# TIBCO Hawk® SNMP Adapter

## User's Guide

*Software Release 6.0*
*February 2018*

Two-Second Advantage®

## Important Information

# Contents

# Preface

This manual is a practical guide to using the TIBCO Hawk® SNMP Adapter.

TIBCO Hawk monitors and manages distributed operating systems and applications. The TIBCO Hawk SNMP Adapter adds SNMP monitoring, management, and integration capabilities to TIBCO Hawk. These capabilities include getting and setting SNMP variables as well as sending and receiving SNMP traps. In addition to the many built-in methods, the adapter provides a MIB browser utility that allows a customer to define custom methods from any SNMP MIB files. These custom methods are stored in XML encoded files that the adapter reads and instantiates at runtime.

This manual assumes you are familiar with the TIBCO Rendezvous and TIBCO Hawk environments, as well as SNMP Concepts.

## Topics

# Related Documentation

This section lists documentation resources you may find useful.

## TIBCO Hawk SNMP Adapter Documentation

The following documents form the TIBCO Hawk® SNMP Adapter documentation set:

- *TIBCO Hawk SNMP Adapter Installation Guide*: Read this book first. It contains step-by-step instructions for installing TIBCO Hawk SNMP Adapter on various operating system platforms.
- *TIBCO Hawk SNMP Adapter User's Guide*: Read this manual for instructions on using the product.
- *TIBCO Hawk SNMP Adapter Release Notes*: Read this document for a summary of features and a description of any issues that may affect installing or using the adapter.

## Other TIBCO Product Documentation

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

- TIBCO Hawk®
  - *TIBCO Hawk Installation, Configuration, and Administration*
- TIBCO ActiveSpaces®
  - *TIBCO ActiveSpaces Developer's Guide*
  - *TIBCO ActiveSpaces Administration*
  - *TIBCO ActiveSpaces Installation*
  - *TIBCO ActiveSpaces C Reference*
- TIBCO Rendezvous™
  - *TIBCO Rendezvous Concepts*
  - *TIBCO Rendezvous Administration*
  - *TIBCO Rendezvous Configuration Tools*
- TIBCO Enterprise Message Service™
  - *TIBCO Enterprise Message Service Installation*
  - *TIBCO Enterprise Message Service User's Guide*

# Typographical Conventions

The following typographical conventions are used in this manual.

*Table 1 General Typographical Conventions*

| Convention | Use |
|---|---|
| *ENV_NAME*<br>*TIBCO_HOME*<br>*HAWK_SNMP_HOME* | 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.<br>An installation environment consists of the following properties:<br>• **Name** Identifies the installation environment. This name is referenced in documentation as *ENV_NAME*. 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.<br>• **Path** The folder into which the product is installed. This folder is referenced in documentation as *TIBCO_HOME*.<br>TIBCO Hawk SNMP installs into a directory within a *TIBCO_HOME*. This directory is referenced in documentation as *HAWK_SNMP_HOME*. The default value of *HAWK_SNMP_HOME* depends on the operating system. For example on Windows systems, the default value is C:\tibco\hawk\<version>\adapters\snmp. |
| code font | Code font identifies commands, code examples, filenames, pathnames, and output displayed in a command window. For example:<br>Use MyCommand to start the foo process. |
| **bold code font** | Bold code font is used in the following ways:<br>• In procedures, to indicate what a user types. For example: Type **admin**.<br>• In large code samples, to indicate the parts of the sample that are of particular interest.<br>• In command syntax, to indicate the default parameter for a command. For example, if no parameter is specified, MyCommand is enabled:<br>MyCommand [**enable** \| disable] |

*Table 1 General Typographical Conventions (Cont'd)*

| Convention | Use |
|---|---|
| *italic font* | Italic font is used in the following ways:<br>• To indicate a document title. For example: See *TIBCO ActiveMatrix BusinessWorks Concepts*.<br>• To introduce new terms For example: A portal page may contain several portlets. *Portlets* are mini-applications that run in a portal.<br>• To indicate a variable in a command or code syntax that you must replace. For example: `MyCommand` *PathName* |
| Key combinations | Key name separated by a plus sign indicate keys pressed simultaneously. For example: Ctrl+C.<br>Key names separated by a comma and space indicate keys pressed one after the other. For example: Esc, Ctrl+Q. |
| | 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, or 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® SNMP Adapter is available on the TIBCO Hawk SNMP Adapter 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 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 Introduction

This chapter gives an overview of the TIBCO Hawk SNMP Adapter software, including the integration between TIBCO Rendezvous and the TIBCO Hawk system.

## Topics

# Overview

TIBCO Hawk monitors and manages distributed operating systems and applications. Using an alert-based model, it receives messages from the TIBCO Hawk agents. In TIBCO Hawk WebConsole or a similar console application, administrators can easily view and respond to exceptional behavior. To be monitored using TIBCO Hawk, applications use the TIBCO Hawk Application Management Interface (AMI). TIBCO Hawk AMI is a protocol that acts as a gateway between the application's management capabilities and the local TIBCO Hawk agent. If an application already employs a management interface other than AMI, an AMI adapter can be built to allow a TIBCO Hawk WebConsole. For more information, see TIBCO Hawk *Programmer's Guide*.

SNMP (Simple Network Management Protocol) is a communication protocol which is one of the most widely accepted methods of managing TCP/IP networks, including individual network devices as well as aggregated devices. SNMP uses a distributed architecture consisting of *agents* and *managers*.

- The SNMP agent is an SNMP application that monitors networks, devices or both, and responds to queries from SNMP manager applications. When significant events occur, the agent notifies the manager by sending a *trap*.
- The SNMP manager is an application that generates queries to SNMP-agent applications and receives traps from SNMP agent applications.

Information about an SNMP agent is stored in its unique Management Information Base (MIB). The SNMP agents and managers exchange information using MIB variables which form the MIB. Each MIB variable is referenced by a unique object identifier (OID).

The TIBCO Hawk SNMP Adapter adds SNMP monitoring, management, and integration capabilities to TIBCO Hawk. This includes getting and setting SNMP variables as well as sending and receiving SNMP traps from Hawk microagent methods. This also includes operating as a Hawk Console application converting Hawk Agent alerts and events to SNMP traps for publishing to other SNMP-enabled management systems. A MIB Browser is provided with the adapter that helps you to create custom TIBCO Hawk microagents for the SNMP agents using its MIB file. For more information, see Using the MIB Browser, page 23.

# Architecture

The TIBCO Hawk SNMP Adapter is a TIBCO Hawk alert subscriber. Like the TIBCO Hawk Display, it receives messages published using subject-based addressing. Multiple instances of the adapter can run on the same network to achieve fault-tolerance, load-balancing, or both.

The adapter is also a TIBCO Hawk microagent. Like other microagents, it uses AMI to communicate directly with the TIBCO Hawk agent on the local node. The adapter microagent exposes an interface with methods and arguments so you can manage its activities from TIBCO Hawk WebConsole. For more information, see Chapter 3, Using Adapter Methods.

The adapter consists of four major components:

- Microagent Trap Handler
- Console Trap Publisher
- AMI Methods and Custom Microagents
- MIB Browser

## Trap Handler

The *Trap Handler*, when enabled, reflects SNMP traps received by the local machine to the TIBCO Hawk agent as asynchronous method calls. These custom asynchronous methods can be created for any SNMP trap using the MIB browser utility. An SNMP trap for which a custom microagent method has not been defined is termed *Unknown Trap*. The asynchronous methods provide a return value for each MIB variable attached to the SNMP trap as well as the return values for the standard SNMP trap header fields. If an unknown trap is received, return values are only provided for the standard SNMP trap header fields. The trap handler can interpret `SNMP version 1` and `version 2c` encoded traps. For more details, see snmp:onUnknownTrap, page 64, and snmp:onHawkUserTrap, page 63.

TIBCO Hawk rules can be defined to respond to asynchronous methods allowing the full power of TIBCO Hawk monitoring to be applied to the SNMP traps. For more information, see Chapter 4, Using Methods in a Rulebase. Methods are also provided to dynamically start (enable) and stop (disable) the trap handler, to return statistics on traps received.

## Trap Publisher

The *trap publisher*, when enabled, monitors all TIBCO Hawk alerts and sends them as SNMP traps to a user-specified host formated using the TIBHAWKSNMP-MIB trap definition. The trap publisher tracks all outstanding alerts for a specific domain. The trap publisher provides a means for sending TIBCO Hawk alerts to other management and monitoring tools, provided those tools can receive SNMP traps. Methods are provided for defining filters to determine which categories of TIBCO Hawk alerts are sent as SNMP traps. Methods are also provided to dynamically start (enable) and stop (disable) the alert publisher, to return statistics on alerts received and traps sent, and to return a table of all currently outstanding (active) alerts.

## AMI Methods and Custom Microagents

The adapter is represented by microagents that communicate with TIBCO Hawk agents. These microagents provide methods for monitoring and managing the SNMP-enabled devices. You can invoke these methods interactively or in a TIBCO Hawk rulebase. For more information on these methods, see Chapter 3, Using Adapter Methods. Using the MIB browser you can create custom TIBCO Hawk microagents and custom methods for the newly created microagent. For more information on creating new microagents, see Creating a Custom Microagent, page 25.

## MIB Browser

In addition to the many built-in methods, the adapter provides an MIB browser utility that allows a user to define custom methods using any SNMP MIB. The MIB browser is a GUI utility that displays a tree view of your custom microagents and of all loaded SNMP MIB files. Point-and-click the MIB entries of interest to create custom microagent methods to get or set the MIB variables, send or receive the SNMP traps. For more information, see Using the MIB Browser, page 23.

# Functional Components

This section provides a general overview of each product component used by the adapter.

## The TIBCO Hawk System

TIBCO Hawk is a tool for monitoring distributed systems and applications based on the TIBCO Rendezvous or TIBCO Enterprise Messaging Service messaging system. The TIBCO Hawk monitoring solution is highly flexible, scalable, and easy to implement.

In a TIBCO Hawk environment, agents on each local computer perform the monitoring work. TIBCO Hawk agents use microagents to represent and interact with managed objects, such as an application or operating system component. Microagents use methods to extract monitoring information and carry out specified tasks internally, in the managed object. This allows a total decoupling of the management data from the management rules or policies. Microagent methods can be accessed and invoked from TIBCO Hawk Display, TIBCO Administrator, or Hawk Console API applications. They are also used with the agent's powerful rules engine in autonomous monitoring.

