ASCII 13).
- There is no support for network-based or dynamic lookup data.

CALLING THE _LOOKUP() FUNCTION WHILE LOADING OR QUERYING

As with other EDW features, lookups can be performed during loads, during queries, or both. To use the `lookup()` function during loading, include it in the SELECT statement in the PTL.

Generally, Hexis Cyber Solutions recommends performing lookups during loads rather than during queries:

- Lookups into giant lookup files require lots of memory. More memory is available during loads.

- The amount of memory required is more consistent during loads. The amount of memory required during queries depends on the details of the query, such as the number of groups that the `GROUP BY` and `SLICE BY` clauses specify.
- If multiple queries use the results of the lookups, they can share the results if the results have been loaded to disk, instead of each performing the lookup.
- Updates to lookup files occur intermittently, not continuously. It can be easier to determine the currency of the lookup data if the lookup occurs during loads. When each query performs its own lookups, it is more difficult to determine which version of the lookup data was used.
- Performing lookups while loading makes the SELECT statement used for queries easier to read.

However, lookups during queries makes sense under these conditions:

- If you want to save storage space, perform lookups at query time to avoid creating additional columns.
- Query-time lookups can improve load performance.

IMPORTANT: Hexis Cyber Solutions recommends that you use your own data to test the advantages of load-time and query-time lookups to determine which approach works best in your situation.

The best reason to perform query-time lookups is the need for “fresher” results. For example, assume you use this function in a query that returns the top 10 users from California. If you use load-time lookups, the precise definition of this query is “top 10 users whose traffic had come from California at the time of the request”. If you use query-time lookups, the precise definition is “top 10 users who currently live in California, regardless of where they lived at the time of the request”.

In some cases, security analysts will want to run both types of queries, and you will have to perform both load-time and query-time lookups.

CENTRALIZING THE LOOKUP FILE

It is possible to store the lookup file on a host that uses a network file system (such as Samba or NFS) to access each host in the EDW instance. This configuration simplifies the updating process because you no longer have to copy the lookup file to every host in the instance (for example, using `clsync`). Generally, centralizing the lookup file makes sense only if the lookups are performed rarely; otherwise, concurrent queries can clog the network or file server.

To estimate the performance impact of centralizing the lookup file, assume that each host in the instance needs to read the file completely for each lookup call. If you have 10 EDW hosts, and each mounts the same file server, and both the EDW hosts and file server are using 100mbps network connections, each EDW host achieves at most 10mbps as they clog the server with requests. This performance concern diminishes if:

- the lookup file is small—because the EDW hosts can cache the whole file
- the file server can keep up with the EDW instance, for example, lots of disks reading in parallel plus faster networking like gigabit ethernet (1gbps) or 10gigE (10gbps).

_rev_dns()

The `_rev_dns()` function performs a kind of reverse DNS look up. Instead of starting with a DNS host name and returning the IP address of the host, the function takes an IP address and returns the DNS host name.

Synopsis

```
_rev_dns( <IP_address> )
```

Description

Returns a string containing the host name for the server with the specified IP address.

Arguments

Argument	Description
<IP_address>	The IP address to look up: • If *<IP_address>* is an `int32` value, the IP address must be in host byte order. • If *<IP_address>* is a `varchar` value, the IP address must be in dotted format.

Return Value

The `_rev_dns()` function returns a `varchar` value containing the host name.

Exceptions

The `_rev_dns()` function raises a SQL processing exception when the data type of *<IP_address>* is neither `int32` nor `varchar`.

_tablematch()

If preceded by the `@` expression, the `_tablematch()` function returns a list of table and view names that match a specified pattern and that are located in the default or specified namespace. If not preceded by the `@` expression, this function returns an integer that represents the number of matching tables and views found.

Synopsis

```
[@]_tablematch( <pattern>[, <namespace>[, <option_string>]]
```

Description

The `_tablematch()` function matches the specified regular expression to return a list of names that represent log tables and views of log tables in the default or specified namespace.

NOTE: The `_tablematch()` function does not return the names of system tables.

The value of *<pattern>* must:

- be a regular expression

For example, to represent a SQL string expression that contains a backslash (\), you must escape the backslash with another backslash, as "\\".

- evaluate to a constant string expression

  The value must represent the actual names of tables. It cannot be a variable expression or other text that requires further processing to return the names of the tables. The names must be represented as strings within quotation marks.

The value for *<namespace>* defaults to an empty string (""). This value is relative to the default namespace, which is determined by the --namespace option of the atquery command. The function looks in the default namespace for table names that match *<pattern>*; specify a value for *<namespace>* to restrict the match to a namespace within the default.

If a value is specified for the namespace option of the _tablematch() function, the specified namespace is concatenated to the default namespace. The table below presents examples of how the query engine evaluates the namespace:

atquery --namespace	**_tablematch(<pattern> <namespace>)**	**Namespace Used**	**Example**
none specified	none specified	default as configured for EDW	default
specified as IT	none specified	value specified by atquery --namespace	IT
none specified	specified as firewalls	default as configured for EDW precedes value specified by <namespace>	default.firewalls
specified as IT	specified as firewalls	value specified by atquery --namespace precedes value specified by <namespace>	IT.firewalls

NOTE:

- When you run the query from HawkEye AP Console, the namespace is the one specified in the report definition or changed at runtime by the user. For more information, see the Options in Chapter 3, “Loading, Querying, and Managing the EDW” in the *Administration Guide* for the atquery utility.
- The value for namespace must be a constant string expression.

The value for *<option_string>* also defaults to an empty string (""). Specify "r" as *<option_string>* to search recursively through all subordinate namespaces, beginning with the namespace stem that concatenates the default namespace and the optional *<namespace>* argument. Like the other arguments, the value for the option string must be a constant string expression.

IMPORTANT: If you specify the third option, you must also specify a value for the namespace option, even to use the default namespace. In other words, to search recursively through all subordinate namespaces within the default namespace, you would specify:

```
FROM @_tablematch("some_pattern", "", "r" )
```

NOTE: All tables and views with names that match the specified pattern must share a common schema. The requirements are the same as those for running a subquery: column names and data types must match.

Arguments

Argument	Description
<pattern>	A regular expression to match the table names.
<namespace>	(Optional.) The namespace in which the tables and views to be matched are located. The namespace is relative to the default namespace. **IMPORTANT:** This option defaults to an empty string (""). You must specify a value for this option if: • You want to limit the match to a branch within the default namespace. In this case, specify the full subordinate namespace. • You specify a value for *<option_string>*. In this case, you must specify a value for the namespace argument. If you do not want to specify a a branch within the default namespace, specify an empty string ("").
<option_string>	(Optional.) Specify "r" to indicate that you want a recursive match, in which the function looks in *<namespace>* and all of its subordinate namespaces.

Return Value

If not preceded by the @ expression, the _tablematch() function returns an integer that represents the number of tables and views that match the regular expression. If preceded by the @ expression, this function returns a list of the matching table and view names. The returned names are fully qualified. You can access the list in the same way that you access the lists returned by list functions.

For more information, see “Working with Lists”, on page 313.

Example

The following SELECT statement uses the _tablematch() function to declare an implicit union of the tables located in the default namespace and its subordinate namespaces.

```
SELECT *
  FROM @_tablematch("syslog_.*", "", "r")
  DURING ALL
;
```

The match above represents all tables that begin with syslog_, such as syslog_ab, syslog_bc, and syslog_cd . This pattern follows standard regular expression syntax. If it had been written without the dot (.) character, it would have returned only names that terminated in the underscore (_) character.

TIP: Specify ".*" as *<pattern>* to match all table and view names.

The following queries uses the `_tablematch()` function to indicate how many tables and views match the search expression.

```
-- The SELECT below returns a count of values for some_field found in each table
   WITH subq1 as (SELECT _fromname() AS src, some_field, count(*) as cnt from
       @_tablematch("syslog_.*", "", "r") GROUP BY 1,2 DURING ALL)

-- The SELECT below returns the # of tables that contain a value for 'some_field'
   WITH subq2 as (SELECT some_field, count(*) as cnt from subq1 GROUP BY 1)

-- The SELECT below returns the # of times a value for 'some_field'
-- was found out of 'total_tables'
   SELECT some_field, cnt, _tablematch("syslog_.*", "", "r") as total_tables
       from subq2;
```

AGGREGATION FUNCTIONS

This section describes functions that perform aggregation on groups of rows.

Function	Purpose	Page
avg()	Computes the average of the values in the group	page 337
count()	Returns the number of rows in the group	page 338
max()	Returns the maximum value in a group	page 339
min()	Returns the minimum value in a group	page 340
median()	Computes the medium of the values in a group	page 341
sum()	Returns the sum of the values in a group	page 342
_first()	Returns the first value in a group	page 342
_last()	Returns the last value in a group	page 343
_strsum()	Returns the concatenation of the string values in a group	page 343

For more information on the use of aggregation functions, see "GROUP BY Clauses and Aggregation Queries", on page 278.

avg()

The `avg()` aggregate function computes the average of the values in the group.

Synopsis

```
avg( [DISTINCT] <column_expression> )
```

Description

Returns the average of the values in *<column_expression>* for all rows in a group.

Arguments

Argument	Description
<column_expression>	The target column with values to average. The allowed data types of the expression are `int32`, `int64`, `float`, and `timestamp`.

Return Value

The return value is the average. The data type of the return value generally matches the data type of *<column_expression>*.

Exceptions

The `avg()` function raises an SQL processing exception under these conditions:

- No column expressions are provided.
- A column expression has a data type that is not allowed.

Example

The SELECT statement returns the average number of bytes for all the records in the specified table:

```
SELECT avg(RespSize)
  FROM example_webserv_100
  DURING ALL;
```

count()

The count() aggregate function returns a count of the rows in the group.

Synopsis

```
count(*)

count( DISTINCT <column_expression> )
```

Description

Returns the number of items in a group.

Arguments

Argument	Description
<column_expression>	The target column with values to count. The argument is ignored unless the DISTINCT modifier keyword is specified

Return Value

The `count()` function returns an `int64` value representing the number of items in the group.

Examples

The following SELECT statement returns the number of records in the specified table:

```
SELECT count(*)
  FROM example_webserv_100
  DURING all;
```

max()

The `_max()` aggregate function returns the maximum value in a group.

Synopsis

```
max( <column_expression> )
```

Description

The `_max()` aggregate function returns the maximum value of the given expression for all the items in a group.

Arguments

Argument	Description
<column_expression>	The target column with values in which to find the maximum value.

Return Values

Generally, the `max()` aggregate function returns the maximum of all values in the group. The data type of the return value is the same as the data type of *<column_expression>*.

The return value of the `max()` aggregate function has special meaning for these data types:“Aggregation Functions”, on page 337“Aggregation Functions”, on page 337“Lookup Functions”, on page 323

Data type of <column_expression>	Meaning of the Return Value
`bool`	Returns a `bool` representing the logical OR of all the values
`timestamp`	Returns a `timestamp` representing the date and time-of-day furthest in the future from midnight, January 1, 1970
`varchar`	Returns a `varchar` containing the maximum

Exceptions

The `max()` aggregate function raises a SQL processing exception if passed less than one expression.

Example

The following query returns the latest record for all the records in the `example_webserv_100` table:

```
SELECT max(ts)
  FROM example_webserv_100
  DURING all;
```

min()

The `_min()` aggregate function returns the minimum value in a group.

Synopsis

```
min( <column_expression> )
```

Description

The `min()` aggregate function returns the minimum value of the given expression for all the items in a group.

Arguments

Argument	Description
`<column_expression>`	The target column with values in which to find the minimum value; data type is `bool` or `varchar`.

Return Values

Generally, the `min()` aggregate function returns the minimum value of all values in the group. The data type of the return value is the same as the data type of *`<column_expression>`*.

The return value of the `max()` aggregate function has special meaning for these data types:

Data type of <column_expression>	Meaning of the Return Value
`timestamp`	Returns a `timestamp` representing the date and time-of-day closest to midnight, January 1, 1970.

Exceptions

The `min()` aggregate function raises a SQL processing exception under any of these conditions:

- No expressions are passed as arguments.
- The data type of *`<column_expression>`* is `bool` or `varchar`.

Example

The following query returns the earliest record for all the records in the `example_webserv_100` table:

```
SELECT min(ts)
  FROM example_webserv_100
  DURING all;
```

median()

The `_median()` aggregate function returns the median value in a group.

Synopsis

```
median( <column_expression> )
```

Description

The `median()` aggregate function returns the median value of the given expression for all the items in a group.

Arguments

Argument	Description
<column_expression>	The target column with values to calculate the median. The allowed data types of the expression are `int32`, `int64`, `float`, and `timestamp`.

Return Value

The return value is the median of the items in the group. The data type of the return value generally matches the data type of *<column_expression>*. If the number of items in the group is even, then median returns the arithmetic means of *<column_expression>* for the two middle items.

Exceptions

The `median()` function raises an SQL processing exception under these conditions:

- No column expressions are provided.
- A column expression has a data type that is not allowed.

Example

The SELECT statement returns the median number of bytes for all the records in the specified table:

```
SELECT median(RespSize)
  FROM example_webserv_100
  DURING ALL;
```

sum()

The _sum() aggregate functions return the sum of values in a group.

Synopsis

```
sum( <column_expression> )
```

Description

The sum() aggregate returns the sum of the given expression for all the items in a group.

Arguments

Argument	Description
<column_expression>	The target column with numeric values to sum; data type is timestamp or varchar.

Return Values

The return value of the sum() aggregate function has special meaning, depending on the data type of *<column_expression>*:

Data type of <column_expression>	Meaning of the Return Value
bool	Returns a bool returns false if all the values in the group are false; otherwise it returns true.
int32 int64	Returns an int64 containing the sum of all the values in the group.
float	Returns a float containing the sum of all the values in the group.

Exceptions

The sum() aggregate function raises a SQL processing exception under any of these conditions:

- No expressions are passed as arguments.
- The data type of *<column_expression>* is varchar.

Examples

The following query returns the total number of bytes returned for all the records in the example_webserv_100 table:

```
SELECT sum(RespSize)
  FROM example_webserv_100
  DURING all;
```

_first()

The _first() aggregate function returns the first value in a group.

Synopsis

```
_first( <column_expression> )
```

Description

Returns the first value of the given expression from the first row in a group.

Generally, you use the `_first()` function in conjunction with an `ORDER BY` clause.

Arguments

Argument	Description
<column_expression>	The target column in the first row in the group from which the return value is taken.

Return Value

The `_first()` function returns the value of *<column_expression>* from the first row in the group.

_last()

The `_last()` aggregate function returns the last value in a group.

Synopsis

```
_last( <column_expression> )
```

Description

Returns the value of the given expression from the last row in a group.

Generally, you use the `_last()` function in conjunction with an `ORDER BY` clause.

Arguments

Argument	Description
<column_expression>	The target column in the last row in the group from which the return value is taken.

Return Value

The `_last()` function returns the value of *<column_expression>* from the last row in the group.

_strsum()

The `_strsum()` aggregate function returns the string concatenation of all the values in a group.

Synopsis

```
_strsum( varchar_column_expression> )
```

Description

The `_strsum()` aggregate function returns the string concatenation of the given expression for all the items in a group.

Arguments

Argument	Description
<varchar_column_expression>	Target column expression

Return values

The `_strsum()` aggregate function returns a `varchar` containing the string concatenation of all the values in the group.

Statistical Aggregate Functions

This section describes functions that perform statistical calculations on the values of a named column. If the query contains a "`group by`" clause, the calculation is made for each group of rows returned by the query; otherwise, the calculation is performed on all rows returned by the query.

Function	Purpose	Page
var_pop()	Returns the population variance of the values in the named column.	page 344
stddev_pop()	Returns the population standard deviation for the values of named column.	page 345
var_samp()	Returns the sample variance of the named column.	page 346
stddev_samp()	Returns the sample standard deviation for the values of named column.	page 347
variance()	Alias for the `var_samp()` function.	page 348
stddev()	Alias for the `stddev_samp()` function.	page 348

var_pop()

Returns the population variance of the values in the named column.

Synopsis

```
var_pop(column_expression)
```

Description

The `var_pop()` function returns the population variance of the values in the named column for each grouping of rows that results from a "`group by`" clause, or for all the rows if there is no "`group by`" clause.

Population Variance Calculation

The population variance of N numbers {x1, x2, x3, ..., xN} is defined as the sum of the squares of the difference between each number and the mean of the numbers, divided by N, as shown in the table below:

Column values	Mean	Difference from mean	Square of difference from mean
10	20	-10	100
20	20	0	0
30	20	10	100
25	20	5	25
15	20	-5	25
Sum of the squares of differences			**250**
Population variance			**50**

Arguments

Argument	Description
`column_expression`	A column expression that evaluates to one of the following datatypes: • int32 • int64 • float • timestamp (calculations are made using the number of seconds since epoch)

Return Value

The function returns the population variance as a FLOAT datatype.

stddev_pop()

Returns the population standard deviation for the values of named column.

Synopsis

```
stddev_pop(column_expression)
```

Description

The `stddev_pop()` function returns the population standard deviation of the values in the named column for each grouping of rows that results from a "`group by`" clause, or for all the rows if there is no "`group by`" clause.

Population Standard Deviation Calculation

The population standard deviation of a group of numbers is the positive square root of the population variance (See “var_pop()”, on page 344. In the example below, the population variance is 50 and the population standard deviation is 7.07106, the square root of 50.

Column values	Mean	Difference from mean	Square of difference from mean
10	20	-10	100
20	20	0	0
30	20	10	100
25	20	5	25
15	20	-5	25
Sum of the squares of differences			**250**
Population variance			**50**
Population standard deviation			**7.0710678**

Arguments

Argument	Description
`column_expression`	A column expression that evaluates to one of the following datatypes: • int32 • int64 • float • timestamp (calculations are made using the number of seconds since epoch)

Return values

The function returns the population standard deviation as a FLOAT datatype.

var_samp()

Returns the sample variance of the named column.

Synopsis

```
var_samp(column_expression)
```

Description

The `var_samp()` function returns the sample variance of the values in the named column for each grouping of rows that results from a "`group by`" clause, or for all the rows if there is no "`group by`" clause.

Sample Variance Calculation

The sample variance of N numbers {x1, x2, x3, ..., xN} is defined as the sum of the squares of the difference between each number and the mean of the numbers, divided by N-1, as shown in the table below:

Column values	Mean	Difference from mean	Square of difference from mean
10	20	-10	100
20	20	0	0
30	20	10	100
25	20	5	25
15	20	-5	25
Sum of the squares of differences			**250**
Sample variance			**62.5**

Arguments

Argument	Description
`column_expression`	A column expression that evaluates to one of the following datatypes: • int32 • int64 • float • timestamp (calculations are made using the number of seconds since epoch)

Return Value

The function returns the sample variance as a FLOAT datatype.

stddev_samp()

Returns the sample standard deviation for the values of named column.

