



TIBCO Hawk[®]

WebConsole User's Guide

*Software Release 6.2
September 2019*



Important Information

SOME TIBCO SOFTWARE EMBEDS OR BUNDLES OTHER TIBCO SOFTWARE. USE OF SUCH EMBEDDED OR BUNDLED TIBCO SOFTWARE IS SOLELY TO ENABLE THE FUNCTIONALITY (OR PROVIDE LIMITED ADD-ON FUNCTIONALITY) OF THE LICENSED TIBCO SOFTWARE. THE EMBEDDED OR BUNDLED SOFTWARE IS NOT LICENSED TO BE USED OR ACCESSED BY ANY OTHER TIBCO SOFTWARE OR FOR ANY OTHER PURPOSE.

USE OF TIBCO SOFTWARE AND THIS DOCUMENT IS SUBJECT TO THE TERMS AND CONDITIONS OF A LICENSE AGREEMENT FOUND IN EITHER A SEPARATELY EXECUTED SOFTWARE LICENSE AGREEMENT, OR, IF THERE IS NO SUCH SEPARATE AGREEMENT, THE CLICKWRAP END USER LICENSE AGREEMENT WHICH IS DISPLAYED DURING DOWNLOAD OR INSTALLATION OF THE SOFTWARE (AND WHICH IS DUPLICATED IN THE LICENSE FILE) OR IF THERE IS NO SUCH SOFTWARE LICENSE AGREEMENT OR CLICKWRAP END USER LICENSE AGREEMENT, THE LICENSE(S) LOCATED IN THE "LICENSE" FILE(S) OF THE SOFTWARE. USE OF THIS DOCUMENT IS SUBJECT TO THOSE TERMS AND CONDITIONS, AND YOUR USE HEREOF SHALL CONSTITUTE ACCEPTANCE OF AND AN AGREEMENT TO BE BOUND BY THE SAME.

This document contains confidential information that is subject to U.S. and international copyright laws and treaties. No part of this document may be reproduced in any form without the written authorization of TIBCO Software Inc.

TIBCO, the TIBCO logo, the TIBCO O logo, TIB, Information Bus, ActiveMatrix BusinessWorks, Enterprise Message Service, Hawk, Rendezvous, TIBCO Administrator, TIBCO Designer, and TIBCO Runtime Agent are either registered trademarks or trademarks of TIBCO Software Inc. in the United States and/or other countries.

Java and all Java based trademarks and logos are trademarks or registered trademarks of Oracle Corporation and/or its affiliates.

All other product and company names and marks mentioned in this document are the property of their respective owners and are mentioned for identification purposes only.

THIS DOCUMENT IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT.

THIS DOCUMENT COULD INCLUDE TECHNICAL INACCURACIES OR TYPOGRAPHICAL ERRORS. CHANGES ARE PERIODICALLY ADDED TO THE INFORMATION HEREIN; THESE CHANGES WILL BE INCORPORATED IN NEW EDITIONS OF THIS DOCUMENT. TIBCO SOFTWARE INC. MAY MAKE IMPROVEMENTS AND/OR CHANGES IN THE PRODUCT(S) AND/OR THE PROGRAM(S) DESCRIBED IN THIS DOCUMENT AT ANY TIME.

THE CONTENTS OF THIS DOCUMENT MAY BE MODIFIED AND/OR QUALIFIED, DIRECTLY OR INDIRECTLY, BY OTHER DOCUMENTATION WHICH ACCOMPANIES THIS SOFTWARE, INCLUDING BUT NOT LIMITED TO ANY RELEASE NOTES AND "READ ME" FILES.

This and other products of TIBCO Software Inc. may be covered by registered patents. Please refer to TIBCO's Virtual Patent Marking document (<https://www.tibco.com/patents>) for details.

Copyright © 1996-2019. TIBCO Software Inc. All Rights Reserved.

Contents

Figures	vii
Tables	ix
Preface	xi
Changes from the Previous Release of this Guide	xii
Related Documentation	xiii
TIBCO Hawk Documentation	xiii
Other TIBCO Product Documentation	xiv
Typographical Conventions	xv
TIBCO Product Documentation and Support Services	xvii
How to Access TIBCO Documentation	xvii
How to Contact TIBCO Support	xvii
How to Join TIBCO Community	xvii
Chapter 1 TIBCO Hawk WebConsole	1
Overview	2
Starting TIBCO Hawk WebConsole	3
Stopping TIBCO Hawk WebConsole And Hawk H2 Database	4
Shutting TIBCO Hawk WebConsole	4
Shutting TIBCO Hawk H2 Database	4
Chapter 2 Working with Hawk Dashboard	5
Overview	6
Agents Portlet	7
Agent HeatMap Portlet	9
HeatMap Hierarchy	9
Repository Portlet	11
Alerts Portlet	12
Dashboard Options	13
Goto-Dashboard	13
Save UI State	13
About	13

Chapter 3 Working with Alert Messages	15
Overview	16
Viewing Alert Messages	17
Suspending Alert Messages	21
Viewing Alert Chart	23
Viewing Warning and Exceptions Messages	25
Chapter 4 Working with Rulebases	27
Overview	28
Sample Rulebase	28
Rulebase Screen Details	29
Creating a Rulebase	32
Using Advanced Rulebase Features	34
Creating a Rule	36
Creating a Test	39
Building Compound Tests	43
Using Advanced Test Features	44
Creating Actions	46
Consequence Action	46
Clear Action	52
Referencing Variables in a Rulebase	54
Referencing External Variables	54
Referencing Internal Variables	55
Referencing Data Source Variables	56
How Variable Substitution Affects Actions	56
Using Posted Conditions	57
Chapter 5 Working with Schedules and Period Groups	58
Overview	59
Sample Schedule and Period Groups	59
Schedules Screen Details	60
Creating Schedules	64
Creating Period Groups	70
Testing Schedules	73
Applying Schedules to Rulebase Objects	75
Chapter 6 Working with Microagents	76
Overview	77

Viewing Microagents and Invoking a Microagent Method	78
Subscribing to a Microagent Method	82
Viewing Subscription Results	83
Moving Subscription Results Window to Custom Dashboards	85
Chapter 7 Working with Network Query	86
Performing Network Queries	87
Performing Network Actions	92
Chapter 8 Working with Rulebase-Maps	94
Overview	95
Sample Rulebase-Map	95
Rulebase-Map Screen Details	96
Creating a Rulebase-Map	99
Creating User-defined Agent Groups	100
Mapping Groups to Agents or Groups	102
Mapping Rulebases to Agents or Groups	104
Mapping Groups to External Commands	106
Distribute Rulebase Map to Repositories	108
Chapter 9 Common Features for Configurable Objects	109
Database Persistence	110
Synchronized, Dirty, and Local States	111
Notification Area	113
Index	115

Figures

Figure 1	Agents Portlet.	7
Figure 2	Agents Portlet.	8
Figure 3	Agent Heatmap Portlet.	9
Figure 4	Repository Portlet.	11
Figure 5	Alert List Tab	17
Figure 6	Detail Information Panel.	19
Figure 7	Chart Tab	24
Figure 8	Console Warning and Exceptions Window	25
Figure 9	Toolbar	29
Figure 10	Rulebase Editing Screen	33
Figure 11	Rule Editing Screen	36
Figure 12	Test Editing Screen	39
Figure 13	Compound Test Fields.	43
Figure 14	Advanced Options	44
Figure 15	Action Editing Screen.	46
Figure 16	Toolbar	60
Figure 17	Microagent Info Screen	79
Figure 18	Invocation Results Window	80
Figure 19	Subscription Results Window - Data Grid View	82
Figure 20	Subscription Results Window - Charts View	84
Figure 21	Rulebase-Map Toolbar.	97
Figure 22	Notification Area.	113

Tables

Table 1	General Typographical Conventions	xv
Table 2	Toolbar icons	29
Table 3	Test Operators For Numeric Method Results.	40
Table 4	Test Operators for Text String Results.	42
Table 5	Test Operators for Boolean Results	42
Table 6	Action Types in the Action Editor	47
Table 7	Toolbar icons	61

Preface

This manual describes the functionality of TIBCO Hawk[®] WebConsole, a web based tool for monitoring and managing applications and operating systems.

Topics

- [Changes from the Previous Release of this Guide, page xii](#)
- [Related Documentation, page xiii](#)
- [Typographical Conventions, page xv](#)
- [TIBCO Product Documentation and Support Services, page xvii](#)

Changes from the Previous Release of this Guide

There are no changes to this guide since the previous release.

Related Documentation

This section lists documentation resources you may find useful.

TIBCO Hawk Documentation

The following documents form the TIBCO Hawk documentation set:

- *TIBCO Hawk Release Notes*: Read the release notes for a list of new and changed features. This document also contains lists of known issues and closed issues for this release.
- *TIBCO Hawk Concepts*: This manual includes basic descriptions of TIBCO Hawk concepts.
- *TIBCO Hawk Installation, Configuration, and Administration*: Read this book first. It contains step-by-step instructions for installing TIBCO Hawk software on various operating system platforms. It also describes how to configure the software for specific applications, once it is installed. An installation FAQ is included.
- *TIBCO Hawk Methods Reference*: A reference to the microagents and methods used by a TIBCO Hawk Agent for system and application monitoring.
- *TIBCO Hawk WebConsole User's Guide*: This manual includes complete instructions for using TIBCO Hawk WebConsole.
- *TIBCO Hawk Programmer's Guide*: All programmers should read this manual. It contains detailed descriptions of Application Management Interface (AMI), Application Programming Interface (API) concepts, and the TIBCO Hawk security framework and its classes. It also contains detailed descriptions of each class and method for the following APIs:
 - AMI API
 - Java, C++ and C API
 - Console API
 - Java API
 - Configuration Object API
 - Java API

Programmers should refer to the appropriate language reference sections for the AMI API details. The TIBCO Hawk Application Management Interface (AMI) exposes internal application methods to TIBCO Hawk.

- *TIBCO Hawk Plug-in Reference Guide*: Contains details about the Enterprise Message Service, Messaging and JVM microagents methods that are used to administer and monitor the TIBCO Enterprise Message Service server.
- *TIBCO Hawk Plug-ins for TIBCO Administrator*: Contains detailed descriptions of the TIBCO Hawk plug-ins accessed via TIBCO Administrator.
- *TIBCO Hawk HTTP Adapter User's Guide*: Contains information about performing discovery, monitoring of agent status, monitoring of agent alerts, method invocation, method subscription, and many more activities on TIBCO Hawk and third-party products.
- *TIBCO Hawk Admin Agent Guide*: Contains basic configuration details for TIBCO Hawk Admin Agent and complete instructions for using the web interface of TIBCO Enterprise Administrator for TIBCO Hawk.
- *TIBCO Hawk Security Guide*: Provides guidelines to ensure security within the components of TIBCO Hawk and within the communication channels between the components.

Other TIBCO Product Documentation

You may find it useful to read the documentation for the following TIBCO products:

- TIBCO® Enterprise Administrator
- TIBCO ActiveSpaces®
- TIBCO Rendezvous®
- TIBCO Enterprise Message Service™

Typographical Conventions

The following typographical conventions are used in this manual.

Table 1 General Typographical Conventions

Convention	Use
<i>ENV_HOME</i>	<p>TIBCO products are installed into an installation environment. A product installed into an installation environment does not access components in other installation environments. Incompatible products and multiple instances of the same product must be installed into different installation environments.</p> <p>An installation environment consists of the following properties:</p> <ul style="list-style-type: none"> • Name Identifies the installation environment. This name is referenced in documentation as <i>ENV_NAME</i>. On Microsoft Windows, the name is appended to the name of Windows services created by the installer and is a component of the path to the product shortcut in the Windows Start > All Programs menu. • Path The folder into which the product is installed. This folder is referenced in documentation as <i>TIBCO_HOME</i>. <p>TIBCO Hawk installs into a directory within a <i>TIBCO_HOME</i>. This directory is referenced in documentation as <i>HAWK_HOME</i>. The default value of <i>HAWK_HOME</i> depends on the operating system. For example on Windows systems, the default value is <code>C:\tibco\hawk\6.0</code>.</p> <p>A TIBCO Hawk configuration folder stores configuration data generated by TIBCO Hawk. Configuration data can include sample scripts, session data, configured binaries, logs, and so on. This folder is referenced in documentation as <i>CONFIG_FOLDER</i>. For example, on Windows systems, the default value is <code>C:\ProgramData\tibco\cfgmgmt\hawk</code>.</p>
<i>TIBCO_HOME</i>	
<i>HAWK_HOME</i>	
<i>CONFIG_FOLDER</i>	
code font	<p>Code font identifies commands, code examples, filenames, pathnames, and output displayed in a command window. For example:</p> <p>Use MyCommand to start the foo process.</p>

Table 1 General Typographical Conventions (Cont'd)

Convention	Use
bold code font	<p>Bold code font is used in the following ways:</p> <ul style="list-style-type: none"> • In procedures, to indicate what a user types. For example: Type admin. • In large code samples, to indicate the parts of the sample that are of particular interest. • In command syntax, to indicate the default parameter for a command. For example, if no parameter is specified, MyCommand is enabled: MyCommand [enable disable]
<i>italic font</i>	<p>Italic font is used in the following ways:</p> <ul style="list-style-type: none"> • To indicate a document title. For example: See <i>TIBCO BusinessWorks Concepts</i>. • To introduce new terms. For example: A portal page may contain several portlets. <i>Portlets</i> are mini-applications that run in a portal. • To indicate a variable in a command or code syntax that you must replace. For example: MyCommand <i>pathname</i>
Key combinations	<p>Key name separated by a plus sign indicate keys pressed simultaneously. For example: Ctrl+C.</p> <p>Key names separated by a comma and space indicate keys pressed one after the other. For example: Esc, Ctrl+Q.</p>
	The note icon indicates information that is of special interest or importance, for example, an additional action required only in certain circumstances.
	The tip icon indicates an idea that could be useful, for example, a way to apply the information provided in the current section to achieve a specific result.
	The warning icon indicates the potential for a damaging situation, for example, data loss or corruption if certain steps are taken or not taken.

TIBCO Product Documentation and Support Services

For information about this product, you can read the documentation, contact TIBCO Support, and join TIBCO Community.

How to Access TIBCO Documentation

Documentation for TIBCO products is available on the TIBCO Product Documentation website mainly in the HTML and PDF formats.

The TIBCO Product Documentation website is updated frequently and is more current than any other documentation included with the product. To access the latest documentation, visit <https://docs.tibco.com>.

Documentation for TIBCO Hawk is available on the [TIBCO Hawk Product Documentation](#) page.

How to Contact TIBCO Support

You can contact TIBCO Support in the following ways:

- For an overview of TIBCO Support, visit <https://www.tibco.com/services/support>.
- For accessing the Support Knowledge Base, viewing the latest product updates that were not available at the time of the release, and getting personalized content about products you are interested in, visit the TIBCO Support portal at <https://support.tibco.com>.
- For creating a Support case, you must have a valid maintenance or support contract with TIBCO. You also need a user name and password to log in to <https://support.tibco.com>. If you do not have a user name, you can request one by clicking **Register** on the website.

How to Join TIBCO Community

TIBCO Community is the official channel for TIBCO customers, partners, and employee subject matter experts to share and access their collective experience. TIBCO Community offers access to Q&A forums, product wikis, and best practices. It also offers access to extensions, adapters, solution accelerators, and tools that extend and enable customers to gain full value from TIBCO products. In addition, users can submit and vote on feature requests from within the [TIBCO Ideas Portal](#). For a free registration, go to <https://community.tibco.com>.

Chapter 1

TIBCO Hawk WebConsole

The TIBCO Hawk WebConsole application is a web application that provides a central view of all the distributed components interacting within TIBCO Hawk System. It is a pictorial view of each of the infrastructure component that is being monitored with the help of Hawk components.

Topics

- [Overview, page 2](#)
- [Starting TIBCO Hawk WebConsole, page 3](#)
- [Stopping TIBCO Hawk WebConsole And Hawk H2 Database, page 4](#)

Overview

TIBCO Hawk WebConsole application enables you to:

- View the statistical data about events, alerts and actions on all the infrastructure nodes that are being monitored, in the entire network as an easy-to-understand color-coded heatmap.
- Allows you to administer Rulebases, Rules and specify appropriate actions to be taken if the rules are triggered upon satisfying the threshold conditions.
- Create your own sets of customized dashboards, without affecting other users.

It provides ability to view multiple Hawk Domains. See the `readme.txt` file available on <https://docs.tibco.com> for list of supported web browsers and their versions.



The recommended screen resolution for Hawk WebConsole is 1280 x 1024.

Starting TIBCO Hawk WebConsole

- Windows** To start TIBCO Hawk WebConsole on Microsoft Windows:
1. Browse to the `HAWK_HOME\webconsole` folder and double-click `tibhawk2db.exe` to start the database.
 2. In the same folder, double-click `startwebconsole.bat` to start the Hawk WebConsole.
 3. After you start the Hawk WebConsole, in your browser's address box, enter a URL of the following format:
`http://<hostname>:<port number>/hawkwebconsole`
where, the default `<port number>` is 8080.
For example, `http://localhost:8080/hawkwebconsole`
In the login window, enter a valid username and password. The default credentials are:
 - Username—**admin**
 - Password—**admin**

- UNIX** To start Hawk WebConsole on UNIX:
1. Open the command prompt and go to the `HAWK_HOME/webconsole` folder.
 2. Run the following scripts:
 - `tibhawk2db`
 - `startwebconsole.sh`
 3. After you start the Hawk WebConsole, in your browser's address box, enter a URL of the following format:
`http://<hostname>:<port number>/hawkwebconsole`
For example, `http://10.97.123.83:8080/hawkwebconsole`

Stopping TIBCO Hawk WebConsole And Hawk H2 Database

You can shut down TIBCO Hawk WebConsole as well as TIBCO Hawk H2 Database using the script/executables provided with TIBCO Hawk.

Shutting TIBCO Hawk WebConsole

Windows To stop TIBCO Hawk WebConsole on Microsoft Windows, browse to the `HAWK_HOME\webconsole` folder and double-click `stopwebconsole.bat`.

UNIX To stop Hawk WebConsole on UNIX:

1. Open the command prompt and go to the `HAWK_HOME/webconsole` folder.
2. Run the `stopwebconsole.sh` script.

Shutting TIBCO Hawk H2 Database

To stop the database on all platforms:

1. Go to the `HAWK_HOME\webconsole` folder and open the `tibhawkh2db.tra` file, in a text editor, for editing.
2. Comment out the code which starts the Hawk H2 database and uncomment the code which stops the Hawk H2 database. Following is the sample code (to stop the database) for the section in the `tibhawkh2db.tra` file, that allows to start/stop the Hawk H2 database:

```

...
...
#
# Specifies the remaining arguments to pass into the application
# For Starting H2 Database, use the following
# application.args -tcp -web -webAllowOthers

# For Shutting down H2 Database, comment above line and uncomment
the following
application.args -tcpShutdown tcp://localhost:9092"
#
...
...