Monitoring logic is stored in rulebases on each agent. Rulebases contain rules, which are user-defined statements about a managed object. Using specialized editing menus in TIBCO Hawk WebConsole, administrators can easily construct rules that carry out specific tests and actions. No special syntax or scripting is required. Rules can then be grouped and distributed to multiple agents with similar monitoring requirements.

The TIBCO Hawk WebConsole or TIBCO Adminsitrator are various console applications available to monitor managed objects within your infrastructure network. The main window of the TIBCO Hawk WebConsole graphically represents the system behavior with each monitored computer represented by a container icon. Alert messages published by TIBCO Hawk Agents on TIBCO Messaging software are also displayed. All WebConsole users can view the same managed objects without complicated configuration schemes, and each user can customize the interface without affecting others. For more information, see TIBCO Hawk *WebConsole User's Guide.*

## AMI Protocol

The TIBCO Hawk Application Management Interface (AMI) is an open protocol for instrumenting an application with a powerful management interface. Using AMI, applications can be monitored and controlled internally using TIBCO Hawk in the same way as any other system component. The protocol provides complete independence between the TIBCO Hawk agent and the managed application with two-way dynamic discovery.

The AMI features are:

- AMI API libraries, which implement the protocol, are provided in C, C++, and Java.
- AMI can be easily back-fitted into the existing applications, even those that do not currently utilize various transport mechanisms for TIBCO Hawk subsystem, and expose existing the internal application methods to a TIBCO Hawk agent.
- Third-party applications can be managed either by writing an AMI wrapper that interfaces with an API, or can be managed by other mechanism provided by the application.

For more information, see the TIBCO Hawk *Programmer's Guide*.

# Chapter 2 Getting Started

This chapter describes how to configure and use the TIBCO Hawk SNMP Adapter.

## Topics

# Configuring the Adapter

The following sections describe the parameters you can set to configure the adapter. The adapter-specific configuration parameters can be set by modifying the default parameters in the adapter configuration file. These parameters specify configurations such as TIBCO Rendezvous or TIBCO DataGrid session and license information.

The microagent descriptor XML file(s), which is created using MIB Browser describes any custom created microagent. The `-xml_file` parameter in the adapter configuration file is used to specify the list of custom microagent XML files to load. All such changes in configuration are effective on the adapter restart.

# Configuring the Transport Mode

The adapter uses three transport modes to listen to alerts sent by the trap publisher from TIBCO Hawk Agents.

The available transport modes are:

- TIBCO DataGrid (distributed as a part of TIBCO Hawk installation)
- TIBCO Rendezvous (RV)
- TCP Transport for TIBCO Hawk
- TIBCO Enterprise Message Service (EMS)

> Atleast one transport mode should be configured in the configuration files of Hawk Agent, Hawk MicroAgent, Hawk Event, and Hawk WebConsole to enable message or event communication among the various Hawk components.

TIBCO Hawk SNMP Adapter has TIBCO Rendezvous configured as the default transport mode between the adapter and agent.

TIBCO Rendezvous and TIBCO Enterprise Message Service are two independent products that need to be installed separately. Refer to the *Installation, Configuration, and Administration* guide of TIBCO Hawk on how to install TIBCO Hawk on TIBCO Rendezvous or TIBCO Enterprise Message Service.

The location of TIBCO Enterprise Message Service and TIBCO Rendezvous installation is configured in the `tibhawksnmp.tra` and `tibhawkmb.tra` files at *HAWK_SNMP_HOME*`\bin`. These transport locations are updated automatically, if TIBCO Hawk SNMP Adapter is installed into the same *TIBCO_HOME*, where TIBCO Enterprise Message Service and TIBCO Rendezvous are already installed.

Otherwise, update the installation path of the TIBCO Enterprise Message Service and TIBCO Rendezvous for the `tibco.env.EMS_HOME` and `tibco.env.RV_HOME` parameters respectively, in the `tibhawksnmp.tra` and `tibhawkmb.tra` files. These locations need to be updated in the case of the following scenarios:

- If you install TIBCO Rendezvous or TIBCO Enterprise Message Service in the same *TIBCO_HOME* after TIBCO Hawk SNMP Adapter.
- If you are using or planning to use a different *TIBCO_HOME* for TIBCO Rendezvous and TIBCO Enterprise Message Service.

## TIBCO Rendezvous Transport

Configure the `-hawk_rvd_session` parameter in the configuration file to reenable the TIBCO Rendezvous as transport back from TIBCO Enterprise Message Service or TIBCO DataGrid.

Comment this option, or let it be commented in the configuration file, if you are using TIBCO Enterprise Message Service or TIBCO DataGrid as the primary transport.

Configure the `-ami_rvd_session` parameter to set up the TIBCO Rendezvous session for the adapter to publish the AMI methods.

TIBCO Hawk connects to the TIBCO Rendezvous daemon by creating a session. In the configuration files, ensure that the `-hawk_as_session` and `-ems_transport` parameters are commented, and then configure the `-hawk_rvd_session` and `-ami_rvd_session` parameters. TIBCO Rendezvous transport creation calls accept three parameters that govern the behavior of the transport: service, network, and daemon.

`-hawk_rvd_session` *<service>* *<network>* *<daemon>*
`-ami_rvd_session` *<service>* *<network>* *<daemon>*

where,

- **service** instructs the Rendezvous daemon to use this service whenever it conveys messages on this transport. You can specify the port number as the service to be used, for example, "`7474`".
- **network** instructs the Rendezvous daemon to use a particular network for all communications involving this transport. The network parameter consists of up to three parts, separated by semicolons: network, multicast groups, and send address.
- **daemon** instructs the transport creation function about how and where to find the Rendezvous daemon and establish communication. For remote daemons, specify two parts (introducing the remote host name as the first part), for example, `tcp:7474`:
  - — Remote host name
  - — Port number

For more details on TIBCO Rendezvous, refer to the TIBCO Rendezvous documentation.

## TIBCO DataGrid Transport

Configure the `-hawk_as_session` parameter in the configuration file to enable the TIBCO DataGrid as transport.

Comment this option, or let it be commented in the configuration file, if you are using TIBCO Enterprise Message Service or TIBCO Rendezvous as the primary transport.

In the configuration file, ensure that the `-ami_rvd_session`, `-hawk_rvd_session`, `-ft_rvd_session`, and `-ems_transport` parameters are commented out, and then configure the `-hawk_as_session` parameter.

`-hawk_as_session` *<listen url>* *<discover url>*

where,

- **listen url** specifies the interface and the TCP port that the process uses to listen for incoming connections from new members to the metaspace, and specified in the form of a URL. To use a listen URL, use a string of the form:

  ```
  tcp://[interface[:port]]
  ```

- **discovery url** determines how the members are discovered. Discovery can be unicast (TCP) or multicast (PGM —Pragmatic General Multicast).

  With PGM, discovery of the current metaspace members is done by using a reliable IP multicast. The attributes of this discovery mechanism are expressed in the form of a URL in the following format:

  ```
  tibpgm://[dport]/[interface];[discovery group address]
  ```

  When multicast discovery is not needed or possible, you can use only the TCP discovery. The discovery URL uses well known addresses. The TCP discovery has the following format:

  ```
  tcp://ip1[:port1];ip2[:port2],...
  ```

Use a semicolon (;) to indicate a null value, or use an empty string. For example, the deault value in the configuration files is:

```
-hawk_as_session "" tibpgm://8989/
```

For more detailed information on TIBCO DataGrid, please refer to the TIBCO ActiveSpaces documentation.

## TCP Transport for TIBCO Hawk

TCP Transport for TIBCO Hawk is a TCP based transport for Hawk components using the Akka clustering designs. The TCP Transport for TIBCO Hawk removes the dependency on an external server or transport (such as, TIBCO Rendezvous and TIBCO Enterprise Messaging Service) as the TCP communication happens peer to peer. TIBCO Hawk can be easily deployed in cloud when TCP Transport for TIBCO Hawk is used.

TCP Transport for TIBCO Hawk is distributed as part of the standard Hawk installation. For more information on TCP Transport for TIBCO Hawk, see *TIBCO Hawk Documentation*.

Configure the `-ami_tcp_session` and `-hawk_tcp_session` parameter in the `tibhawksnmp.cfg` file to enable the TCP Transport for TIBCO Hawk to be used for AMI communication. See `-ami_tcp_session` and `-hawk_tcp_session` parameters on Table 2, Adapter Configuration Parameters in tibhawksnmp.cfg, on page 14 for more details.

## TIBCO Enterprise Message Service (EMS) Transport

Configure the `-ems_transport` parameter for connecting to the TIBCO Enterprise Message Service server as transport.

Comment this option if you are using TIBCO DataGrid or TIBCO Rendezvous as the primary transport.

In the configuration file, ensure that the `-hawk_rvd_session`, `-ft_rvd_session` and `-hawk_as_session` parameters are commented, and then configure the `-ems_transport` parameter.

However, you also need to configure the `-ami_rvd_session` parameter along with the `-ems_transport` parameter for the adapter to publish the AMI methods to agents. TIBCO Hawk SNMP Adapter uses the Enterprise Message Service transport only when you start the TIBCO Hawk SNMP trap publisher. Thus, Enterprise Message Service logs are created only when the publisher methods are used: isPublisherRunning(), startPublisher(), and stopPublisher().

The two ways to specify the TIBCO EMS transport parameters are:

1. Specify only the location of the EMS server.

   For example,

   ```
   -ems_transport tcp://server1:7222
   ```

   If communicating with the EMS server using SSL, specify the location of the EMS server as follows for the preceding example:

   ```
   -ems_transport ssl://server1:7222
   ```

   Also, specify the additional options outlined as follows.

2. Specify the location of the EMS server and a valid user name and password for the EMS server.

   These parameters are separated by a space and can be an empty string to indicate a null value.