Synopsis

```
stddev_samp(column_expression)
```

Description

The `stddev_samp()` function returns the sample standard deviation of the values in the named column for each grouping of rows that results from a "`group by`" clause, or for all the rows if there is no "`group by`" clause.

Sample standard deviation Calculation

The sample standard deviation of a group of numbers is the positive square root of the sample variance (See "var_samp()", on page 346). In the example below, the sample variance is 62.5 and the sample standard deviation is 7.9056, the square root of 62.5.

Column values	Mean	Difference from mean	Square of difference from mean
10	20	-10	100
20	20	0	0
30	20	10	100
25	20	5	25
15	20	-5	25
Sum of the squares of differences			**250**
Sample variance			**62.5**
Sample standard deviation			**7.9056**

Arguments

Argument	Description
`column_expression`	A column expression that evaluates to one of the following datatypes: • int32 • int64 • float • timestamp (calculations are made using the number of seconds since epoch)

Return values

The function returns the sample standard deviation as a FLOAT datatype.

variance()

The `variance()` function is an alias for the `var_samp()` function. The functions may be used interchangeably. For details, see "var_samp()", on page 346.

stddev()

The `stddev()` function is an alias for the `stddev_samp()` function The functions may be used interchangeably. For details, see "stddev_samp()", on page 347.

Logarithmic and Exponential Functions

This section describes functions that perform logarithmic and exponential calculations.

Function	Purpose	Page
_log()	Returns the natural logarithm of a value	page 34 9
_log10()	Returns the base 10 logarithm of a value	page 34 9
_pow()	Returns a power of a value	page 35 0
_exp()	Returns a power of e	page 35 0

_log()

The `_log()` function returns the natural logarithm of a value.

Synopsis

```
_log( <numeric_expression> )
```

Description

The `_log()` function returns the natural logarithm of *<numeric_expression>*.

Arguments

Argument	Description
<numeric_expression>	An expression that evaluates to an `int32`, `int64`, or `float` value.

Return Value

The data type of the return value from the `_log()` function is `float`.

_log10()

The `_log10()` function returns the base 10 logarithm of a value.

Synopsis

```
_log10( <numeric_expression> )
```

Description

The `_log10()` function returns the base 10 logarithm of *<numeric_expression>*.

Arguments

Argument	Description
<numeric_expression>	An expression that evaluates to an `int32`, `int64`, or `float` value.

Return Value

The data type of the return value from the `_log10()` function is `float`.

_pow()

The `_pow()` function returns the power of a value.

Synopsis

```
_pow( <numeric_expression>, <power> )
```

Description

The `_pow()` function returns the *<power>* of *<numeric_expression>*.

Arguments

Argument	Description
<numeric_expression>	An expression that evaluates to an `int32`, `int64`, or `float` value. This value is the base in the computation.
<power>	An expression that evaluates to an `int32`, `int64`, or `float` value. This value is the exponent in the computation.

Return Value

The data type of the return value from the `_pow()` function is `float`.

_exp()

The `_exp()` function returns a power of e.

Synopsis

```
_exp( <power> )
```

Description

The `_exp()` function returns e raised to the power specified by *<power>*.

Arguments

Argument	Description
<power>	An expression that evaluates to an `int32`, `int64`, or `float` value. This value is the exponent in the computation; e is the base.

Return Value

The data type of the return value from the `_exp()` function is `float`.

NUMERIC ROUNDING FUNCTIONS

This section describes functions that perform numeric rounding.

Function	Purpose	Page
_abs()	Computes the absolute value of an expression	page 35 1
_ceil()	Rounds a numeric value up to the nearest integer value	page 35 1
_floor()	Rounds a numeric value down to the nearest integer value	page 35 2
_round()	Performs fractional rounding to a specified precision	page 35 2

_abs()

The `_abs()` function computes the absolute value of an expression.

Synopsis

```
_abs( <expression> )
```

Description

The `_abs`() function computes the absolute value of *<expression>*.

Arguments

Argument	Description
<expression>	An expression that evaluates to an `int32`, `int64`, or `float` value.

Return Value

The `_abs()` function returns the absolute value of *<expression>*, in the same data type.

Exceptions

The `_abs()` function raises an SQL processing exception if the data type of <expression> is not `int32`, `int64`, or `float`.

_ceil()

The `_ceil()` function rounds a numeric value up to the nearest integer.

Synopsis

```
_ceil( <expression> )
```

Description

The `_ceil()` function rounds *<expression>* up to the nearest integer.

Arguments

Argument	Description
<expression>	An expression that evaluates to an `int32`, `int64`, or `float` value.

Return Value

If the data type of *<expression>* is `float`, the `_ceil()` function returns a `float` value that rounds *<expression>* up to the nearest integral value. If the data type is `int32` or `int64`, the `_ceil()` function returns *<expression>* without altering its value.

_floor()

The `_floor()` function rounds a numeric value down to the nearest integer.

Synopsis

```
_floor( <expression> )
```

Description

The `_floor()` function rounds *<expression>* down to the nearest integer.

Arguments

Argument	Description
<expression>	An expression that evaluates to an `int32`, `int64`, or `float` value.

Return Value

If the data type of *<expression>* is a `float` value, The `_floor()` function returns a `float` value that rounds that *<expression>* down to the nearest integral value. If the data type is `int32` or `int64`, the `floor()` function returns *<expression>* without altering its value.

Example

The following query returns the integral value of the average size of the requests in the log table:

```
SELECT _floor( avg(RespSize) )
  FROM example_webserv_100
  DURING ALL;
```

_round()

The `_round()` function performs rounding to a specified decimal or whole-number precision.

Synopsis

```
_round( <expression> [,<digits>] )
```

Description

The `_round()` function rounds *<expression>* to the decimal or whole-number precision specified by *<digits>*. The function rounds to the nearest integer value when *<digits>* is omitted.

Arguments

Argument	Description
<expression>	An expression that evaluates to an `int32`, `int64`, or `float` value.
<digits>	Optional. An `int32` value in the range -16 through 16. Positive numbers specify decimal precision. Negative numbers specify whole-number precision. The value 0 specifies that rounding does not occur. If this parameters is omitted, rounding occurs to the nearest integer.

Return Value

If the data type of *<expression>* is a `float` value, the `_round()` function returns a `float` the closest to the 10 ^ (-*<digits>*) value. If the data type is `int32` or `int64`, the function returns its argument in the same data type respectively.

Exceptions

The `_round()` function raises an SQL processing exception if the data type of *<expression>* is not `int32`, `int64`, or `float`.

STRING FUNCTIONS

This section describes functions that operate on string values.

Function	Purpose	Page
_strlowercase(), _lc()	Converts text to all lower-case letters	page 354
_struppercase(), _uc()	Converts text to all upper-case letters	page 355
_strmd5(), _md5()	Computes the MD5 hash value for a large block of text as a `varchar`	page 355
_strmd5_64(), _md5_64()	Computes the MD5 hash value for a large block of text as an `int64`	page 356
_strlen()	Counts the number of characters in a `varchar` value	page 357
_strstr()	Determines if a text value can be found as part of another text value	page 358

Function	Purpose	Page
_strmatch()	Returns a portion of text that matches a regular expression	page 357
_strmatchlist()	Parses portions of text that match a regular expression into a list variable	page 359
_strsplit()	Parses text into a list of text values using a single separation character	page 360
_strsplitxsv()	Parses text into a list of text values using complex separation logic	page 361
_strleft()	Returns a specified number of leading characters from text	page 361
_strright()	Returns a specified number of trailing characters from text	page 362
_strmiddle(), substr()	Returns all text from a starting character for a specified length	page 362
_strrepeat()	Returns a varchar expression repeated a specified number of times	page 363
_strlpad()	Prepends a space or the value of an optional varchar expression to a varchar expression up to the length specified in an int32 expression.	page 364
_strrpad()	Appends a space or the value of an optional varchar expression to the varchar expression up to the length specified in an int32 expression.	page 366
_strtrim()	Returns all but a specified number of leading and trailing characters from text	page 368
_strlink()	Creates an HTML anchor tag	page 369
_strcat()	Concatenates a list of text arguments	page 371
_strjoin()	Concatenates a list of text arguments with a separator between each value	page 371
_strformat(), _sprintf()	Formats a string from a list of substitution values	page 372

_strlowercase(), _lc()

The `_strlowercase()` function converts text to all lower-case letters.

Synopsis

```
_strlowercase( <string> )

_lc( <string> )
```

Description

The `_strlowercase()` function converts each letter in `<string>` to lower case, if possible.

The `_lc()` function is an alias for `strlowercase()`.

Arguments

Argument	Description
<string>	A varchar value containing the string to convert.

Return Value

The _strlowercase() function returns a varchar value with the result of the conversion.

Examples

The following query returns all the values of the URL column of the example_webserv_100 table in lower case.

```
SELECT _strlowercase( Url )
  FROM example_webserv_100
  DURING ALL;
```

_struppercase(), _uc()

The _struppercase() function converts text to all upper-case letters.

Synopsis

```
_struppercase( <string> )

_uc( <string> )
```

Description

The _struppercase() function converts each letter in *<string>* to upper case, if possible.

The _uc() function is an alias for struppercase().

Arguments

Argument	Description
<string>	A varchar value containing the string to convert.

Return Value

The _struppercase() function returns a varchar value that contains the result of the conversion.

_strmd5(), _md5()

The _strmd5() function computes the MD5 hash value for a large block of text.

NOTE: These functions are not recommended for cryptographic use.

Synopsis

```
_strmd5( <string> )

_md5( <string> )
```

Description

The `_strmd5()` function computes the MD5 hash value for a large block of text and returns the hash value as a `varchar` value. MD5 hash values are computed by an algorithm that produces a 128-big message digest, or “signature” for a text block of arbitrary size. Generally, one assumes that MD5 hash values are unique unless text blocks are identical.

The `_md5_64()`function is an alias for `_strmd5_64()`.

Arguments

Argument	Description
<string>	A `varchar` value containing the text block to hash.

Return Value

The `_strmd5()` function returns a `varchar` value that contains the hash value.

Exceptions

The `_strmd5()` function raises an SQL processing exception if the data type of *<string>* is not `varchar`.

_strmd5_64(), _md5_64()

The `_strmd5_64()` function computes the MD5 hash value for a large block of text.

NOTE: These functions are not recommended for cryptographic use.

Synopsis

```
_strmd5_64( <string> )

_md5_64( <string> )
```

Description

The `_strmd5_64()` function computes the MD5 hash value for a large block of text and returns the hash value as an `int64` value. MD5 hash values are computed by an algorithm that produces a 128-big message digest, or “signature” for a text block of arbitrary size. Generally, one assumes that MD5 hash values are unique unless text blocks are identical.

The `_md5_64()`function is an alias for `_strmd5_64()`.

Arguments

Argument	Description
<string>	A `varchar` value containing the text block to hash.

Return Value

The `_strmd5_64()` function returns an `int64` value that contains the first 8 bytes of the 128-bit hash value.

Example

The following query returns the MD5 hash value of the URL column for each record in the `example_webserv_100` table.

```
SELECT _md5( Url )
  FROM example_webserv_100
  DURING ALL;
```

_strlen()

The `_strlen()` function counts the number of characters in a `varchar` value.

Synopsis

```
_strlen( <string> )
```

Description

The `_strlen()` function counts the number of characters in *<string>*.

Arguments

Argument	Description
<string>	A `varchar` value.

Return Value

The `_strlen()` function counts the number of characters in *<string>* up to the first 0x0 character.

Examples

The following query returns the length of the largest URL in the `example_webserv_100` table:

```
SELECT max( _strlen(Url) )
  FROM example_webserv_100
  DURING ALL;
```

_strstr()

Synopsis

```
_strstr( <string>, <string_to_find> )
```

Description

The `_strstr()` function searches *<string>* for the first occurrence of *<string_to_find>*.

Arguments

Argument	Description
<string>	A `varchar` value that contains the string to search
<string_to_find>	A `varchar` value that contains the string to find

Return Value

The data type of the return value from the `_strstr()` function is `int32`. The return value contains the 0-based offset of the beginning *<string_to_find>* `within` *<string>*. The return value is -1 if *<string_to_find>* is not found within *<string>*.

Examples

The following query returns the number of records in the table where the `UserAgent` column contains the string 'Windows'.

```
SELECT count(*)
  FROM example_webserv_100
  WHERE _strstr(UserAgent,'Windows') <> -1
  DURING ALL;
```

The following query returns the number of records in the table where the `Url` column begins with the sequence '/images':

```
SELECT count(*)
  FROM example_webserv_100
  WHERE _strstr(Url,'/images') = 0
  DURING ALL;
```

_strmatch()

Synopsis

```
_strmatch( <string>, <pattern>, <default_value> )
```

Description

The `_strmatch()` function searches *<string>* for a sequence of characters that matches *<pattern>*, which you write as a Perl regular expression. The function returns the matching sequence of characters if a match is found; otherwise, the function returns *<default_value>*.

Arguments

Argument	Description
<*string*>	A varchar value that contains the string to search
<*pattern*>	A varchar value that contains the matching pattern, expressed as a Perl regular expression **NOTE:** When a pattern contains a backslash (\) character, you must escape it with a second backslash character; for example: \\d.
<*default_value*>	A varchar value that contains the string to return if the match fails

Return Value

The _strmatch() function returns a varchar that corresponds to the first parenthesized elements that matched the string. The function returns <*default_value*> if no match is found.

Exceptions

The _strmatch() function raises an SQL processing exception when the value of <*pattern*> is not a valid regular expression.

Examples

The following query returns a portion of the UserAgent column of each row in the example_webserv_100 table with the string 'MSIE'. If the UserAgent column does not contain the value, the default value of 'MSIE 5.0' is used.

```
SELECT _strmatch(UserAgent, '(MSIE [^ ]*)', 'MSIE 5.0')
  FROM example_webserv_100
  DURING ALL;
```

The following query uses the _strmatch() function in Boolean expression to return rows that contain ’eclipse’ anywhere within the referrer column:

```
SELECT *
  FROM mytable
  WHERE _strmatch(referrer, ’*.eclipse.*’, ’-’) <> ’-’
  DURING ALL
;
```

_strmatchlist()

Synopsis

```
_strmatchlist( <string>, <pattern>[, <fallback>] )
```

Description

The _strmatchlist() function attempts to match the specified <*string*> against the specified <*pattern*> and makes the matching portions of the string available as an INTO variable.

For more information, see “INTO Keyword”, on page 315.

Arguments

Argument	Description
<string>	A varchar value that contains the string to match
<pattern>	A varchar value that contains the matching pattern, expresses as a Perl regular expression
<fallback>	Optional. A varchar value that contains a string to match against *<pattern>* if *<string>* fails to match.

Return Values

The return value from the _strmatchlist() function is an int32 that corresponds to the number of parenthesized elements in *<pattern>* that find matches in *<string>*. If there are no parentheses used in *<pattern>*, _strmatchlist() returns 1 if there is a match and 0 if the pattern does not match.

Also, the _strmatchlist() makes available the actual matches as list elements. You can return the list as a list variable in place of the return value, or you can return it as an INTO variable. If there are no parenthesized elements used in *<pattern>*, the entire matched substring is returned as the first list element.

For more information on accessing return lists in place of return values, see “Multiple Values as Lists”, on page 313.

Example

Here is an example of using INTO to capture multiple matches from _strmatchlist():

```
SELECT match[1], match[2]
  FROM example_webserv_100
  WHERE _strmatchlist(UserAgent, '([^(]*)[(]([^)]+)') INTO match
  DURING time('Feb 01 00:00:00 2002'), time('Mar 31 23:59:59 2002');
```

_strsplit()

Synopsis

```
_strsplit( <separator_string>, <expression>[, <throw>] )
```

Description

The _strsplit() function parses *<expression>* into a list of strings, with the separator between the elements specified by *<separator_string>*.

The return value of _strsplit() is the number of strings parsed from *<expression>*. To access the list, use INTO or @. For more information about lists, see “List Functions”, on page 321.

Arguments

Argument	Description
<separator_string>	A varchar value that specifies the parsing separator.

Argument	Description
<expression>	A `varchar` value that contains the string to parse
<throw>	Optional. A value of any data type that will cause an exception to be thrown with an explanation when *<splitstring>* is rejected

Return Values

The `_strsplit()` function returns the number of strings parsed from *<expression>*.

Also, the `_strsplit()` function makes available the actual matches as list elements. You can return the list as a list variable in place of the return value, or you can return the list as an `INTO` variable.

For more information on accessing return lists in place of return values, see "Multiple Values as Lists", on page 313.

_strsplitxsv()

Synopsis

```
_strsplitxsv( <parsing_options>, <expression>[, <throw>])
```

Description

The `_strsplitxsv()` function is similar to `_strsplit()`, but instead of using a literal string to specify the separator, it allows you to specify more complex parsing options, such as quoting and escapification. The *<parsing_options>* argument contains a semicolon-delimited list of options.

For details about the permitted values in *<parsing_options>*, see "Basic Parsing Options", on page 326.

Return Values

The `_strsplitxsv()` function returns the number of strings parsed from *<expression>*.

Also, the `_strsplitxsv()` function makes available the parsed strings as list elements. You can return the list as a list variable in place of the return value, or you can return the list as an `INTO` variable.

For more information on accessing return lists in place of return values, see "Multiple Values as Lists", on page 313.

_strleft()

Synopsis

```
_strleft( <string>, <count> )
```

Description

The `_strleft()` function returns the leading *<count>* of characters from *<string>*.

Arguments

Argument	Description
<string>	A varchar value that contains the string to return
<count>	A non-negative int32 value that specifies the number of characters to return, starting from the left of *<string>*

Return Value

The _strleft() function returns a varchar value with the result.

_strright()

Synopsis

```
_strright( <string>, <count> )
```

Description

The _strright() function returns the trailing *<count>* of characters from *<string>*.

Arguments

Argument	Description
<string>	A varchar value that contains the string to return
<count>	A non-negative int32 value that specifies the number of characters to return, starting from the right of *<string>*

Return Value

The _strright() function returns a varchar value with the result.

_strmiddle(), substr()

Synopsis

```
_strmiddle( <string>, <offset>, <length> )

_substr( <string>, <offset>, <length> )
```

Description

The _strmiddle() function extracts a substring that starts at *<offset>* in *<string>* and extends the specified *<length>*.

The _substr() function is an alias for _strmiddle().

Arguments

Argument	Description
<string>	A varchar value that contains the string to return
<offset>	A non-negative int32 value that specifies the position of the first character to return.
<length>	A non-negative int32 value that specifies the number of characters to return.