```

3. Save and close the `tibhawkh2db.tra` file.
4. Now run `tibhawkh2db` to stop the database.

Chapter 2 **Working with Hawk Dashboard**

This section discusses the Hawk Dashboard features.

Topics

- [Overview, page 6](#)
- [Agent HeatMap Portlet, page 9](#)
- [Repository Portlet, page 11](#)
- [Alerts Portlet, page 12](#)
- [Dashboard Options, page 13](#)

Overview

When you login to the TIBCO Hawk WebConsole, it automatically discovers machines running TIBCO Hawk agents on the network. The **Agent** portlet on the Hawk Dashboard list all the agents. By default, agents are clustered in tabular form and grouped by the Hawk domains.

Each agent has a set of default microagents, that are discovered by agent when it is started. If you install and start an adapter or gateway, or instrument an application with AMI, microagents for these objects are dynamically added to the agent. In TIBCO Hawk WebConsole, you can view microagents and their methods for any discovered TIBCO Hawk agents. For more details, see *TIBCO Hawk Concepts Guide*.

The Hawk Dashboard screen is divided into four default portlets.

- Agents
- Agent HeatMap
- Repository
- Alerts

The following sections provide details about these portlets.

Agents Portlet

The Agents portlet displays all the agents grouped by domains in tabular view.

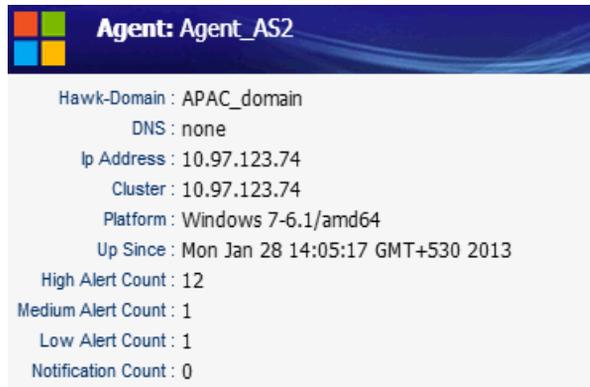
Figure 1 Agents Portlet



OS	Hawk Domain	Agent Name	Cluster	Status
APAC_domain				
	APAC_domain	Agent_AS2	10.97.123.74	Alive
	APAC_domain	Agent_AS3	10.97.112.74	Alive
	APAC_domain	Agent3	10.97.123.0	Alive
	APAC_domain	Agent4	10.97.123.0	Offline
	APAC_domain	Agent_AS1	10.97.123.0	Alive
				1 Offline, 4 Alive

- Each row represents the following details for each agent:
 - Operating System of the agent (OS)
 - Hawk Domain of the agent (Hawk Domain)
 - Agent Name
 - Cluster
 - Status: The "Alive" and "Offline" status of the agents are displayed.
- The filter options for the agent list are as follows:
 - Filter-as-you-type on every column.
 - For every column you can sort the table in ascending or descending order.
 - You can auto fit columns.
 - Group by the column.
- Grouping is intact even if the sort order is changed.
- A brief description of every agent is displayed when you hover over the agents in the **Agents** portlet

Figure 2 Agents Portlet

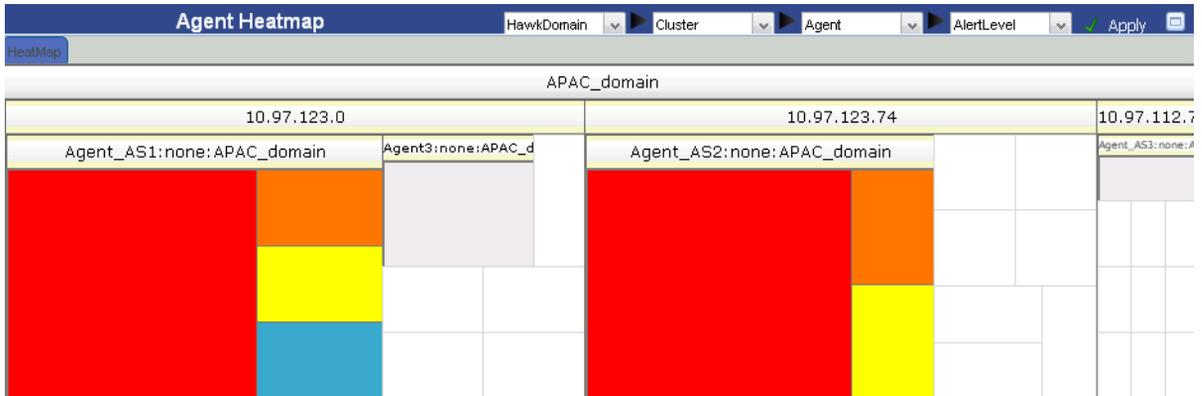


- **Agents** portlet status bar displays a summary of "Alive" and "Offline" agents count.
- Clicking on an agent that is "Alive", a new **Agent** tab is opened. The name of the **Agent** tab is same as the selected agent name *<agent_name>*.

Agent HeatMap Portlet

HeatMap are a graphical representation of alerts and notifications in the entire monitoring ecosystem (across domains and clusters).

Figure 3 Agent Heatmap Portlet



Individual cells in the map are the alert levels represented as different colors and the size of the individual cell is directly proportional to the number of alerts/notifications of that type.

The color scheme of the alerts indicate the following type of alerts:

- - High
- - Medium
- - Low
- - Notification

HeatMap Hierarchy

HeatMap implementation in Hawk WebConsole is a treemap representation. Treemaps display hierarchical (tree-structured) data as a set of nested rectangles. Each branch of the tree is given a rectangle, which is then tiled with smaller rectangles representing sub-branches. A leaf node's rectangle has an area proportional to a specified dimension on the data. The leaf nodes are colored to show a separate dimension of the data.

The default dimensions of hierarchy is:

- HawkDomain
- Cluster
- Agent
- AlertLevel

The default pivoting order of the hierarchy is: **HawkDomain ->Cluster ->Agent ->Alert Level**.

You can change the pivoting hierarchy as desired and click the  button to apply changes. The heatmap will change accordingly.

Drill-Down Capability

Top level heat map shows all alerts in all agents in all domains. You can drill down to any level of the hierarchy (dimension) to see the details. For example: if there are 4 domains, you can drill down to one domain to see all the agents in the domain in an expanded form. You can further drill down to an agent in the domain to see all alerts expanded view. You can drill down to the last level in the hierarchy.

HeatMap Realtime Updates

The HeatMap gets updated in based on the configuration provided in `web.xml`. It can either be configured in `auto` mode or in `timer` mode.

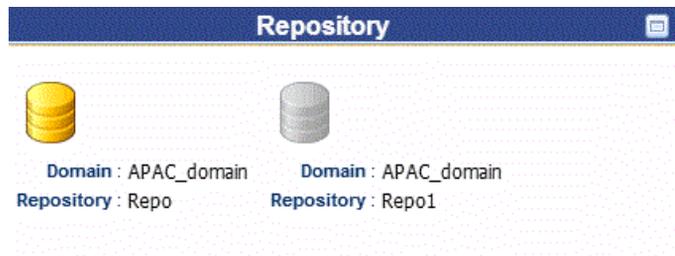
- **auto** mode: The heatmap updates are near real time. This mode can be used for optimized performance.
- **timer** mode: The heatmap is updated as per the time configured in `web.xml`. This mode can be used if the load is high.

If agent previously had less number of high alerts, the size of high alerts cell for the agent was small as compared to other cells. However, if the agent starts generating large number of high alerts, then the size of high alerts cell for the agent starts growing dynamically and all other cells in the entire heatmap are automatically adjusted accordingly.

Repository Portlet

This portlet displays the repositories in the network. Clicking on every repository icon opens a new tab for the repository.

Figure 4 Repository Portlet

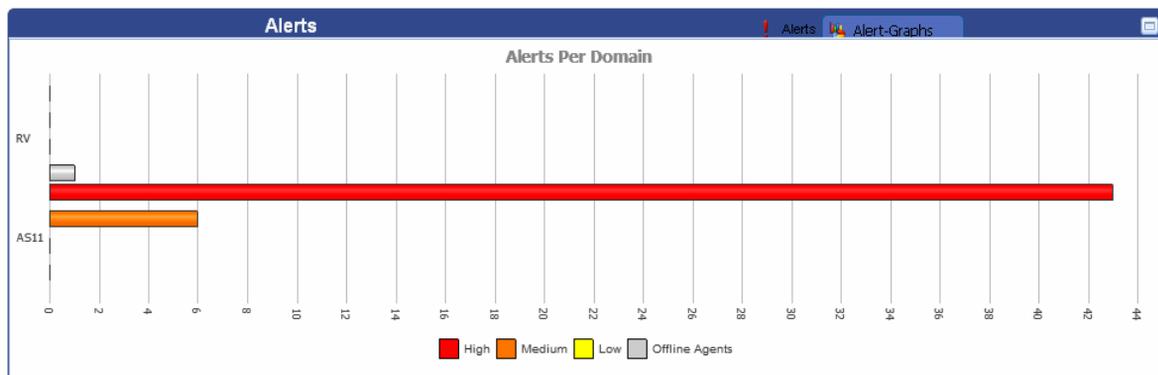


For more information on setting up a Repository, see *TIBCO Hawk Installation Configuration and Administration Guide*.

For more details, see [Working with Rulebase-Maps, page 94](#).

Alerts Portlet

The Alerts portlet displays the alert list as well as bar chart representation of alerts. It displays all alerts across domains.



This chart displays high, medium, low alerts and dead agents counts group by each domain.

For every agent, a separate bar chart graph is displayed. The graph is displayed with the category axis as "time" and value axis as "number of alerts".



In the alert chart if the number of agents are less, then the X-axis values are adjusted to properly use the space. In such a scenario, the X-axis values can be fractional.

Dashboard Options

Dashboard options are available on the top right corner of the Hawk Dashboard screen.



The following options are available:

Goto-Dashboard

Selecting this option takes the control back to the Hawk Dashboard tab.

Save UI State

Selecting this option saves the current UI tab status in the database. All the open tabs are saved and restores the state after login.



The UI state functionality depends on the persistence mode setting in the `web.xml`:

- **true**: If persistence mode is set to `true`, the UI state is saved in the database on logout.
- **false**: If the persistence mode is set to `false`, then the **Save UI State** option is not available in the **Options** list.

About

Selecting this option provides detailed information about the Hawk WebConsole, Hawk Domain, and Hawk Components.

Chapter 3 **Working with Alert Messages**

This chapter contains simple examples that demonstrate the operations supported on the alerts tab.

Topics

- [Overview, page 16](#)
- [Viewing Alert Messages, page 17](#)
- [Suspending Alert Messages, page 21](#)
- [Viewing Alert Chart, page 23](#)
- [Viewing Warning and Exceptions Messages, page 25](#)

Overview

Alerts are messages an agent sends to TIBCO Hawk WebConsole. Alerts originate from rulebases when a specified condition occurs that enforce your monitoring logic. The **Alerts** screen is divided into two sections:

- **Alert List:** Displays the alert and notification messages in tabular format.
- **Alert Chart:** Displays the alert message data as charts.

Viewing Alert Messages

The following steps describe how to view alert messages for a particular agent.

1. Click an agent listed in the **Agent** portlet on the Hawk Dashboard tab. The **Agent** <agent_name> tab opens.
2. Click the **Alerts** tab. The alert messages for the selected agent are displayed in the **Alert List** section.

Figure 5 Alert List Tab

The screenshot shows the 'Alert List' interface. At the top right is a 'Clear All Alerts' button. Below it is a filter bar with a search input field. The main area is a table with columns: Read, Cleared, State, Alert Text, Alert Time, and Rulebase. The table contains four rows with different colors: yellow (Low), red (High), orange (Medium), and blue (Notification). Callouts point to various elements: 'Filter Bar' points to the search input; 'Click on any column to sort by that column' points to the 'Alert Time' column header; 'Type in the textbox to filter alerts based on the entered text.' points to the search input; 'Read checkbox indicates whether the alert has been read and Cleared checkbox indicates whether the alert has been cleared. Cleared is a read-only field.' points to the 'Read' and 'Cleared' checkboxes in the first row; 'Colour of the row indicates the severity of the alert.' points to the blue row.

Read	Cleared	State	Alert Text	Alert Time	Rulebase
<input type="checkbox"/>	<input type="checkbox"/>	Low	This is LOW priority alert	Mon Dec 10 15:13:40	TestRuleBase
<input type="checkbox"/>	<input type="checkbox"/>	High	This is HIGH priority alert	Mon Dec 10 15:13:40	TestRuleBase
<input type="checkbox"/>	<input type="checkbox"/>	Medium	This is MEDIUM priority alert	Mon Dec 10 15:13:40	TestRuleBase
<input type="checkbox"/>	<input type="checkbox"/>	Notification	This is a NOTIFICATION	Mon Dec 10 15:13:40	TestRuleBase

The **Alert List** displays alert and notification. When a test with an Alert or Notification action becomes `true`, TIBCO Hawk WebConsole receives the alert message unless advanced options delay it. For more information, see [Creating Actions, page 46](#).



The number of alerts displayed in the **Alert List** can be configured in `web.xml`.

Each line in the Alert List represents an alert sent by a single message published by a TIBCO Hawk agent.



Any alert cannot be deleted from the alert portlet. To remove the alert from the portlet, suspend it or delete the rulebase which generated that alert.

3. Use the filter bar available right above the each column heading to filter the alerts list. The filter options available are as follows:
 - **Read:** Check the checkbox to view the alert messages that are read.
 - **Cleared:** Check the checkbox to view the alert messages that are cleared.
 - **State:** Select from the drop-down list to view the alert messages in the selected alert state.
 - **Alert Text:** Type the alert text in the text box to view the alert messages with the specified alert text content.
 - **Alert Time:** Type the alert time in the text box to view the alert messages generated at the specified time.
 - **Rulebase:** Type the rulebase name in the text box to view alert messages generated by the specified rulebase.



If **Read** checkbox is toggled to filter the alerts, it has 3 states of filtering the alerts:

- Selected
 - Unselected
 - Both
4. Click the column heading to sort the alert list on the selected column.
 5. Click the  icon to open the detail information panel for the alert message.

Figure 6 Detail Information Panel

The screenshot shows a web application window with an orange border. The title bar contains 'Medium', 'OS is : Windows 7', and 'Tue Jan 29 15:10:40 GMT New-Rulebase'. The main content area is divided into three sections:

- Rulebase Details:**
 - Rulebase Name : New-Rulebase
 - Data-Source : COM.TIBCO.hawk.microagent.SysInfo:0
 - Rule : SysInfo:getOperatingSystem():5
 - Test : (OS Name Contains Win)
 - Action : 0
 - DataIndex : _
- Alert Details:**
 - Alert Text: OS is : Windows 7
 - Alert ID: 18
- Suspend Alert:**
 - Suspend Alert For (in minutes): 5
 - Reason For Suspend: (empty text area)
 - Suspend Alert button

The following details are displayed:

- **Rulebase Details:** Rulebase details are provided.
 - **Alert Details:** **Alert Text** and **Alert ID** details are provided.
 - **Suspend Alert:** This section is used to specify details for temporarily suspending an alert. See [Suspending Alert Messages on page 21](#) for further details.
6. After reviewing the alert message, click the  icon to close the detailed information panel.
 7. Click **Clear All Alerts** to remove alert entries from Alert List.

Viewing details for an alert message automatically selects the **Read** checkbox in the Alert List. You can also manually select or unselect this field. As the Alert List continuously updates, this feature helps you track alerts that have already been reviewed.



To view alerts for all agents, go to the **Alerts** portlet on the Hawk Dashboard.

Suspending Alert Messages

You can temporarily suspend an alert message, to prevent it from interfering with other monitoring tasks. For example, if a condition such as process failure is generating a high-level alert with a warning bell and the problem is being worked on, you can suspend the alert until the problem is resolved. Suspension details are added to the properties of the message. These details are visible to you and other TIBCO Hawk WebConsole users, as well as all Console API applications.

Suspending an alert message affects only the action of the generated the alert. If the condition that generates the alert message also generates another type of action, such as attempting to restart the process, that action is unaffected.

To suspend an alert:

1. Click an agent listed in the **Agent** portlet on the Default Hawk Dashboard tab. The **Agent** *<agent_name>* tab opens.
2. Click **Alerts** tab. The alert messages for the selected agent are displayed in the **Alert List** tab.
3. Click the  icon to open the detailed information panel for an alert message.
4. Specify a suspension interval by using up and down arrow keys or typing a value in the **Suspend Alert For** text box. The default value is 5 minutes.
5. Type a reason for suspension in the **Reason For Suspend** text area. This is used to communicate to other users.
6. Click **Suspend Alert**.

The **Cleared** checkbox for this alert is selected. The Alert Suspend details are displayed in the **Alert Clear/Suspend Details** section of the alert information panel. The following information is displayed:

Alert Clear Time - Fri Jan 11 11:17:14 GMT: +530 2013

Alert Clear Reason - Suspended: <Reason specified by the user>

Alert Suspended By - <User Name>

Alert Suspended For - 5 minute(s)



You can also use the `RulebaseEngine:suspendAlert` method to suspend specific alerts. Refer to the *TIBCO Hawk Microagent Reference* for details.

Alert suspension is lifted either when the suspension interval ends or when a user invokes the `resumeSuspendedAlerts` method of the `RuleBaseEngine` microagent. For more information on microagent methods, see the *TIBCO Hawk Microagent Reference*.

Once suspended, the alert is cleared. Once cleared, it can be purged from the Alert List, whether or not the problem condition is resolved. If the condition still exists when the suspension interval ends, the agent generates a new alert message for the condition.

Viewing Alert Chart

The **Alert Chart** is a graphical representation of the alert messages generated for a selected agent. The alert message data displayed on the **Alert List** section appears as charts on the **Alert Chart**.

The alert chart value never decreases, it always increases. This happens because even if an alert is cleared, it is still displayed in the chart since it was generated at some point of time.