   For example,

   ```
   -ems_transport tcp://server1:7222 admin "#!NhAD1NBC"
   ```

   If communicating with the EMS server using SSL, specify the location of the EMS server as follows for the preceding example:

```
-ems_transport ssl://server1:7222 admin "#!NhAD1NBC"
```

# Editing the Adapter Configuration File

When the adapter is installed, an adapter configuration file named `tibhawksnmp.cfg` is copied to your *HAWK_SNMP_HOME*`\config` directory. This file contains parameters that determine how the adapter communicates with TIBCO Rendezvous, and the parameters that specify the locations of the required files.

The following table lists the adapter parameters and their default values. You can modify the adapter configuration by editing the file, and modifying one or more parameter values. Changes take effect the next time the adapter is started.

In the adapter configuration file, java property values can be substituted into command options and parameters using the format `${file.separator}`. The java property `hawk.hawk_root` is passed in by the adapter script and the Microsoft Windows service and contains the TIBCO Hawk installation directory.

In the adapter configuration file, all parameters that begin with the string `_filter_` are used in conjunction with console trap publisher.

*Table 2 Adapter Configuration Parameters in tibhawksnmp.cfg*

| Parameter Name | Description |
|---|---|
| `-hawk_domain` *<domain name>* | *<domain name>* specifies the TIBCO Hawk domain to be monitored for TIBCO Hawk alerts by the trap publisher.<br>*<domain name>* has no default value. |
| `-ami_rvd_session` *<service>* *<network>* *<daemon>* | Name of the TIBCO Rendezvous session the adapter uses to publish its AMI methods to the local TIBCO Hawk Agent.<br>Session name must be in the format *<service>* *<network>* *<daemon>*.<br>The default value is `7474 "" tcp:7474` where `7474` is the UDP service, `""` represents a NULL value for the primary network, and `tcp:7474` listens on port `7474` using the TCP/IP protocol. For details, see *TIBCO Rendezvous Concepts*. |

*Table 2 Adapter Configuration Parameters in tibhawksnmp.cfg (Cont'd)*

| Parameter Name | Description |
|---|---|
| `-ami_tcp_session` | Configures the TIBCO Hawk SNMP Adapter to use TCP connection to publish its AMI methods to TIBCO Hawk agent.<br>The syntax for the property is:<br>`-ami_tcp_session` *<self_IP>:<port>* *<hawk_agent_IP>:<AMI_session_port>*<br>where,<br>• *<self_IP>:<port>* - The unique socket address of TIBCO Hawk SNMP Adapter for AMI communication. The default port value is zero, that is, an available port is automatically assigned to the microagent during initialization.<br>• *<hawk_agent_IP>:<AMI_session_port>* - The socket address of the Hawk agent for AMI communication. This socket address is same as *<self_IP>:<port>* specified for the `-ami_tcp_session` parameter in `hawkagent.cfg`.<br>The default values are:<br>`-ami_tcp_session localhost:0 localhost:2571` |
| `-hawk_as_session` | Comment this option if you are using TIBCO Enterprise Messaging Server or TIBCO Rendezvous as the primary transport.<br>The format is `-hawk_as_session` *<listen url>* *<discover url>*.<br>See the TIBCO Hawk documentation for more details. Use a semicolon (;) to indicate a null value, or use an empty string, for example,<br>`-hawk_as_session "" tibpgm://8989/` |
| `-xml_file` *<xmlfile>*`...` | *<xmlfile>*is the fully specified name of the TIBCO Hawk SNMP microagent definition (XML) file(s) to be loaded at startup.<br>*<xmlfile>* has no default value.<br>See Creating a Custom Microagent, page 25, for more information on creating microagent definition files. |
| `-publisher_host` *<hostname>* | *<hostname>* specifies the name or IP address of the target machine for SNMP traps sent by the TIBCO Hawk SNMP console trap publisher.<br>*<hostname>* has no default value. |
| `-publisher_port` *<port>* | *<port>* specifies the UDP port number on the target machine for receiving SNMP traps sent by the TIBCO Hawk SNMP console trap publisher.<br>*<port>* has a default value of "162". |

*Table 2 Adapter Configuration Parameters in tibhawksnmp.cfg (Cont'd)*

| Parameter Name | Description |
|---|---|
| `-publisher_community` *<community>* | *<community>* specifies the SNMP community to be used for SNMP traps sent by the TIBCO Hawk SNMP console alert publisher.<br>*<community>* has a default value of "public". |
| `-trap_handler_port` *<port>* | *<port>* specifies the UDP port number used by the adapter trap handler for monitoring SNMP traps sent to the local machine.<br>*<port>* has a default value of "162". |
| `-trap_handler_community` *<community>* | *<community>* specifies the SNMP community to be used by the adapter trap handler for authenticating SNMP traps sent to the local machine.<br>*<community>* has a default value of "public". |
| `-default_port` *<port>* | *<port>* specifies the default UDP port number to be used when sending SNMP requests.<br>*<port>* has a default value of "161". |
| `-default_community` *<community>* | *<community>* specifies the default SNMP community to be used when sending SNMP requests.<br>*<community>* has a default value of "public". |
| `-default_trap_port` *<port>* | *<port>* specifies the default UDP port number to be used when sending SNMP traps.<br>*<port>* has a default value of "162". |
| `-start_trap_handler` | Specifies that the SNMP trap handler should be started automatically when the SNMP Microagent is started. Methods are also provided by the SNMP Microagent to dynamically start and stop the trap handler. |
| `-start_trap_publisher` | Specifies that the SNMP trap publisher should be started automatically when the SNMP Microagent is started. Methods are also provided by the SNMP Microagent to dynamically start and stop the trap publisher. |
| `-file` *<filename>* | *<filename>* is the fully specified name of the file containing the command line options.<br>The contents of the specified file are read in and processed as if they had been entered directly on the command line. Comments can be placed in the file by proceeding them with the '#' character.<br>*<filename>* has no default value. |

*Table 2 Adapter Configuration Parameters in tibhawksnmp.cfg (Cont'd)*

| Parameter Name | Description |
|---|---|
| `-hawk_rvd_session` *<service>* *<network>* *<daemon>* | Name of the TIBCO Rendezvous session the adapter uses to listen for alerts from TIBCO Hawk Agents by the trap publisher.<br>Session name must be in the format *<service>* *<network>* *<daemon>*.<br>The default value is `7474 "" tcp:7474` where `7474` is the UDP service, `""` represents a NULL value for the primary network, and `tcp:7474` listens on port `7474` using the TCP/IP protocol. For details, see *TIBCO Rendezvous Concepts*.<br>If using TIBCO Rendezvous as the transport, make sure you commenting out the `-ems_transport` parameter in the configuration file. |
| `-hawk_tcp_session` | Configures the TIBCO Hawk SNMP Adapter to use TCP connection to listen for alerts from TIBCO Hawk Agent by the trap publisher.<br>The syntax for the property is:<br>`-hawk_tcp_session` *<self_IP>:<port>* *<hawk_daemon_IP>:<daemon_port>*<br>where,<br>• *<self_IP>:<port>* - The unique socket address of the Hawk Event service for joining the cluster.<br>• *<hawk_daemon_IP>:<daemon_port>* - The socket address of the Hawk TCP daemon acting as the seed for the cluster. This socket address is same as *<daemon_IP>:<port>* specified for the `-tcp_session` parameter in Hawk TCP daemon configuration file (`hawktcpdaemon.cfg`).<br>The default values are:<br>`-hawk_tcp_session localhost:2583 localhost:2561` |

*Table 2 Adapter Configuration Parameters in tibhawksnmp.cfg (Cont'd)*

| Parameter Name | Description |
|---|---|
| `-ems_transport` *<serverURL> <username> <password>* | Specifies the TIBCO EMS transport parameters used by adapter.<br>*<serverURL>* specifies the location of the EMS server.<br>*<username>* and *<password>* are used to connect to the EMS server.<br>`-ems_transport tcp://dev_server:7222 admin "#!NhAD1U"`<br>If communicating with the EMS server using SSL, specify the location of the EMS server as follows:<br>`-ems_transport ssl://server1:7222 admin "#!BhAD1NB2"`<br>For fault tolerant setup, list configured fault tolerant EMS servers in a comma-separated list as follows:<br>`-ems_transport tcp://dev_server:7222, tcp://dev_server2:7444 admin "#!Phhq1t"`<br>If the servers are configured with no authentication, substitute the *<username>* and *<password>* parameters with double quotes as follows:<br>`-ems_transport tcp://qa_emsserver:7222 "" ""` |
| `-ssl_vendor` *<name of the vendor>* | Specifies the name of the vendor of the SSL implementation. Valid choices are `j2se` and `entrust61`. The default value is `j2se.` |
| `-ssl_no_verify_host` | Specifies that the adapter should not verify the server. Conversely, if this option is not included in the configuration file, it indicates that the adapter component should verify the server. |
| `-ssl_trusted` *<file-name>* | Specifies the file name of the server certificates. This option can be repeated if more than one certificate file is used. |
| `-ssl_expected_hostname` *<host-name>* | Specifies the name that is expected in the name of the CN field of the server certificates.<br>The value specified by this option is used when the `-ssl_no_verify_hostname` option is absent from the configuration file. |
| `-ssl_identity` *<file-name>* | Specifies the digital certificate of the adapter. |
| `-ssl_private_key` *<file-name>* | Specifies the private key of the adapter.<br>If the key is included in the digital certificate in `-ssl_identity`, then you may omit this parameter. |
| `-ssl_password` *<string>* | Specifies the password to decrypt the identity file of the adapter.<br>For details, see Encrypting Passwords on page 21. |