Return Value

The _strmiddle() function returns a varchar value with the result.

_strrepeat()

Synopsis

```
_strrepeat( <varchar_expression>, <int32_expression> )
```

Description

The _strrepeat() function repeats the varchar expression as many times as the value specified in the integer expression.

Arguments

Argument	Description
<varchar_expression>	An expression that evaluates to a varchar value; can be a column value, any general expression that includes column values, and/or constants, provided that the data type of the expression evaluates to varchar.
<int32_expression>	An expression that evaluates to a non-negative int32 value that specifies the number of times to repeat the value in the first argument; can be a column value, any general expression that includes column values, and/or constants, provided that the data type of the expression evaluates to int32. NOTE: If the integer is negative, the EDW treats the negative value as zero (0) and returns an empty varchar value.

Return Value

The _strrepeat() function returns a varchar value that repeats the varchar-expression value as many times as specified by the int32 value.

Exceptions

The _strrepeat() function raises a SQL processing exception when the data type of *<varchar_expression>* is not varchar and the data type of *<int32_expression>* is not int32.

Examples

The following query repeats "`Page`" four times in each row:

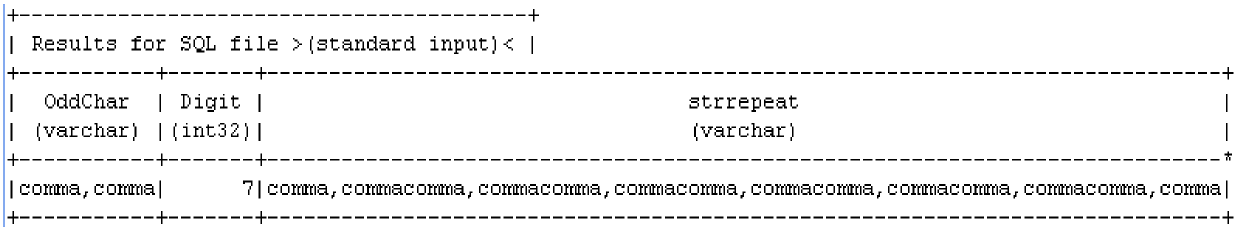
```
SELECT _strrepeat( 'Page', 4 )
  FROM someTable
  DURING ALL;
```

The query above returns: `PagePagePagePage`.

The following query repeats the value of the `OddChar` column as many times as specified by the value of the `Digit` column:

```
SELECT top 1 OddChar, Digit, _strrepeat( OddChar, Digit )
  FROM analyzer_types
  DURING ALL;
```

The graphic below illustrates the results.

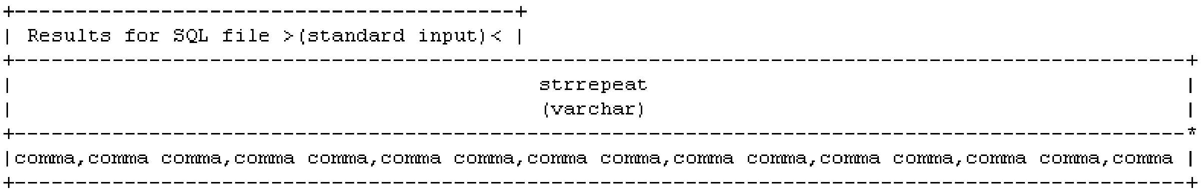


```
+----------------------------------------+
| Results for SQL file >(standard input)< |
+-----------+-------+-----------------------------------------------------------------------------+
|  OddChar  | Digit |                                  strrepeat                                  |
| (varchar) |(int32)|                                  (varchar)                                  |
+-----------+-------+-----------------------------------------------------------------------------*
|comma,comma|      7|comma,commacomma,commacomma,commacomma,commacomma,commacomma,commacomma,comma|
+-----------+-------+-----------------------------------------------------------------------------+
```

The following query appends a trailing space to the value of the `OddChar` column, which it repeats as many times as specified by the value of the `Digit` column plus 1:

```
SELECT top 1 _strrepeat( OddChar+" ", Digit+ _int32(1))
  FROM analyzer_types
  DURING ALL;
```

The graphic below illustrates the results.



```
+----------------------------------------+
| Results for SQL file >(standard input)< |
+------------------------------------------------------------------------------------------------+
|                                           strrepeat                                            |
|                                           (varchar)                                            |
+------------------------------------------------------------------------------------------------*
|comma,comma comma,comma comma,comma comma,comma comma,comma comma,comma comma,comma comma,comma |
+------------------------------------------------------------------------------------------------+
```

To display the full results of the `_strrepeat()` function, the query above includes only the `_strrepeat()` function in the SELECT clause. Therefore, it does not return values from the `OddChar` and `Digit` columns.

_strlpad()

Synopsis

```
_strlpad( <varchar_expression>, <int32_expression>
[,<optional_varchar_expression>])
```

Description

The _strlpad() function prepends the value of the optional varchar expression to the varchar expression until the result value becomes the length specified in the int32 expression. If the length of the result varchar expression is longer than the specified length, the function truncates the text from the right. If no optional varchar expression is included, the function prepends a space.

Arguments

Argument	Description
<varchar_expression>	An expression that evaluates to a varchar value; can be a column value, any general expression that includes column values, and/or constants, provided that the data type of the expression evaluates to varchar.
<int32_expression>	An expression that evaluates to a non-negative int32 value that specifies the length of the returned value; can be a column value, any general expression that includes column values, and/or constants, provided that the data type of the expression evaluates to int32. NOTE: If the integer is negative, the EDW treats the negative value as zero (0) and returns an empty varchar value.
<optional_varchar_expression>	An expression that evaluates to a varchar value; can be a column value, any general expression that includes column values, and/or constants, provided that the data type of the expression evaluates to varchar.

Return Value

The _strlpad() function returns a varchar value that includes the value of the varchar expression prepended with the value of the optional varchar expression, or a space if the optional expression is missing, up to the length specified in the int32 expression. If the length of this result varchar expression is longer than the specified length, the function truncates the text from the right.

Exceptions

The _strlpad() function raises a SQL processing exception when the data type of *<varchar_expression>* or *<optional_varchar_expression>* is not varchar and the data type of *<int32_expression>* is not int32.

Examples

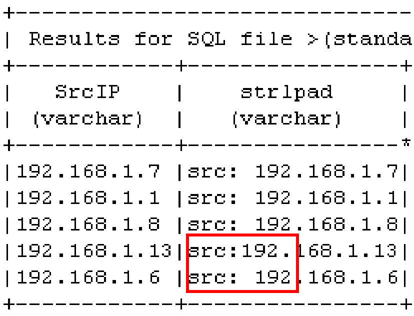
The following query returns "xyxhi":

```
SELECT _strlpad( 'hi', 5, 'xy')
  FROM someTable
  DURING ALL;
```

The following query prepends the value of the optional expression to the value in the SrcIP column until the result value becomes the length specified in the int32 expression:

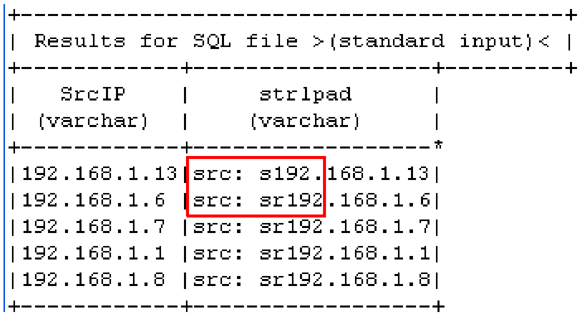
```
SELECT top 5 SrcIP,  _strlpad(SrcIP,16,"src: ")
  FROM analyzer_types
  DURING ALL;
```

The graphic below illustrates the results.



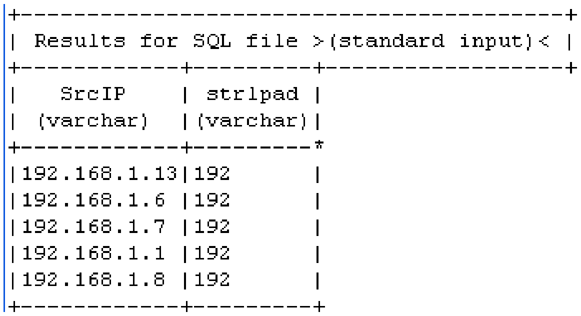
```
+-----------------------------------------+
| Results for SQL file >(standard input)< |
+------------+----------------+-----------+
|   SrcIP    |    strlpad     |
| (varchar)  |   (varchar)    |
+------------+----------------*
|192.168.1.7 |src: 192.168.1.7|
|192.168.1.1 |src: 192.168.1.1|
|192.168.1.8 |src: 192.168.1.8|
|192.168.1.13|src:192.168.1.13|
|192.168.1.6 |src: 192.168.1.6|
+------------+----------------+
```

NOTE: In the results above, the fourth row does not include the space before the IP address because this address is longer than the others. If you modify the function to return 18 characters instead of 16, four of the five rows returned repeat the first two characters of the optional varchar expression whereas the row with the longer value repeats only the first character, as shown below.



```
+-----------------------------------------+
| Results for SQL file >(standard input)< |
+------------+------------------+---------+
|   SrcIP    |     strlpad      |
| (varchar)  |    (varchar)     |
+------------+------------------*
|192.168.1.13|src: s192.168.1.13|
|192.168.1.6 |src: sr192.168.1.6|
|192.168.1.7 |src: sr192.168.1.7|
|192.168.1.1 |src: sr192.168.1.1|
|192.168.1.8 |src: sr192.168.1.8|
+------------+------------------+
```

If you modify the function to return only 3 characters, which is shorter than the values in the `SrcIP` column, no additional characters are prepended to the result. Instead, the query truncates the column values from the right to 3 characters, as shown below.



```
+-----------------------------------------+
| Results for SQL file >(standard input)< |
+------------+---------+------------------+
|   SrcIP    | strlpad |
| (varchar)  |(varchar)|
+------------+---------*
|192.168.1.13|192      |
|192.168.1.6 |192      |
|192.168.1.7 |192      |
|192.168.1.1 |192      |
|192.168.1.8 |192      |
+------------+---------+
```

_strrpad()

Synopsis

```
_strrpad( <varchar_expression>, <int32_expression>
[,<optional_varchar_expression>])
```

Description

The _strrpad() function appends the value of the optional varchar expression to the varchar expression until the result value becomes the length specified in the int32 expression. If the length of the result varchar expression is longer than the specified length, the function truncates the text from the right. If there is no optional varchar expression, the function appends a space.

Arguments

Argument	Description
<varchar_expression>	An expression that evaluates to a varchar value; can be a column value, any general expression that includes column values, and/or constants, provided that the data type of the expression evaluates to varchar.
<int32_expression>	An expression that evaluates to a non-negative int32 value that specifies the length of the returned value; can be a column value, any general expression that includes column values, and/or constants, provided that the data type of the expression evaluates to int32. NOTE: If the integer is negative, the EDW treats the negative value as zero (0) and returns an empty varchar value.
<optional_varchar_expression>	An expression that evaluates to a varchar value; can be a column value, any general expression that includes column values, and/or constants, provided that the data type of the expression evaluates to varchar.

Return Value

The _strrpad() function returns a varchar value that includes the value of the varchar expression appended with the value of the optional varchar expression, or a space if the optional expression is missing, up to the length specified in the int32 expression. If the length of this result varchar expression is longer than the specified length, the function truncates the text.

Exceptions

The _strrpad() function raises a SQL processing exception when the data type of *<varchar_expression>* or *<optional_varchar_expression>* is not varchar and the data type of *<int32_expression>* is not int32.

Examples

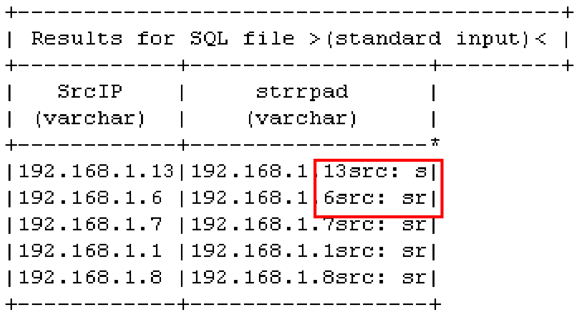
The following query returns "hixyx":

```
SELECT _strrpad( 'hi', 5, 'xy')
  FROM someTable
  DURING ALL;
```

The following query appends the value of the optional expression to the value in the SrcIP column until the result value becomes the length specified in the int32 expression:

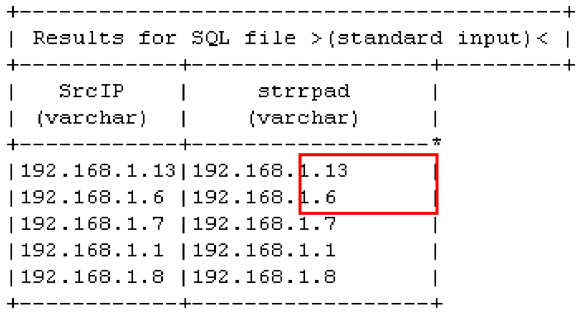
```
SELECT top 5 SrcIP,  _strrpad(SrcIP,18,"src: ")
  FROM analyzer_types
  DURING ALL;
```

The graphic below illustrates the results.



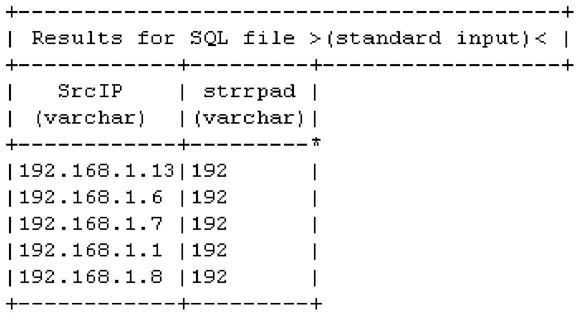
```
+------------------------------------------+
| Results for SQL file >(standard input)< |
+------------+-----------------+---------+
|   SrcIP    |     strrpad      |
| (varchar)  |    (varchar)     |
+------------+-----------------*
|192.168.1.13|192.168.1.13src: s|
|192.168.1.6 |192.168.1.6src: sr|
|192.168.1.7 |192.168.1.7src: sr|
|192.168.1.1 |192.168.1.1src: sr|
|192.168.1.8 |192.168.1.8src: sr|
+------------+-----------------+
```

NOTE: In the results above, the first row appends one fewer character from the optional expression because this IP address is longer than the others. If you modify the query so that it does not include an optional expression, the results append spaces after the IP address, as shown below.



```
+------------------------------------------+
| Results for SQL file >(standard input)< |
+------------+-----------------+---------+
|   SrcIP    |     strrpad      |
| (varchar)  |    (varchar)     |
+------------+-----------------*
|192.168.1.13|192.168.1.13      |
|192.168.1.6 |192.168.1.6       |
|192.168.1.7 |192.168.1.7       |
|192.168.1.1 |192.168.1.1       |
|192.168.1.8 |192.168.1.8       |
+------------+-----------------+
```

If you modify the function to return only 3 characters, which is shorter than the values in the `SrcIP` column, no additional characters are appended to the result. Instead, the query truncates the column values from the right to 3 characters, as shown below.



```
+------------------------------------------+
| Results for SQL file >(standard input)< |
+------------+--------+-----------------+
|   SrcIP    | strrpad |
| (varchar)  |(varchar)|
+------------+--------*
|192.168.1.13|192      |
|192.168.1.6 |192      |
|192.168.1.7 |192      |
|192.168.1.1 |192      |
|192.168.1.8 |192      |
+------------+--------+
```

_strtrim()

Synopsis

```
_strtrim( <string>, <left>[, <right>] )
```

Description

The `_strtrim()` function creates a substring by truncating the *<left>* number of characters from the left side of *<string>* and the *<right>* number of characters from the right side.

Arguments

Argument	Description
<string>	A varchar value that contains the string to return
<left>	A non-negative int32 value that specifies the number of characters on the left of *<string>* to remove. – or – A varchar value that identifies the leading characters to remove. The function scans the left portion of *<string>* until it finds a character not in *<left>* and then removes the characters up to that point
<right>	Optional. A non-negative int32 value that specifies the number of characters on the right of *<string>* to remove. – or – Optional. A varchar value that identifies the trailing characters to remove. The function scans backwards through the right portion of *<string>* until it finds a character not in *<right>* and then removes the characters up to that point.

Return Value

The _strtrim() function returns a varchar value containing the result.

Examples

The following query returns all the values of the Referrer column of the table with any slashes removed from the left side:

```
SELECT _strtrim( Referrer, '/' )
  FROM example_webserv_100
  DURING ALL;
```

The following query returns all the values of the Referrer column of the table with the trailing character removed:

```
SELECT _strtrim( Referrer, 0, 1 )
  FROM example_webserv_100
  DURING ALL;
```

The following query returns all the values of the Url column of the table with the leading 5 characters removed, if they begin with 'http:':

```
SELECT _strtrim( Url, _if( _strstr(Url,'http:')=0,5,0 ) )
  FROM example_webserv_100
  DURING ALL;
```

_strlink()

Synopsis

```
_strlink( <url>, <text>[, <new_window>] )
```

Description

The `_strlink()` function returns an HTML anchor tag suitable for use in HTML documents. The returned tag contains *<text>* as the link text to display, with *<url>* as the HREF attribute.

This function is useful for query results you intend to display in your own HTML or other third-party application.

NOTE: Do not use this function to display an active link in a HawkEye AP Console report. Although HawkEye AP Console provides the special URL data type from a dropdown in the Columns tab, the URL data type does not display the text as clickable links; instead it sorts the text as URLs.

Arguments

Argument	Description
<url>	A `varchar` value that contains the URL referenced by the `href` attribute of the link
<text>	A `varchar` value that contains the text content of the link
<new_window>	Optional. Value of any data type, which if equal to 1, indicates that the link should display the target URL in a new browser window.

Return Value

The `_strlink()` function returns a `varchar` value containing a valid HTML anchor tag.

Examples

Assume an event-log table contains a `varchar` `url` column, with values similar to the following:

```
Url
----------------------
http://www.acme.com
```

The following query returns a list of HTML anchor tags for the URLs contained in the `Url` column of the `example_webserv_100` table:

```
SELECT _strlink( Url, 'Click Here' ) AS anchor_tag
  FROM example_webserv_100
  DURING ALL;
```

The output looks similar to the following:

```
anchor_tag
----------------------
<a href="http://www.acme.com">Click Here</a>
```

The following query returns a list of HTML anchor tags encoded so that the link destination is displayed in a new browser window instead of replacing the page in the current browser window:

```
SELECT _strlink( Url, 'Click Here', 1) AS anchor_tag
  FROM example_webserv_100
  DURING ALL;
```

The output looks similar to the following:

```
anchor_tag
----------------------
<a href="http://www.acme.com" target=new>Click Here</a>
```

_strcat()

Synopsis

```
_strcat ( <argument>[, <argument>[...]] )
```

Description

The `_strcat()` function concatenates the string representations of the *<argument>* values.