To view the chart for alert messages generated for an agent:

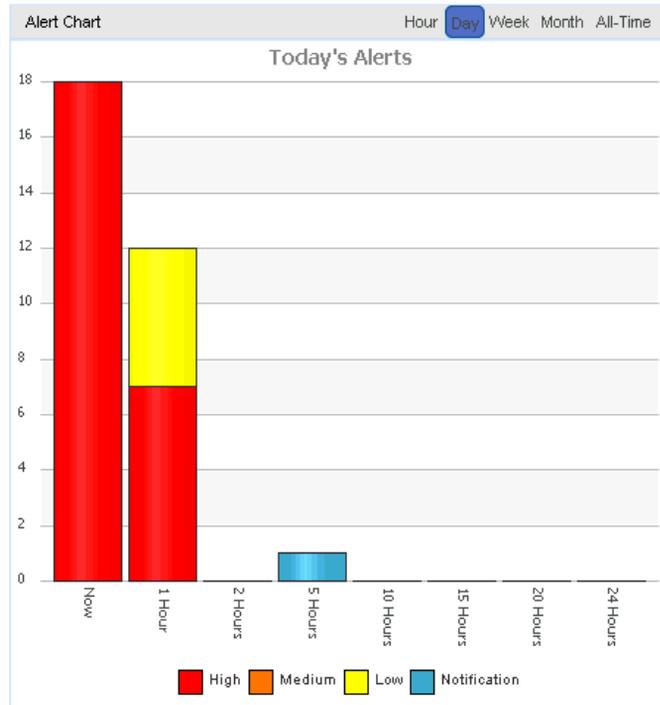
1. The **Alert Chart** section plots the state of the alert message data for the specified time period.
2. Specify the period for which the alert messages should be plotted by clicking on the time period options available on the top right corner of the **Alert Chart** section. The options are:
 - Hour
 - Day (Default)
 - Week
 - Month
 - All-time

Each of these options are divided into sub-intervals for plotting the graph. For example the **Day** option is divided into:

- Now
- 1 Hour
- 2 Hours
- 5 Hours
- 10 Hours
- 15 Hours
- 20 Hours
- 24 Hours

Alerts are displayed under each of these sub-intervals. One common sub-interval for **Hour, Day, Week, Month** and **All-time** options is called the **Now** sub-interval. All live alerts are displayed in the **Now** sub-interval until they can be transferred to the next available sub-interval.

Figure 7 Chart Tab



For example: You choose the **Day** option. The moment an alert is generated, it is displayed under the **Now** sub-interval. The next available sub-interval is **1 Hour**. As soon as alerts is 1 hour old, it is transferred from the **Now** sub-interval to the **1 Hour** sub-interval. Similarly, alerts which were earlier displayed in **1 Hour** sub-interval are transferred to **2 Hours** sub-interval and so on.

Alert charts are automatically refreshed for the following conditions:

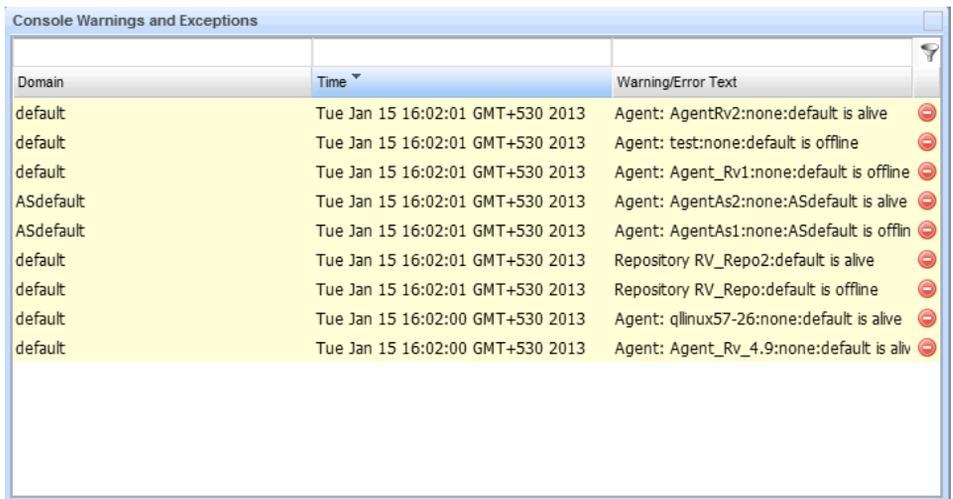
1. When a new alert is generated.
2. Alerts get older and are ready to be transferred to next available sub-interval.
3. When you move to the **Alerts** tab under the parent `<agent_name>` tab. The **Alert Chart** section is refreshed only when you are working on the **Alerts** tab.

Viewing Warning and Exceptions Messages

Items such as dissimilar rulebase duplications, rulebase application errors and agent discovery errors are displayed in the Console Warning Window. When TIBCO Hawk WebConsole generates or receives these messages, the Warning and Exceptions messages icon  starts glowing.

Clicking on this icon will open the Console Warning and Exceptions window to the foreground. The following figure shows the Console Warning and Exceptions window:

Figure 8 Console Warning and Exceptions Window



Domain	Time	Warning/Error Text	
default	Tue Jan 15 16:02:01 GMT+530 2013	Agent: AgentRv2:none:default is alive	
default	Tue Jan 15 16:02:01 GMT+530 2013	Agent: test:none:default is offline	
default	Tue Jan 15 16:02:01 GMT+530 2013	Agent: Agent_Rv1:none:default is offline	
ASdefault	Tue Jan 15 16:02:01 GMT+530 2013	Agent: AgentAs2:none:ASdefault is alive	
ASdefault	Tue Jan 15 16:02:01 GMT+530 2013	Agent: AgentAs1:none:ASdefault is offlin	
default	Tue Jan 15 16:02:01 GMT+530 2013	Repository RV_Repo2:default is alive	
default	Tue Jan 15 16:02:01 GMT+530 2013	Repository RV_Repo:default is offline	
default	Tue Jan 15 16:02:00 GMT+530 2013	Agent: qlinux57-26:none:default is alive	
default	Tue Jan 15 16:02:00 GMT+530 2013	Agent: Agent_Rv_4.9:none:default is aliv	

Chapter 4 Working with Rulebases

This chapter contains simple examples that demonstrate the operations supported on the rulebases tab.

Topics

- [Overview, page 28](#)
- [Creating a Rulebase, page 32](#)
- [Creating a Rule, page 36](#)
- [Creating a Test, page 39](#)
- [Creating Actions, page 46](#)
- [Referencing Variables in a Rulebase, page 54](#)
- [Using Posted Conditions, page 57](#)

Overview

A rulebase is a collection of one or more rules. A rule is a user-defined monitoring policy. It specifies:

- A data source in the form of a microagent method
- One or more tests that check for conditions
- One or more actions to perform if a test result is `true`.

To know more about Rule, Test, and Actions, see *TIBCO Hawk Concepts Guide*.

Sample Rulebase

Sample rulebases are bundled with TIBCO Hawk installation. They are installed in `<HAWK_HOME>\6.0\examples\rulebases` directory.

The `HawkAgent-UNIX.hrb` rulebase has a rule that monitors the `Hawk.log` file. If TIBCO Hawk is installed in a directory other than the default, this rule will fail and send out high alerts for `noDataSourceErrors`. If TIBCO Hawk is installed in a directory other than the default, you must edit the rule to use the actual directory path of the `Hawk.log` file.



Any tag missing in the rulebase XML will lead to error in use of the artifact. All the tags are mandatory in `.hrb` file.

Rulebase Screen Details

This section provides a detailed description of the rulebase screen.

Rulebase Screen Sections

The rulebase screen is divided into three sections:

- **Deployed Rulebases:** Lists the rulebases deployed to the agent/repository.
- **User Rulebases:** Lists the rulebases created and saved in the database by the user.
- **Rulebase / Rule / Test / Action** editing screen: The screen displayed in this section is based on the selected configurable object from the Rulebase Tree. You can edit the following objects:
 - Rulebase: Refer to [Creating a Rulebase on page 32](#) for further details.
 - Rule: Refer to [Creating a Rule on page 36](#) for further details.
 - Test: Refer to [Creating a Test on page 39](#) for further details.
 - Action: Refer to [Creating Actions on page 46](#) for further details.

Toolbar Icons

The toolbar is located in the top right corner of the rulebase screen.

Figure 9 Toolbar



When you hover over the icons in the toolbar, a popup tool tip describes the tool. The functionality of each icon is described in the following table:

Table 2 Toolbar icons

Toolbar Icons	Description
	Displays alerts for the selected rulebase.

Toolbar Icons	Description
	Creates a new rulebase.
	Creates a new rule under current rulebase.
	Creates a new test under current rule.
	Creates a new consequence action under current test.
	Creates a new clear action under current test.
	Deletes the selected configurable object.
	Derives the selected configurable object.
	Deploys selected rulebase.

Toolbar Icons	Description
	<p>Saves the selected rulebase. The functionality of this icon in the deployed and user sections is as follows:</p> <ul style="list-style-type: none"> • Deployed Rulebases section: Saves the rulebase to the User Rulebases section and to the local database. • User Rulebases section: Saves the rulebase to the local database.
	<p>Synchronizes your copy of the selected item with the copy on the agent.</p>
	<p>Deploys one or more rulebases on agents or repository.</p>
	<p>Deletes rulebases from multiple agents or repositories.</p>
	<p>Import a rulebase.</p>
	<p>Export a rulebase.</p>



The toolbar icons are enabled or disabled based on the selected configuration object.

Creating a Rulebase

To create a new rulebase using TIBCO Hawk WebConsole:

1. Click an agent listed in the **Agents** portlet on the Hawk Dashboard. The **Agent** tab opens. The name of the **Agent** tab is same as the selected agent name `<agent_name>`. The following tabs are available on the top right corner of the screen:
 - Alerts (Default)
 - Rulebase
 - Schedules
 - Microagents
2. Click the **Rulebase** tab. The Rulebase screen lists the rulebases loaded by the selected agent and provides a toolbar for performing rulebase operations.
3. Click the **Create new Rulebase** icon from the toolbar to create a new rulebase. A new rulebase is created with the default name "New-Rulebase". Type **Sample** in the **Rulebase Name** field.



The rulebase name must be unique on the agent. It may contain only numeric digits, underscore (_), hyphen (-), or a letter as defined by the UNICODE 2.0 standard. The latest version of the UNICODE specification can be found at www.unicode.org/ucd.

Figure 10 Rulebase Editing Screen

Rulebase Name :

Author : admin

Last Modified : Mon Feb 04 14:47:57 GMT+530 2013

Comments :

Schedule :

[Hide Advanced Options](#)

Included Rulebases

Included Rulebases

No items to show.

Included Commands

Command

No items to show.

Apply & Add Rule  

Apply Changes

4. Enter comments in the **Comments** field, if required.
5. Select the name of an existing schedule from the **Schedule** dropdown list to apply to this rulebase. This is an optional field. By default, the rulebase is always active. For more information, see [Working with Schedules and Period Groups on page 58](#).
6. To work on the advanced options, refer [Using Advanced Rulebase Features on page 34](#).
7. Click the **Apply & Add Rule** button to save the changes and proceed to the rule building process. Refer [Creating a Rule on page 36](#) for further instruction.
8. Click the **Apply Changes** button to save the current changes. These changes are available in the browser till the user logs out.

A rulebase must have at least one rule, a rule must have at least one test, and a test must have at least one action.

Using Advanced Rulebase Features

A rulebase can include other rulebases which can automatically load when the rulebase is loaded by an agent. The Included Rulebases feature is useful for maintaining modular rulebase sets. For example, if rulebase A should always be loaded with rulebase B, rulebase B can have A as a member of its include list.

Similarly, the Included Commands feature can be used to add any command to a rulebase, but rulebase commands are typically related to the resources or activities that the rulebase monitors. For example, if a rulebase is monitoring the `httpd` daemon, the commands could execute scripts such as: `start_httpd` or `kill_httpd`.

To access advanced rulebase options:

1. In the rulebase editing screen, click **Show Advanced Options** link. The Advanced options are displayed.

[Hide Advanced Options](#)



2. Click  icon available on the **Included Rulebases** section title bar. A new row is added to the Included Rulebases table.
3. Select an existing rulebase from the dropdown list to include.
4. Click  icon available on the **Included Commands** section title bar. A new row is added to the Command table.

5. Type a command or the name of a script to execute. You can specify an absolute or relative path. To specify multiple commands, separate them with a semicolon (;). The Command List displays the string you typed.
6. Click the **Apply & Add Rule** button to save the changes and proceed to the rule building process. Refer [Creating a Rule on page 36](#) for further instruction.
7. Click the **Apply Changes** button to save the current changes. These changes are available in the browser till the user logs out.

Creating a Rule

The following example uses the `Process` microagent as a representative data source.

To create a new rule:



These instructions begin in the rulebase definition section. For instructions on accessing this screen, go to [Creating a Rulebase, page 32](#) section.

1. On the Rule Editing screen, choose a microagent and method.

Figure 11 Rule Editing Screen

The screenshot shows the 'Microagent Info' dialog box with the following details:

- Microagent Info** (Title Bar)
- Methods** (Section Header)
 - onRvDaemonStatus (Selected)
 - onRvLicenseExpire
- Description** (Section Header)
 - Method :** onRvDaemonStatus
 - Description :** Report TIBCO Rendezvous daemon status
- Arguments** (Section Header)
 - Data Interval (secs) : 5
 - Service :
 - Network :
 - Daemon :
 - Interval : 360
- Microagent List** (Left Panel)
 - 1 TIBCO Rendezvous
 - 2 Process (Selected)
 - 3 FileSystem
 - 4 FileStat
 - 5 Custom
 - 6 SysInfo
 - 7 Network
 - 8 Self
 - 9 Logfile
 - 10 Repository:RV_Repo2
 - 11 RuleBaseEngine
 - 12 System
- Buttons**
 - Refresh

Below the dialog box, there are additional controls:

- Schedule :
- Over-Ruling : 0

2. This screen displays a list of microagents you can use as a data source for the rule. TIBCO Hawk default microagents are listed along with microagents for AMI adapters, gateways, and instrumented applications.
3. Select the **Process** microagent. The **Methods** panel displays the methods for the microagent.
4. Select the **getProcess** method. The **Description** panel displays a detailed description of the method, including arguments and return values. The **Arguments** panel displays the fields for method arguments and a data interval.
5. In the **Process Name** field, type `rvd.*`.

When invoked, the `getProcess()` method returns information about all running processes. It takes the name of a specific process as an optional argument and with no argument specified, the entire process table is returned. In this case, the method returns all process instances that begin with the string `rvd`.

The `getProcess()` method is a synchronous method. When the rule is active, the agent subscribes to this method and receives data every 5 seconds, by default. The data interval can be changed, if required. For asynchronous methods, such as `LogFile: onNewLine()`, no collection interval is required.



You can reference both external and internal variables within data source arguments. These must be typed in manually using the correct format, for example, `${Internal.Agent Name}` or `${External.HAWK_ROOT}`. For more information on referencing external and internal variables, see [Referencing Variables in a Rulebase](#), page 54.

6. Select the name of an existing schedule from the **Schedule** dropdown list to apply to this rule. This is an optional field. By default, the rule is always active. To know more, see [Working with Schedules and Period Groups](#) on page 58.
7. Specify the **Over-Ruling** value by using the up and down arrow keys. This is an optional field. For know more about **Over-Ruling** feature, see *TIBCO Hawk Concepts Guide*.
8. Click the **Apply & Add Test** button to save the changes and proceed to the defining the test process. Refer [Creating a Test](#) on page 39 for further instruction.
9. Click the **Apply Changes** button to save the current changes. These changes are available in the browser till the user logs out.

The rule is now configured to use the `getProcess()` method of the `Process` microagent as a data source. The Rule screen displays `COM.TIBCO.hawk.Process:getProcess(Name=rvd.*):5` in the title bar.

Creating a Test

The following procedure shows how to build a test expression by specifying a test parameter and test operator in the Test screen. This example uses the MemUsage microgent method result field and a numeric operator.

To create a new test in the current rule:

1. Go to the **Test** screen.



These instructions begin in the rule definition section. For instructions on accessing this screen, go to [Creating a Rule](#) section.

Figure 12 Test Editing Screen

Test Name: new test: choose a parameter and operator

The screenshot shows the 'Test Editing Screen' interface. At the top, there is a 'Match Any' dropdown menu. Below it, there are three test conditions, each consisting of a red minus sign button, a dropdown menu with 'Process Name' selected, a dropdown menu with 'Equals' selected, and an 'Equal to:' text input field. A green plus sign button with a plus sign and a plus sign in parentheses is located below the test conditions. Below the test conditions, there is a 'Schedule:' dropdown menu. Underneath, there is a link 'Hide Advanced Options'. Below that, there is a 'True Test Counter:' label followed by a text input field containing '1'. Below the counter, there are three radio button options: 'First False' (selected), 'Clear Timer', and 'Clear Test'. To the right of the 'First False' option, there is a link 'Clear on the first time the test becomes False' and a text input field containing '1'. Below the 'Clear Test' option, there is a button labeled 'Edit Clear Test'. At the bottom of the screen, there are two radio button options: 'Clear Action' and 'Consequence Action' (selected). To the right of these options, there are two buttons: 'Apply & Add Action' with a green play button icon, and 'Apply Changes'.

The **Test** screen helps you create multiple tests.

2. You can specify if you want to match all, match any, or match none.
3. Select **Mem Usage** from the **Process Name** dropdown list. This is used as the test parameter. (On UNIX, use **Virtual KBytes**.)

Mem Usage is one of 10 result fields returned by the `getProcess()` method, the data source for this rule. The text area in the Microagents, Methods and Arguments screen displays a short description of each parameter. In the description for this method, you can see that `Process Name` is a string value, and `Mem Usage` is an integer.

4. Select the **>** operator from the Operator dropdown list. Only operators that apply to the current parameter are included in the list. **Mem Usage** is an integer field, so numeric operators are listed. All test operators are described on the following pages.
5. In the **Greater than** field, type the number **1000**.
6. Select the name of an existing schedule from the **Schedule** dropdown list to apply to this test. This is an optional field. By default, the test is always active. For more information, see [Working with Schedules and Period Groups on page 58](#).
7. To specify Advanced option, click on the **Show Advanced Options** link. Refer to [Using Advanced Test Features on page 44](#) for further details.
8. Click the **Apply Changes** button to save the test expression.

This test now checks for instances of the process `rvd` using more than 1000 Kilobytes, or 1 MB, of memory when the test is evaluated. If all instances found are using less than 1 MB of memory, no problem condition exists and the test is false. Since `getProcess` is a synchronous method, the agent evaluates the test every 60 seconds by default.

The following tables describe the test operators you can apply to numeric, text and Boolean parameters.

Table 3 Test Operators For Numeric Method Results

Operator	Description
== ! =	The test expression is true when the value of the test parameter is (equal to, not equal to) the operator value.
< <= > >=	The test expression is true when the value of the test parameter is (less than, less than or equal to, greater than, greater than or equal to) the operator value.
inRange	The test expression is true when the value of the test parameter is between two extremes of a range. Endpoints are included.