*Table 2 Adapter Configuration Parameters in tibhawksnmp.cfg (Cont'd)*

| Parameter Name | Description |
|---|---|
| `-ssl_no_verify_hostname` | Specifies that the client should not verify the name in the CN field of the server certificate. Conversely, if this option is not included in the configuration file, it indicates that the adapter should verify the name in the CN field of the server certificate. |
| `-ssl_ciphers` *<suite-name(s)>* | Specifies the cipher suites that can be used. When specifying this option, use the ^ qualifier instead of a - qualifier. For more information on specifying cipher suites, refer to the *TIBCO Enterprise Message Service* documentation. |
| `-security_policy` *<security class>* | *<security class>* is the fully qualified Java Class which implements the TIBCO Hawk security policy in effect.<br>*<security class>* has no default value. |
| `-filter_agent_alive` *<switch>* | Specifies filtering of agent alive alerts. Specify ON/OFF to enable/disable the publishing of agent alive alerts.<br>*<switch>* is either `ON` or `OFF`. The default value is `ON`. |
| `-filter_agent_dead` *<switch>* | Specifies filtering of agent dead alerts. Specify ON/OFF to enable/disable the publishing of agent dead alerts.<br>*<switch>* is either `ON` or `OFF`. The default value is `ON`. |
| `-filter_high_alerts` *<switch>* | Specifies filtering of high level alerts. Specify ON/OFF to enable/disable the publishing of high level alerts.<br>*<switch>* is either `ON` or `OFF`. The default value is `ON`. |
| `-filter_medium_alerts` *<switch>* | Specifies filtering of medium level alerts. Specify ON/OFF to enable/disable the publishing of medium level alerts.<br>*<switch>* is either `ON` or `OFF`. The default value is `ON`. |
| `-filter_low_alerts` *<switch>* | Specifies filtering of low level alerts. Specify ON/OFF to enable/disable the publishing of low level alerts.<br>*<switch>* is either `ON` or `OFF`. The default value is `ON`. |
| `-filter_information` *<switch>* | Specifies filtering of informational alerts. Specify ON/OFF to enable/disable the publishing of informational alerts.<br>*<switch>* is either `ON` or `OFF`. The default value is `ON`. |

*Table 2 Adapter Configuration Parameters in tibhawksnmp.cfg (Cont'd)*

| Parameter Name | Description |
|---|---|
| `-filter_rulebase` *<rulebase name>* | Specifies a rulebase name used to filter alerts. Only alerts generated by this rulebase will be published. ALL indicates that alerts for all rulebases should be published. This filter is not applied to agent alive and agent dead alerts.<br>*<rulebase name>* is either a rulebase name, the string `'ALL'`, or a Perl5 expression.<br>For more information on using Perl5 expressions, see Perl5 Pattern Matching Syntax, page 103. |
| `-filter_host_name` *<host name>* | Specifies the TIBCO Hawk agent host name used to filter alerts. Alerts originating from the TIBCO Hawk agent associated with the specified host name will be published. ALL indicates that alerts for all host names should be published.<br>*<host name>* is either an agent host name, the string `'ALL'`, or a Perl5 expression.<br>For more information on using Perl5 expressions, see Perl5 Pattern Matching Syntax, page 103. |
| `-filter_dns_name` *<dns name>* | Specifies the TIBCO Hawk agent host DNS name used to filter alerts. Alerts originating from the TIBCO Hawk agent associated with the specified host DNS name will be published. ALL indicates that alerts for all host DNS names should be published.<br>*<dns name>* is either an agent DNS name, the string `'ALL'`, or a Perl5 expression.<br>For more information on using Perl5 expressions, see Perl5 Pattern Matching Syntax, page 103. |
| `-log_dir` *<directory>* | Specifies the fully specified directory, where log and trace files are created.<br>The default value is "*HAWK_SNMP_HOME*`/logs`". |
| `-log_max_size` *<maxsize>* | Specifies the maximum size of the rotating log file in kilobytes. When a trace file reaches this size, the adapter rolls over to a new log file. You may apply a suffix `m` or `M` for indicating MB values, for example, `10m`.<br>The default value is "`10M`". |
| `-log_max_num` *<maxnum>* | Specifies the maximum number of rotating log files. If number of log files exceeds this number, the oldest log file is deleted.<br>The default value is "`10`". |

*Table 2 Adapter Configuration Parameters in tibhawksnmp.cfg (Cont'd)*

| Parameter Name | Description |
|---|---|
| `-log_level` *<loglevel>* | Specifies the level of diagnostic information stored in the logs. The following are the logging levels:<br>• 4 - Indicates that the error level messages should be logged.<br>• 6 - Indicates that the warning level messages should be logged.<br>• 7 - Indicates that the information level messages should be logged.<br>• 8 - Indicates that the debug level messages should be logged.<br>• 16 - Indicates that the AMI level messages should be logged.<br>• A value of zero turns all tracing off.<br>• A value of -1 turns all tracing on.<br>The default value is "`7`". |
| `-snmp_version_trap_publisher` *<SNMP_version>* | *<SNMP_version>* specifies the SNMP version that will be used by the trap publisher. Valid values are `Version 1` and `Version 2(C).`<br>*<SNMP_version>* has a default value of `Version 1`.<br>To dynamically change the SNMP version used by the trap publisher, use the method snmp:setSnmpVersionTrapPublisher. |
| `-ft` | Fault Tolerant weight for TibrvFtMember used by the trap publisher. The default is no fault tolerance. |
| `-ft_rvd_session` | The TIBCO Rendezvous session used for fault tolerance. If this option is used, all three parameters must be present and separated by whitespace but any of them may be an empty string to indicate a null value.<br>When using TIBCO Rendezvous as the primary transport, if this option is not specified, the primary Rendezvous session is used for fault tolerance. This option is ignored if the `-ft` option is not specified. |

### Encrypting Passwords

Follow these steps to encrypt your password:

1. Invoke the script from the command line using the syntax. The script is located in the directory *TIBCO_HOME*`\hawk\`*<version>*`\bin`.

   On Microsoft Windows, `tibhawkpassword.exe -encrypt` *`<string>`*

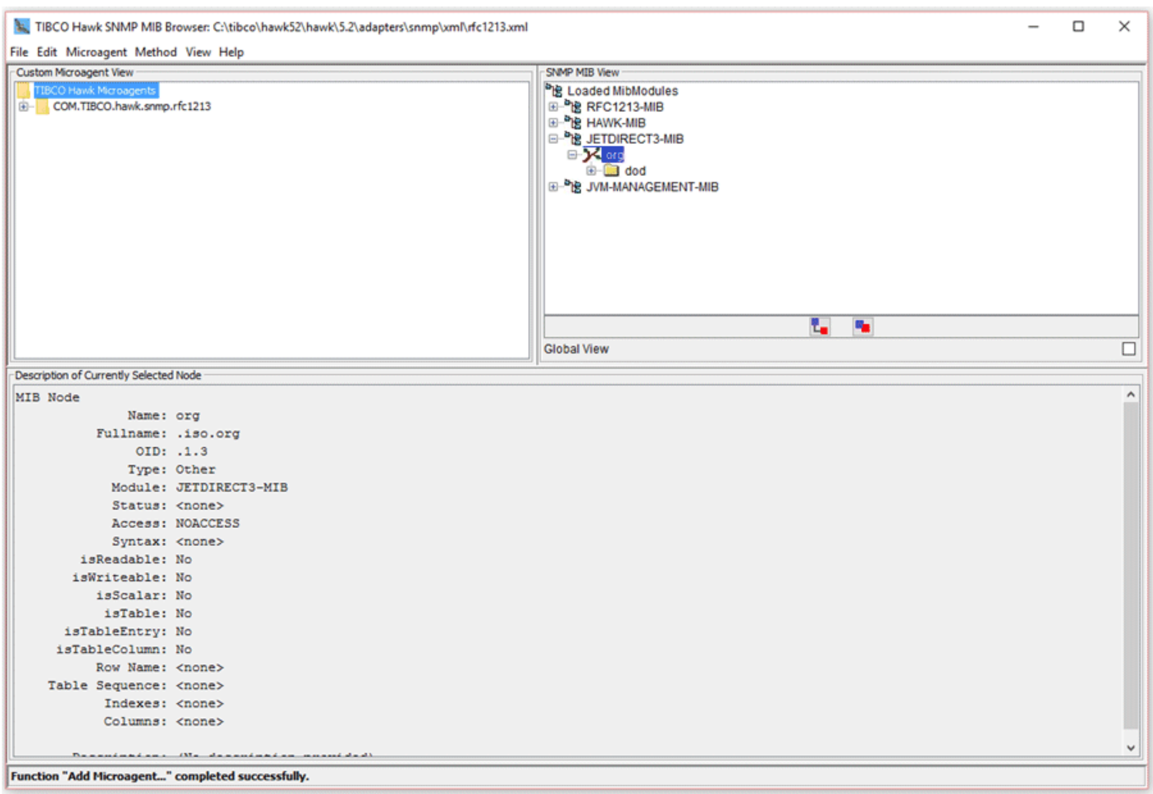
   On UNIX, `tibhawkpassword -encrypt` *`<string>`*

   where *`<string>`* is the password you want to encrypt.

2. Copy and paste the output of the script into the configuration file. The encrypted password should be placed within quotes ("").

# Using the MIB Browser

The MIB browser is used to create custom TIBCO Hawk microagents and methods directly from SNMP MIB files. The MIB browser window, shown in the following figure, consists of three main panels. The upper left-hand panel shows a tree view of the currently defined custom microagents and their methods. The upper right-hand panel shows a tree view of the currently loaded SNMP MIB files. Additional SNMP MIB files can be loaded and added to the tree view at any time. The lower panel shows detailed information about the currently selected node from either of the tree views mentioned earlier. Right-click various tree nodes to display popup menus. Using these pop-up menus, you can create, modify, and delete microagents and methods based on the SNMP MIB tree. These custom microagent definitions can be saved and loaded from XML encoded files. The adapter can load these XML files on startup and instantiate the defined microagents.



The SNMP MIB can be presented in either MIB or trap view. The MIB view tree shows the SNMP variables defined in the loaded MIBs and is used for creating get and set methods. The trap view tree shows the SNMP traps defined in the loaded MIBs and is used for creating `onTrap` (to receive traps) and `sendTrap` methods. You can switch between views using the View menu.

The following sections cover the various MIB browser operations in detail.