Arguments

Argument	Description
<argument>	A value of any data type. The function converts the value to a `varchar` before concatenating.

NOTE: The `_strcat()` function may be passed list variables as arguments. For more information, see "Working with Lists", on page 313.

Return Value

The `_strcat()` function returns a `varchar` value containing the result.

Example

For rows with `Url="/www.acme.com"` and `ClientDNS = "66.127.84.10"`, the following query will return the value `"http://www.acme.com?dns=66.127.84.10"`.

```
SELECT _strcat('http:/', Url, '?dns=', ClientDNS)
  FROM example_webserv_100
  DURING ALL;
```

_strjoin()

Synopsis

```
_strjoin( <separator_string>, <argument>[, <argument>[...]] )
```

Description

The `_strjoin()` function concatenates the string representations of the *<argument>* values, but separates each string with *<separator_string>*. This function is similar to the Perl function of the same name.

Arguments

Argument	Description
<separator_string>	A `varchar` value that contains the characters that separate the concatenated string values. The value must be constant; that is, it cannot vary from row to row.
<argument>	A value of any data type. The function converts the value to a `varchar` before concatenating.

NOTE: The `_strjoin()` function may be passed list variables as arguments. For more information, see “Working with Lists”, on page 313.

Return Value

The `_strjoin()` function returns a `varchar` value containing the result.

Example

The following query will return the value “a,b,c,d”

```
SELECT TOP 1 _strjoin(",", "a", "b", "c", "d")
  FROM example_webserv_100
  DURING ALL;
```

_strformat(), _sprintf()

Synopsis

```
_strformat( <format_string>, <argument>[, <argument>[...]] )

_sprintf( <format_string>, <argument>[, <argument>[...]] )
```

Description

The `_strformat()` function creates a string by replacing the *format specifiers* in *<format_string>* with the string representations of the corresponding arguments. A format specifier is a special sequence of characters that is replaced by the `varchar` value of a corresponding *<argument>*.

For more information, see “Format Specifiers”, on page 373.

The `_sprintf()` function is an alias for `_strformat()`.

Arguments

Argument	Description
<format_string>	A `varchar` value that specifies the format of the output.
<argument>	A value of any data type to use in place of the corresponding format specifier in *<format_string>*. The data type must match the data type its corresponding format specifier.

Format Specifiers

Use format specifiers in the *<format_string>* argument as placeholders for values that you provide as additional arguments. Format specifiers have the following syntax:

```
%[<flags>][<width>][<precision>]<format>
```

The `_strformat()` function replaces each format specifier with a formatted value, leaving the other text unchanged. Use format specifiers that are appropriate for the data type of the corresponding arguments.

Formats to Specify

These are the types of formats you can specify.

%<format>	**Allowed Data Types**	**Meaning**
`%d`	`int32`, `int64`, or `timestamp`	Format the value as a decimal (base 10) number
`%x`	`int32`, `int64`, or `timestamp`	Format the value as a hexadecimal (base 16) number
`%o`	`int32`, `int64`, or `timestamp`	Format the value as an octal (base 8) number
`%f`	`float` or `timestamp`	Format the value as a floating-point number
`%e` `%E`	`float` or `timestamp`	Format the value as a floating-point number with exponential notation
`%g`	`float` or `timestamp`	Format the value as though `%f` were used, unless the exponent is less than -4 or larger than the specified precision, in which case format the value as though `%e` were used.
`%s`	`varchar` or `timestamp`	Format the value appropriately
`%c`	`bool`, `int32`, or `int64`	Format the value as a single character
`%%`		Replace with a single `%` character

Flags in Format Specifiers

<flag>	**Meaning**
–	Minus sign – Left-justify the field
=	Equal sign – Center-justify the field
0	Zero – Pad numeric values with leading zeros
+	Plus sign – Prefix positive numeric values with '+'

Field Widths in Format Specifiers

<width>	**Meaning**
<n>	A positive, base-10 integer that specifies a field width
*	Asterisk – obtain the width from the corresponding argument

Precision or Length in Format Specifiers

<precision>	**Meaning**
.<*n*>	A period followed by a base-10 integer that specifies the precision for a numeric argument or the maximum number of characters for a text argument
.*	A period followed by an asterisk – obtain the precision or length from the corresponding argument

Return Value

The `_strformat()` function returns a `varchar` value containing the result.

Exceptions

The `_strformat()` function raises an SQL processing exception if the format specifier used for an argument is invalid.

Examples

For rows with a `RespSize` equal to `10240`, the following query returns the value `"size: 10240"`:

```
SELECT _strformat("size: %d", RespSize)
  FROM example_webserv_100
  DURING ALL;
```

For rows with a `RespSize` equal to `10240`, the following query returns two columns containing the value `"size: 10.0k"`. The first use of `_strformat()` specifies the field width and precision in the format string, and the second specifies them in the argument list.

```
SELECT _strformat("size:%6.1fk", _float(RespSize)/1024.0),
       _strformat("size:%*.*fk", 6, 1, _float(RespSize)/1024.0)
  FROM example_webserv_100
  DURING ALL;
```

The space after `"size:"` in the result occurs because the specified length is six, and `"10.0k"` is only five characters wide. The function expands the value to six characters by padding the left with a space.

Time Functions

This section describes functions that operate on `timestamps`.

Function	Purpose	Page
_now(), now()	Returns the current system time as a timestamp	page 375
_time(), time()	Creates timestamps in various ways	page 375
_timeadd()	Adds units of time to a timestamp	page 378
_timediff()	Computes the difference between two timestamps	page 379
_timeformat(), _timef()	Creates a formatted string from a timestamp	page 380

Function	Purpose	Page
_timeparse(), _strptime()	Creates a timestamp from a formatted string	page 383
_timestart()	Rounds a timestamp down to specified precision	page 385

_now(), now()

The _now() function returns the current system time in GMT as a `timestamp`.

Synopsis

```
_now()

now()
```

Description

The `_now()` function returns the current system time.

The `now()` function is an alias for `_now()`.

Return Value

The `_now()` function returns a `timestamp` representing the current system time.

_time(), time()

The `_time()` function creates a timestamp from a variety of specifications.

Synopsis

```
_time( <time_specification>, <adjustment>[, <adjustment>[...]] )

time( <time_specification>, <adjustment>[, <adjustment>[...]] )
```

Description

The `_time()` function returns a timestamp based on the *<time_specification>* you provide. Generally you specify a character representation of the timestamp value you want. The function recognizes character-based timestamps in a fixed set of character formats. In addition, you can specify that you want the current time, or you can specify that you want the minimum or maximum timestamps that the EDW instance allows.

With *<adjustment>* arguments, you can adjust the *<time_specification>* to an earlier or later timestamp.

The `time()` function is an alias for `_time()`.

Arguments

Argument	Description
<time_specification>	A varchar value that specifies the timestamp you want; for example: _time('Oct 15 07:18:09 1997') For more information, see "Time Specifications for the _time() Function", next.
<adjustment>	A varchar value that specifies an amount and unit of time by which to adjust the *<time_specification>*; for example: _time('now', '-1wk') The value is expressed is expressed as '*<n><unit>*', where *<n>* is a positive or negative integer and *<unit>* is a longhand or shorthand notation for different units of time. For more information, see "Units of Time for the _time() and _timeadd() Functions", on page 377.

Time Specifications for the _time() Function

The `_time()` functions recognizes the following time specifications.

Time Specification	Pattern and/or Description	Example
Minimum allowed	The earliest timestamp allowed by the EDW instance	'min'
Maximum allowed	The latest timestamp allowed by the EDW instance	'max'
Current system time	The current timestamp	'now'
ISO 8601 format	YYYY-MM-DDTHH:MM:SS.NNNNNNZ The function always returns ISO 8601 timestamps in GMT, regardless of the WITH TIMEZONE directive.	'1997-11-15T07:18:09.000000Z'
Unix date format	MMM DD HH:MM:SS YYYY	'Oct 15 07:18:09 1997'
Unix date format with time zone	MMM DD HH:MM:SS ZZZ YYYY	'Oct 15 07:18:09 GMT 1997'
Unix date format with day of the week	WW MMM DD HH:MM:SS ZZZ YYYY	'Wed Oct 15 07:18:09 GMT 1997'
American century format with time	MM/DD/YYYY HH:MM:SS	'10/15/1997 17:18:09'
American decade format with time	MM/DD/YY HH:MM:SS	'10/15/97 17:18:09'
American century format	MM/DD/YYYY	'10/15/1997'
American decade format	MM/DD/YY	'10/15/97'

UNITS OF TIME FOR THE _TIME() AND _TIMEADD() FUNCTIONS

The `_time()` and `_timeadd()` functions recognize the following longhand and shorthand notations for specifying units of time. Trailing ”s” characters are ignored.

Longhand Time Unit	Shorthand Time Unit	Usage Notes
microsecond	usec, microsec	Specifies one microsecond.
millisecond	msec, millisec	Specifies one thousand microseconds.
second	sec	Specifies one second worth of microseconds.
minute	min	Specifies one minute worth of microseconds.
hour	hr	Specifies one hour worth of microseconds.
day		Specifies 24 hours worth of microseconds.
week	wk	Specifies seven days worth of microseconds.
month	mon	Specifies 30 days-worth of microseconds.
year	yr	Specifies 365 days worth of microseconds.

Return Value

The `_time()` function returns a `timestamp` containing the value you specified.

Examples

The following query returns rows with timestamps between the time the query executes and one hour prior:

```
SELECT count(*)
  FROM example_webserv_100
  DURING _time( 'now', '-1hr' ), _now()
;
```

The following query returns rows with timestamps between the time the query executes and three months plus five days prior:

```
SELECT count(*)
  FROM example_webserv_100
  DURING _time( 'now', '-3months', '-5days' ), _now();
```

The following query returns rows with timestamps between the beginning of the current year and the end of the first quarter:

```
WITH $tz AS 'PDT'

SELECT count(*)
  FROM example_webserv_100
  DURING      // current year begin
          _time( 'Jan 1 00:00:00' +
                $tz +
                _timeparse( (_now(), '%C')),
              // current year 1st quarter end
          _time( _timeparse( (_now(), '%C') + '-03-31T23:59:59:999999' +
                $tz)
```

```
;
```

_timeadd()

The `_timeadd()` function adds a length of time to a timestamp.

Synopsis

```
_timeadd( <timestamp>, <amount>, <time_unit>[, <time_zone>] )
```

Description

The `_timeadd()` function adds or subtracts a length of time to or from a `timestamp`.

The `_timeadd()` function accepts negative values for *<amount>*, which may yield timestamps that fall before the earliest timestamp that the EDW can store—generally, January 1, 1970.

Arguments

Argument	Description
<timestamp>	The timestamp to adjust by adding or subtracting a length of time.
<amount>	An `int32` that specifies the number of time units to add or subtract.
<time_unit>	A `varchar` value that indicates the kind of time units to add or subtract. The value is expressed in longhand or shorthand notation. For more information, see "Units of Time for the _time() and _timeadd() Functions", on page 377.
<time_zone>	Optional. A `varchar` value that indicates the time zone in which to perform the calculation. Use this argument to override the default time zone. You control the default time zone with the TIMEZONE setting directive. For more information, see "The TIMEZONE Setting", on page 311. For a list of allowed time-zone values, see "Appendix B: Time Zones" in the *Administration Guide*.

Return Value

The `_timeadd()` function returns a `timestamp` adjusted with the length of time added or subtracted.

Examples

The following query returns rows with timestamps between the time the query executes and one hour prior:

```
SELECT count(*)
  FROM example_webserv_100
  DURING _timeadd( _now(), -1, 'hr' ), _now()
;
```

The following query returns rows with timestamps between the time the query executes and three months prior, in a specified time zone:

```
WITH $tz AS 'PDT'

SELECT count(*)
  FROM example_webserv_100
  DURING _time( _now(), '-3', 'mon', $tz ), _now();
```

_timediff()

The `timediff()` function computes the difference between two `timestamps`.

Synopsis

```
_timediff( <timestamp_1>, <timestamp_2> )
```

Description

The `_timediff()` function computes the difference in microseconds between *<timestamp_1>* and *<timestamp_2>*.

Arguments

Argument	Description
<timestamp_1>	The base `timestamp`
<timestamp_2>	The timestamp for which the difference with *<timestamp_1>* is computed.

Return Value

The `_timediff()` function returns an `int64` value, which is the absolute value of the difference between the two timestamps in microseconds. The return value is always positive. For a signed result, use `"_INT64(ts1) - _INT64(ts2)"`.

Example

The following query searches the table for users who visited a page under `'/company'` two or more times, and returns the elapsed time between their first and last visit.

```
SELECT ClientDNS, _timediff( min(ts), max(ts) )
  FROM example_webserv_100
  WHERE _strstr(Url,'/company') = 0
  GROUP BY 1
  HAVING count(*) > 1
  DURING ALL
```

```
+------------------------------------------+
| Results for SQL file >example-time-03.sql< |
+--------------+---------+-------------------+
|  ClientDNS   |timediff |
|  (varchar)   | (int64) |
+--------------+---------*
```

```
output is post-sorted
+--------------+---------*
|216.239.46.200|363000000|
+--------------+---------*
```

_timeformat(), _timef()

The `_timeformat()` function creates a formatted string from a timestamp.

Synopsis

```
_timeformat( <time_format_string>, <timestamp>[, <time_zone>] )

_timef( <time_format_string>, <timestamp>[, <time_zone>] )
```

Description

The `_timeformat()` function creates a date-and-time-of-day `varchar` from a `timestamp` by replacing the formatting directives in <*time_format_string*> with the appropriate fields from <*timestamp*>.

The `_timef()` function is an alias for `_timeformat()`.

Arguments

Argument	Description

<*time_format_string*>	A `varchar` value that specifies the format of the output date and time-of-day. For example, the following time format string specifies month-day-year format:
	`%m-%d-%y`
	It produces results similar to the following:
	`11-16-06`
	For more information, see “Format Strings and Formatting Directives for the _timeformat() and _timeparse() Functions”, on page 380.

<*timestamp*>	A `timestamp` value to be formatted.
<*time_zone*>	Optional. A `varchar` value that indicates the time zone in which to perform the calculation. You control the default time zone with the TIMEZONE setting directive. For more information, see “The TIMEZONE Setting”, on page 311. For a list of allowed values, see “Appendix B: Time Zones” in the *Administration Guide*

FORMAT STRINGS AND FORMATTING DIRECTIVES FOR THE _TIMEFORMAT() AND _TIMEPARSE() FUNCTIONS

A <*time_format_string*> specifies the format you want for character-based timestamps. You can use text literals, such as hyphens (`-`) and colons (`:`), and you can use formatting directives, which begin with a percent sign (`%`). Enclose time format strings in quotation marks when you pass them as arguments to the `_timeformat()` and `_timeparse()` functions.

For example, the following time format string uses two text literals and three directives to specify character-based timestamps in month-day-year format:

```
'%m-%d-%y'
```

These are the formatting directives you can use in time format strings.

Formatting Directive	Meaning
`%a`	The abbreviated weekday name
`%A`	The full weekday name
`%b`	The abbreviated month name
`%B`	The full month name
`%c`	The preferred date and time representation for the current locale
`%C`	The century number (year/100) as a 2-digit integer; single digits are preceded by a zero. See also `%y` and `%Y`.
`%d`	The day of the month as a decimal number (range 01 to 31)
`%D`	Equivalent to: `%m/%d/%y` **NOTE:** In countries other than the United States, `%d/%m/%y` is the standard date format. To avoid the ambiguity of these two date formats, use the ISO standard for date formats: `%Y/%m/%d`. In addition to avoiding ambiguity, the ISO format sorts in a reasonable way.
`%e`	Like `%d`, the day of the month as a decimal number, but a leading zero is replaced by a space
`%E`	POSIX locale extensions. The sequences %Ec %EC %Ex %EX %Ey %EY %Od %Oe %OH %OI %Om %OM %OS %Ou %OU %OV %Ow %OW %Oy are supposed to provide alternate representations. Additionally %OB implemented to represent alternative months names (used standalone, without day mentioned).
`%G`	The ISO 8601 year with century as a decimal number. The 4-digit year corresponding to the ISO week number (see `%V`). This has the same format and value as `%y`, except that if the ISO week number belongs to the previous or next year, that year is used instead.
`%g`	Like `%G`, but without century, that is, with a 2-digit year (00-99)
`%h`	Equivalent to `%b`
`%H`	The hour as a decimal number using a 24-hour clock (range 00 to 23)
`%I`	The hour as a decimal number using a 12-hour clock (range 01 to 12)
`%j`	The day of the year as a decimal number (range 001 to 366)
`%k`	The hour (24-hour clock) as a decimal number (range 0 to 23); single digits are preceded by a blank. See also `%H`
`%l`	The hour (12-hour clock) as a decimal number (range 1 to 12); single digits are preceded by a blank. See also `%I`
`%m`	The month as a decimal number (range 01 to 12)
`%M`	The minute as a decimal number (range 00 to 59)
`%n`	A newline character
`%O`	Same as `%E`

Formatting Directive	Meaning
%p	Either 'AM' or 'PM' according to the given time value, or the corresponding strings for the current locale. Noon is treated as 'PM' and midnight as 'AM'.
%P	Like %p but in lowercase: 'am' or 'pm' or a corresponding string for the current locale
%r	The time in a.m. or p.m. notation. In the POSIX locale this is equivalent to '%I:%M:%S %p'.
%R	The time in 24-hour notation (%H:%M) For a version including the seconds, see %T below.
%s	The number of seconds since the epoch defined for the EDW instance.
%S	The second as a decimal number (range 00 to 61)
%t	A tab character
%T	The time in 24-hour notation (%H:%M:%S)
%u	The day of the week as a decimal, range 1 to 7, Monday being 1. See also %w
%U	The week number of the current year as a decimal number, range 00 to 53, starting with the first Sunday as the first day of week 01. See also %V and %W.
%V	The ISO 8601:1988 week number of the current year as a decimal number, range 01 to 53, where week 1 is the first week that has at least 4 days in the current year, and with Monday as the first day of the week. See also %U and %W.
%w	The day of the week as a decimal, range 0 to 6, Sunday being 0. See also %u.
%W	The week number of the current year as a decimal number, range 00 to 53, starting with the first Monday as the first day of week 01.
%x	The preferred date representation for the current locale without the time
%X	The preferred time representation for the current locale without the date
%y	The year as a decimal number without a century (range 00 to 99)
%Y	The year as a decimal number including the century
%z	The time-zone as a numeric offset from GMT. Required to emit dates that conform with RFC822 (using '%a, %d %b %Y %H:%M:%S %z'). Valid for the following functions: • _timeformat() • _timef()
%Z	The time-zone as name or abbreviation. Valid for the folloiwng functions: • _timeparse() • _strptime() • _timeformat() • _timef()
%+	The date and time in C <date(1)> format
%%	A literal '%' character

Return Value

The _timeformat() function returns a varchar with a formatted date and time-of-day.