Table 3 Test Operators For Numeric Method Results (Cont'd)

Operator	Description
OutOfRange	The test expression is true when the value of the test parameter is outside the range of two operator values. Endpoints are excluded.
Increase	The test expression is true when the value of the test parameter has increased at least by the operator value between two successive test evaluations. For example, the amount of disk space in use has increased by more than 10 MB in a sample period.
%Increase	The test expression is true when the value of the test parameter increases by at least the operator value as a percentage (the increase divided by the previous value times 100) between two successive test evaluations. For example, the amount of disk space in use has increased by more than 10 percent in a sample period.
Decrease	The test expression is true when the value of the test parameter decreases by at least the operator value between two successive tests.
%Decrease	The test expression is true when the value of the test parameter decreases by at least the operator value as a percentage (the decrease divided by the previous value times 100) between two successive test evaluations.
NetChange	The test expression is true when the value of the test parameter increases or decreases by at least the operator value between two successive test evaluations. The operator value specifies the absolute value of the increase or decrease.
%NetChange	The test expression is true when the value of the test parameter increases or decreases by at least the operator value as a percentage (the increase or decrease divided by the previous value times 100) between two successive test evaluations. The operator value specifies the absolute value of the percentage increase or decrease.
postedConditionExists	The test expression is true when the specified posted condition exists. This operator displays when a posted condition is selected in the parameter list. <code>PostedConditionExists</code> is equivalent to <code>Posted.x > 0</code> . For more information, see Using Posted Conditions, page 57 .

Table 3 Test Operators For Numeric Method Results (Cont'd)

Operator	Description
<code>!PostedConditionExists</code>	The test expression is true when the specified posted condition does not exist. This operator displays when a posted condition is selected in the parameter list. <code>!PostedConditionExists</code> is equivalent to <code>\${Posted.x} == 0</code> when x is the posted condition name. For more information, see Using Posted Conditions, page 57 .

Table 4 Test Operators for Text String Results

Operator	Description
<code>Equals</code>	The test expression is true when the value of the test parameter exactly matches the operator value. This is a case-sensitive match.
<code>!Equals</code>	The test expression is true when the value of the test parameter does not exactly match the operator value. This is a case-sensitive match.
<code>StartsWith</code>	The test expression is true when the value of the test parameter starts with the operator value. This is a case-sensitive match.
<code>Contains</code>	The test expression is true when the value of the test parameter contains the operator value. This is a case-sensitive match.
<code>!Contains</code>	The test expression is true when the value of the test parameter does not contain the operator value. This is a case-sensitive match.
<code>Perl5 PatternMatch</code>	The test expression is true when a match is found using a regular expression as an operator value.

Table 5 Test Operators for Boolean Results

Operator	Description
<code>isTrue</code>	The test expression is true when the value of the test parameter is true.

Table 5 Test Operators for Boolean Results

Operator	Description
isFalse	The test expression is true when the value of the test parameter is false.

Building Compound Tests

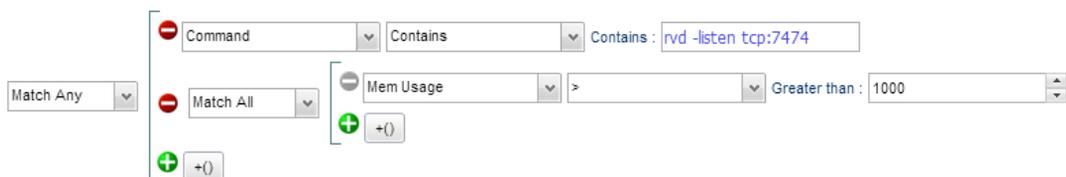
A compound test uses the same operators as a simple test, but allows you to combine multiple expressions using the logical operators AND, NOT, and OR. You can group expressions and insert operators in the compound test editor.

The following procedure shows how to build a compound test expression by modifying a simple expression. This example adds a second condition, using the Command microagent method result field and a text string operator, to the sample test expression. These instructions are provided in the test definition section (step 7 of [Creating a Test](#) section). Both conditions in the new test expression must be satisfied for the test to evaluate to true.

To build a compound test:

1. In the Test screen dialog, click . The compound test fields are added below the current test expression. The current text expression is highlighted in the Test Expression field, for example:
If (Command Contains rvd -listen tcp:7474)
This test checks for rvd processes started with specific command options.
2. Select **Match All** from the dropdown list to group the highlighted expression and add the AND operator.
3. In the compound test fields, click the **Mem Usage** parameter and the > operator.
4. In the **Greater than** field, type **1000**.

Figure 13 Compound Test Fields



5. Click the **Apply Changes** button to insert the expression into the highlighted set of parentheses. The compound test now looks like the following:
((Command Contains rvd -listen tcp:7474) AND (Mem Usage > 1000))

This test evaluates to true when the specified `rvd` process uses more than 1000 Kilobytes, or 1 MB of memory. If the memory threshold is exceeded, the test could trigger an action for restarting the process and notifying the system administrator. For more information, see [Creating Actions on page 46](#).

Processes named `rvd` that were started using different parameters, such as `-listen tcp:7475` are unaffected.

6. Click the **Apply Changes** button to save the test.

Using Advanced Test Features

A test includes the test expression (such as `Processes > 10`) and any extra conditions, such as counters, timers and additional tests. These advanced options add extra requirements for a test to evaluate as true or false.

To access advanced test options:

1. In the **Test** screen, click **Show Advanced Options** link. The Advanced options are displayed:

Figure 14 Advanced Options

Hide Advanced Options

True Test Counter : 1

First False Clear on the first time the test becomes False
 Clear Timer
 Clear Test Edit Clear Test

2. To specify a true test counter, type the number of true evaluations in the **True Test Counter** field. The default value is 1.

With a true test counter, the action is triggered only after the test expression has been `true` for the specified number of test evaluations. For example, to check for consistently high CPU usage and ignore any brief spikes, you could set the true test counter for the test to five. The action is triggered when the test expression (`CPU use high`) is `true` for five consecutive test evaluations.

3. To specify a clear condition, click the **First False**, **Clear Timer** or **Clear Test** radio button.
 - **First False:** After the test becomes `true`, the test is cleared when the first time the test changes from `true` to `false`. This is the default behavior for a test with a synchronous data source.
 - **Clear Timer:** Specify a wait interval in seconds. After the test becomes `true` it remains `true` until this interval has passed without an additional `true`

test. This is the default behavior for a test with an asynchronous data source, and the default wait interval is 900 seconds (15 minutes).

- **Clear Test:** Click **Edit Clear Test** to specify an extra test expression for clearing the test. After the test becomes `true`, it becomes `false` only when the clear test expression becomes `true`. The clear test uses the microagent method result fields of the data source as input.

For example, a test monitors each line in a log file for the string `Feed Line Down`. If this string is found, an alert is generated. A clear test for the original test checks for a log file line that signals the condition is resolved, such as `Feed Line Up`. When the clear test evaluates to `true`, the original alert message is cleared.

4. To specify action, select one of the following radio buttons for action type:
 - **Clear Action**
 - **Consequence Action**

then click the **Apply & Add Action** button.

Creating Actions

Each test has one or more related actions. An action can be of two types:

- Consequence Action
- Clear Action

Consequence Action

This action is the consequence of a rule, such as an alert message or a custom script. The details are described below.

Creating an Alert Message with Variable Substitution

The following example shows how to create an alert message with variable substitution as a representative action.

To define an action:

1. Go to **Action** screen.

Figure 15 Action Editing Screen

The screenshot displays the 'Action Editing Screen' with the following elements:

- At the top, there are radio buttons for selecting the action type: Alert, Execute, Notification, Method, E-mail, and Post Condition. To the right is a link labeled [Show Advanced Options](#).
- Below the radio buttons, there is an 'Alert Level' dropdown menu set to 'Medium' and an 'Insert Variable' button.
- The 'Alert Message' field is a large text area containing the variable substitution `${nextLine}`.
- At the bottom, there is a 'Schedule' dropdown menu.

- Select an action type from the radio buttons across the top of the screen. These buttons correspond to the following TIBCO Hawk action types:

Table 6 Action Types in the Action Editor

Action Type	Result	Usage Notes
Alert (default)	Sends an alert message to TIBCO Hawk WebConsole	In the Message field, type the alert text that you want to appear in the TIBCO Hawk WebConsole Alerts portlet. Specify an alert level: high (default), medium or low.
Execute	Executes a command on the TIBCO Hawk agent machine	In the Execute field, type the entire command line to send to the operating system of the agent machine.
Notification	Sends a notification message to TIBCO Hawk WebConsole	In the Notification field, type the notification text that you want to appear in the TIBCO Hawk WebConsole Alerts portlet.
Method	Invokes a microagent method on the TIBCO Hawk agent machine	Select a microagent and method from the Microagent Info panel. Specify any required arguments.
Email	Sends an email message.	Specify a message recipient as <i>recipient@domain.com</i> . Specify a subject string, an SMTP mail server for sending the message, and message text.
Post Condition	Creates a posted condition to use in another rule in the same rulebase	In the Posted Condition field, type a label for the posted condition. For more information, see Using Posted Conditions, page 57 .

3. In the **Alert Message** field, type the following:
Process `${Process Name}` is using `${Mem Usage}` KBytes
4. You can insert an internal variable, an external variable, or a data source.

Alert
 Execute
 Notification
 Method
 E-mail
 Post Condition
 [Show Advanced Options](#)

Alert Level :

Alert Message :

Schedule :

Insert Variable ▾

- Local Variable ▶
- Internal Variable ▶
- External Variable ▶
- Posted Condition ▶



Substituting variables in string fields is supported for all action types except Post Condition. For more information on using variables in a rulebase, see [Referencing Variables in a Rulebase, page 54](#).

- To insert an internal variable, click **Insert Variable** and select **Internal Variable** from drop-down menu.

Internal variables available are:

- Agent Name
- Hawk Domain
- Current RuleBase
- Current Rule
- Current Test
- Current Action
- Condition True Time
- TimeZone

- To insert an *external variable*, click **Insert Variable** and select **External Variable** from the menu. `${External . <var name>}` is inserted in the active string field. Replace `<var name>` in the syntax string with the name of the external variable defined in the properties file.

External variables are obtained from the variable file specified in the `-variable` option for the rulebase engine (suboption of `-M RuleBaseEngine`) when the agent is started.

- To insert a *local variable* in a string argument, select a variable name from the menu.

Alert
 Execute
 Notification
 Method
 E-mail
 Post Condition
 [Show Advanced Options](#)

Alert Level : High ▼

Alert Message : Process \${Process Name} Usage} KBytes

Schedule : ▼

Insert Variable ▼

- Local Variable ▶
- Internal Variable ▶
- External Variable
- Posted Condition ▶

- Process Name
- ID Process
- Parent Process ID
- Command
- CPU Time
- Class
- User Name
- Mem Usage
- Peak Working SetSize
- Page File Usage
- Page Fault Count
- Start time

Variable syntax is added to the string field at the cursor location. The syntax does not require modification. You can also manually type the syntax `${<return-field-name>}` in the string field for an action, where *return-field-name* is the label for a value returned by the method. The microagent method that returns this field must be the data source for the current rule.



You can get more information for specific method return fields by viewing descriptive help text for the method in the Microagents, Methods and Arguments dialog. For instructions on accessing this dialog, see [Working with Alert Messages, page 15](#).

5. Select the name of an existing schedule from the **Schedule** dropdown list to apply to this test. This is an optional field. By default, the test is always active. For more information, see [Working with Schedules and Period Groups on page 58](#).
6. Click one the following buttons:
 - **Save But do not Deploy** : To save until user logs out.
 - **Save and Deploy**: To save it and deploy it to the agent.

Using Advanced Action Features

Advanced action options add flexibility in timing when an action is performed. For example, using advanced options you can automate problem escalation procedures.

To use advanced action features:

1. Click **Show Advanced Options** in the Action screen. The Advanced Action options are displayed:

[Hide Advanced Options](#)

Escalation Period : 0

Perform Action Policy

Maximum 1 : Times, No sooner than every 1 : sec
 Once until the alert message changes
 Once Only
 Always

Properties Template

Key	Value
No items to show.	

2. To specify how actions are performed, click the **Maximum**, **Once until the alert message changes**, **Once Only**, or **Always** radio buttons.
 - For **Maximum**, specify the maximum number of times the action can be performed, no matter how long the associated test continues to remain `true`. If the test becomes `false`, the counter is reset. Specify the number of seconds to wait between actions as long as the test is `true`. The related action can only be triggered at a test evaluation, so the actual interval between actions may be longer than the specified interval.

This option is useful when the action executes a paging script. A single page might be lost, but paging at each test evaluation (such as once per

minute) is too often. With this option you can send the page every five minutes until it is likely to be received.

- With **Once until the alert message changes**, the first time this action is triggered by a test, the action is performed. On subsequent `true` evaluations the action is performed only until there is a change in the alert message.

This option is applicable only if the associated action creates an alert message with some string variables. The action is performed each time the value of the string variable changes resulting in change in the alert message.

Substituting variables in alert messages overrides this feature. For more information, see [How Variable Substitution Affects Actions](#), page 56.

- With **Once Only**, the first time this action is triggered by a test, the action is performed. On subsequent `true` evaluations the action is not performed. The action is not performed again until the test becomes `false` and then `true` again. This is the default behavior for all actions.

Substitution of variables in alert messages will have no impact on this feature.

- With **Always**, the action is performed each time the associated test is evaluated as `true`, even if the test was `true` in the last evaluation.

3. To escalate a problem, type a wait interval in seconds in the Escalation Period field.

The action is not performed the first time the associated test is true, but instead starts an internal timer. When the action is triggered on a test transition from false to true, the timer is started. If the associated test remains true for another evaluation after the specified interval, then the action is performed. You can use an escalation period to respond to continuing or deteriorating conditions.

4. To add properties to the rulebase actions:
 - a. To create a new property, click the **Create New Property** button in the Properties area. A new table row appears.

Normally, when an action is executed, internal properties (such as Rule, DataSource, Test, and so on) are assigned to the action. You can define additional properties for an action here. If an action causes an alert to be generated, these properties (internal and user-defined) are passed to the `PostAlertEvent`. For user-defined properties in the action, the properties are

pre-appended with "Action" in the key. These properties can also be viewed in the alert details in the **Alerts** portlet.

To add properties to rulebase actions, type in the key and value. The key and value must be strings. You can modify, copy and delete keys and values that you have added.

- b. To load a previously-defined template, click **Template** and select **Load**, then select the file to be loaded.
- c. If you want to save your new action properties as a template, click **Template** and select **Save**, then specify where the file should be created and enter a file name. The template is saved as an XML file, and the extension `.xml` is added to the filename if not specified. The template can be loaded later.
- d. You can also configure a saved template as the default template by clicking **Set Default** from the **Template** menu and selecting the appropriate template.

When a default template is set, the properties from the default template are saved in the local database and loaded to the Advanced Options Editor when an action is being edited. If a property from the default template exists in the current action, the property from the default template is ignored. The default template is on a per-user basis.



For existing rulebases, even if a default template is set, the actions in the rulebase are NOT updated with properties from the default template unless the action is edited using the Advanced Action Editor.

5. Click one the following buttons on the Action Editing screen:
 - **Save But Do Not Deploy** : To save in the browser until user logs out.
 - **Save and Deploy**: To save it and deploy it to the agent or repository.

Clear Action

A clear action is an action that takes place only when a test makes the transition from `true` to `false`. After the test becomes `true`, whenever it becomes `false` again one or more actions are performed. Clear actions behave like regular actions but do not support advanced options except schedules, and cannot generate alerts. They are useful for sending notifications or all-clear messages.

A clear action can reference the same data source variables as the associated test. For more information, see [Referencing Data Source Variables on page 56](#). For instructions on using this screen, see [Creating Actions on page 46](#).



A clear action generated from a test cleared by using clear timer will not have data source parameter variables available to the clear action, since it is a timer and not a data source evaluation creating the clear event.

Referencing Variables in a Rulebase

You can reference several kinds of variables in a rulebase. By referencing variables, the rulebase can adapt to changes on multiple machines. For example, not all machines store log files or temporary files in the same directory. Also, rulebases used on multiple platforms need to accommodate subtle differences in how path names are expressed. You can use variables rather than specifying this information manually.

When an action contains variable substitution, a new alert is generated each time the test is true and the value of the variable changes. Variable substitution is most useful for values that are either slowly changing, very important, or both.



Variable substitution affects the performance of rulebase processing. Therefore, you should reference a variable only when it provides a clear benefit.

Supported Types of Variables

The following types of variables are supported in a TIBCO Hawk rulebase:

- External, such as user-defined variables
- Internal, such as the name of a test in a rule
- Data source, such as a microagent method result field (Data source variables can be referenced in actions only)

Referencing these variables outside of a rulebase is not supported.

Referencing External Variables

External variables are variables defined by a user on the machine where the TIBCO Hawk agent runs.

First, you define the variable values in a properties file on the local machine. Then you specify the variable file using the `-variable` option when starting Hawk Agent. Then you can reference the external variable in a rulebase. For more information on agent startup parameters, see *TIBCO Hawk Installation Configuration and Administration Guide*.

After variable values are defined and the properties file is specified to the agent, you can reference external variables in a rulebase using the following syntax:

```
${External . <variable-name> }
```

where *variable-name* is the name of an environment variable defined in the properties file. The file uses a standard Java property file format, with one line per variable defined. Each entry is a name-value pair in the following format:
`<variable-name>=<value>`

You can reference external variables in string arguments of actions and in data source method string arguments. For example, the Hawk Services sample rulebase provides a rule for sending a high-level alert. Without variable substitution, the text of the alert is generic. With variable substitution, the alert includes information specific to the generating condition.

Restrictions

In Microsoft Windows, the following restrictions apply to external variables:

- The variables file to support External variables in the agent must conform to the Java properties file format.
- Variables and variable names cannot include spaces or any of the following characters: equals sign (=), period (.), or forward slash(\).
- Any special characters must be escaped to be evaluated properly.

On UNIX systems, the `env` command outputs environment values in the correct format.

Referencing Internal Variables

Internal variables refer to elements of the current rulebase. This type of variable is defined internally by the TIBCO Hawk agent, and requires no properties file. Values are assigned to variables when the rule is processed.

Like external variables, internal variables can be referenced in string arguments of methods used as a rule's data source or in string arguments of actions. You can manually type internal variable syntax in the string argument of a method, or, for action arguments, TIBCO Hawk WebConsole provides a dropdown list of internal variables.

Manually entering variables

To manually enter internal variables, specify the variable using the following syntax:

```
${Internal.<variable>}
```

where `<variable>` can be `Agent Name`, `Agent IP Address`, or so on.

The variables will be substituted with the appropriate value before the command is executed. For example, the command `Telnet ${Internal.Agent Name}` will be executed as `Telnet kimyou` if the command is executed for agent `kimyou` from the Agent View.

Referencing Data Source Variables

Data source variables are TIBCO Hawk variables that represent the return fields of a microagent method. The method must be used as the data source of the current rule. You can reference data source variables only in actions.

For example, the Hawk Services sample rulebase provides a rule for monitoring an event log and sending a high-level alert message when an error is written to the log. The Alert action type used in this rule allows you to specify a text string for the alert message. In this example, the text string is:

```
Hawk Agent : ${nextLine}
```

where `${nextLine}` is the text of the error message in the log. `nextLine` is a label for values returned by the microagent method that gets information from the log file. Without variable substitution, you could include only static text, such as `High level alert` or a similar string, in the alert message.

How Variable Substitution Affects Actions

Action text strings can include variable references, where you include pertinent information from the data source in the alert text.