## MIB Browser Functions

The following table describes the items in the MIB browser pull-down menu.

| Function | Type | Description |
|---|---|---|
| File | | |
| | New | Starts a new microagent definition with an empty tree. |
| | Open | Opens an existing microagent definition file for editing. |
| | Save | Saves changes to the current microagent definition file. |
| | Save As | Saves the current microagent definition file to a user specified file. |
| | Load MIB | Loads MIB files and adds them to the MIB tree. |
| | Exit | Exits the MIB browser utility. |
| Edit | | |
| | Properties | Edits the properties of the selected microagent tree node. |
| Microagent | | |
| | Add Microagent | Adds a new microagent to the tree. |
| | Remove Microagent | Removes the currently selected microagent from the tree. |
| Method | | |

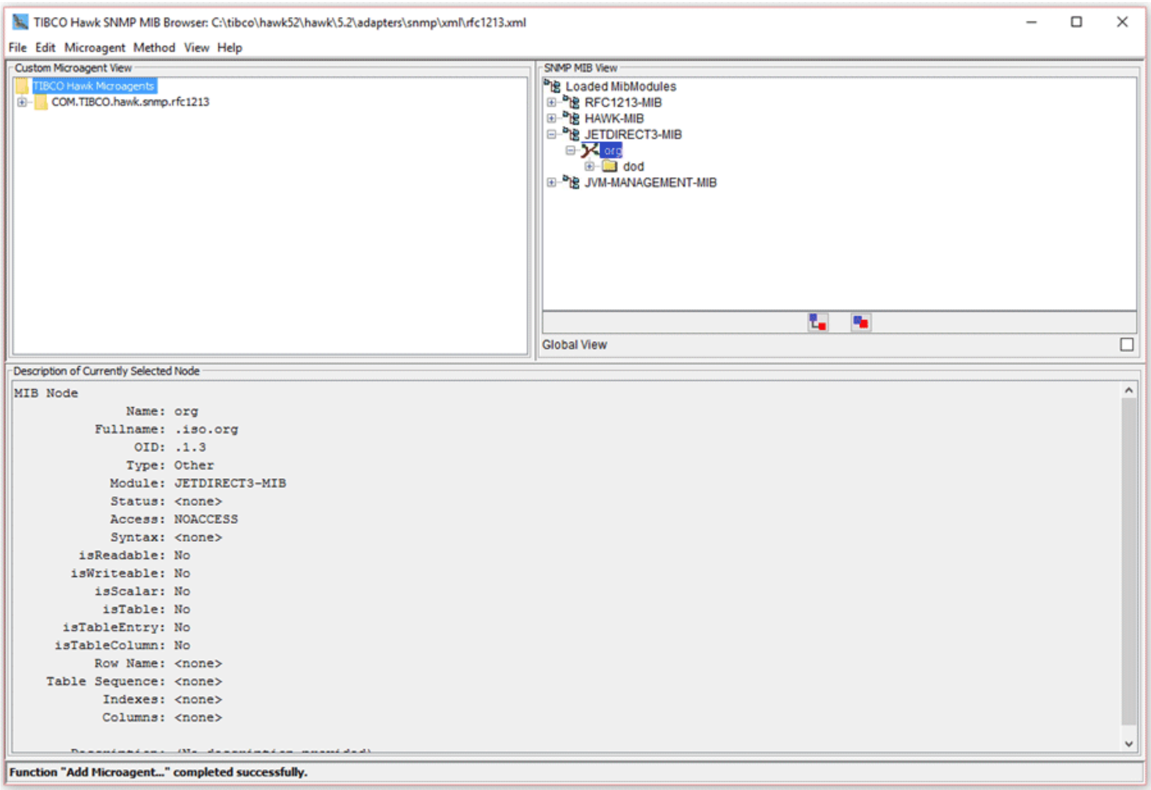
| Function | Type | Description |
|---|---|---|
| | Add Get Method | Adds a new get method for the currently selected MIB tree node. |
| | Add Set Method | Adds a new set method for the currently selected MIB tree node. |
| | Ad Send Trap Method | Adds a new send trap method for the currently selected MIB tree node. |
| | Add On Trap Method | Adds a new on trap method for the currently selected MIB tree node. |
| | Remove Method | Removes the currently selected method from the microagent tree. |
| View | | |
| | Trap View/ MIB View | Switches between the Trap View and MIB View. |
| Help | | |
| | About | Displays a dialog containing product description information. |

## Creating a Custom Microagent

This section describes the steps involved in creating a custom microagent 'HP_PRINTERS' and a microagent descriptor file for a newly created microagent. This microagent monitors Hewlett Packard Jet Direct laser printers. This microagent is used in later examples to monitor the printer for 'paper jam' conditions.

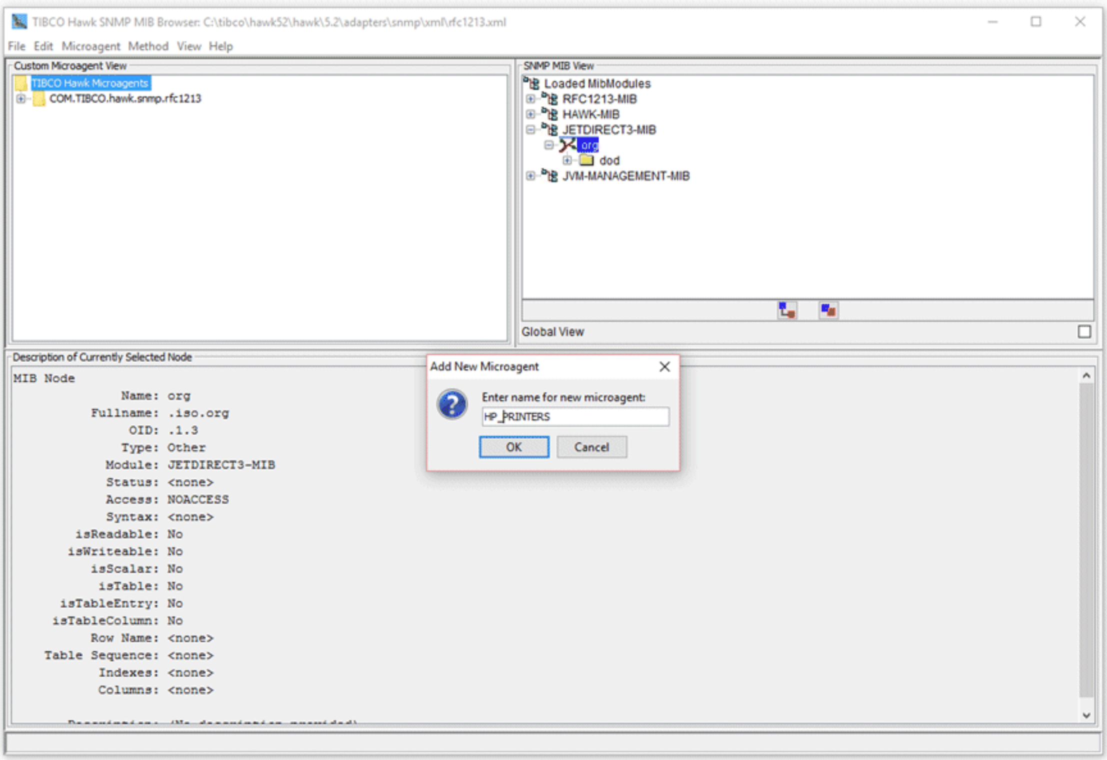
1. If the Hewlett Packard JetDirect3 MIB file has not been previously loaded, select the **File->Load MIB** options, navigate to the directory which contains the Hewlett Packard JetDirect3 MIB file.

All the MIB files in the directory are displayed. Double-click the MIB file corresponding to the printer to load it into the MIB browser. Once loaded, the managed data are displayed on the right-side of the screen.




2. To create a new microagent, select **Microagent->Add Microagent**.

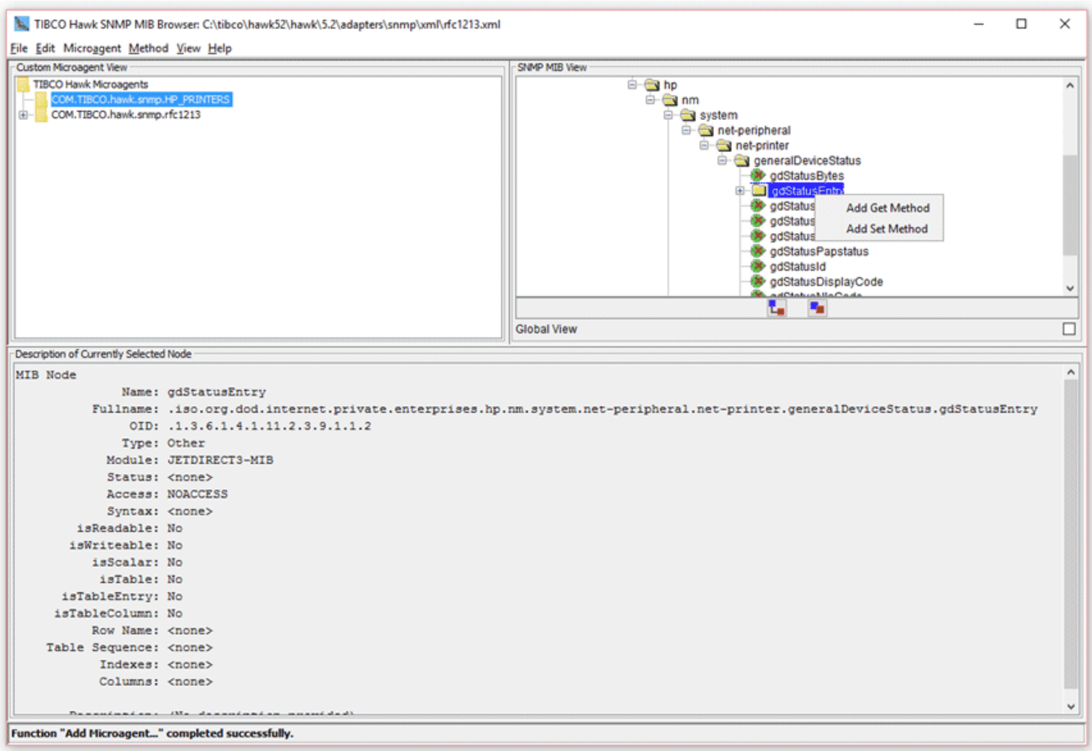
3. Type '**HP_PRINTERS**' and click **OK**.




4. Expand the tree-structure of the **JETDIRECT3** MIB.
5. Right-click the **gdStatusEntry** folder.

   A pop-up menu displays the available options for selecting the type of method to be added.