Exceptions

The `_timeformat()` function raises an SQL processing exception if <*time_format_string*> does not conform with the defined syntax.

Example

The following query returns the epoch of the EDW instance, formatted as a string. The epoch is January 1, 1970. It is always represented internally as 0 microseconds.

```
WITH $epoch AS timestamp '0usec'

SELECT _timeformat('%b %e %H:%M:%S %Y', $epoch)


+--------------------------------------------+
| Results for SQL file >example-time-04.sql< |
+--------------------+-----------------------+
|    timeformat      |
|     (varchar)      |
+--------------------*
output is post-sorted
+--------------------*
|Jan  1 00:00:00 1970|
+--------------------*
```

The following query returns the URLs visited by '65.194.51.154', along with the hour and minute the URLs were visited:

```
SELECT ts, _timeformat('%H:%M', ts) AS time, url
  FROM example_webserv_100
  WHERE ClientDNS = '65.194.51.154'
  ORDER BY 1
  DURING ALL


+--------------------------------------------+
| Results for SQL file >example-time-07.sql< |
+---------------------------+---------+-------------+
|             ts            |  time   |     url     |
|        (timestamp)        |(varchar)|  (varchar)  |
+---------------------------+---------+-------------*
|2002-02-01T08:02:07.000000Z|08:02    |/summary.html|
|2002-02-01T08:17:05.000000Z|08:17    |/summary.html|
|2002-02-01T08:32:22.000000Z|08:32    |/summary.html|
|2002-02-01T09:17:12.000000Z|09:17    |/summary.html|
|2002-02-01T10:16:54.000000Z|10:16    |/summary.html|
|2002-02-01T11:01:51.000000Z|11:01    |/summary.html|
+---------------------------+---------+-------------+
```

_timeparse(), _strptime()

The `_timeparse()` function creates a `timestamp` from a date-and-time-of-day `varchar`.

Synopsis

```
_timeparse( <date_and_time>, <time_format_string>[, <time_zone>] )

_strptime( <date_and_time>, <time_format_string>[, <time_zone>] )
```

Description

The `_timeparse()` function creates a timestamp value from a character-based timestamp. It uses the specification in *<time_format_string>* to interpret the date and time-of-day components in *<date_and_time>*.

The `_strptime()` function is an alias for `_timeparse()`.

Arguments

Argument	Description
<date_and_time>	A `varchar` value with a date and time-of-day to be parsed and converted to a `timestamp`

<time_format_string>	A `varchar` value that specifies the format of the input date and time-of-day. For example, the following time format string specifies month day, year hour:minute:second format:
	`%b %d, %Y %H:%M:%S`
	It expects the value in *<date_and_time>* to be similar to the following:
	`Jul 8, 2009 19:01:36`
	For more information, see "Format Strings and Formatting Directives for the _timeformat() and _timeparse() Functions", on page 380.

<time_zone>	Optional. A `varchar` value that indicates the time zone in which to perform the calculation. You control the default time zone with the TIMEZONE setting directive. For more information, see "The TIMEZONE Setting", on page 311. For a list of allowed values, see "Appendix B: Time Zones" in the *Administration Guide*.

Return Value

The `_timeparse()` function returns a `timestamp`.

Exceptions

`The _timeparse()` function raises an SQL processing exception if:

- *<time_format_string>* does not conform with the defined syntax
- input value and time-format string omit hour, minute, and second

Examples

The following query returns the specified timestamp in IS0 8601 format:

```
WITH $timestamp AS '11-16-06:00:00:00'

SELECT _timeparse($timestamp, '%m-%d-%y:%H:%M:%S')


+--------------------------------------------+
| Results for SQL file >example-time-05.sql< |
+---------------------------+----------------+
|         timeparse         |
|        (timestamp)        |
+---------------------------*
|2006-11-16T00:00:00.000000Z|
+---------------------------*
```

_timestart()

The `_timestart()` rounds `timestamp` down to a specified unit of precision.

Synopsis

```
_timestart ( <timestamp>, <unit>[, <time_zone>]] )
```

Description

The `_timestart()` function rounds down *<timestamp>* to the *<unit>* of time specified.

Arguments

Argument	Description
<timestamp>	A `timestamp` value to be rounded down
<unit>	A `varchar` value that indicates the time unit to round to. The value is expressed in longhand (microsecond, millisecond, second, hour, minute, day, week, month, year) or in shorthand (usec, microsec, msec, millisec, sec, min, hr, wk, mon, yr) and must be enclosed within quotation marks. Sunday is the beginning of the week.
<time_zone>	Optional. A `varchar` value that indicates the time zone in which to perform the calculation. You control the default time zone with the TIMEZONE setting directive. For more information, see "The TIMEZONE Setting", on page 311. For a list of allowed values, see "Appendix B: Time Zones" in the *Administration Guide*.

Return Value

The `_timestart()` functions returns a `timestamp`.

Exceptions

The _timestart() function raises an SQL processing exception the value of *<unit>* does not conform to the specified syntax.

Network Address Functions

This section describes these functions that operate on network address values.

Function	Purpose	Page
_abbrev()	Abbreviated display format as text	page 387
_broadcast()	Broadcast address for network	page 388
_family()	Extract family of address: 4 for IPv4, 6 for IPv6	page 388
_host()	Extract IP address as text	page 389
_hostmask()	Construct host mask for network	page 390
_masklen()	Extract netmask length	page 390
_netmask()	Construct netmask for network	page 391
_set_masklen()	Set netmask length for inet value	page 391
_mapto_ipv4, _mapto_ipv6()	Maps an IPv6 address to an IPv4 address	page 392
_inet_plus()	Addition operator function	page 392
_inet_minus()	Substraction operator function	page 393
_inet_and(), _inet_not(), _inet_or()	Bitwise AND operator function Bitwise NOT operator function Bitwise OR operator function	page 394

_abbrev()

The `_abbrev()` function produces an abbreviated display format of a network address as text.

Synopsis

```
_abbrev(<expression>)
```

Description

The `_abbrev()` function produces an abbreviated display of *<expression>* as text, where *<expression>* is a column expression that evaluates to an `inet` value representing a valid IPv4 or IPv6 network address.

Arguments

Argument	Description
<expression>	A column expression that evaluates to an `inet`value.

Return Value

The `_abbrev()` function returns a `varchar` value. The value will be "(invalid)" if the column expression did not evaluate to a valid `inet` value.

Examples

The following query returns all values of the addr1 column of the `inet_test` table appended to the string "Network Address: "

```
SELECT _strcat('Network Address:',_abbrev(addr1))FROM inet_test DURING ALL;
```

If the addr1 column held an `inet` value `10.1.0.0/16` then the `_abbrev()` function returns the varchar value `"10.1.0.0/16"`.

_broadcast()

The `_broadcast()` function produces a broadcast address from a specified network address.

Synopsis

```
_broadcast(<expression>)
```

Description

The `_broadcast()` function produces a broadcast address from *<expression>* as an `inet` value, where *<expression>* is a column expression that evaluates to an `inet` value representing a valid IPv4 or IPv6 network address.

Arguments

Argument	Description
<expression>	A column expression that evaluates to an `inet`value.

Return Value

The `_broadcast()` function returns an `inet` value.

_family()

The `_family()`function returns the family of the specified network address.

Synopsis

```
_family(<expression>)
```

Description

The `_family()` function returns the family of the specified network address; 4 for IPv4, 6 for IPv6 address.

<expression> is a column expression that evaluates to an `inet` value representing a valid IPv4 or IPv6 network address.

Arguments

Argument	Description
<expression>	A column expression that evaluates to an inetvalue.

Return Value

The _family() function returns one of the following int32 values:

Value	Description
0	the <*expression*> evaluated to an empty inet value.
1	invalid (the <*expression*> evaluated to a value that was neither a valid IPv4 nor IPv6 representation
4	IPv4 address
6	IPv6 address

_host()

The _host() function extracts just the IP address as text from a specified network address.

Synopsis

```
_host(<expression>)
```

Description

The _host() function extracts an IP address from *<expression>* as a varchar value, where <expression> is a column expression that evaluates to an inet value representing a valid IPv4 or IPv6 network address.

Arguments

Argument	Description
<expression>	A column expression that evaluates to an inetvalue.

Return Value

The _host()function returns an varchar value.

Examples

The following query returns all values of the addr1 column of the inet_test table appended to the string "Host Address: "

```
SELECT _strcat('Host Address: ',_host(addr1)) FROM  inet_test DURING ALL;
```

If the addr1 column held an `inet` value `10.1.0.0/16` then the `_host()` function returns the `varchar` value "`10.1.0.0`".

_hostmask()

The `_hostmask()` function produces a host mask for the specified network address.

Synopsis

```
_hostmask(<expression>)
```

Description

The `_hostmask()` function produces a host mask from <*expression*> as an `inet` value, where <*expression*> is a column expression that evaluates to an `inet` value representing a valid IPv4 or IPv6 network address.

Arguments

Argument	Description
<*expression*>	A column expression that evaluates to an `inet`value.

Return Value

The `_hostmask()`function returns an `inet` value.

Examples

If the column expression held an `inet` value `192.168.1.0/16`, then the `_hostmask()` function returns the `inet` value `0.0.255.255`.

_masklen()

The `_masklen()` function extracts the network mask length for a specified network address.

Synopsis

```
_masklen(<expression>)
```

Description

The `_masklen()` function extracts the network mask length from <*expression*> as an `int32` value, where <*expression*> is a column expression that evaluates to an `inet` value representing a valid IPv4 or IPv6 network address.

Arguments

Argument	Description
<*expression*>	A column expression that evaluates to an `inet`value.

Return Value

The `_masklen()` function returns an `int32` value.

Examples

If the column expression held an `inet` value `192.168.1.0/16`, then the `_masklen()` function returns the `int32` value `16`.

_netmask()

The `_netmask()`function produces a network mask for a specified network address.

Synopsis

```
_netmask(<expression>)
```

Description

The `_netmask()`function produces a network mask from <*expression*> as an `inet` value, where <*expression*> is a column expression that evaluates to an `inet` value representing a valid IPv4 or IPv6 network address.

Arguments

Argument	**Description**
<*expression*>	A column expression that evaluates to an `inet`value.

Return Value

The `_netmask()` function returns an `inet` value.

Examples

If the column expression held an `inet` value `192.168.1.0/16`, then the `_netmask()` function returns the `inet` value `255.255.0.0`.

_set_masklen()

The `_set_masklen()` function sets the network mask length for a specified network address.

Synopsis

```
_set_masklen(<expression>,<length>)
```

Description

The `_set_masklen()` function sets the network mask length for <*expression*> as an `inet` value, where <*expression*> is a column expression that evaluates to an `inet` value representing a valid IPv4 or IPv6 network address.

Arguments

Argument	Description
<expression>	A column expression that evaluates to an inetvalue.
<length>	A column expression that evaluates to an int32 value, representing the network mask length

Return Value

The _set_masklen()function returns an inet value.

Examples

If the column expression held an inet value 192.168.1.5/24, then the _set_masklen() function returns the inet value **192.168.1.5/16** if the <*length*> column expression contained the int32 value 16.

_mapto_ipv4, _mapto_ipv6()

The _mapto_ipv4() and _mapto_ipv6() functions return a specified network address as an IPv4 or IPv6 address.

Synopsis

```
_broadcast(<expression>)
```

Description

The _mapto_ipv4() function produces an IPv4 address from <*expression*> as an inet value, where <*expression*> is a column expression that evaluates to an inet value representing a valid IPv4-mapped IPv6 network address (see http://tools.ietf.org/html/rfc4291#section-2.5.5.2).

The _mapto_ipv6() function produces an IPv4-mapped IPv6 network address from <*expression*> as an inet value, where <*expression*> is a column expression that evaluates to an inet value representing a valid IPv4 network address.

Arguments

Argument	Description
<expression>	A column expression that evaluates to an inetvalue.

Return Value

The mapto_ipv4() and _mapto_ipv6() functions returns an inet value.

_inet_plus()

The _inet_plus() function provide the addition operator for inet values.

Synopsis

```
_inet_plus(<operand1>,<operand2>)
```

Description

The `_inet_plus()` and `_inet_minus()` functions provide addition and subtraction capability in the absence of support for the "+" and "-" operators on `inet` values within Sensage SQL.

Arguments

Argument	Description
<operand1>	A column expression that evaluates to an `inet`value.
<operand2>	A column expression that evaluates to an `int32` or `int64` value

Return Value

The `inet_plus()` function returns an `inet` value.

Examples

If the first operand held an inet value `127.0.0.1`, then the `_inet_plus()` function returns the `inet` value `127.0.1.2` if the second operand contained the `int32` value `257`.

_inet_minus()

The `_inet_minus()` function provides the subtraction operator for `inet` values.

Synopsis

```
_inet_minus(<operand1>,<operand 2>)
```

Description

The `_inet_plus()` and `_inet_minus()` functions provide addition and subtraction capability in the absence of support for the "+" and "-" operators on inet values within Sensage SQL.

Arguments

Argument	Description
<operand1>	A column expression that evaluates to an `inet` value.
<operand2>	A column expression that evaluates to an `int32`, `int64` or inet value

Return Value

The `_inet_minus()`function returns an `inet` value unless both operands are `inet` values, that is, we are subtracting two `inet` values, in which case an `int64` value is returned. Consequently, subtracting two `inet` values means that the `int64` value cannot properly represent the delta between all pairs of IPv6 addresses. Under this condition, Sensage SQL does not produce an overflow error but simply wraps the integers/addresses and proceeds (this is consistent with how math operations on simple integers are performed in Sensage SQL).

Examples

If the first operand held an `inet` value `192.168.1.2`, then the `_inet_minus()` function returns the `inet` value `192.168.0.1` if the second operand contained the `int32` value `257`.

If the first operand held an `inet` value `192.168.1.2`, then the `_inet_minus()` function returns the `int64` value `257` if the second operand contained the `inet` value `192.168.0.1`.

_inet_and(), _inet_not(), _inet_or()

The `_inet_and()`,`_inet_not()` and `_inet_or()` functions provide the bitwise AND, NOT and OR operators for `inet` values.

Synopsis

```
_inet_and(<expression>,<operand>)
_inet_not(<expression>,<operand>)
_inet_or (<expression>,<operand>)
```

Description

The `_inet_and()`,`_inet_not()` and `_inet_or()` functions provide the bitwise AND, NOT and OR operator capability in the absence of support for the "&", "^" and "|" operators on inet values within Sensage SQL.

Arguments

Argument	Description
<expression>	A column expression that evaluates to an `inet`value.
<operand>	A column expression that evaluates to an `int32`, `int64` or inet value

Return Value

The `_inet_and()`,`_inet_not()` and `_inet_or()` functions return an `inet` value.

Examples

If the first operand held an `inet` value `127.0.0.1`, then the `_inet_plus()` function returns the `inet` value `127.0.1.2` if the second operand contained the `int32` value `257`.

Miscellaneous Functions

This section describes miscellaneous functions.

Function	Purpose	Page
_quantize()	Performs distribution analysis	page 395
_fifo()	Pushes a value onto a queue	page 399
_lms_taskid()	Returns the task ID of the SQL request	page 399
_lms_buildinfo()	Returns information about the version of the EDW	page 401
_fromname()	Returns the name of the table that is being queried	page 402
_fromindex()	Returns the index of the table that is being queried	page 403

_quantize()

The `quantize()` performs distribution analysis on the values in a column.

Synopsis

```
_quantize( <expression>, <low_format>, <limit>, <high_format> )

_quantize( <expression>, <low_format>[, <start>, <format>, <increment>
           [, <start>, <format>, <increment> [...]], <limit>, <high_format> )
```

Description

The `_quantize()` function performs distribution analysis on numeric values returned in a column of a result set. The function returns formatted strings that describe the distribution ranges in which the numeric values fall. You specify the distribution ranges with *<limit>* and optionally with *<start>* and *<increment>* arguments. These arguments specify the thresholds that delimit one distribution range from the next.

You can specify thresholds with:

- a single value that delimits two ranges

  —or—

- a set of values that delimit multiple ranges

The function returns a formatted string that is derived from one of the format arguments. Each format argument must be a `varchar` that includes a threshold variable. The threshold variable is expressed as `%1` for low thresholds and `%2` for high thresholds. For example, a low-range format argument might be `"below %1"`, a high-range format argument might be `"%2 and above"`, and a mid-range format argument might be `"%1 up to %2"`.

The formatted string that `_quantize()` returns indicates the range in which *<expression>* falls. The function compares the value of each expression in the result set against values specified for *<limit>*, *<start>*, and *<increment>*. When the shorter syntax is used, the function operates with only two ranges. When the longer syntax is used, the function operates with three or more ranges.

Arguments

Argument	Description
<expression>	A column expression that evaluates to an `int32`, `int64`, or `float` value
<low_format>	A `varchar` that includes `%1` as the low-value threshold
*<start>**	Optional. An `int32`, `int64`, or `float` value
*<format>**	Optional. A `varchar` that includes `%1` as low-value threshold and `%2` as high-value threshold
*<increment>**	Optional. An `int32`, `int64`, or `float` value
<limit>	An `int32`, `int64`, or `float` value
<high_format>	A `varchar` that includes `%2` as the high-value threshold

* The optional arguments operate as a set. You cannot specify one without specifying the others. Multiple sets of the optional arguments are allowed.

DISTRIBUTING DATA BETWEEN TWO RANGES

When only *<limit>* is specified, `the _quantize()` function compares the value of *<expression>* to the value of *<limit>*. If *<expression>* is less than *<limit>*, the function replaces `%1` in the *<low_format>* argument with *<limit>* and returns the formatted string. If *<expression>* is equal to or greater than *<limit>*, the function replaces `%2` in the *<high_format>* argument with *<limit>* and returns the formatted string.