For example, the alert text:

```
Disk space on ${Instance} is at ${% Free Space}%
```

might display as:

```
Disk space on C: is at 10.2%
```

when generated. Or, if you call a script named `ClearTempFiles.exe` in an action whose data source provides information on disk partitions, you could specify the following command syntax:

```
ClearTempFiles.exe ${Instance}
```

and the agent will insert the name of the logical drive into the command line.

Variable substitution can cause actions to be taken more than once. If an action raises an alert with a variable reference, a new alert is generated at each test evaluation when the text message is different until the alert is cleared, even if the action that raises the alert was configured to take place only once.

Using Posted Conditions

A posted condition is an internal status message, similar to an alert message. Posted conditions are the result of actions in a rule, and can pass status information to other rules in the same rulebase. Each rule uses only a single data source for input, so the posted condition serves as a link between rules with different data sources. This allows you to test for conditions in more than one managed object.

To know more about Posted Conditions, see *TIBCO Hawk Concepts Guide*.

Chapter 5 **Working with Schedules and Period Groups**

This chapter contains simple examples that demonstrate the operations supported by schedules and period groups.

Topics

- [Overview, page 59](#)
- [Creating Schedules, page 64](#)
- [Creating Period Groups, page 70](#)
- [Testing Schedules, page 73](#)
- [Applying Schedules to Rulebase Objects, page 75](#)

Overview

A schedule is a configuration object that defines when a rulebase, rule, test or action is active. The schedule screen enables you to define a schedule and deploy the schedule. Then you can send the schedule to one or more TIBCO Hawk agents or Repositories, and apply the schedule to rulebase objects.

To know more about Schedules and Period Groups, see *TIBCO Hawk Concepts Guide*.

Sample Schedule and Period Groups

Sample schedules are bundled with TIBCO Hawk installation. They are installed in `<HAWK_HOME>\6.0\examples\rulebases` directory.



Any tag missing in the schedules XML will lead to error in use of the artifact. All the tags are mandatory in `schedules.hsf` file.

Schedules Screen Details

This section provides detailed description of the schedules screen.

Schedule Screen Sections

The schedules screen is divided into three sections:

- **Deployed Schedule:** Supports the following two tabs:
 - Schedule Items: Lists the schedules deployed to the agent.
 - Period Groups: Lists the period groups deployed to the agent.
- **User Schedule:** Supports the following two tabs:
 - Schedule Items: Lists the schedules created and saved locally by the user in the database.
 - Period Groups: Lists the period groups created and saved locally by the user in the database.
- **Schedule Item or Period Group:** This section displays schedule item details if you select Schedule Items tab in either the **Deployed Schedule** or **User Schedule** section. Similarly, it displays period item details if you select the Period Items tab in either the **Deployed Schedule** or **User Schedule** section.

This section is further divided in four sections:

- Time Of Day
- Week Day of Month
- Day of Month
- Month of Year

Toolbar Icons

The toolbar is located in the top right corner of the schedules screen.

Figure 16 Toolbar



When you hover over the icons in the toolbar, a popup tool tip describes the tool. The functionality of each icon is described in the following table:

Table 7 *Toolbar icons*

Toolbar Icons	Description
	Creates a new period group.
	Create a new schedule item.
	Creates a new period item.
	Refers to a existing period group.
	<p>Deletes either a schedule item or period item. It is enabled when the focus is on a schedule item and period item. The functionality is based on the active configuration object from the schedule items tree.</p> <ul style="list-style-type: none"> Deletes a schedule item if the selected object is a schedule item. Deletes a period item if the selected object is a period item.
	<p>Derives either a schedule item or period item. It is enabled when the focus is on a schedule item and period item. The functionality is based on the selected configuration object from the schedule items tree.</p> <ul style="list-style-type: none"> Derives a schedule item if the selected object is a schedule item. Derives a period item if the selected object is a period item.

Toolbar Icons	Description
	Edits schedule comment.
	Saves changes locally in the browser cache. It is enabled when the selected object is modified.
	Distributes one or more schedules on agents or repositories.
	Deploys the selected schedule or period group on the agent it belongs to.
	Saves the selected schedule item or period group. The functionality of this icon in the deployed and user sections is as follows: <ul data-bbox="244 916 776 1057" style="list-style-type: none">• Deployed section: Saves the schedule item or period group to the User section.• User section: Saves the schedule item or period group to the local database.
	Synchronizes your copy of the selected schedule item or period group with the copy on the agent.
	Tests the validity of schedule-items.
	Imports a schedule.

**Toolbar
Icons**

Description



Exports a schedule.

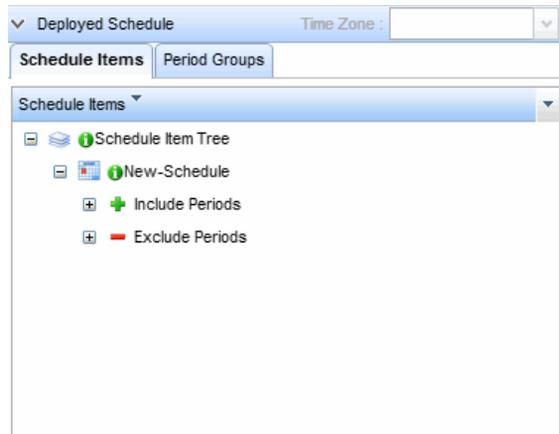
Creating Schedules

You define schedules in TIBCO Hawk WebConsole, then send schedules to agents and Repositories on your network.

To create a schedule:

1. Click an agent listed in the **Agents** portlet on the Hawk Dashboard tab. The **Agent** tab opens. The name of the **Agent** tab is same as the selected agent name *<agent_name>*. The following tabs are available on the top right corner of the screen:
 - a. Alerts (Default)
 - b. Rulebases
 - c. Schedules
 - d. Microagents
2. Click the **Schedules** tab. The Schedules screen displays the schedules defined for the selected agent.
3. Click the  **Create new schedule item** icon from the toolbar to create a new schedule item. Alternatively, click on the schedule item name in the Schedule Item Tree to edit an existing schedule item.
4. A new schedule item is created with the default name "New-Schedule". Double-click the default name field of the schedule item to edit the schedule name. This name must be unique within the schedule items list and cannot contain spaces.
5. Specify the Time Zone for the schedule item. By default, the time zone is set to GMT.

6. You can define inclusion periods and exclusion periods for a schedule-item. Click the arrow icon to expand the "New Schedule" tree. You can see the **Include Periods** and **Exclude Periods** options.



7. To define new inclusion period, select **Include Periods** and click the  **Create new period item** icon from the toolbar.
8. The **Schedule Item** section of the screen is enabled.



For a period item to be valid, you must select at least one interval in each category in the Schedule Item section: Time of Day, WeekDay of Week, Day of Month, and Month. If one or more categories has no selections, the schedule is not used.

9. The schedule item is defined in four parts. The detailed steps are provided in the following sections:
 - a. [Define Time of Day, page 66](#)
 - b. [Define Week Day of Month, page 67](#)
 - c. [Define Day of Month, page 68](#)
 - d. [Define Month of Year, page 69](#)
10. When the inclusion period is completely defined, click **Apply** button to save the changes locally. The **Schedule Item Tree** is updated with the new schedule item details.
11. You can define an exclusion period, if required. Select **Exclude Periods** from the **Schedule Item Tree** and click the  **Create new period item** icon from the toolbar.

12. Then repeat this procedure from [step 8](#) to [step 10](#), selecting the times, days, dates and months that the schedule item should not include.

For example, to create a schedule item of 9:00 AM to 11:59 AM and 1:00 PM to 4:59 PM, you can define an inclusion period of 9:00 AM to 4:59 PM, then add an exclusion period of 12:00 PM to 12:59 PM.

13. Click **Apply** to save the changes locally. The **Schedule Item Tree** is updated with the new schedule item details. At this point the schedule definition is stored in browser cache on the current machine.
14. Click the  **Deploy** icon from the toolbar to deploy the schedule updates on the corresponding agent or in the Repository. When the schedule is updated on the agent, the schedule will take effect immediately.

Define Time of Day

Perform the following steps to define the **Time of Day** values in the **Schedule Item** section:

1. Select one or more **Time of Day** intervals. There are four pre-defined intervals available:
 - **All Day** 12:00 AM to 11:19 PM
 - **AM** 12:00 AM to 11:59 AM
 - **PM** 12:00 PM to 11:59 PM
 - **9 to 5** 9:00 AM to 4:59 PM

By default, the 9 to 5 interval is displayed when the a new period item is added. You can select a different pre-defined interval by clicking on the **Options** drop-down list. The selected interval is displayed in the **Time of Day** section title bar and in the **Start** and **End** fields. (Clicking on a pre-defined interval also resets the **Start** and **End** times to the selected interval.)

2. To create a custom interval, double-click the time interval entry and type new values for **Start** and **End** fields.
3. To add another interval, click the  **Add new time period** icon on the **Time of Day** section title bar. Another set of Start and End intervals are added. If an interval is selected when you click the  **Add new time period** icon, the new interval is placed after the selected interval. If no interval is selected when you click the  **Add new time period**, the new interval is placed before the existing intervals.

By default, the new interval is the largest possible interval that can be specified. You can click in these fields to modify the interval, if desired.

- To delete an interval, select the interval and click the  **Delete** icon.



You can have as many intervals as desired, but the intervals cannot overlap in time and cannot span more than 24 hours. For example, if the first interval starts at 12:00 AM, the last interval must end at 11:59 PM or earlier.

Go back to [step 9](#) of [Creating Schedules](#) section.

Define Week Day of Month

Perform the following steps to define the **Week of Day** values in the **Schedule Item** section:

- Select the days of the week and weeks in the month for the interval in the **Week Day of Month** section.
- There are five pre-defined sets of days available on the Options drop-down list. You can also click an individual day to select it:
 - **Weekday** Mondays through Fridays
 - **Weekend** Saturdays and Sundays
 - **Select All** Every day
 - **Unselect** Unselects all days
 - **Toggle Selection** Selects the days that are currently not selected, and unselects the selected day
- There are five weeks available, selected by clicking the appropriate checkbox:
 - **First** The first occurrence in the month of the checked day(s)
 - **Second** The second occurrence in the month of the checked day(s)
 - **Third** The third occurrence in the month of the checked day(s)
 - **Fourth** The fourth occurrence in the month of the checked day(s)
 - **Last** The last occurrence in the month of the checked day(s).

The following selection shows the first and third Mondays in the month are selected:

Example 2, below, shows the last Saturday of every month is selected:

Go back to [step 9](#) of [Creating Schedules](#) section.

Define Day of Month

Perform the following steps to define the **Day of Month** values in the **Schedule Item** section:

1. Select the dates of the month for the schedule from the **Day of Month** section. There are three pre-defined sets of dates:
 - **Select All** The first through the last date of the month
 - **Unselect** Unselects all dates
 - **Toggle Selection** Selects the dates that are currently not selected, and unselects the selected dates
2. You can also select **Last** to indicate the final day of the month. The last day of the month may be the 28th, 29th, 30th, or 31st, depending on the month and if the year is a leap year.

By default, all dates are selected when a new schedule item is added.

3. The WeekDay of Month and Day of Month selections must overlap in order for a day to be selected.

For example, Example 1 of the previous step shows the first and third Mondays of each month selected. Because these days fall on a different date each month, the selected Days of Month would have to include the 1st through 22nd in order to make sure the first and third Mondays fell on selected dates.

In Example 2 of the previous step, the selected Days of Month would have to include the 22nd through the 31st to ensure the last Saturday fell on a selected date.

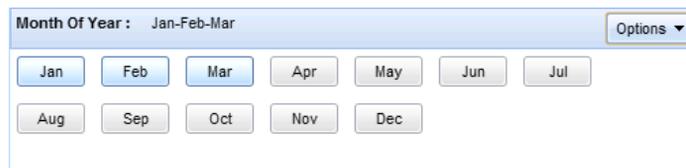
Go back to [step 9](#) of [Creating Schedules](#) section.

Define Month of Year

Perform the following steps to define the **Month of Year** values in the **Schedule Item** section:

1. Select the month to be included from the **Month of Year** section. There are seven pre-defined sets of months:
 - **First Quarter** January through March
 - **Second Quarter** April through June
 - **Third Quarter** July through September
 - **Fourth Quarter** October through December
 - **Select All** All twelve months
 - **Unselect** Unselects all months
 - **Toggle Selection** Selects the months that are currently not selected, and unselects the selected months

In the following example, January, February and March are selected.



The screenshot shows a web interface for selecting months. At the top, it says "Month Of Year : Jan-Feb-Mar" with an "Options" dropdown menu. Below this, there are two rows of buttons representing the months of the year. The first row contains buttons for Jan, Feb, Mar, Apr, May, Jun, and Jul. The second row contains buttons for Aug, Sep, Oct, Nov, and Dec. The buttons for Jan, Feb, and Mar are highlighted in blue, indicating they are selected. The other buttons are in a light gray color.

Go back to [step 9](#) of [Creating Schedules](#) section.

Creating Period Groups

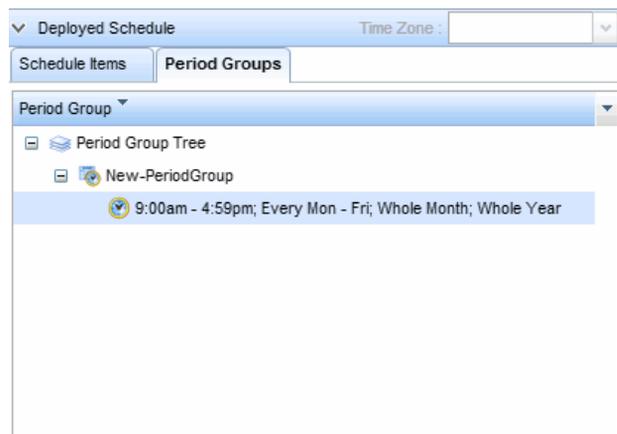
Period groups are useful when you use a set of periods regularly in defining schedules. It also eases the maintenance of those schedules because you can make a change in the period group and have it automatically reflected in all the schedules that use it.

For example, you can create a period group for Holidays and use it in any schedule that is affected by holidays, such as a work schedule or a delivery schedule. If any holidays change or are added throughout the year, you only need to update the Holidays period group. All schedules that use the Holidays period group will get the updated list of holidays.

Period groups can be added to an inclusion or exclusion period. When used in the inclusion or exclusion period, a period group is in-schedule only if all of its periods are in-schedule. Otherwise, a period group is out-of-schedule.

When a period group is added to the list, the periods in the period group are listed indented below the period group name in the list. The period group in the inclusion or exclusion list cannot be derived but can be edited or deleted from list. Periods in a period group in the list cannot be added or be deleted.

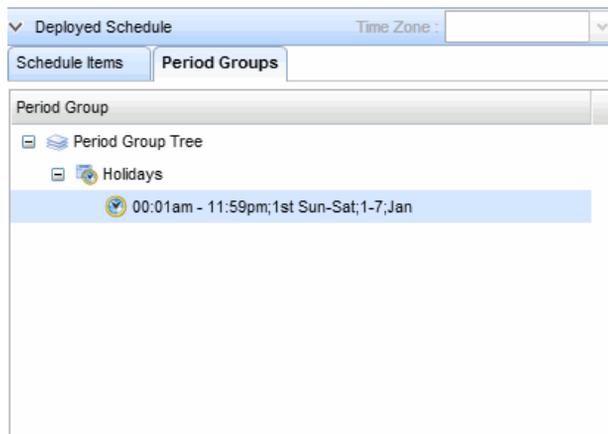
1. On the **Deployed Schedule** section, click the **Period Groups** tab to display the current period groups.



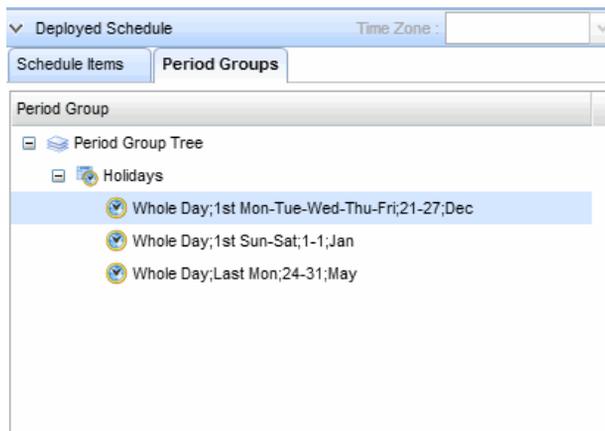
2. Click the  **Create new period group** icon on the toolbar to create a new period group. Alternatively, click the period group name from the Period Group Tree to edit an existing period group.

3. A new period group is created with the default name "New-PeriodGroup". Double-click the default name of the period group to edit the name. This name must be unique within the Period Group list and cannot contain spaces.
4. Click the  **Create new period item** icon from the toolbar. A new period item is added to the Period Group Tree and the **Period Group** section of the schedules screen is enabled for editing the period item.
5. Set the schedule item of the first item in the period group. For example, you might set Whole Day; 1st Sun-Sat; 1 - 7; Jan to schedule every January 1.

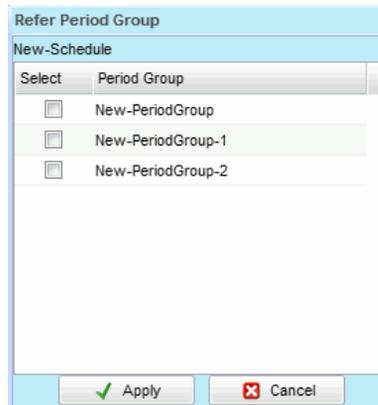
The Period Group Tree contains the first period group, such as:



6. Click the  **Create new period item** again to create the next schedule item. Continue until all scheduled items are created. For example, if you are creating a Holidays period group, the list may look like this:



7. Click the **Apply** button to save changes locally. At this point the period group definition is stored in browser cache on the current machine.
8. Click the  **Deploy** icon from the toolbar to deploy the period group updates to the current agent. When the period group is updated on the agent, it will take effect immediately.
9. To apply the period group to a schedule item, display the existing schedule items by clicking the **Schedule Items** tab. The Schedule Item Tree is displayed.
10. Select a schedule item and click either **Include Periods** or **Exclude Periods**. Note that the  **Refer to a period group** icon is enabled on the toolbar.
11. Include or exclude the period group:
 - To include the period group in the existing schedule item, select **Include Periods** and click the  **Refer to a period group** icon on the toolbar.
 - To exclude the period group, select **Exclude Periods** and click the  **Refer to a period group** icon on the toolbar. In this example, the period group Holidays is excluded from the WorkSchedule schedule item.
12. Select the appropriate period group from the **Refer Period Group** dialog and click **Apply**.