6. Select the **Add Get Method** option.




A new method named `getgdStatusEntry` is added to the selected microagent in the microagent tree. The method name is derived from the operation (`get`) and the MIB item name (`gdStatusEntry`). Expanding the `getgdStatusEntry` tree node, note that method arguments and returns have been automatically generated from the MIB definition.

The standard arguments `Host`, `Community`, `Port`, and `SNMP` are automatically added where appropriate to allow these values to be specified when the method is invoked. These standard arguments have default values of `localhost`, `public`, `161`, and `version 1`, respectively. The `Port` argument defaults to `162` for `sendTrap` methods.

These standard arguments can be removed and they take on default values on method invocation.

For the get() methods, a return is defined for each MIB variable from the selected MIB node. If the node represents a terminal node, a single return is added for that variable. If the node represents a non-terminal node (for example, `gdStatusEntry`), a return is added for each terminal node contained in that node. If the node represents a table, a return is added for each column of the table.

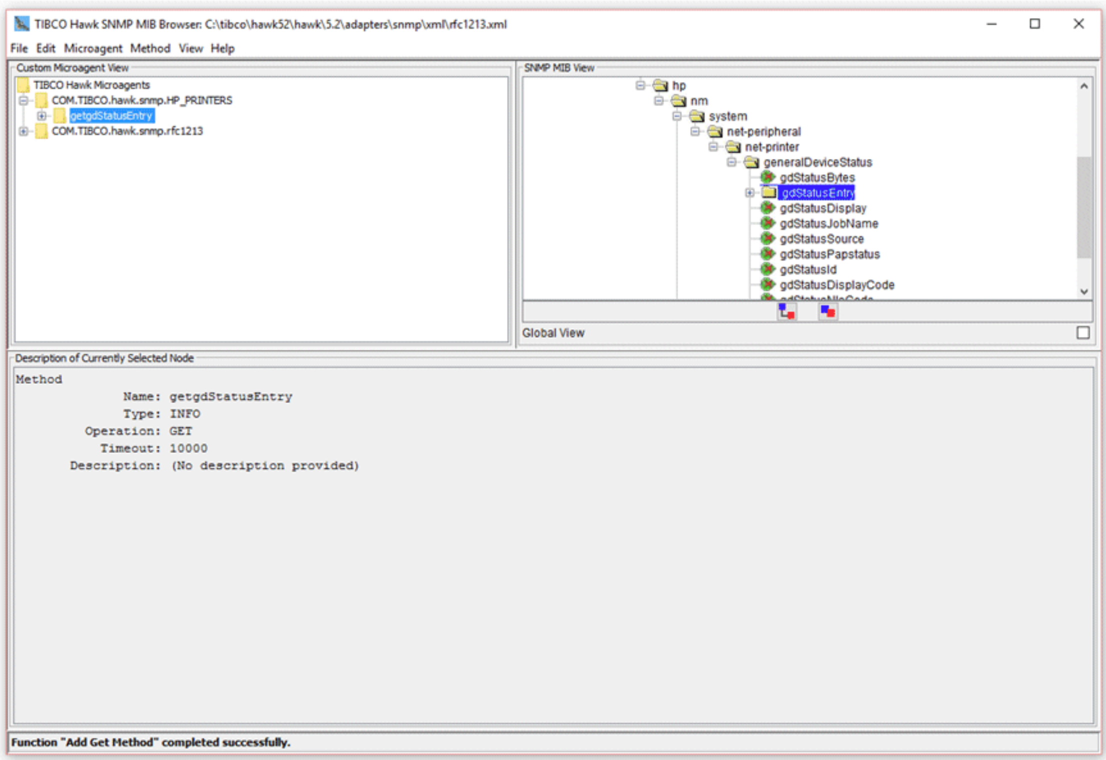
For set methods there are no returns, only arguments. Arguments are added based on the type of node selected in the same manner as returns are handled for get methods (see above). Arguments, however, are only added for writable nodes. If no writable variables are defined for the selected node then an error message is displayed and the create method fails. These arguments are used to specify the new value for the variable when the method is invoked. If the SNMP variable is an enumerated type then a legal value choice is defined for the argument containing the list of enumerated types.

For `onTrap` methods a return is added for each variable defined for the selected trap.

For `sendTrap` methods an argument is added for each variable associated with the selected trap.

Filter arguments can be added to get methods which return tabular data and to `onTrap` methods. A filter argument corresponds to a particular return and is used to filter the data returned by a method. For tabular get methods they are used to filter which table rows are to be returned. For `onTrap` methods they are used to filter out unwanted traps. Filter arguments are automatically added for all table index columns in tabular get methods. A filter argument can be added for any return. If the return is an enumerated type then a legal value choice is defined for the filter argument containing the list of enumerated types.

Arguments and returns can be removed as desired with two exceptions. Tabular get methods must return all index columns and tabular set methods must have an argument for each index column. An error is displayed if an attempt is made to delete an index return from a tabular get method or an index argument from a tabular set method.



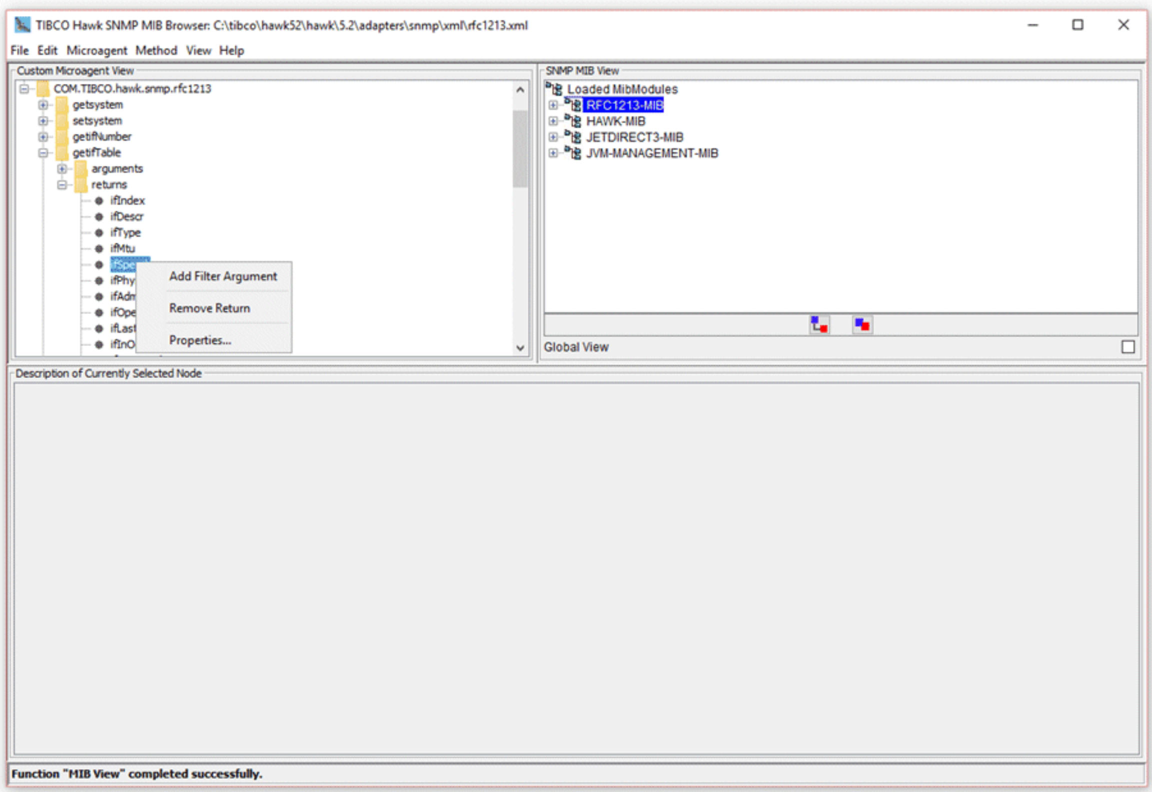

7. Select the **File->Save** options to save the microagent descriptor file.

## Adding Filter Arguments

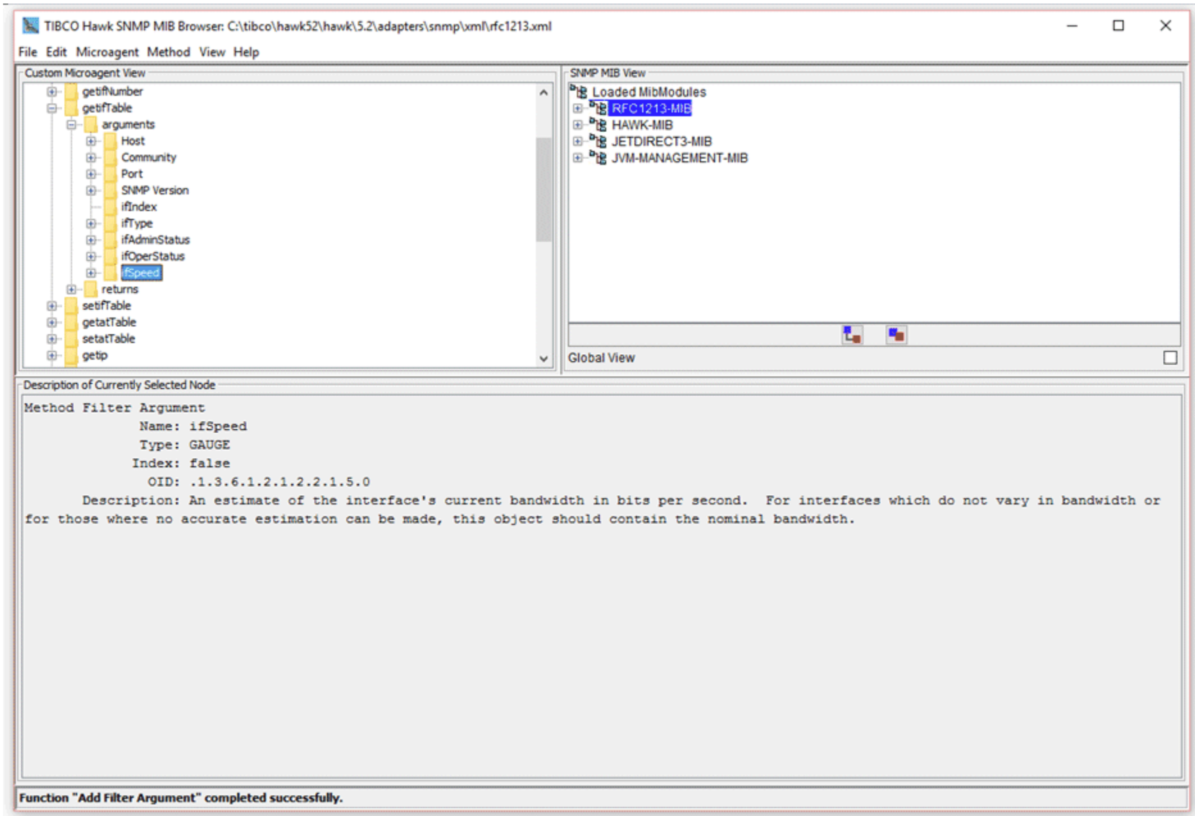
When working with `OnTrap` methods or methods that return tabular data you can specify any output parameter of the method as a filter argument. This adds an output filter argument corresponding to the selected output parameter. You can now filter the returned data using specific values for this output filter argument.