For example, assume you run the following query:

```
SELECT RespSize, _quantize(RespSize, "under %1", 16000, "%2 or above") as range
  FROM example_webserv_100
  DURING ALL;
```

The example below illustrates a subset of the rows that the `_quantize()` function returns.

```
+-------------------------------------------+
| Results for SQL file >(standard input)<   +
+--------+-------------+-------------------+
|RespSize|    range     |
|(int32) |   (varchar)  |
+--------+--------------+
|   37361|16000 or above|
|     261|under 16000   |
|       0|under 16000   |
|    7121|under 16000   |
|   37361|16000 or above|
|    3172|under 16000   |
|   17252|16000 or above|
|   12203|under 16000   |
|       0|under 16000   |
|    7121|under 16000   |
```

DISTRIBUTING DATA AMONG MULTIPLE RANGES

When *<start>*, *<format>*, and *<increment>* arguments are specified, in addition to *<limit>*, the `_quantize()` function compares the value of *<expression>* to *<limit>* and the other intermediate thresholds, which it computes. The additional *<format>* argument must include both

`%1` as the low threshold variable and `%2` as the high threshold variable; for example, it might be `"%1 up to %2"`.

The `_quantize()` function evaluates the syntax with one start value as follows:

- If *`<expression>`* is equal to or greater than *`<limit>`*, `_quantize()` replaces “`%2`” in *`<high_format>`* with *`<limit>`* and returns the resulting string.
- If *`<expression>`* is less than *`<start>`*, `_quantize()` replaces “`%1`” in *`<low_format>`* with *`<start>`* and returns the resulting string.
- If *`<expression>`* is greater than *`<start>`* but less than *`<limit>`*, `_quantize()` operates with a set of intermediate ranges to which it applies *`<format>`*.

  The first intermediate range begins with the start value and ends with the threshold computed by adding the increment value to the start value. The `quantize()` function computes additional intermediate ranges by sequentially adding the increment value to the upper threshold of the previous range. It then compares each expression to the values in these intermediate ranges. When *`<expression>`* is greater than *`<start>`* but less *`<limit>`*, `_quantize()` determines which intermediate range it falls within. It replaces `%1` in *`<format>`* with the lower threshold of the range and replaces `%2` with the higher threshold and returns the resulting string.

For example, assume you run the following query:

```
SELECT RespSize, _quantize(RespSize, "under %1",
       1000, "%1 up to %2", 5000,
       16000, "%2 or above") as range
  FROM example_webserv_100
  DURING ALL;
```

The example below illustrates a subset of the rows that the `_quantize()` function returns.

```
+-----------------------------------------+
| Results for SQL file >(standard input)<  +
+--------+----------------+---------------+
|RespSize|      range      |
|(int32) |    (varchar)    |
+--------+----------------+
|   37361|16000 or above   |
|     261|under 1000       |
|       0|under 1000       |
|    7121|6000 up to 11000 |
|   37361|16000 or above   |
|    3172|1000 up to 6000  |
|   17252|16000 or above   |
|   12203|11000 up to 16000|
|       0|under 1000       |
|    7121|6000 up to 11000 |
```

IMPORTANT: The `_quantize()` function returns the high-format string for any value that equals or exceeds *`<limit>`*. To avoid confusing results, ensure that *`<start>`* and *`<increment>`* define intermediate ranges such that an upper threshold eventually coincides with *`<limit>`*.

Specifying Multiple Start Values

You can specify more than one set of *<start>*, *<format>*, and *<increment>* arguments to the `_quantize()` function. The `_quantize()` function evaluates the syntax with multiple start values as follows:

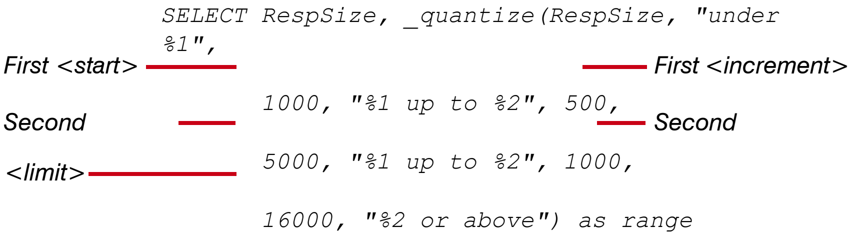
- If *<expression>* is equal to or greater than *<limit>*, `_quantize()` replaces "`%2`" in *<high_format>* with *<limit>* and returns the resulting string.
- If *<expression>* is less than the first *<start>*, `_quantize()` replaces "`%1`" in *<low_format>* with the first *<start>* and returns the resulting string.
- If *<expression>* is greater than the first *<start>* but less than the second *<start>*, `_quantize()` places the expression value within the appropriate intermediate range between the two.
- If *<expression>* is greater than the second *<start>* but less than the third *<start>*, or *<limit>* if there are no additional *<start>* arguments, `_quantize()` places the expression value within the appropriate intermediate range.

IMPORTANT: If there are multiple start values, you must list them in increasing order for the result to be meaningful.

For example, assume you run the following query:

```
SELECT RespSize, _quantize(RespSize, "under %1",
       1000, "%1 up to %2", 500,
       5000, "%1 up to %2", 1000,
       16000, "%2 or above") as range
  FROM example_webserv_100
  DURING ALL;
```

The graphic below illustrates the purpose of the numeric values in the above query.



```
SELECT RespSize, _quantize(RespSize, "under
%1",

         1000, "%1 up to %2", 500,

         5000, "%1 up to %2", 1000,

         16000, "%2 or above") as range
```

The example below illustrates a subset of the rows that the `_quantize()` function returns.

```
+----------------------------------------+
| Results for SQL file >(standard input)<  +
+--------+-----------------+---------------+
|RespSize|      range      |
|(int32) |    (varchar)    |
+--------+-----------------+
|   37361|16000 or above   |
|     261|under 1000       |
|       0|under 1000       |
|    8132|8000 up to 9000  |
|   37361|16000 or above   |
|    3172|3000 up to 3500  |
|   17252|16000 or above   |
|    5208|5000 up to 6000  |
|   12203|12000 up to 13000|
|      43|under 1000       |
|    3894|3500 up to 4000  |
|    2421|2000 up to 2500  |
|    4765|4500 up to 5000  |
|    1313|1000 up to 1500  |
```

GRAPHING THE RESULTS

Because the `_quantize()` function labels each range, you can graph its result set in HawkEye AP Console. To make the graph meaningful, put the statement with the `_quantize()` function in a subquery and then count the ranges in the main query. Run the query in HawkEye AP Console and display the results as a bar graph.

For example, the following query enables graphing in HawkEye AP Console:

```
WITH subquery as (
  SELECT RespSize, _quantize(RespSize, "under %1",
         5000, "%1 to %2", 10000,
         15000, "%1 to %2", 25000,
         30000, "%2 or above") as range
    FROM example_webserv_100
    DURING ALL
)
SELECT range, count(*)
  FROM subquery
  GROUP BY range
  ORDER BY 1
```

Return Value

The return type of the `_quantize()` function is a `varchar`.

Exceptions

The `_quantize()` function raises a SQL processing exception under any of these conditions:

- When the data types of the *<expr>* or *<limit>* arguments are not `int32`, `int64`, or `float` values.
- When the data types of the *<expr>* and *<limit>* arguments are not identical.
- When the data type of format arguments are not `varchar`.
- If you pass an incorrect number of arguments.

_fifo()

The `_fifo()` function pushes a value onto a queue.

Synopsis

```
_fifo( <list_name>, <value>, <default> )
```

Description

If the queue designated by `<list_name>` contains at least one element, `_fifo()` shifts off the first value and adds the specified `<value>` to the end of the queue, returning the removed value.

If the queue designated by `<list_name>` is empty, `_fifo()` adds the specified `<value>` to the end of the queue and returns the value of the `<default>` argument, if specified.

If the queue designated by `<list_name>` is empty and no `<default>` is specified, `_fifo()` raises an exception.

This function is often used with the SLICE BY clause. For more information, see "SLICE BY Clauses", on page 282.

Arguments

Argument	Description
`<list_name>`	The name of a list such as 'myfifo'. The name <*list_name*>may be derived from a column so that each group (those rows with the same value in a certain column) has its own FIFO queue.
`<value>`	A value of any data type that designates the value to be pushed into the queue
`<default>`	Mandatory. The value to be returned when the designated queue is empty.

Return Value

The return type is the type of the `<default>` argument.

Exceptions

The `_fifo()` function raises an SQL processing exception if the data types of `<value>` and `<default>` are not the same.

_lms_taskid()

The `_lms_taskid()` function returns the task ID of the SQL request.

Synopsis

```
_lms_taskid()
```

Description

The `_lms_taskid()` function returns the internal task ID generated by the system when it receives a SQL request.

Return Value

The `_lms_taskid()` function returns a `varchar` value that contains the internal task ID.

Example

The following query returns the value of the task ID:

```
SELECT TOP 1 _lms_taskid()
  FROM example_webserv_100
  DURING ALL

Results:

+-------------------------------------------+
| Results for SQL file >example-lms-01.sql< |
+--------------------------------+----------+
|           lms_taskid           |
|           (varchar)            |
+--------------------------------*
output is post-sorted
+--------------------------------*
|955BC97C2E68F83F797E0AEFF3BC0307|
+--------------------------------*
```

_lms_buildinfo()

The `_lms_buildinfo()` function returns information about the version of the EDW.

Synopsis

```
_lms_buildinfo()
```

Description

The `_lms_buildinfo()` function returns a string describing the version of the EDW.

Return Value

The `_lms_buildinfo()` function returns a varchar that contains the build date, the ID of the most recent committed software change, the name of the build client specification, the name and IP address of the machine on which the build occurred, the directory containing the build files, and a list of the files compiled with VERBOSE logging enabled.

Example

The following query returns the build information for the EDW.

```
SELECT TOP 1 _lms_buildinfo()
  FROM example_webserv_100
  DURING ALL;
```

The following shows an example of the output:

```
2008-09-25T03:44:06+0000 Change 56459 Client name:
source.quattroloop.2006.main.56460 Client host: eng.hq.sensage.com Client root:
/export/services/buildloop/tmp/67c1ae309a349577d03d1973ae9d7d21/ootb-analytics
verbose:
```

_fromname()

The `_fromname()` function returns the fully qualified name of each table or view that is being queried.

Synopsis

```
_fromname()
```

Description

HawkEye AP supports a `FROM @<list_expression>` construct that concatenates the rows of one or more tables and/or views and provides the result as the source data for the `SELECT` statement. The result of the query is equivalent to the results of a `UNION ALL` query.

You can use the `_fromname()` function to identify the source object (table or view) of each row in the result set of a `SELECT` statement whose `FROM` clause uses the @`<list_expression>` construct. The `_fromname()` function evaluates to a string that represents the name of the object in the `FROM @<list_expression>` construct for each returned row.

For more information, see:

- "FROM Clauses", on page 273
- "Working with Lists", on page 313
- "_tablematch()", on page 334

Return Value

The `_fromname()` function returns a `varchar` with the name of each table or view that is being queried.

Examples

The following statement queries two tables and returns the earliest timestamp, the latest timestamp, and the number of records in each table, grouped by tablename. The name of the table is returned in the first column of the result set.

```
SELECT _fromname(), min(ts), max(ts), count(*)
  FROM @_list('example_webserv_100', 'example_syslog')
  GROUP BY 1
  DURING ALL;
```

The following statement queries all tables in the default namespace and returns a count of each url in each table returned. In this case, the table names are not known at the time the query is written.

```
SELECT _fromname(), url, count(*)
  FROM @_tablematch('.*')
  GROUP BY 1,2
  DURING ALL;
```

_fromindex()

The `_fromindex()` function returns the index of the table or view that is being queried.

Synopsis

```
_fromindex()
```

Description

HawkEye AP supports a `FROM @<list_expression>` construct that concatenates the rows of one or more tables and/or views and provides the result as the source data for the `SELECT` statement. The result of the query is equivalent to the results of a `UNION ALL` query.

You can use the `_fromindex()` function to identify the source object (table or view) of each row in the result set of a `SELECT` statement whose `FROM` clause uses the @*<list_expression>* construct. The `_fromindex()` function evaluates to an int64. The position is zero-based, which means that `0` represents the first name, `1` the second, and so on.

For more information, see:

- "FROM Clauses", on page 273
- "Working with Lists", on page 313
- "_tablematch()", on page 334

Return Value

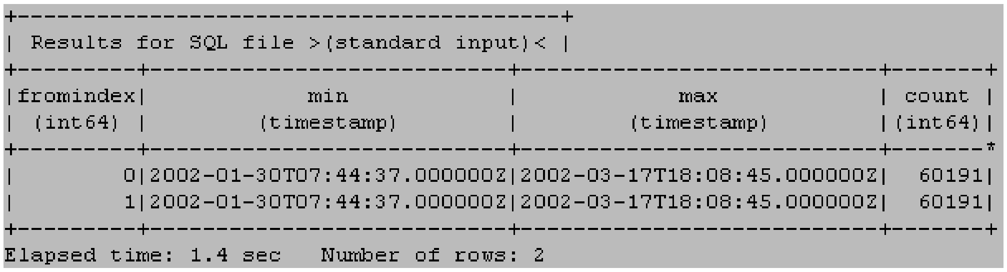
The `_fromindex()` function returns an `int64` with the 0-based index of the table that is being queried.

Examples

The following statement queries two tables and returns the earliest timestamp, the latest timestamp, and the number of records in each table, grouped by the position of the table in the result set. The position of the first table is represented as `0`.

```
SELECT _fromindex(), min(ts), max(ts), count(*)
  FROM @_list('example_webserv_100', 'example_syslog')
  GROUP BY 1
  DURING ALL;
```

The graphic below illustrates the results of this query.



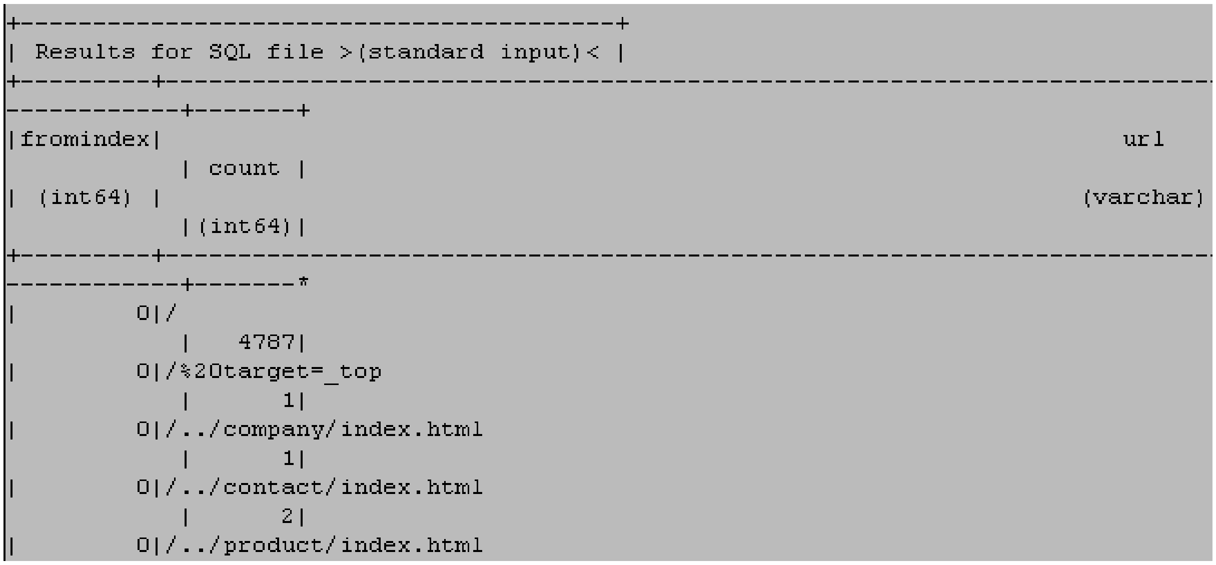
```
+----------------------------------------+
| Results for SQL file >(standard input)< |
+---------+---------------------------+---------------------------+-------+
|fromindex|            min            |            max            | count |
| (int64) |        (timestamp)        |        (timestamp)        |(int64)|
+---------+---------------------------+---------------------------+-------*
|        0|2002-01-30T07:44:37.000000Z|2002-03-17T18:08:45.000000Z|  60191|
|        1|2002-01-30T07:44:37.000000Z|2002-03-17T18:08:45.000000Z|  60191|
+---------+---------------------------+---------------------------+-------+
Elapsed time: 1.4 sec   Number of rows: 2
```

The following statement queries all tables in the default namespace and returns a count of each url in each table, grouped by the position of the table in the result set.

```
SELECT _fromindex(), url, count(*)
  FROM @_tablematch('.*')
  GROUP BY 1,2
  DURING ALL;
```

NOTE: In this case, the table names are not known, either at the time the query is written or in the result set. This query, which enables you to aggregate the results by table position rather than by the full result set, is useful for aggregating the result data when you do not need to know the table names.

The graphic below illustrates a few of the results of this query.



```
+------------------------------------------+
| Results for SQL file >(standard input)< |
+---------+----------------------------------------------------------------------
------------+-------+
|fromindex|                                                          url
            | count |
|  (int64) |                                                      (varchar)
            |(int64)|
+---------+----------------------------------------------------------------------
------------+-------*
|        0|/
            |   4787|
|        0|/%20target=_top
            |      1|
|        0|/../company/index.html
            |      1|
|        0|/../contact/index.html
            |      2|
|        0|/../product/index.html
```

CHAPTER 12

Perl Subroutines

HawkEye AP SQL can perform simple transformation and analysis of event-log data, but it has limitations in performing complex transformations, also known as *business logic*. The HawkEye AP SQL engine includes a Perl subsystem that lets you declare and use Perl code in your HawkEye AP SQL statements.

This chapter includes these sections:

About Perl Subroutines in HawkEye AP SQL

Perl subroutines can be declared as either *functions* or *aggregates*. Perl Functions are passed arguments for each row being processed, and they return a single value for each row. For example, `_strformat()` is an SQL function. You pass in arguments and it returns a string. With Perl functions, you can write your own routines like `_strformat()`.

Perl aggregates behave like SQL aggregation functions, such as `sum()` and `avg()`. Perl aggregates are passed arguments for each group created by a GROUP BY clause, and they return one result per group. Perl aggregates are useful for analyzing multiple records, such as when performing session analysis. For example, you might declare a Perl aggregate to detect suspicious user behavior. Use a GROUP BY clause to create a group for each user, with subgroups for each session. Then apply your Perl aggregate to match against suspicious patterns.

How Perl Processing Works

The basic purpose of the SQL engine is to consume streams of columns from a stream parser and send streams of columns to a stream formatter. Without displaying its processing to the user, the SQL engine rewrites calls to Perl functions to call a built-in function named `_perl()`. This function invokes the Perl interpreter once per column value.

For each query, the Perl engine is initialized, which compiles the Perl code and checks for errors. Assuming there are no errors, the Perl code is evaluated, registering the Perl functions and initializing any global variables.

For a Perl function, the SQL engine performs these steps

1 The SQL engine evaluates each parameter to the Perl function (per row).

2 For each argument, the value is converted into a `varchar` if necessary.

3 `The _perl()` SQL function is called with these arguments.