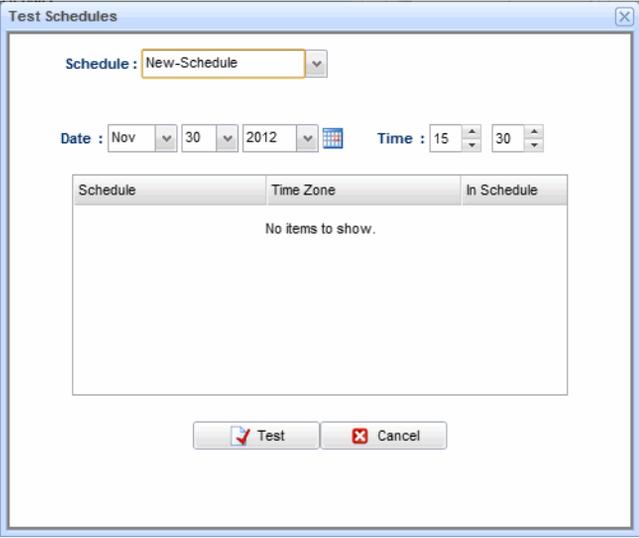
13. The period group is added to the existing schedule item in the **Include Periods** or **Exclude Periods** section. The scheduled items in the period group are indented under the period group name.
14. Click the **Apply** button to save the changes locally. At this point the changes to the schedule definition are stored in browser cache on the current machine.
15. Click the  **Deploy** icon on the toolbar to deploy the schedule updates to the current agent.

Testing Schedules

After creating a schedule, you can test it dynamically before applying it to rulebase objects. Testing allows you to specify a time and date, then determine which schedules are in-schedule or out-of-schedule at that particular time.

To test a schedule:

1. Select a schedule item from the Schedule Item Tree. This enables the  **Test Schedules** icon. Click the  **Test Schedules** icon on the toolbar.
2. The Test Schedules dialog displays:



Schedule	Time Zone	In Schedule
No items to show.		

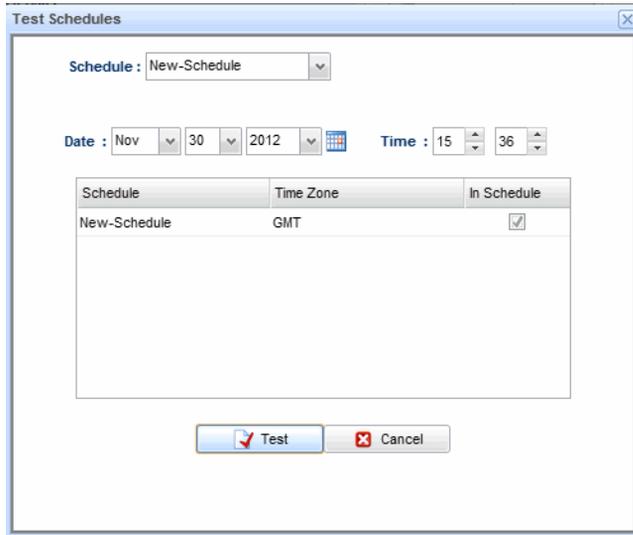
3. Select the name of a schedule to test from the **Schedule** drop-down list.
4. Specify the date to test in the **Date** field. You can also use the date chooser to select a date. The test will indicate whether each schedule is in-schedule on the selected date. The default value is the current date.
5. Specify the time to test in the **Time** field. The test will indicate whether each schedule is in-schedule at the selected time. The time is in a 24-hour format of hh:mm where hh is 00 to 24 and the default value is the current time.



While testing schedule items, the time zone setting for schedule item is considered.

6. Click **Test**.

The **Test Schedules** dialog displays the results of the test. Schedules that will be in-schedule at the specified date and time are marked with a check in the **In Schedule** column as shown:



7. Click **Cancel** to close the **Test Schedules** dialog.

Applying Schedules to Rulebase Objects

After defining a schedule in the Schedule Editor dialog, you can apply it to any rule, test or action in a rulebase. Rulebase objects are related through a hierarchy, so a schedule you apply to one object also affects objects below it in the hierarchy. For example, when you apply a schedule to a rulebase, its inclusion and exclusion periods affect all rules, tests and actions defined in that rulebase.

You apply schedules in the Rulebase, Rule, Test, and Action editors.

Chapter 6 **Working with Microagents**

This chapter contains simple examples that demonstrate the operations supported for microagents.

Topics

- [Overview, page 77](#)
- [Viewing Microagents and Invoking a Microagent Method, page 78](#)
- [Subscribing to a Microagent Method, page 82](#)
- [Viewing Subscription Results, page 83](#)

Overview

Microagents represent managed objects such as operating system subsystems, agent components, log files, event logs or applications. Each microagent exposes a set of methods to the agent that the agent uses to collect information and take action.

To know more about Microagents, see *TIBCO Hawk Concepts Guide*.

In TIBCO Hawk WebConsole, you can view microagents and their methods for any discovered TIBCO Hawk agents. The following operations are supported for microagents:

- Viewing Microagents and Invoking Microagent Method
- Subscribing to a Microagent Method
- Viewing Subscription results

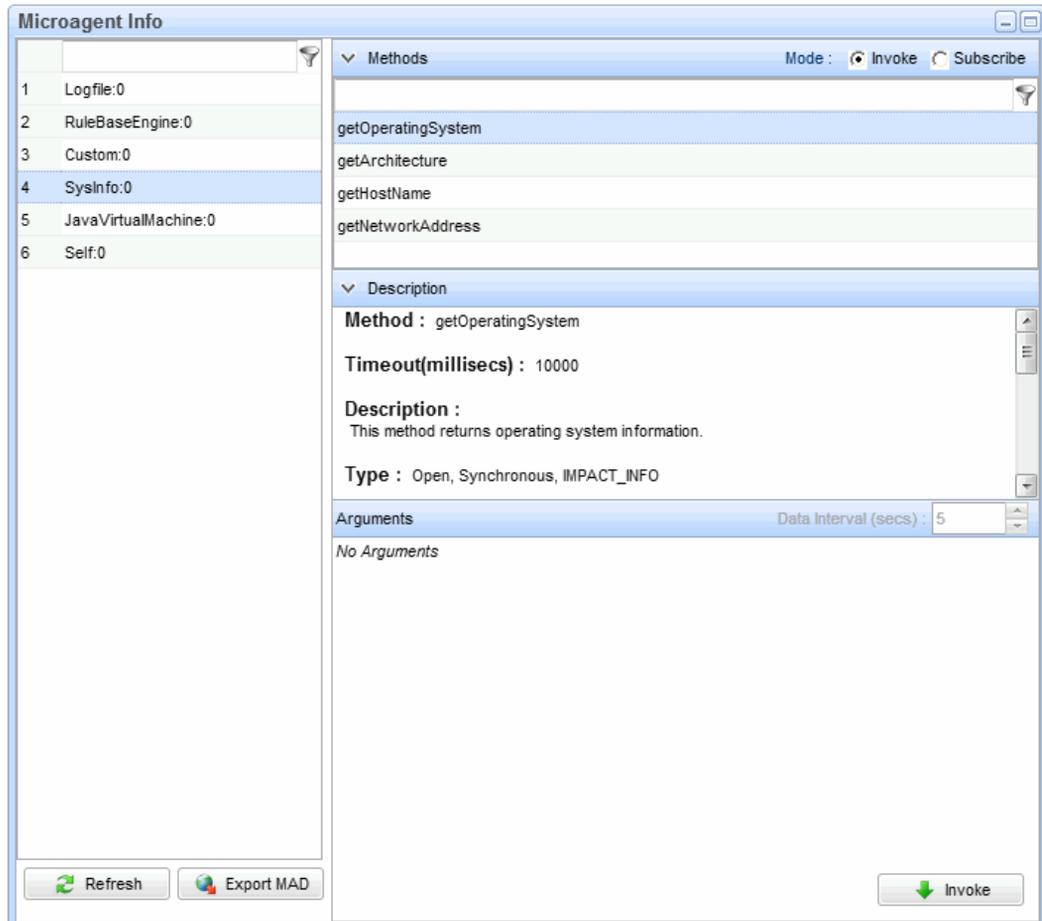
The examples discussed in the following sections demonstrates these operations.

Viewing Microagents and Invoking a Microagent Method

The following example invokes the `getMicroAgentInfo()` method of the `Self` microagent:

1. Click an agent listed in the **Agent** portlet on the Hawk Dashboard. The **Agent** tab opens. The name of the **Agent** tab is same as the selected agent name `<agent_name>`. The following tabs are available on the top right corner of the screen:
 - a. Alerts (Default)
 - b. Rulebases
 - c. Schedules
 - d. Microagents
2. Click the **Microagents** tab. The **Microagent Info** screen is displayed. This screen is divided in four panels.
 - **Microagent Info**: Lists the microagents you can access on the current agent.
 - **Methods**: List the methods supported by the selected microagent and the mode of operation.
 - **Description**: Provides detailed help description for the selected method.
 - **Arguments**: Provides the arguments required for the microagents.

Figure 17 Microagent Info Screen



3. Click **Self** from the microagent list to display the list of methods in the **Methods** panel. The **Self** microagent represents the TIBCO Hawk agent itself. Its methods generally provide information about the agent.
4. Click the **getMicroagentInfo** method from the list displayed in the **Methods** panel. A detailed help description is displayed in the **Description** panel.

Methods can return information about a managed object, take an action that modifies the managed object, or both.

The **getMicroAgentInfo** method returns the names of microagents currently active on this agent. It takes a test string for microagent name as an optional argument, but if no name is specified it returns a list of all microagents for this agent. Invoking this method without arguments returns a list similar to the

microagent list in the dialog. For detailed descriptions of default microagent methods, see the *TIBCO Hawk Microagent Reference*.

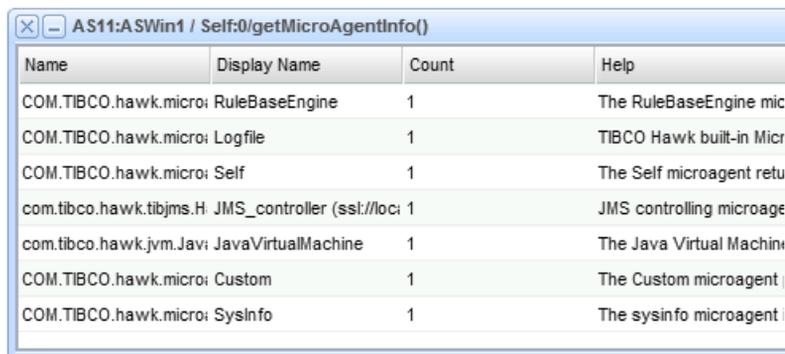
5. You need to specify the mode of operation of microagents. The following two modes are available on the **Methods** panel title bar:
 - **Invoke**: Use the Invoke mode to immediately view the results. Invoking is useful when you want to test a method before using it in a rule, or to check a return value for troubleshooting purposes. This example demonstrates the Invoke mode of operation.
 - **Subscribe**: Use the Subscribe mode to view microagent method results over time. Creating a subscription is useful when you want to test a range of return values before specifying boundaries in a rule, or to identify general patterns of activity. The example for Subscribe mode is demonstrated in section [Subscribing to a Microagent Method, page 82](#).

Select the **Invoke** radio button from the **Methods** panel title bar. For more information on any synchronous and asynchronous methods, see the *TIBCO Hawk Microagent Reference*.

6. Click the  button.

The Invocation Results window displays the results returned by the method:

Figure 18 Invocation Results Window

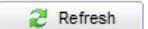


Name	Display Name	Count	Help
COM.TIBCO.hawk.micro: RuleBaseEngine	RuleBaseEngine	1	The RuleBaseEngine mic
COM.TIBCO.hawk.micro: Logfile	Logfile	1	TIBCO Hawk built-in Micr
COM.TIBCO.hawk.micro: Self	Self	1	The Self microagent retu
com.tibco.hawk.tibjms.H JMS_controller (ssl://loc	JMS_controller (ssl://loc	1	JMS controlling microage
com.tibco.hawk.jvm.Javi: JavaVirtualMachine	JavaVirtualMachine	1	The Java Virtual Machin
COM.TIBCO.hawk.micro: Custom	Custom	1	The Custom microagent
COM.TIBCO.hawk.micro: SysInfo	SysInfo	1	The sysinfo microagent

Methods can return either a single record with one or more fields, or tabular data with one or more columns and rows. This method returns tabular data with each row corresponding to a separate component.



To sort tabular results in the window, click on a column header. There are multiple column display options available.

7. Click the  button in the **Microagent Info** panel to refresh the list of microagents.

8. Click the  button to export the microagent definition on the local machine. The microagent definition is saved as a `logfile.hmd`.

Subscribing to a Microagent Method

To subscribe to `getSystemInfo()` method of the **System** microagent:

1. Click an agent listed in the **Agent** portlet on the Hawk Dashboard. The **Agent** `<agent_name>` tab opens.
2. Click the **Microagents** tab. The **Microagent Info** screen is displayed.
3. Click **System** microagent from the microagent list. The supported methods are displayed in **Methods** panel.
4. Click the `getSystemInfo` method from the **Methods** panel. A detailed help description is displayed in the **Description** panel.

This method returns the process management information, including process swapping and run queuing on the current TIBCO Hawk agent machine. The method is synchronous, so you must specify a time interval for collecting data points. The default interval is 5 seconds.

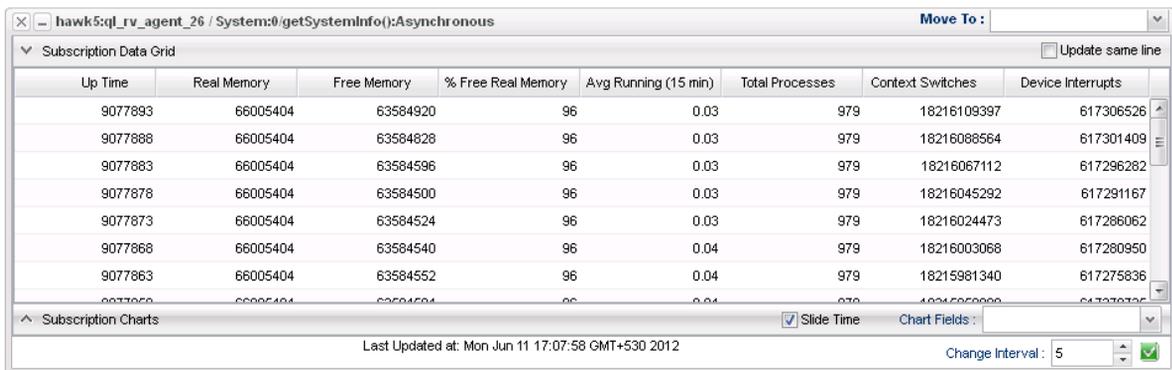
5. Specify a **Data Interval** (how often you want to call the method) of 15 seconds. You can change the value by using the spinner or type the value.

Asynchronous methods do not require a **Data Delivery Interval**.

Asynchronous methods continue to send information as it becomes available until the subscription is ended.

6. Verify that the **Subscribe** radio button is selected on the **Methods** panel title bar.
7. Click the  button. The Subscription Results window is displayed:

Figure 19 Subscription Results Window - Data Grid View



Up Time	Real Memory	Free Memory	% Free Real Memory	Avg Running (15 min)	Total Processes	Context Switches	Device Interrupts
9077893	66005404	63584920	96	0.03	979	18216109397	617306526
9077888	66005404	63584828	96	0.03	979	18216088564	617301409
9077883	66005404	63584596	96	0.03	979	18216067112	617296282
9077878	66005404	63584500	96	0.03	979	18216045292	617291167
9077873	66005404	63584524	96	0.03	979	18216024473	617286062
9077868	66005404	63584540	96	0.04	979	18216003068	617280950
9077863	66005404	63584552	96	0.04	979	18215981340	617275836
9077858	66005404	63584554	96	0.04	979	18215959990	617270722

Subscription Charts Slide Time Chart Fields:

Last Updated at: Mon Jun 11 17:07:58 GMT+530 2012 Change Interval: 5

Viewing Subscription Results

You can view the results of a method subscription in a data grid or as a chart. These viewing modes are useful for identifying patterns over time.

To view a results of a subscription, we will continue working on the same example used to subscribe to a microagent method.

1. The Subscription Results window (Figure 19) has two viewing options:
 - **Subscription Data Grid:** Displays the subscription results in tabular form.
 - **Subscription Charts:** Displays the subscription results in form of charts.

These options are displayed as cascading windows. Only one option is active at a time.



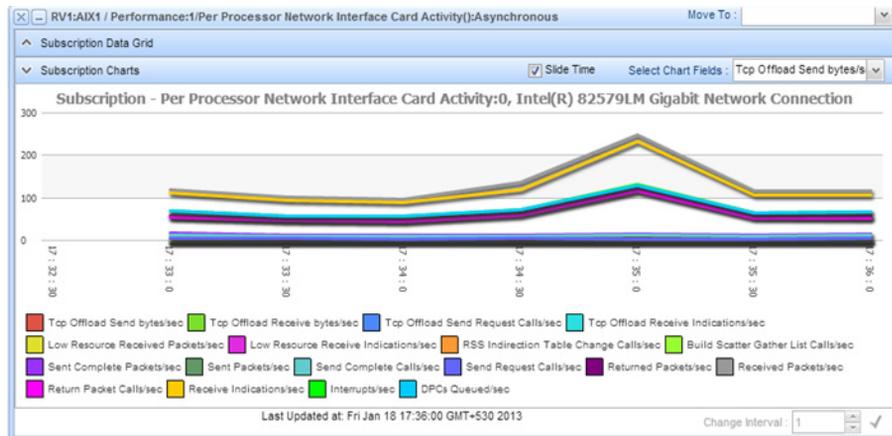
If the Subscription Results windows contains more than one row, each row usually describes one instance in a set — one storage device, one process, one application.

2. Click the **Subscription Data Grid** window title bar to view the tabular results of the subscription. The subscription results appear at regular intervals (if the method you called was synchronous) or as information becomes available (if the method you called was asynchronous).

When new results are retrieved, they replace the previous results in the dialog. Over time, the values will fluctuate in all fields except Instance.

3. Select the **Update same line** checkbox to view the subscription results updates in the same line. This feature can be used for microagents that return subscription results in a single row (as composite data). If the **Update same line** is not checked, the subscription results are displayed in multiple rows.
4. Click the **Subscription Charts** window title bar to view the subscription results as a chart.
5. The **Subscription Charts** window title bar has the **Select Chart Fields** drop-down list. Select the rows to chart on the **Select Chart Fields** drop-down list by selecting the row checkbox. Subscribed data begins to appear in the charts.

Figure 20 Subscription Results Window - Charts View



Data points are added dynamically, as they are returned by the method. The chart automatically assigns a range to the Y axis based on the values collected; the X axis always represents time.

Since the field to chart is a numeric result field, both Chart View and Tabular View tabs are active. If the result contained only text, you would see a table.