When used with `OnTrap` methods, the output filter is used to send only those traps that pass the filter criteria. Similarly, with methods that return tabular data, the output filter is used to filter rows of the return data.

To add a filter argument, right click on the selected output parameter and choose **Add Filter Argument**.

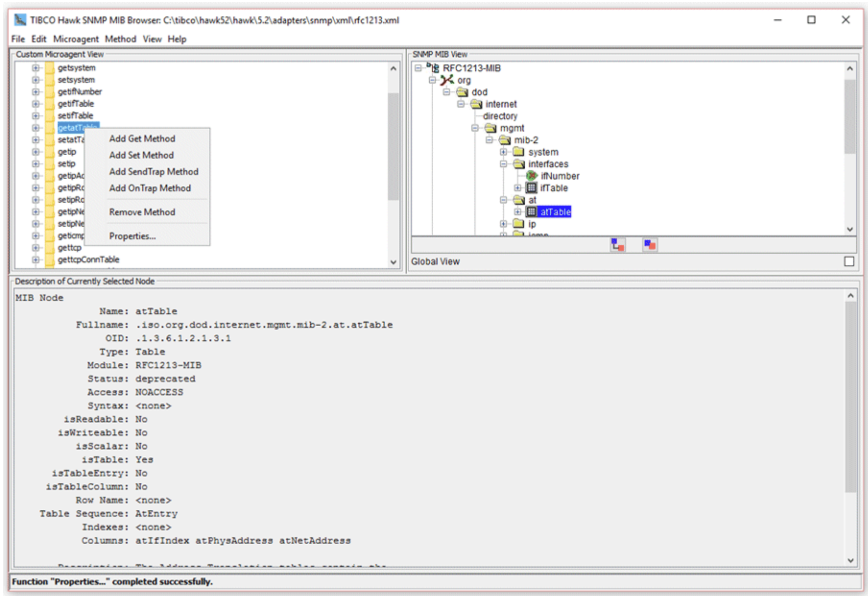


The selected output parameter is now added as an output filter. When using this method as a data source for a rulebase, the output filter is now an additional method argument.

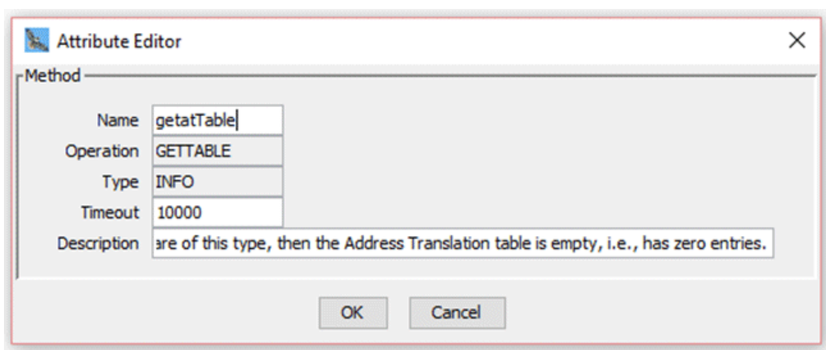


## Editing Microagent Properties

To edit the properties of a node in the custom microagent tree, right-click on that node and choose the **Properties...** option.




This brings up the **Attribute Editor** containing the properties of the selected node. The properties that can be edited are displayed with a white background:



# The SNMP Microagent Definition File

The following tables describe each configuration item of the microagent definition file created by the MIB browser.

The following table describes the top-level *<TIBHAWK_SNMP>*, which defines the document type definition (DTD) version that defines the syntax rules for the microagent definition file. More information, see Appendix A, Product Document Type Definition (DTD).

*Table 3 TIBHAWK_SNMP Element Definition*

| Element Name | Description | |
|---|---|---|
| *<TIBHAWK_SNMP>* | This element is a container for the *<microagent>* element. | |
| | **Attribute Name** | **Description** |
| | `dtd_version` | The version of the SNMP DTD used. |
| | **Contained Elements:** *<microagent>* | |

The following table describes the *<microagent>* element, which defines microagents for this instance of the adapter. Multiple microagents can be defined in a single configuration file.

*Table 4 Microagent Element Definition*

| Element Name | Description | |
|---|---|---|
| *<microagent>* | This element is a container for the *<microagent>* element. | |
| | **Attribute Name** | **Description** |
| | `name` | The name of the microagent. |
| | `display_name` | The given name of the microagent. This name will appear in TIBCO Hawk WebConsole. |
| | `help` | A help string that describes the parameter. If no value is specified, the `name` value of this element is used. Optional. |
| | | **Contained Elements:** *<method>* |

The following table describes the *<method>* element, which defines methods for a microagent. Methods are used to perform to perform the Get, Set, Sendtrap and Ontrap operations.

*Table 5 Method Element Definition*

| Element Name | Description | |
|---|---|---|
| *<method>* | This element is a container for the *<method>* element. | |
| | **Attribute Name** | **Description** |
| | `name` | The name of the method. |
| | `operation` | The SNMP operation being performed. Valid values:<br>`GET`<br>`SET`<br>`ONTRAP`<br>`SENDTRAP`<br>`GETTABLE`<br>`SETTABLE` |
| | `timeout` | The maximum number of milliseconds to wait for method results after invocation. The default value is 10000. If results are not returned in this interval, an error is generated. |
| | `help` | A help string that describes the parameter. If no value is specified, the `name` value of this element is used. Optional. |
| | `enterprise` | The SNMP enterprise ID associated with the trap. This attribute is only used for the `sendtrap` and `ontrap` operations. |
| | `traptype` | This attribute is only used for the `sendtrap` and `ontrap` operations. |
| | `trapspecifictype` | The specific trap type. |
| | **Contained Elements:** *<inputParameter>*, *<output Filter>*, *<outputParameter>* | |

*Table 6 Inputparameter Element Definition*

| Element Name | Description |
|---|---|
| *<inputParameter>* | This element is a container for the *<valueChoice>* element. |

*Table 6 Inputparameter Element Definition*

| Element Name | Description | |
|---|---|---|
| | **Attribute Name** | **Description** |
| | `name` | The name of the input parameter. |
| | `type` | The type of data the parameter accepts. Types are defined by the `valueType` entity in the adapter DTD. |
| | `oid` | The MIB object identifier. |
| | `help` | A help string that describes the parameter. If no value is specified, the `name` value of this element is used. Optional. |
| | `index` | Indicates if this output parameter is an index for tabular data. One of more `index` output parameters can be specified and should be in order of precedence with the most significant index first. |
| | **Contained Elements:** *<valueChoice>* , *<legalChoice>* | |

The following table describes the *<valueChoices>* element, which defines some acceptable values for the *<inputParameter>* element. These values are included in the dropdown list for method arguments in TIBCO Hawk WebConsole. Users can also type a value not in the list. This element is optional, and can only be used if *<legalValueChoices>* is not specified.

*Table 7 valueChoice Element Definition*

| Element Name | Description | |
|---|---|---|
| *<valueChoice>* | This element defines a set of acceptable input values for input parameters. Other values can be specified. | |
| | **Attribute Name** | **Description** |
| | `value` | One or more acceptable input values. Separate multiple values with commas. |
| | **Contained Elements:** `None` | |

*Table 8 LegalChoice Element Definition*

| Element Name | Description | |
|---|---|---|
| *<legalChoice>* | This element defines a set of legal input values for input parameters. Only defined values can be specified. | |
| | **Attribute Name** | **Description** |
| | `value` | One or more acceptable input values. Separate multiple values with commas. |
| | `label` | Description of the input value. |
| | **Contained Elements:** `None` | |

*Table 9 outputParameter Element Definition*

| Element Name | Description | |
|---|---|---|
| *<outputParameter>* | This element defines the method output parameters. | |
| | **Attribute Name** | **Description** |
| | `name` | The name of the output parameter. |
| | `type` | The type of the data the parameter returns. Types are defined by the `valueType` entity in the adapter DTD. |
| | `oid` | The MIB object identifier. |
| | `help` | A help string that describes the parameter. If no value is specified, the `name` value of this element is used.<br>Optional. |
| | `index` | Indicates if this output parameter is an index for tabular data.<br>One of more `index` output parameters can be specified and should be in order of precedence with the most significant index first. |
| | **Contained Elements:** `None` | |

*Table 10 outputFilter Element Definition*

| Element Name | Description | |
|---|---|---|
| *<outputFilter>* | This element defines the method argument which allows a filter entry to screen the returned data. | |
| | Attribute Name | Description |
| | `name` | The method argument name which will be used to screen the returned data. |
| | `type` | The type of data the parameter accepts.<br>Types are defined by the `valueType` entity in the adapter DTD. |
| | `oid` | The MIB object identifier. |
| | `index` | Indicates if this output parameter is an index for tabular data.<br>One of more `index` output parameters can be specified and should be in order of precedence with the most significant index first. |
| | `help` | A help string that describes the parameter. If no value is specified, the `name` value of this element is used.<br>Optional. |
| | **Contained Elements**: `None` | |

# Chapter 3 Using Adapter Methods

The adapter is represented by a microagent that communicates with TIBCO Hawk agents. The microagent exposes an interface with methods for monitoring and managing the adapter. This chapter describes adapter methods, which you can invoke directly or in a TIBCO Hawk rulebase.