4 The result of the `_perl()` expression is a column of `varchar` values corresponding to the values returned by the Perl function.

For a Perl aggregate, the SQL engine performs these steps

1 The SQL engine saves up argument-records for each unique group (buffering).

2 When enough records have been buffered, `_perlagg()` is called with these arguments, similar to `_perl()`, including the conversion to `varchar` when necessary.

3 `The _perlagg()` SQL function loops through each argument record and invokes the Perl aggregate.

4 After all values for a group have been processed (not just the last value for this one buffer of records), the routine whose name is derived by adding `_final` to the name specified by the first argument to the `_perlagg()` expression is invoked.

5 The result of the `_perlagg()` expression is a column of `varchar` values that correspond to the values returned by the final Perl subroutine.

DECLARING PERL FUNCTIONS

You declare Perl functions as processing directives in a WITH clause. For example, the following SQL Select statement declares a Perl function named `maxarg()`, which returns the larger of two values.

```
-- declare a Perl function
WITH maxarg AS BUILTIN 'perl5' FUNCTION <<EOF
  sub maxarg {
    my($x,$y) = @_;

    if ($x > $y) {
      return $x;
    }

    return $y;
  }
EOF

-- use the Perl function in the query
SELECT maxarg( _strlen(Url), _strlen(Referrer) ),
      _strlen(Url),
      _strlen(Referrer)
  FROM example_webserv_100
  DURING ALL;
```

Because the Perl declaration spans lines, it is bounded by here document syntax `<<EOF ... EOF`. Here documents tell the SQL parser to treat the contents as a single `varchar` literal, including the newline characters.

For more information on declaring Perl functions, see “User-Defined Subroutines”, on page 306

Declaring Perl Aggregates

You declare Perl aggregates as processing directives in a WITH clause. Within a Perl aggregate declaration, you specify a pair of subroutines and global variables that the two subroutines can access. One subroutine manages the incremental state of groups in an aggregate, and the other subroutine returns the final aggregate value for a group. The second subroutine, which returns the final value, has the same name as the first, but with a suffix of `_final`. The SQL query engine calls each subroutine when appropriate.

The following SQL Select statement implements a custom Perl aggregate named `my_max()`:

```
WITH my_max AS BUILTIN 'perl5' AGGREGATE <<EOF
# -- global variables
my %state;

# -- incremental subroutine, called for each value in a group
sub my_max {
  my $call  = $_[0];  # -- is this the first call?
  my $group = $_[1];  # -- which group is the value part of?
  my $value = $_[2];  # -- what is the value being passed?

  if (!defined($state{$group})) {       # -- save the first value in the group
    $state{$group} = $value;
  } elsif ($value > $state{$group}) {  # -- or save a higher value in the group
      $state{$group} = $value;
  }
}

# -- final subroutine, called after the end of a group is reached
sub my_max_final {
  my $group = $_[1];
  return $state{$group};  # -- return the highest value from the group
}

EOF

SELECT ClientDNS, my_max(RespSize)
  FROM example_webserv_100
  GROUP BY 1
  DURING ALL;
```

The `my_max()` subroutine keeps track of the incremental state, and the `my_max_final()` `subroutine` returns the final value. Note that `my_max()` does not return any value; instead, it manages the global list variable `%state`.

The arguments to the custom Perl aggregate are saved in the global variables `$call`, `$group`, and `$value`. You reference arguments to custom Perl aggregates by their ordinal numbers.

The first aggregate argument, `$_[0]`, the *call* argument, indicates with the string values `first` and `other` whether this is the first invocation of the aggregate. Many custom Perl aggregates, like `my_max()`, do not use the call argument. The call argument has the string value `final` whenever the final subroutine is called.

The second aggregate argument, `$_[1]`, the *group* argument, indicates which values are combined in the GROUP BY clause. The third argument, $_[2], the *value* argument, has a value from a row in the group identified by `$_[1]` when the incremental subroutine is called.

The `%state` variable is a Perl hash table. This table receives its key values from the group values in `$_[1]`, which the Perl aggregate passes into the SQL statement. Each entry in the hash table represents one GROUP BY group. The subroutines access entries in %state with the expression `$state{$group}`.

IMPORTANT: Because the EDW uses parallel processing when aggregating data, when an aggregate function is called, the EDW creates multiple instances of the code. Each instance of the code is receives a subset of the group keys. Therefore, you cannot write functions that perform calculations *among* groups.

For more information on declaring Perl aggregates, see “User-Defined Subroutines”, on page 306.

Perl Execution Environment

This section describes these topics:

- "Exiting from Perl Subroutines", next
- "List Support and Perl Functions", on page 409
- "Using Macros in Perl Subroutines", on page 410
- "Understanding Parallelism and Side Effects", on page 411

Exiting from Perl Subroutines

The SQL query engine prohibits the use of the Perl `exit` operation. Subroutines and modules that include it may fail to load. Use the Perl `die` function instead. For example:

```
WITH Digit AS BUILTIN 'perl5' FUNCTION <<EOF

  my %digits = (
    "0" => "zero", "1" => "one",  "2" => "two", "3" => "three",
    "4" => "four", "5" => "five", "6" => "six", "7" => "seven",
    "8" => "eight", "9" => "nine"
  );

  sub Digit {
    return $digits{ $_[0] } || die "invalid value $_[0]";
  }

EOF
```

When a Perl function or aggregate calls `die` during query execution, an SQL processing exception with the message passed to `die` is raised, and the query terminates.

List Support and Perl Functions

Perl subroutines can accept lists as arguments. In doing so, the `@_` argument is an array filled with the list values. Perl subroutines can also return lists of values. Use the `addamark::setInto()` function to populate the entries in the returned list. The function has the following syntax:

```
addamark::setInto( <index>, <value> )
```

The `<value>` is stored in a temporary list variable, at the entry specified by `<index>`. The temporary list is returned as the INTO variable when the function is invoked with the INTO keyword:

```
<list_function>( <arguments> ) INTO <list_variable>
```

For more information on INTO variables, see "Working with Lists", on page 313.

Under normal evaluation, the following `test()` function returns its first argument. With the INTO keyword, the function also returns a list. The list contains three entries: the second argument, the third argument, and the constant `'three'`.

```
WITH Test AS BUILTIN 'perl5' FUNCTION <<EOF
  sub Test {

    # -- make the list available
    addamark::setInto(1, $_[1]);
    addamark::setInto(2, $_[2]);
    addamark::setInto(3, 'three');

    # -- return a single value
    return $_[0];
  }
EOF

SELECT TOP 1
    , foo[1]
    , foo[2]
    , foo[3]
    , Test('a', 'b', 'c') INTO foo
  FROM example_webserv_100
  DURING ALL;
```

The result of running this query is a four-column row containing the values 'b', 'c', 'three', and 'a':

```
+-------------------------------------------------+
| Results for SQL file >example-ref-perl-02.sql< |
+---------+---------+---------+-------------+----+
|  foo_1  |  foo_2  |  foo_3  |perl_into_foo|
|(varchar)|(varchar)|(varchar)|  (varchar)  |
+---------+---------+---------+-------------*
output is post-sorted
+---------+---------+---------+-------------*
|b        |c        |three    |a            |
+---------+---------+---------+-------------*
```

Using Macros in Perl Subroutines

To facilitate the sharing of information between Perl subroutines and SQL expressions, the SQL query engine defines global Perl variables for each expression macro declared in the SELECT statement. The names of these global variables have the following form:

```
$sql_<macro_identifier>
```

In the following statement, the `$sql_start` and `$sql_end` Perl variables correspond to the `$start` and `$end` expression macros used in the DURING clause:

```
WITH $start AS _timeadd( _now(), -1, 'hour' )
WITH $end as _now()

WITH Last2 AS BUILTIN 'perl5' AGGREGATE <<EOF
  my %state;
  sub Last2 {
    my $group = $_[1];
    my $value = $_[2];
    $state{$group} = $value;
  }
  sub Last2_final {
    my $group = $_[1];
    return "From $sql_start to $sql_end: $state{$group}";
  }
EOF

SELECT _perlagg('Last2', ClientDNS)
  FROM example_webserv_100
  DURING $start, $end


+------------------------------------------------+
| Results for SQL file >example-ref-perl-03.sql< |
+-------------------------------------------------------+
|                        perlagg                        |
|                       (varchar)                       |
+-------------------------------------------------------*
output is post-sorted
+-------------------------------------------------------*
|From 1012558131000000 to 1012561731000000: 212.9.190.79|
+-------------------------------------------------------*
```

For more information, see “Macros”, on page 302.

Understanding Parallelism and Side Effects

While a single `_perl()` expression is free to make use of side effects, the combination of the distributed nature of the SQL query engine and the column-wise evaluation order make it very difficult for two separate `_perl()` expressions to share state in a meaningful way.

For example, consider the following simple query:

```
WITH Count AS BUILTIN 'perl5' FUNCTION <<EOF
  my $count = 0;
  sub Count {
    $count++;
    return $count;
  }
EOF

SELECT _perl('Count') as cnt1, _perl('Count') as cnt2
  FROM Table
  DURING ALL;
```

When this query runs against a table that contains a few thousand rows spread across a three-host instance, the following results might be returned:

```
cnt1 cnt2
---- ----
1    24
1    322
1    366
10   33
10   331
10   375
100  421
100  465
1000 1300
1001 1301
```

```
1002 1302
1003 1303
101  422
101  466
  ...
```

The output is interesting for several reasons:

- Duplicate count values occur because the SQL engine distributes the query across then hosts in the EDW instance. Separate processes on each host have their own private copies of `$count`.
- The differences between values returned by the first and second `_perl()` expressions occur because the SQL query engine makes an arbitrary number of calls to the '`cnt1`' expression before it makes any calls to the '`cnt2`' expression.
- The actual number of column values processed at any given instant is based on network buffering and is not generally predictable.

Accessing External Modules

This section describes these topics to help you access external modules from within Perl subroutines in HawkEye AP SQL:

- "The use Directive", next
- "The @INC Variable", on page 413
- "The Inline.pm Perl Module", on page 413

The use Directive

The Perl `use` directive loads external Perl modules into the Perl interpreter that is embedded in the EDW. For example, the following statement declares a Perl function named `FormatBytes()` that formats numbers as bytes. The statement invokes the function on the sum of the `RespSize` column, which contains byte counts.

The statement uses the `Number::Format` Perl module to implement the custom Perl function. The Perl `use` directive makes the module available within the custom Perl function. Depending on the number of bytes in the expression `sum(RespSize)`, the result of `FormatBytes()` might be `744.6K`.

```
WITH FormatBytes AS BUILTIN 'perl5' FUNCTION <<EOF
  use Number::Format;  # -- access the external Perl module from the Perl library
  my $fmt = Number::Format->new;

  sub FormatBytes {
    return $fmt->format_bytes($_[0])
  }
EOF

SELECT _perl('FormatBytes', sum(RespSize) )
  FROM example_webserv_100
  DURING ALL;
```

NOTE: As an alternative, you can access Perl modules that have been installed in the Perl interpreter. For more information, see "Installing Perl Modules", on page 415.

The @INC Variable

The Perl `use` directive causes queries to fail if the Perl interpreter cannot load the specified module from one of the places identified by the `@INC` variable. The following query will show you the contents of the `@INC` variable:

```
WITH inc AS BUILTIN 'perl5' FUNCTION <<EOF
  sub inc {
    return join " ", @INC;
  }
EOF

SELECT TOP 1 _perl('inc')
  FROM example_webserv_100  -- use any table containing at least one record
  DURING ALL;
```

If `@INC` does not reflect where your Perl modules are located, use the `perldir=` directive in the `athttpd.conf` file to specify up to 36 additional places where perl modules may be found. Separate locations with colons (`:`). For example:

```
perldir=/usr/local/lib/perl5/site_perl:/usr/lib/perl5/site_perl
```

You can find the `athttpd.conf` file in this location:

<*SenSage_Home*>latest/etc/sls/instance/<*instance_name*>

The Inline.pm Perl Module

The `Inline.pm` module allows you to call functions written in other languages, including C, C++, and Java, from within your custom Perl subroutines. Languages like C/C++ provide greater performance than Perl, and they allow you to reuse existing code written in these other languages.

TESTING AND DEBUGGING PERL SUBROUTINES

This section describes these topics:

- "Running Perl Subroutines in Test Scripts", next
- "Printing Debugging Messages in Utility Logs", on page 414
- "Printing Debugging Messages to External Files", on page 415

Running Perl Subroutines in Test Scripts

You can test the code of your Perl routines by placing it in command line scripts. For example, to test the `Mean()` Perl aggregate, copy the body of the aggregate declaration as shown below to a file named `mean.pl`, and run the command `perl mean.pl`.

```
  Use Statistics::Descriptive;
  my %state;
  sub Mean {
    my $call  = $_[0];
    my $group = $_[1];
    my $value = $_[2];

    if ($call eq "first") {
      $state{$group} = Statistics::Descriptive::Full->new();
    }
    $state{$group}->add_data( $value );
    }

  sub Mean_final {
    my $group = $_[1];
    return $state{$group}->mean();
  }

  # simulate the operation of _perlagg()
  #
  Mean("first", 0, 1000);
  Mean("other", 0, 2000);
  Mean("other", 0, 3000);
  print Mean_final("final", 0), "\n");
```

The last six lines in the test-script version simulate how the SQL query engine invokes the final subroutine when your Perl aggregate runs within a SELECT statement.

Printing Debugging Messages in Utility Logs

Use the `addamark::dbgPrint()` Perl function to print debugging messages that appear in the log files of the `atquery` and `atload` EDW utilities. The debug printing function has the following syntax:

```
addamark::dbgPrint( <message> )
```

For example:

```
WITH sleeper AS BUILTIN 'perl5' FUNCTION <<EOF

  my $counter=0;

  sub sleeper {
    $counter++;
    if ($counter % 2 == 0) { sleep(1); }   # -- sleep a second every other call

    my $t = localtime();
    addamark::dbgPrint("$t: $counter");
  }
EOF
```

NOTE:

- The EDW runs the debug printing function in parallel on several computers at once, interleaving the messages in the results. This causes messages to appear in a different orders each time.
- If you have many debugging messages to print, it may be easier to write them to your own log files. For more information, see "Printing Debugging Messages to External Files", next.

Printing Debugging Messages to External Files

Use Perl `open()` and `print()` functions to print debugging messages to your own, external files. For example:

```
WITH MyCount AS BUILTIN 'perl5' function <<EOF
    my $count = 0;
    sub MyCount {
        open(LOG, ">>/tmp/log.$$.txt");
        $count++;
        print LOG  "Count is ",  $count, "\n" ;
        close(LOG);
        return $count;
    }
EOF
SELECT _perl('MyCount')
  FROM example_webserv_100
  DURING ALL;
```

Logfile Locations and Filenames

Generally, you should open and write debugging messages to files in the `/tmp` directory. This well-known directory exists on every host in an EDW instance. Perl subroutines run in parallel on each host in the instance, so each host will a version of your log file with a unique set of messages.

Generally, you should include the special symbols `$$` in the filenames of you log files. The symbols expand to the to current process ID. Embedding the process ID in filenames prevents multiple processes from overwriting the log files of each other.

Installing Perl Modules

The EDW embeds a fully-functional Perl interpreter, `atperl`, within its SQL query engine. The interpreter lets you use Perl code in HawkEye AP SQL Select statements while loading or querying the EDW. As with any Perl interpreter, you can install Perl modules into `atperl` to extend its functionality.

Perl modules are distributed in files with names like the following:

```
<module_name>.tar.gz
```

The EDW distribution includes these Perl modules:

Module	Distribution File	Description
`addamark::dbgPrint`		Pre-installed. Prints debugging messages in the log files of the `atload` and `atquery` commands.
`addamark::setInto`		Pre-installed. Supports implementing Perl subroutines that work with the INTO keyword. For more information, see Chapter 12: Perl Subroutines in the *Reporting Guide*.
`Archive::Tar`	`Archive-Tar-0.22.tar.gz`	Pre-installed. Load logs from `.tar` files.

Module	Distribution File	Description
`Archive::Zip`	`Archive-Zip-1.01.tar.gz`	Pre-installed. Load logs from `.zip` files.
`Compress::Zlib`	`Compress-Zlib-1.19.tar.gz`	Pre-installed. Archive loaded logs in `.tar` files.

In addition, you can download and install third-party Perl modules, such as the `Crypt::Rijndael` module for strong encryption. Rijndael is the U.S. government standard for strong encryption, designed to replace DES.

To install a Perl module in the EDW

1 Download or copy the distribution file to `/usr/local/src`.

2 Run `clsync` to copy the file to every host in the EDW cluster:

```
clsync <module_filename>.tar.gz
```

3 Run the following `clssh` command to install the module on every host in the EDW cluster:

```
clssh "cd /usr/local/src; \
  tar zxf <module_filename>; cd <module_filename>;
  ..<Sensage_Home>/latest/bin/atperl Makefile.PL; make; make install"
```

4 If the Perl module supports test, create a test by running the following `clssh` command:

```
clssh "cd //usr/local/src; \
  cd <module_filename>; \
  ..<Sensage_Home>/latest/bin/atperl make test"
```

NOTE: When you install Perl modules you must also have installed the GCC compiler.

CHAPTER 13

Using the DBD Driver

The DBD driver (DBD::Addamark) implements the DBI (Database Independent) API over the Perl API of the EDW. Only queries (that is, SQL SELECT statements) are supported.

This chapter contains these sections:

NOTE: For more information about the DBI API, including full documentation, visit:

http://search.cpan.org/~timb/

and click the DBI link under Registered Modules near the bottom of the page.

INSTALLATION

Requirements:

- Perl 5.x — (note that there is an issue with 5.8.2 as described in the Bundle::DBI `README`)
- DBI 1.32 or later — download the distribution file for the Bundle::DBI Perl module from:

  http://search.cpan.org/~timb/

  and click the DBI link under Registered Modules near the bottom of the page.
- DBD::Addamark — provided in the HawkEye AP software distribution in the following location: `<`*`SenSage_Home`*`>/latest`

Install the DBI and DBD modules according to the instructions for "Installing Perl Modules", on page 415.

INTERNATIONAL SUPPORT IN THE DBD DRIVER

The DBD driver does not support international characters in text data. The driver operates on ASCII text data only.