6. You can select a different row in the **Select Chart Fields** drop-down list and view the subscription for that instance at the same time. This allows you to compare data between different agents or microagent methods. For example, if you have two versions of an instrumented application (one with a new, faster algorithm and one with an older algorithm), you can compare performance of the versions. All charts are active for as long as the subscription exists.
7. Select the **Slide Time** checkbox to see the graphs in a sliding window with the facet representing time sliding. If the **Slide Time** checkbox is cleared, you can see a consolidated average result over all time periods in the window.
8. To change the data interval use the **Change Interval** text box. You can either type a new value or change the value using the spinner.
9. Click the  icon to apply the changes.

Moving Subscription Results Window to Custom Dashboards

Hawk WebConsole provides the ability to create custom dashboards. You can move the microagent **Subscription Results** window to the custom dashboards.

To create a custom dashboard:

1. Go to the Hawk dashboard screen. Click the  tab. The **Enter dashboard name** dialog is displayed.
2. Specify a unique name for the new dashboard and click **OK**.
3. A new dashboard tab is created with the specified name.

To move Subscription Results Window to the custom dashboard:

1. On the **Subscription Results** window, click the **Move to** drop-down list. The user created custom dashboards are listed.
2. Select the required dashboard from the drop-down list to move the **Subscription Results** window to a user created custom dashboard.

Chapter 7 **Working with Network Query**

This chapter describes how to perform network queries and network actions on multiple remote TIBCO Hawk agents using TIBCO Hawk WebConsole.

Topics

- [Performing Network Queries, page 87](#)
- [Performing Network Actions, page 92](#)

Performing Network Queries

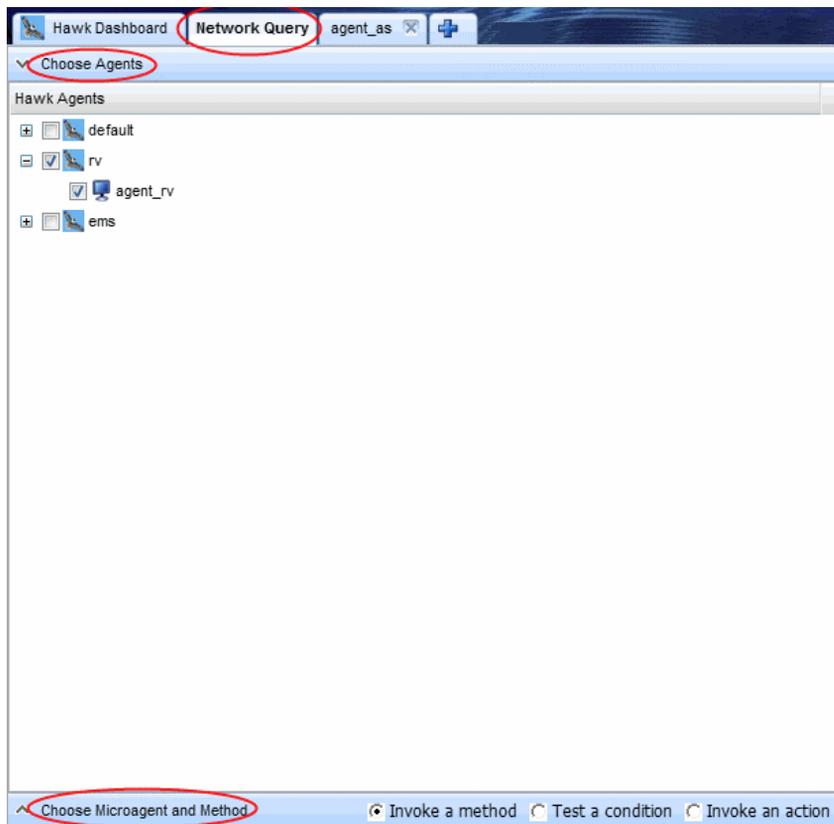
Network query allows you to communicate with multiple TIBCO Hawk agents at one time. Network query feature enables you to ask multiple agents on the network any question which you can ask an individual agent. Agents can be grouped by domains or by the response to a prior question. The query is broadcast to every agent on the network, but only specific agents on the target list process the query and respond.

A network query has some of the same structural elements as a rule. However, parameters are specified dynamically, not stored in a rulebase. In TIBCO Hawk WebConsole, you specify a data source for the query, which is a microagent method with optional arguments. You can also specify an optional test to filter the result set.

The following example performs a network query by invoking the `SysInfo:getOperatingSystem` method on a agent. It includes a test that checks the `OS Name` parameter to determine if the agent is running on Linux operating system.

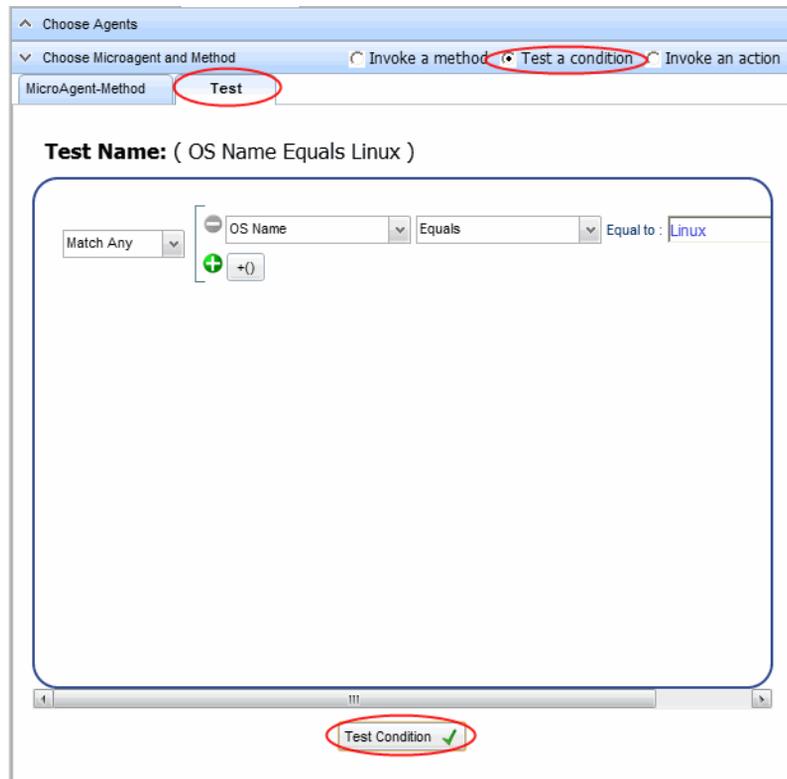
To query remote agents on the network:

1. Select the **Network Query** tab from the TIBCO Hawk WebConsole screen. The Network Query screen is displayed.



2. Go to the **Choose Agents** panel to select one or more Hawk agents. The list of discovered agents is displayed. Click to expand the domains until a list of individual agents is displayed.
3. Select the agent checkbox to include an agent in the target list. To remove an agent, clear the checkbox.
4. Click **Choose Microagent and Method** panel. TIBCO Hawk WebConsole looks up microagents and methods for the selected agent, and displays them in the **MicroAgent-Method** tab.
5. Select **Invoke a method** radio button from **Choose Microagent and Method** panel title bar to invoke a method. This radio button is the default selection.
6. Select the **SysInfo** microagent and **getOperatingSystem** method. This method returns operating system details for the target agents. For instructions on invoking a microagent, see [Viewing Microagents and Invoking a Microagent Method, page 78](#).

7. Click the  **Execute** icon to invoke the method. The results are displayed in the **Results** section of the screen.
8. You can add a test to filter the results of the method invocation. To add a test to a network query, select the **Test a condition** radio button from **Choose Microagent and Method** panel title bar. The **Test** tab is enabled.
9. Click the **Test** tab to specify the test parameters. The **Test** tab screen looks like the following:



A test is applied to microagent method results for every agent in the target list. This test expression specified in the above example returns all agents running on Linux operating system. For instructions on building test expressions, [Creating a Test, page 39](#).



Advanced test options, such as clear timers and clear tests, are not supported for network query.

10. Click **Test Condition**. While agents are queried, the message **Request Sent** is displayed in the **Query Result** column of the results table. It might take several seconds for all agents in the target list to respond.
11. The results of the network query are displayed in the **Results** section of the screen:

Results Set Targets

MicroAgent: COM.TIBCO.hawk.microagent.SysInfo

Method: getOperatingSystem

Test: (OS Name Equals Linux)

T	Agent Name	Hawk Domain	Query Result
▲	Anirudh Test Result : true	default	Success
▲	Agent_rv Test Result : false	default	Success
▲	test Test Result : false	default	Success
▲	AgentRv1 Test Result : true	default	Success
▲	naman	default	Timeout

Success = 4 , Error = 0 , Timeout = 1

12. The Results screen displays the following details:
 - **MicroAgent**
 - **Method**
 - **Test**
 - **Results Table:** The list of agents responding to the query are displayed in a table. The table displays the Agent Name, Hawk Domain, and Query Result.

The status bar displays the number of agents with **Success**, **Error**, and **Timeout** status.

- **Success**: Agents that received the query and responded successfully.
 - **Error**: Agents that could not process the query.
 - **TimeOut**: Agents that did not respond, usually due to a network problem.
13. For network queries with no test, a list of agents responding to the query are displayed. Click the  icon for any agent in the list to view query results. Results can be a simple value or a table with result records (rows) and result fields (columns).
 14. For network queries with a test, click the  icon for any agent in the list to view the test result details. Like tests in a rule, only a `true` or `false` value is returned.
 15. You can mark any agents listed in the results table as the subject of a second query or a network action. Select the agents from the Results section and click **Set Targets**. The selected agents replace the agents on the target list. The next query or action is sent only to those agents.

Performing Network Actions

A network action is similar to a network query, except that instead of gathering information, you specify an action to perform on one or more remote agents.

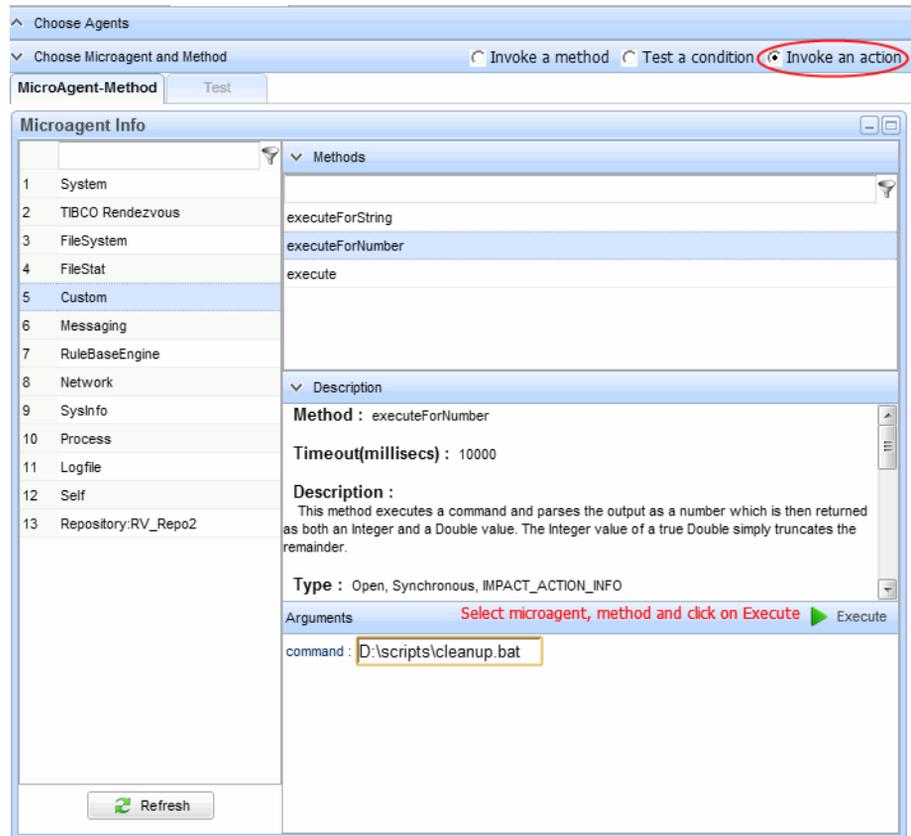
The action can be any task a microagent method of type `IMPACT_ACTION` or `IMPACT_ACTION_INFO` can perform. Examples of tasks you might perform as a network action include loading rulebases, starting and stopping processes, or executing custom scripts.



Since performing a single network action can have a widespread effect, use this feature with caution.

To perform a network action:

1. Follow the steps specified in [Performing Network Queries, page 87](#) section for accessing the Network Query screen and specifying a list of target agents.
2. Click **Set Targets** to add agents that tested true during the network query to the target list for a network action.
3. Click the **Invoke an action** radio button from the **Choose Microagent and Method** panel title bar. The **Microagent-Method** tab lists only methods that perform an action (`IMPACT_ACTION` and `IMPACT_ACTION_INFO`).
4. Click the **Custom** microagent and the **executeForNumber()** method.
5. Type the full path of a script to execute, such as `cleanup.bat`, in the **Command** field. This might be a script that deletes core and temporary files on the local machine.



6. Click  **Execute**. The results screen details are same as **Invoke a method** results.
7. Click the  icon for any agent in the list to view action results.

Network query and action are interactive tools, so conditions you identify and solve in this manner might return without warning. If a particular network query repeatedly yields a true result, you should consider formalizing query and action steps in a rulebase. The rulebase could contain a single rule to check for disk space usage greater than 90 percent, then call the cleanup script. A second action could notify you by email that the script was run. You could distribute this rulebase to all agents, or just agents with a pattern of high disk space usage. For more information, see [Working with Rulebases, page 27](#).

Chapter 8 Working with Rulebase-Maps

This chapter describes the advanced TIBCO Hawk rulebase feature like Rulebase-Map and shows how to create different type of mappings using TIBCO Hawk WebConsole.

Topics

- [Overview, page 95](#)
- [Rulebase-Map Screen Details, page 96](#)
- [Creating a Rulebase-Map, page 99](#)
- [Creating User-defined Agent Groups, page 100](#)
- [Mapping Groups to Agents or Groups, page 102](#)
- [Mapping Rulebases to Agents or Groups, page 104](#)
- [Mapping Groups to External Commands, page 106](#)
- [Distribute Rulebase Map to Repositories, page 108](#)

Overview

A rulebase-map is a configuration object that maps rulebases to TIBCO Hawk agents on your network. It directs TIBCO Hawk agents or groups of agents on your network to load particular rulebases at startup. For example, using a rulebase-map you can instruct an agent to load a rulebase designed specifically for the operating system where it runs.

To efficiently manage agent configuration, an entire enterprise should use the same rulebase-map. Configuration objects access the rulebase-map using the same configuration source they use to access rulebases.



Rulebase-maps are supported only when running in Manual Configuration mode. For more information, see *TIBCO Hawk Installation Configuration and Administration Guide*.

When creating a rulebase-map, you typically group agents on your network according to rulebase requirements. Then you map individual rulebases to agents and groups of agents and distribute the rulebase-map to other repositories. Agents in Manual Configuration mode load the rulebase-map when started to determine which rulebases they require. Agents then proceed to load these rulebases, if they exist in the configuration source.

Sample Rulebase-Map

Sample rulebase-map is bundled with TIBCO Hawk installation. They are installed in `<HAWK_HOME>\6.0\examples\rulebases` directory.

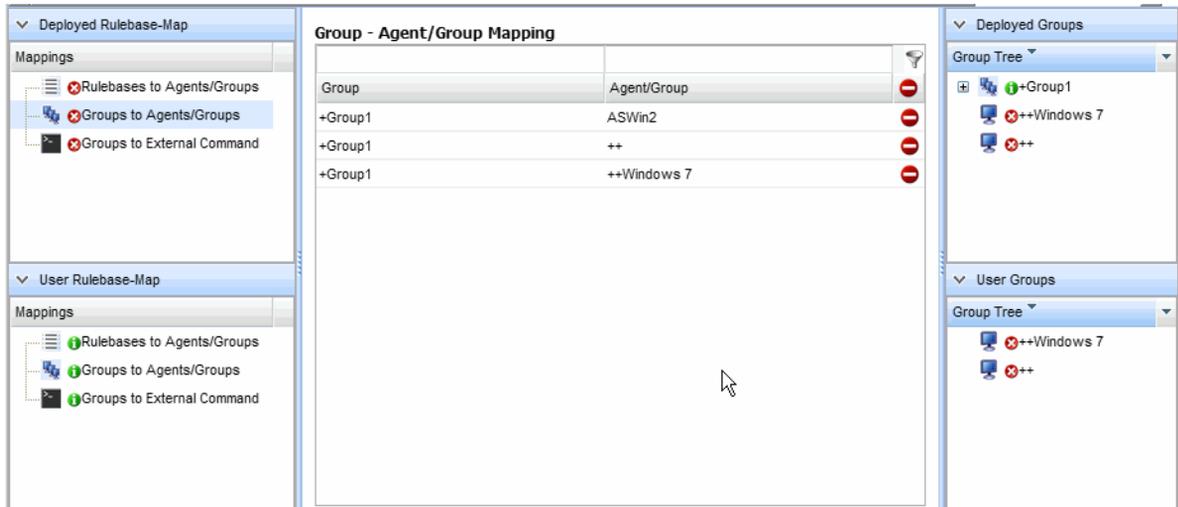


Any tag missing in the rulebase-map XML will lead to error in use of the artifact. It is mandatory to include the `<Name></Name>` tag in the `.hrm` file of rulebase-map.

Rulebase-Map Screen Details

The Rulebase-Map screen is divided into following parts:

- **Deployed Rulebase-Map**
- **User Rulebase-Map**
- Mapping section: This section displays the mapping options based on the mapping object selected in either the **Deployed Rulebase-Map** section or **User Rulebase-Map** section. The following mapping options are available:
 - Rulebase to Agent/Groups: Refer [Mapping Rulebases to Agents or Groups, page 104](#) for further instruction.
 - Groups to Agents/Groups: Refer [Mapping Groups to Agents or Groups, page 102](#) for further instruction.
 - Group to External Command: Refer [Mapping Groups to External Commands, page 106](#) for further instruction.
- **Deployed Groups**
- **User Groups**



Toolbar Icons

Toolbar is located in the top right corner of the screen.

Figure 21 Rulebase-Map Toolbar



When you hover over the icons in the toolbar, a popup tool tip describes the tool. The functionality of each icon is described in the following table:

Toolbar Icons	Description
	Creates a new agent group on the repository.
	Creates groups to groups/agents mapping.
	Creates rulebases to groups/agents mapping.
	Creates groups to external command mapping.
	Saves changes locally in the browser cache.
	Deploys the rulebase-map on the repository.
	<p>Deployed section: Saves the rulebase-map to the User section.</p> <p>User section: Saves the rulebase-map to the local database.</p>

Toolbar Icons	Description
	Synchronizes your copy of the rulebase-map with the copy on the WebConsole server.
	Deploys the rulebase-map to other repositories from various domains.
	Import a rulebase-map.
	Export a rulebase-map.

Creating a Rulebase-Map

You create and deploy a rulebase-map using TIBCO Hawk WebConsole. Agents running in Manual Configuration mode with the Repository or Configuration Path options can use the rulebase-map the next time they are started.