Sample DBI Program

```
#!/usr/bin/env perl
use strict;
use v5.6.0;
use DBD::Addamark;
#DBI->trace(3);

my $port = shift;
# print "$port\n";

my $namespace = shift;
# print "$namespace\n";

my $sqlfile = shift;
# print "$sqlfile\n";

my $sql = "";
open(FILE, $sqlfile) || die "could not open $sqlfile";

while (<FILE>) {
    $sql .= $_;
}
close(FILE);

my $dbh = DBI->connect('dbi:Addamark:host.domain.com:1234',
                       'username', 'password');
my $sth = $dbh->prepare("select * from syslog during all");
$sth->execute();

while(my $d = $sth->fetch) {
    print "@$d\n";
}
```

Explanation of the Sample DBI Program

Perl Code	Explanation
`#!/usr/bin/env perl`	A special syntax that tells the shell to run the rest of the script using Perl.

`use strict;`
`use v5.6.0;`
	Set up the Perl environment (v5.6.0) and use strict to restrict unsafe constructs and limit constructs to using Perl version 5.6.0.

`use DBD::Addamark;`	Specify that DBI should use the Addamark DBD driver.
`#DBI->trace(3);`	A comment line. Uncomment to set DBI tracing to level three. Trace level three traces DBI calls returning with results or errors, method entries with parameters and returning with results, and adds high-level information from the DBD driver and internal information from DBI.
`my $port = shift;`	Declares `$port` as a local variable and sets the value. The `$port` variable stores the TCP/IP port to be used when connecting to the database server.

Perl Code	Explanation
`# print "$port\n";`	A comment line. Uncomment to display the port number to be used.
`my $namespace = shift;`	Declares `$namespace` as a local variable and sets the value. The `$namespace` variable stores the namespace to be used when connecting to the database server.
`# print "$namespace\n";`	A comment line. Uncomment to display the Addamark namespace.
`my $sqlfile = shift;`	Declares `$sqlfile` as a local variable and sets the value. The `$sqlfile` variable stores the name of the SQL file. The SQL file contains the SQL statement or statements (query or queries) to send to the database server.
`# print "$sqlfile\n";`	A comment line. Uncomment to display the name of the sqlfile.
`my $sql = "";`	Declares `$sql` as a local variable and initializes (clears) the value. The `$sql` variable stores a SQL statement to be passed to the database server after the connection is made.

`open(FILE, $sqlfile) ||`
`    die "could not open $sqlfile";`
	Try to open the SQL file. If the file cannot be opened, exits the program and displays an error message saying, “could not open <*sqlfile*>”, where <*sqlfile*> is the name of the SQL file.

`while (<FILE>) {`
`    $sql .= $_;`
`}`
	Iteratively read the contents of the SQL file and puts the content into the `$_` global variable.

`close(FILE);`	Closes the SQL file.

`my $dbh =`
`    DBI->connect(`
`        'dbi:Addamark:' . $port,`
`        '', ''`
`    );`
	Declares `$dbh` as a local variable and sets the value to `DBI->connect('dbi:Addamark:' . $port, '', '')`, where port is the value previously stored in the `$port` variable. This entire variable is used to construct the database handle, which contains the information needed to connect with the database.

`$dbh->{addamark_tableNamespace} =`
`    $namespace;`
	The completely constructed database handle connects to the Addamark database in the addamark_tableNamespace. The connection to the database server uses the TCP/IP port contained in the `$port` variable, and DBI uses the DBD::Addamark driver.

`my $sth = $dbh->prepare( $sql );`	Declares `$sth` as a local variable, which is used to prepare the SQL statement from the SQL file for execution. A statement handle is returned, and that handle contains the SQL statement that may be executed later using the execute method.

`$sth->execute ||`
`    warn $sth->errstr;`
	Executes the SQL statement (from the SQL file) contained in the statement handle. If an error occurs, an error message is displayed, showing the errors from the last SQL statement executed.

`while(my $d = $sth->fetch) {`
`    print "@$d\n";`
`}`
	Iteratively retrieve the results from the query and stores them in the local variable `$d`, then displays the results, one to a line.

DBI ELEMENTS SUPPORTED BY HAWKEYE AP

DBD::Addamark supports most of the database and statement handle attributes; however, not all the attributes are relevant to this driver. As of DBI v.1.40, the supported attributes are:

- **Database handle attributes**
  - `AutoCommit`
  - `Driver`
  - `Name`
  - `Statement`
  - `RowCacheSize`
  - `Username`

  For more information about these attributes, see:

  http://search.cpan.org/~timb/DBI-1.40/DBI.pm#Database_Handle_Attributes

- **Statement handle attributes**
  - `NUM_OF_FIELDS`
  - `NUM_OF_PARAMS`
  - `NAME`
  - `NAME_lc`
  - `NAME_uc`
  - `NAME_hash`
  - `NAME_lc_hash`
  - `NAME_uc_hash`
  - `TYPE`
  - `CursorName`
  - `Database`
  - `Statement`

  For more information about these attributes, see

  http://search.cpan.org/~timb/DBI-1.40/DBI.pm#Statement_Handle_Attributes

UNSUPPORTED DBI FEATURES

The following DBI features *are not* supported by HawkEye AP:

- Question marks (?) as placeholders (and the ParamValues statement handle attribute will return “undef”) are not supported.
- The following database schema related methods are not supported:
  - `table_info()`
  - `column_info()`
  - `primary_key_info()`
  - `primary_key()`
  - `foreign_key_info()`
  - `tables()`
  - `type_info_all()`
  - `type_info()`
  - `get_info()`

  For more information about these methods, see

  http://search.cpan.org/~timb/DBI-1.40/DBI.pm#Database_Handle_Methods

HAWKEYE AP DBD ATTRIBUTES

In addition to the previously mentioned attributes, DBD::Addamark implements the following private driver attributes:

- addamark_tableNamespace
- addamark_rawOutputHandle
- addamark_dbgprintRequest
- addamark_oob_callback
- addamark_tableNamespace

addamark_tableNamespace

String, required. Sets the tableNamespace value in the request.

```
$dbh->{addamark_tableNamespace} = "default";
```

addamark_rawOutputHandle

IO handle, optional. Sets the `rawOutputHandle` for dumping internal request data. Useful for debugging. Must be specified when `addamark_dbgprintRequest` is enabled.

addamark_dbgprintRequest

Boolean, optional, default is “F” if not specified. When true, the request to the server is printed out, and the program terminates. Requires `addamark_rawOutputHandle` to be specified.

addamark_oob_callback

Subroutine reference, optional. A subroutine that is called with any OOB data sent back during the execution of the request. The subroutine should have the following structure:

```
sub oob_callback {
    my $code = shift;
    my $msg = shift;
    my $type = shift;
    my $data = shift;
    .... do any processing here ....
}
```

Use a callback subroutine to code progress meters, for example.

CHAPTER A

TIME ZONES

This appendix lists every time zone supported by the EDW and the HawkEye AP Console. When you enter a time value in HawkEye AP Console, use one of the time-zone formats listed below in “Supported Time Zones”, on page 423.

Because the EDW handles raw data from logs, which does not always use the supported time zones, it must accommodate some shortenings of time zone indicators found in that data (such as PDT for Pacific Daylight Time and CST for Central Standard Time). For more information, see “Time-Zone Conversion”, next.

TIME-ZONE CONVERSION

Sensage SQL provides functions that recognize specific time-zone shortenings and maps them to standard time-zone strings, as shown in the table below.

Shortened Time Zone	Standard Time Zone	Description
`PST`	`PST8PDT`	Pacific Standard Time and Pacific Daylight Time are interpreted as either standard time or daylight savings depending on the time of year of the actual date.
`PDT`	`PST8PDT`	
`MDT`	`MST7MDT`	Mountain Daylight Time is interpreted as either standard time or daylight savings depending on the time of year of the actual date.
`CST`	`CST6CDT`	Central Standard Time and Central Daylight Time are interpreted as either standard time or daylight savings depending on the time of year of the actual date.
`CDT`	`CST6CDT`	
`EDT`	`EST5EDT`	Eastern Daylight Time is interpreted as either standard time or daylight savings depending on the time of year of the actual date.

NOTE:

- Because Arizona does not support MDT, Mountain Standard Time is NOT converted to Mountain Daylight Time.
- Because Indiana does not consistently support EDT, Eastern Standard Time is NOT converted to Eastern Daylight Time.

SUPPORTED TIME ZONES

```
Africa/Addis_Ababa
Africa/Algiers
Africa/Asmera
Africa/Bangui
Africa/Blantyre
Africa/Brazzaville
Africa/Bujumbura
Africa/Cairo
```

```
Africa/Ceuta
Africa/Dar_es_Salaam
Africa/Djibouti
Africa/Douala
Africa/Gaborone
Africa/Harare
Africa/Johannesburg
Africa/Kampala
Africa/Khartoum
Africa/Kigali
Africa/Kinshasa
Africa/Lagos
Africa/Libreville
Africa/Luanda
Africa/Lubumbashi
Africa/Lusaka
Africa/Malabo
Africa/Maputo
Africa/Maseru
Africa/Mbabane
Africa/Mogadishu
Africa/Nairobi
Africa/Ndjamena
Africa/Niamey
Africa/Porto-Novo
Africa/Tripoli
Africa/Tunis
Africa/Windhoek
America/Adak
America/Anchorage
America/Anguilla
America/Antigua
America/Araguaina
America/Argentina/Buenos_Aires
America/Argentina/Catamarca
America/Argentina/ComodRivadavia
America/Argentina/Cordoba
America/Argentina/Jujuy
America/Argentina/La_Rioja
America/Argentina/Mendoza
America/Argentina/Rio_Gallegos
America/Argentina/San_Juan
America/Argentina/Tucuman
America/Argentina/Ushuaia
America/Aruba
America/Asuncion
America/Atka
America/Bahia
America/Barbados
America/Belem
America/Belize
America/Boa_Vista
America/Bogota
America/Boise
America/Buenos_Aires
America/Cambridge_Bay
America/Campo_Grande
America/Cancun
America/Caracas
```

```
America/Catamarca
America/Cayenne
America/Cayman
America/Chicago
America/Chihuahua
America/Coral_Harbour
America/Cordoba
America/Costa_Rica
America/Cuiaba
America/Curacao
America/Dawson
America/Dawson_Creek
America/Denver
America/Detroit
America/Dominica
America/Edmonton
America/Eirunepe
America/El_Salvador
America/Ensenada
America/Fort_Wayne
America/Fortaleza
America/Glace_Bay
America/Godthab
America/Goose_Bay
America/Grand_Turk
America/Grenada
America/Guadeloupe
America/Guatemala
America/Guayaquil
America/Guyana
America/Halifax
America/Havana
America/Hermosillo
America/Indiana/Indianapolis
America/Indiana/Knox
America/Indiana/Marengo
America/Indiana/Vevay
America/Indianapolis
America/Inuvik
America/Iqaluit
America/Jamaica
America/Jujuy
America/Juneau
America/Kentucky/Louisville
America/Kentucky/Monticello
America/Knox_IN
America/La_Paz
America/Lima
America/Los_Angeles
America/Louisville
America/Maceio
America/Managua
America/Manaus
America/Martinique
America/Mazatlan
America/Mendoza
America/Menominee
America/Merida
America/Mexico_City
```

```
America/Miquelon
America/Monterrey
America/Montevideo
America/Montreal
America/Montserrat
America/Nassau
America/New_York
America/Nipigon
America/Nome
America/Noronha
America/North_Dakota/Center
America/Panama
America/Pangnirtung
America/Paramaribo
America/Phoenix
America/Port-au-Prince
America/Port_of_Spain
America/Porto_Acre
America/Porto_Velho
America/Puerto_Rico
America/Rainy_River
America/Rankin_Inlet
America/Recife
America/Regina
America/Rio_Branco
America/Rosario
America/Santiago
America/Santo_Domingo
America/Sao_Paulo
America/Shiprock
America/St_Johns
America/St_Kitts
America/St_Lucia
America/St_Thomas
America/St_Vincent
America/Swift_Current
America/Tegucigalpa
America/Thule
America/Thunder_Bay
America/Tijuana
America/Toronto
America/Tortola
America/Vancouver
America/Virgin
America/Whitehorse
America/Winnipeg
America/Yakutat
America/Yellowknife
Antarctica/Casey
Antarctica/Davis
Antarctica/DumontDUrville
Antarctica/Mawson
Antarctica/McMurdo
Antarctica/Palmer
Antarctica/Rothera
Antarctica/South_Pole
Antarctica/Syowa
Antarctica/Vostok
Arctic/Longyearbyen
```

```
Asia/Aden
Asia/Almaty
Asia/Amman
Asia/Anadyr
Asia/Aqtau
Asia/Aqtobe
Asia/Ashgabat
Asia/Ashkhabad
Asia/Baghdad
Asia/Bahrain
Asia/Baku
Asia/Bangkok
Asia/Beirut
Asia/Bishkek
Asia/Brunei
Asia/Calcutta
Asia/Choibalsan
Asia/Chongqing
Asia/Chungking
Asia/Colombo
Asia/Dacca
Asia/Damascus
Asia/Dhaka
Asia/Dili
Asia/Dubai
Asia/Dushanbe
Asia/Gaza
Asia/Harbin
Asia/Hong_Kong
Asia/Hovd
Asia/Irkutsk
Asia/Istanbul
Asia/Jakarta
Asia/Jayapura
Asia/Jerusalem
Asia/Kabul
Asia/Kamchatka
Asia/Karachi
Asia/Kashgar
Asia/Katmandu
Asia/Krasnoyarsk
Asia/Kuala_Lumpur
Asia/Kuching
Asia/Kuwait
Asia/Macao
Asia/Macau
Asia/Magadan
Asia/Makassar
Asia/Manila
Asia/Muscat
Asia/Nicosia
Asia/Novosibirsk
Asia/Omsk
Asia/Oral
Asia/Phnom_Penh
Asia/Pontianak
Asia/Pyongyang
Asia/Qatar
Asia/Qyzylorda
```

```
Asia/Rangoon
Asia/Riyadh
Asia/Riyadh87
Asia/Riyadh88
Asia/Riyadh89
Asia/Saigon
Asia/Sakhalin
Asia/Samarkand
Asia/Seoul
Asia/Shanghai
Asia/Singapore
Asia/Taipei
Asia/Tashkent
Asia/Tbilisi
Asia/Tehran
Asia/Tel_Aviv
Asia/Thimbu
Asia/Thimphu
Asia/Tokyo
Asia/Ujung_Pandang
Asia/Ulaanbaatar
Asia/Ulan_Bator
Asia/Urumqi
Asia/Vientiane
Asia/Vladivostok
Asia/Yakutsk
Asia/Yekaterinburg
Asia/Yerevan
Atlantic/Bermuda
Atlantic/Canary
Atlantic/Cape_Verde
Atlantic/Faeroe
Atlantic/Jan_Mayen
Atlantic/Madeira
Atlantic/South_Georgia
Atlantic/Stanley
Australia/ACT
Australia/Adelaide
Australia/Brisbane
Australia/Broken_Hill
Australia/Canberra
Australia/Currie
Australia/Darwin
Australia/Hobart
Australia/LHI
Australia/Lindeman
Australia/Lord_Howe
Australia/Melbourne
Australia/NSW
Australia/North
Australia/Perth
Australia/Queensland
Australia/South
Australia/Sydney
Australia/Tasmania
Australia/Victoria
Australia/West
Australia/Yancowinna
Brazil/Acre
```

```
Brazil/DeNoronha
Brazil/East
Brazil/West
CET
CST6CDT
Canada/Atlantic
Canada/Central
Canada/East-Saskatchewan
Canada/Eastern
Canada/Mountain
Canada/Newfoundland
Canada/Pacific
Canada/Saskatchewan
Canada/Yukon
Chile/Continental
Chile/EasterIsland
Cuba
EET
EST
EST5EDT
Egypt
Eire
Europe/Amsterdam
Europe/Andorra
Europe/Athens
Europe/Belfast
Europe/Belgrade
Europe/Berlin
Europe/Bratislava
Europe/Brussels
Europe/Bucharest
Europe/Budapest
Europe/Chisinau
Europe/Copenhagen
Europe/Dublin
Europe/Gibraltar
Europe/Helsinki
Europe/Istanbul
Europe/Kaliningrad
Europe/Kiev
Europe/Lisbon
Europe/Ljubljana
Europe/London
Europe/Luxembourg
Europe/Madrid
Europe/Malta
Europe/Mariehamn
Europe/Minsk
Europe/Monaco
Europe/Moscow
Europe/Nicosia
Europe/Oslo
Europe/Paris
Europe/Prague
Europe/Riga
Europe/Rome
Europe/Samara
Europe/San_Marino
Europe/Sarajevo
```

```
Europe/Simferopol
Europe/Skopje
Europe/Sofia
Europe/Stockholm
Europe/Tallinn
Europe/Tirane
Europe/Tiraspol
Europe/Uzhgorod
Europe/Vaduz
Europe/Vatican
Europe/Vienna
Europe/Vilnius
Europe/Warsaw
Europe/Zagreb
Europe/Zaporozhye
Europe/Zurich
GB
GB-Eire
GMT
HST
Hongkong
Indian/Antananarivo
Indian/Chagos
Indian/Christmas
Indian/Cocos
Indian/Comoro
Indian/Kerguelen
Indian/Mahe
Indian/Maldives
Indian/Mauritius
Indian/Mayotte
Indian/Reunion
Iran
Israel
Jamaica
Japan
Kwajalein
Libya
MET
MST
MST7MDT
Mexico/BajaNorte
Mexico/BajaSur
Mexico/General
Mideast/Riyadh87
Mideast/Riyadh88
Mideast/Riyadh89
NZ
NZ-CHAT
Navajo
PRC
PST8PDT
Pacific/Apia
Pacific/Auckland
Pacific/Chatham
Pacific/Easter
Pacific/Efate
Pacific/Enderbury
Pacific/Fakaofo
```

```
Pacific/Fiji
Pacific/Funafuti
Pacific/Galapagos
Pacific/Gambier
Pacific/Guadalcanal
Pacific/Guam
Pacific/Honolulu
Pacific/Johnston
Pacific/Kiritimati
Pacific/Kosrae
Pacific/Kwajalein
Pacific/Majuro
Pacific/Marquesas
Pacific/Midway
Pacific/Nauru
Pacific/Niue
Pacific/Norfolk
Pacific/Noumea
Pacific/Pago_Pago
Pacific/Palau
Pacific/Pitcairn
Pacific/Ponape
Pacific/Port_Moresby
Pacific/Rarotonga
Pacific/Saipan
Pacific/Samoa
Pacific/Tahiti
Pacific/Tarawa
Pacific/Tongatapu
Pacific/Truk
Pacific/Wake
Pacific/Wallis
Pacific/Yap
Poland
Portugal
ROK
Singapore
Turkey
US/Alaska
US/Aleutian
US/Arizona
US/Central
US/East-Indiana
US/Eastern
US/Hawaii
US/Indiana-Starke
US/Michigan
US/Mountain
US/Pacific
US/Samoa
UTC
W-SU
WET
```

INDEX