To create or edit a rulebase-map:

1. Click the repository listed in the **Repository** portlet on the Hawk Dashboard tab.
2. A new repository tab opens. The name of the repository tab is same as the selected repository name *<repository_name>*. The following tabs are available on the top right corner of the screen:
 - a. Rulebase (Default)
 - b. Schedules
 - c. Rulebase-Map
3. Click the **Rulebase-Map** tab. The Rulebase-Map screen is displayed.
4. You can create the following mappings from this screen:
 - Rulebase to Agent/Groups: Refer to [Mapping Rulebases to Agents or Groups](#)
 - Groups to Agents/Groups: Refer to [Mapping Groups to Agents or Groups](#) for further instructions.
 - Groups to External Commands: Refer to [Mapping Groups to External Commands](#) for further instructions.
5. When the rulebase-map updates are complete, click  **Deploy** icon from the toolbar to deploy the rulebase-map changes to the repository.

Creating User-defined Agent Groups

Agent groups are sets of TIBCO Hawk agents on the same network with similar rulebase needs. Groups are more efficient when mapping rulebases to agents in a rulebase-map. Instead of assigning a rulebase to every individual agent that needs it, you can assign it once to the group.

Every rulebase-map can have two types of groups: user-defined and automatic. Automatic groups, defined for you in TIBCO Hawk, consist of operating system groups. The set of operating system groups includes one group for each operating system.

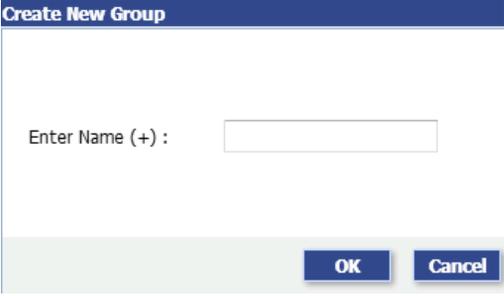
You can also define groups in TIBCO WebConsole. User-defined groups are optional, and can include any combination of individual agents and other agent groups. For example, a user-defined group might include one or more automatic groups.

All group names begin with a plus (+) character. Operating system groups are named ++<OS>, where *OS* is the name of the operating system.

To create a user-defined agent group:

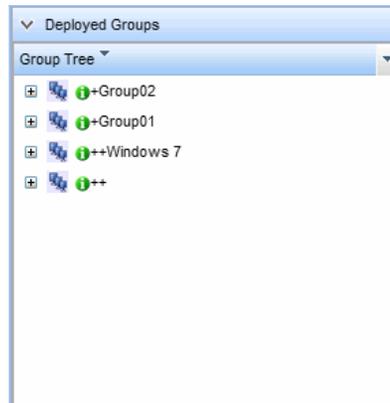
1. Click the repository listed in the **Repository** portlet on the Hawk Dashboard tab.
2. A new repository tab opens. The name of the repository tab is same as the selected repository name <repository_name>. The following tabs are available on the top right corner of the screen:
 - a. Rulebase (Default)
 - b. Schedules
 - c. Rulebase-Map
3. Click the **Rulebase-Map** tab. The Rulebase-Map screen is displayed.

4. Select **Group to Agent/Groups** option from the Mappings tree and click  **Create new group** icon from the toolbar. The **Create New Group** dialog is displayed.



The image shows a dialog box titled "Create New Group". It has a dark blue header bar with the title. Below the header is a light gray area with the text "Enter Name (+) : " followed by a white text input field. At the bottom of the dialog, there are two buttons: "OK" and "Cancel", both with dark blue backgrounds and white text.

5. Type the name of the new group and click **OK** to create the new group.
6. The new groups are also displayed in the Group Tree of **Deployed Groups** section.



Ensure that every user-defined agent group has an active mapping to either an agent, rulebase, or command. All user-defined agent groups without any mapping are deleted when a rulebase-map is deployed to the repository.

Mapping Groups to Agents or Groups

Mapping groups with agents allows you to add individual agents to one or more agent groups. You should add agents to groups that consist of agents with similar rulebase requirements.

To map a group to an agent or group:

1. On the Rulebase-Map screen, select the **Groups to Agents/Groups** option from the Mappings tree and click the  **New Groups-Agents/Groups Mapping** icon from the toolbar. The Group to Agent/Group mapping options are displayed.
2. The **Group** drop-down list displays all the existing groups. You can select a single group or multiple groups by selecting the group name checkbox from the drop down list.



3. Similarly, the **Agent/Group** drop-down list displays all the existing agents and groups.

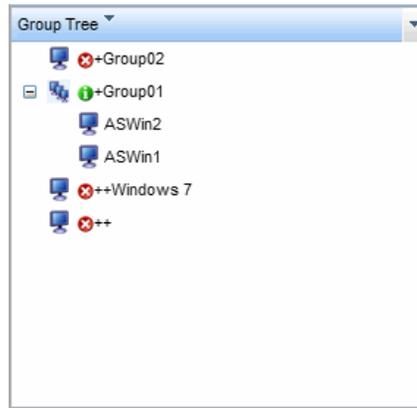


Select a single agent/group or multiple agents/groups to map to the group selected in the **Group** drop-down list.

4. Click the  icon from the toolbar to save the new mapping locally. The new mappings are added to the list.

Group	Agent/Group	
+Group01	ASWin1	
+Group01	ASWin2	

5. Click the  icon to remove a group to agent/group mappings from the list.
6. The new group to agent/group mappings are also displayed in the Group Tree of the **Deployed Groups** section.



The groups tree also indicates the mapping status of the groups.

-  - Represents that the group is mapped to any rulebase, command, or agent.
 -  - Represents that the group is not mapped.
7. Add other mappings to complete the rulebase-map.
 8. When the rulebase-map is complete, click  **Deploy** icon from the toolbar to deploy the rulebase-map updates to the repository.

Mapping Rulebases to Agents or Groups

After defining any agent groups, you map rulebases to agents and groups of agents. Mapping a rulebase to a single agent directs the agent to load that rulebase when it is started. Mapping a rulebase to a group of agents directs all agents in the group to load the rulebase when started.

To map a group to an agent or group:

1. On the Rulebase-Map screen, select the **Rulebases to Agents/Groups** option from the Mappings tree, click the  **New Rulebase-Agents/Groups Mappings** icon from the toolbar. The Rulebase to Agents/Groups mapping options are displayed.
2. The **Rulebase** drop-down list displays all the existing rulebases. You can select a single rulebase or multiple rulebases by selecting the rulebase name checkbox from the drop down list.



If the rulebase does not exist yet, click the **Rulebase** tab to create rulebases for the repository. For more information on creating rulebases, see [Creating a Rulebase, page 32](#).

3. Similarly, the **Agent/Group** drop-down list displays all the existing agents and groups.



Select a single agent/group or multiple agents/groups that should load the selected rulebase.

- Click  **Save changes locally** icon from the toolbar to save the new mapping locally. The new mappings are added to the list.

Rulebase	Agent/Group	
 New-Rulebase-1		
New-Rulebase-1	+Group01	
New-Rulebase-1	++Windows 7	
New-Rulebase-1	+Group02	

- Click  icon to remove a rulebase to agent/group mappings from the list.
- Add other mappings to complete the rulebase-map.
- When the rulebase map is complete, click  **Deploy** icon to deploy the rulebase map updates to the repository.



You need to make the following settings in the `hawkagent.cfg` file to ensure that the agent to rulebase mapping is functional.

- Omit the `-auto_config_dir` option.
- Set the `-repository_path` option to the repository where the rulebase is deployed.

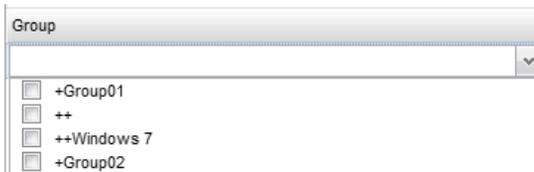
For more information on `hawkagent.cfg` file options, see *TIBCO Hawk Installation Configuration and Administration Guide*.

Mapping Groups to External Commands

An external command is an alternative or a complimentary mechanism to rulebase maps for identifying the rulebases to be loaded. You can map external commands to a group or a single agent. The output of the external command must return a list of rulebase names, which are the additional rulebases that will be loaded. An agent can use multiple such external commands, the results of which are merged into one list with the duplicates removed.

To map group to a command:

1. On the Rulebase-Map screen, select the **Groups to External Command** option from the Mappings tree, click the  icon. The Group to Command mapping options are displayed on the screen.
2. Select a single group or multiple groups from the **Group** drop-down list.



3. Type the external command in the command text box.



4. Click  **Save changes locally** icon from the toolbar to save the new mapping. The new mappings are added to the rulebase map.

Group	Command	
+Group01	D:\scripts\cleanup.exe	
++Windows 7	D:\scripts\cleanup.exe	

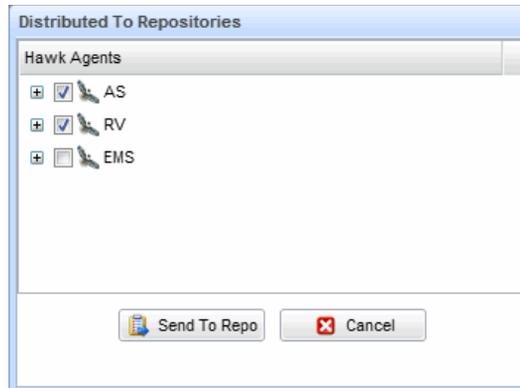
5. Click  icon to remove a group to external command mappings from the list.

6. Add other mappings to complete the rulebase map.
7. When the rulebase map is complete, click  **Deploy** icon to deploy the rulebase map updates to the repository.

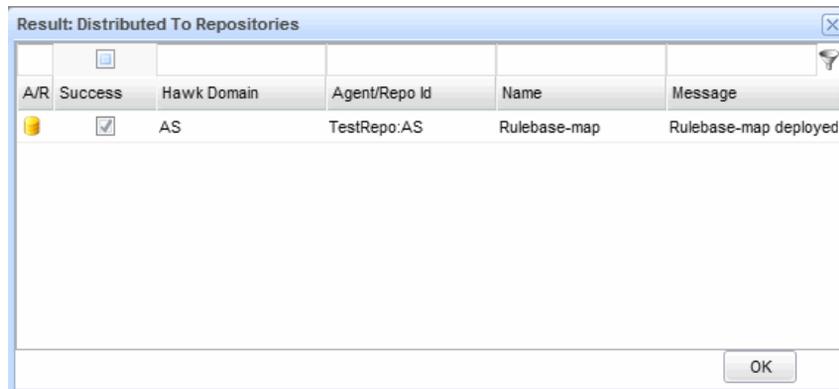
Distribute Rulebase Map to Repositories

To distribute a rulebase map to other repositories:

1. Click  **Distribute rulebase maps to other repositories** icon on the toolbar. The **Distributed To Repositories** dialog is displayed.



2. Select one or more repositories to send the rulebase map to. Click **Send To Repo** button. The results screen is displayed.



3. Click **OK** after viewing the results.

The rulebase map is sent to the selected Repository or Repositories. Agents that use these Repositories use the rulebase map the next time they are started.

Chapter 9

Common Features for Configurable Objects

This chapter describes the common features that are available for the configurable objects.

Topics

- [Database Persistence, page 110](#)
- [Synchronized, Dirty, and Local States, page 111](#)
- [Notification Area, page 113](#)

Database Persistence

If the persistence mode is enabled for the WebConsole deployment, all TIBCO Hawk configuration objects can be saved locally to the database before deploying them to the agents. Refer to the *TIBCO Hawk Installation, Configuration, and Administration Guide* for enabling the persistence mode.

The configuration objects enabled for database persistence are:

- RuleBases
- RuleBase Map
- Schedules

For each of these objects, there are two separate sections displayed on the editing screen:

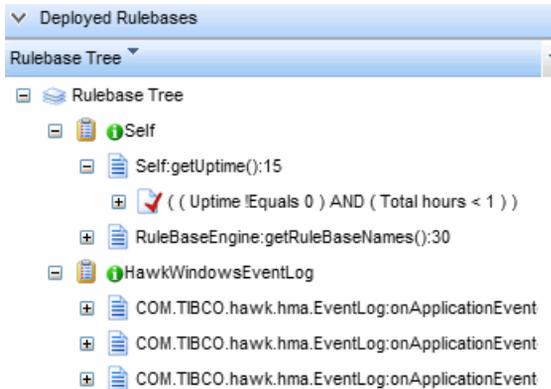
- **Deployed** <component>: Lists the objects that are deployed to the agents.
- **User** <component>: Lists the objects that are saved on the local database.

For example: **User** RuleBases section will show the rulebases persisted in the database by the current user.

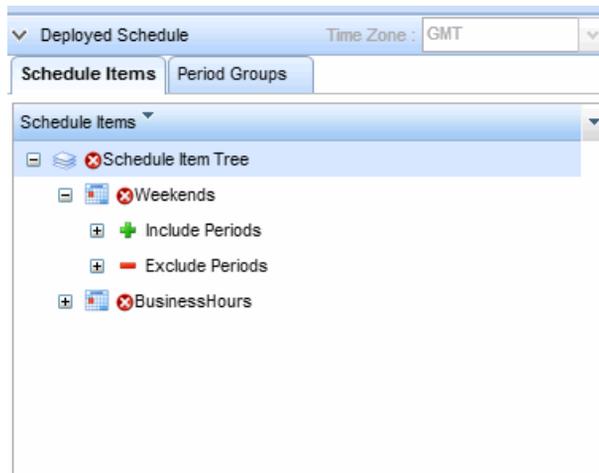
Synchronized, Dirty, and Local States

The configurable objects can have three possible states in TIBCO Hawk WebConsole.

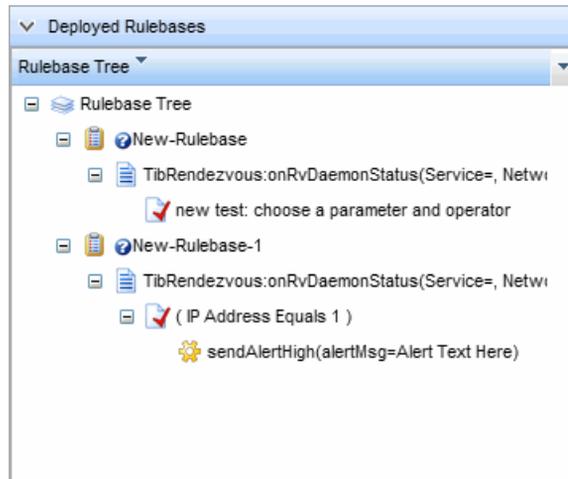
- Synchronized state - Indicated by an **i** icon. In this state, the object is synchronized with the Agent/Repository.



- Dirty state - Indicated by an **x** icon. If the object is updated and saved locally, the state of the object changes to "Dirty State". In this state the changes made to the object are available only till user logs out.



- Local - Indicated by an  icon. This state is supported only for rulebases. In this state, the object is not deployed and is available in the browser till the user logs out.



Notification Area

The configurable object editing screens provides the notification status information. The top left corner of the toolbar strip displays the last notification received.

Figure 22 Notification Area



The notification area displays:

- All the notifications regarding the configurable object
- Server notifications

Index

A

- accessing
 - Action Editor [46](#)
- Action Editor, accessing [46](#)
- actions
 - advanced options [50](#)
 - creating [46](#)
 - escalating [51](#)
 - network [92](#)
 - specifying variables in [54](#)
 - substituting variables in [56](#)
- advanced
 - action options [50](#)
 - test options [34, 44](#)
- Advanced Action Editor, accessing [50](#)
- Advanced Test Editor, accessing [34, 44](#)
- agents
 - performing actions on multiple [92](#)
 - querying multiple [86](#)
 - querying remote [87](#)
 - viewing alerts for [17, 17, 23, 23](#)
- Alert Display
 - description of [17, 23](#)
- alerts
 - marking all [19](#)
 - suspending [21](#)
 - viewing for an agent [17, 23](#)

B

- Boolean test operators [42](#)

C

- clear conditions, specifying [44](#)
- clear test option [44](#)
- clear timer option [44](#)
- compound tests [43](#)
- creating
 - actions [46](#)
 - rulebase maps [99](#)
 - schedules [64](#)
 - tests in a rule [39](#)

D

- data source variables [56](#)

E

- email, sending with actions [47](#)
- escalating actions [51](#)
- evaluation intervals, test [38](#)
- expressions
 - building test [39](#)
- external variables [54](#)
 - referencing in a rulebase [54, 55](#)

F

- false host list [91](#)
- first false option [44](#)

I

- including rulebases [34](#)
- in-schedule [73](#)
- internal variables [55](#)
- intervals
 - test evaluation [38](#)

M

- marking alerts [19](#)
- memory usage, testing [40](#)
- methods, microagent
 - invoking remotely [??-92](#)
 - referencing in actions [56](#)
 - results of [80](#)
- microagents
 - default platform-independent [15](#)

N

- network actions [92](#)
- network queries
 - adding tests to [89](#)
 - definition [86](#)
 - sending [87](#)
 - viewing results of [91](#)
- numeric test operators [40](#)

O

- operators, test
 - specifying [40](#)
 - specifying values for [40](#)

P

- Period Groups [58](#)
- Perl5PatternMatch
 - test operator for [42](#)
- Post Condition, action type [47](#)
- posted conditions
 - creating with actions [47](#)
 - test operators for [41](#)
- problem escalation [51](#)

Q

- queries, network [87](#)
- querying multiple agents [86](#)

R

- rulebase map
 - creating [99](#)
 - defining agent groups [??-101](#)
 - definition [95](#)
- rulebases
 - applying schedules to [15, 27, 58, 76](#)
 - including [34](#)
 - referencing internal variables in [55](#)

S

- schedules
 - creating [64](#)
 - definition [15, 27, 58, 76](#)
 - testing [73](#)
- sending
 - email with actions [47](#)
- specifying
 - external variables in actions [48, 48](#)
- suspending alerts [21](#)

syntax

- data source variable [56](#)
- external variable rulebase [54](#)
- internal variable [55](#)

T

Test Builder, accessing [39](#)

Test Editor, accessing [39](#)

testing

- schedules [73](#)

tests

- adding to network queries [89](#)
- advanced options [34](#), [44](#)
- Boolean operators [42](#)
- compound [43](#)
- creating in a rule [39](#)
- description [38](#)
- evaluation intervals [38](#)
- numeric operators [40](#)
- scheduling [44](#)
- specifying clear conditions for [44](#)
- specifying expressions in [39](#)

TIBCO support

- TIBCOcommunity [xvii](#)

TIBCO_HOME [xv](#)

true host list [91](#)

V

variables

- effect on actions [56](#)
- referencing data source [56](#)
- referencing external [54](#)
- referencing internal [55](#)
- specifying in actions [54](#)

viewing

- alert messages [17](#), [23](#)
- alerts for an agent [17](#), [23](#)