## Topics

# COM.TIBCO.hawk.snmp

*Microagent*

The SNMP microagent exposes the following methods for monitoring the adapter.

*Table 11 SNMP Microagent Methods*

| Method | Description |
|---|---|
| snmp:getVariable | Gets the value of the specified MIB variable. |
| snmp:setVariable | Sets the value of the specified MIB variable. |
| snmp:getSystem | Gets the iso.org.dod.internet.mgmt.mib-2.system group variables. |
| snmp:onDiscovery | Checks all the IP addresses specified by the IP wildcard entered in the Host field to determine if they are SNMP enabled devices. |
| snmp:isTrapHandlerRunning | Checks if the adapter trap handler is currently running. |
| snmp:startTrapHandler | Starts the adapter trap handler. |
| snmp:stopTrapHandler | Stops the adapter trap handler. |
| snmp:getTrapStatistics | Gets statistics regarding SNMP traps processed. |
| snmp:isPublisherRunning | Checks if the adapter Publisher is currently running. |
| snmp:startPublisher | Starts the adapter publisher. |
| snmp:stopPublisher | Stops the adapter publisher. |
| snmp:getActiveAlerts | Gets the active alerts for the specified alert type, agent host name, and agent host DNS name. |
| snmp:getAlertFilters | Gets the current alert filter settings for the TIBCO Hawk alert publisher. |
| snmp:setAlertFilters | Sets the alert filter settings for the TIBCO Hawk alert publisher. |
| snmp:getEventStatistics | Gets statistics related to monitored TIBCO Hawk events. |
| snmp:sendHawkUserTrap | Sends a hawkUserTrap to the specified target. |
| snmp:onHawkUserTrap | Returns hawkUserTrap SNMP traps received by this host. |
| snmp:onUnknownTrap | Returns any unknown SNMP traps received by this host. |
| snmp:getReleaseVersion | Gets version information about the current release of this microagent. |
| snmp:setTraceLevel | Sets the trace level settings. |

*Table 11 SNMP Microagent Methods*

| Method | Description |
|---|---|
| snmp:getTraceLevel | Gets the current trace level settings. |
| snmp:setTraceParameters | Modifies the current trace configuration. |
| snmp:getTraceParameters | Gets the current trace configuration. |
| snmp:setSnmpVersionTrapPublisher | Sets the SNMP version used by the trap publisher. |
| snmp:_onUnsolicitedMsg | Returns any unsolicited notifications that are sent from this microagent. |

# snmp:getVariable

*Method*

This method returns the value(s) associated with the specified SNMP object ID (OID).

**Type** SYNC, IMPACT_INFO

**Arguments**

| Name | Type | Description |
|---|---|---|
| `Host` | String | The host name or IP address of target host for this SNMP request. |
| `Community` | String | The SNMP community to be used for the SNMP request. |
| `Port` | Integer | The port number on the target host for this SNMP request. |
| `SNMP Version` | String | The SNMP version to be used for this SNMP request. |
| `OID` | String | Specifies the object ID (OID) of the SNMP MIB variable to be retrieved. |

**Returns**

| Name | Type | Description |
|---|---|---|
| `OID` | String | The OID of the returned MIB variable. |
| `Data Type` | String | The data type of the returned MIB variable. |
| `Value as String` | String | The value of the MIB variable as a string (always returned). |
| `Value as Integer` | Integer | The value of the MIB variable as an integer (if applicable). |

**Usage Notes** This method returns the value associated with the specified SNMP MIB variable. The value is always returned as a string and also returned as an integer where appropriate. If the specified OID represents a non-terminal OID this method will return all OIDs in the specified subtree (that is, walk the MIB).

# snmp:setVariable

*Method*

Sets the value of the specified MIB variable.

**Type** SYNC, IMPACT_ACTION

**Arguments**

| Name | Type | Description |
|---|---|---|
| `Host` | String | The host name or IP address of target host for this SNMP request. |
| `Community` | String | The SNMP community to be used for the SNMP request. |
| `Port` | Integer | The port number on the target host for this SNMP request. |
| `SNMP Version` | String | The SNMP version to be used for this SNMP request. |
| `OID` | String | Specifies the object ID (OID) of the SNMP MIB variable being set. |
| `Data Type` | String | Specifies the data type of the SNMP MIB variable being set. |
| `Data Value` | String | Specifies the new data value to set. |

**Returns** None

**Usage Notes** This method sets the value of the specified SNMP MIB variable. The OID for non-indexed variables must be terminated with a '`.0`'. If the specified OID does not begin with a period (.) then the prefix `.1.3.6.1.2.1` will be automatically prepended to the entered OID.

## snmp:getSystem

*Method*

Returns the iso.org.dod.internet.mgmt.mib-2.system group variables.

**Type** SYNC, IMPACT_INFO

**Arguments**

| Name | Type | Description |
|---|---|---|
| `Host` | String | The host name of the SNMP agent from which the MIB variable is to be retrieved |
| `Community` | String | The SNMP community to be used for the SNMP request. |
| `Port` | Integer | The port number on the target host for this SNMP request. |
| `SNMP Version` | String | The SNMP version to be used for this SNMP request. |

**Returns**

| Name | Type | Description |
|---|---|---|
| `SysDescr` | String | The textual identification of the contact person for this managed node. |
| `sysName` | Integer | An administratively-assigned name for this managed node. By convention, this is the node's fully-qualified domain name. |
| `sysLocation` | String | The physical location of this node (for example,`telephone closet, 3rd floor'). |

| Name (Cont'd) | Type (Cont'd) | Description (Cont'd) |
|---|---|---|
| sysServices | Integer | A value which indicates the set of services that this entity primarily offers. The value is a sum. This sum initially takes the value zero, Then, for each layer, L, in the range 1 through 7, that this node performs transactions for, 2 raised to (L - 1) is added to the sum. For example, a node which performs primarily routing functions would have a value of 4 (2^(3-1)). In contrast, a node which is a host offering application services would have a value of 72 (2^(4-1) + 2^(7-1)). Note that in the context of the Internet suite of protocols, values should be calculated accordingly:<br>**layer functionality**<br>1 physical (for example., repeaters)<br>2 datalink/subnetwork (for example., bridges)<br>3 internet (for example., IP gateways)<br>4 end-to-end (for example., IP hosts)<br>7 applications (for example., mail relays)<br>For systems including OSI protocols, layers 5 and 6 may also be counted. |

## snmp:onDiscovery

*Method*

This method checks all the IP addresses specified by the IP wildcard entered in the Host field to determine if they are SNMP enabled devices. For each SNMP device discovered the iso.org.dod.internet.mgmt.mib-2.system group variables are returned. The IP wildcard can contain an asterix (*) character in any position where all addresses (0-255) are to be checked (for example, `160.101.*.*`).

**Type** Asynchronous, IMPACT_INFO

**Arguments**

| Name | Type | Description |
|---|---|---|
| `Host` | String | Host name or IP address of target host for this SNMP request. |
| `Community` | String | The SNMP community to be used for the SNMP request. |
| `Port` | Integer | The port number on the target host for this SNMP request. |
| `SNMP Version` | String | The SNMP version to be used for this SNMP request. |
| `Processing Threads` | Integer | Specifies the number of threads to be used to process this invocation. Increasing the number of threads allows multiple SNMP get requests to be executed in parallel increasing performance.<br>The default number of processing threads is 1. The maximum number of processing threads could be 100 only. If the specified threads are less than 1, the default value of 1 is used. Similarly, if the specified thread number is more than 100, the default maximum value will be used, that is, 100. |

**Returns**

| Name | Type | Description |
|---|---|---|
| `Host` | String | Host name or IP address of the discovered device. |
| `sysDescr` | String | A textual description of the entity. This value should include the full name and version identification of the system's hardware type, software operating-system, and networking software. It is mandatory that this only contain printable ASCII characters. |

| Name (Cont'd) | Type (Cont'd) | Description (Cont'd) |
|---|---|---|
| `sysObject ID` | String | The vendor's authoritative identification of the network management subsystem contained in the entity. This value is allocated within the SMI enterprises subtree (1.3.6.1.4.1) and provides an easy and unambiguous means for determining `what kind of box' is being managed. For example, if vendor `Flintstones, Inc.' was assigned the subtree 1.3.6.1.4.1.4242, it could assign the identifier 1.3.6.1.4.1.4242.1.1 to its `Fred Router'. |
| `sysUpTime` | String | The time (in hundredths of a second) since the network management portion of the system was last re-initialized. |
| `sysContact` | String | The textual identification of the contact person for this managed node, |
| `sysName` | Integer | An administratively-assigned name for this managed node. By convention, this is the node's fully-qualified domain name. |
| `sysLocation` | String | The physical location of this node (for example,`telephone closet, 3rd floor'). |
| `sysServices` | Integer | A value which indicates the set of services that this entity primarily offers. The value is a sum. This sum initially takes the value zero, Then, for each layer, L, in the range 1 through 7, that this node performs transactions for, 2 raised to (L - 1) is added to the sum. For example, a node which performs primarily routing functions would have a value of 4 (2^(3-1)). In contrast, a node which is a host offering application services would have a value of 72 (2^(4-1) + 2^(7-1)). Note that in the context of the Internet suite of protocols, values should be calculated accordingly:<br>**layer functionality**<br>1 physical (for example., repeaters)<br>2 datalink/subnetwork (for example., bridges)<br>3 internet (for example., IP gateways)<br>4 end-to-end (for example., IP hosts)<br>7 applications (for example., mail relays)<br>For systems including OSI protocols, layers 5 and 6 may also be counted. |

## snmp:isTrapHandlerRunning

*Method*

Checks if the TIBCO Hawk SNMP trap handler is currently running.

**Type** SYNC, IMPACT_INFO

**Arguments** None

**Returns**

| Name | Type | Description |
|---|---|---|
| `isRunning` | Boolean | A return value of TRUE means the adapter is currently running. A value of FALSE means the adapter is not currently running. |

**Usage Notes** This method is used to determine whether the SNMP trap handler is currently running or not. When the SNMP trap handler is running, SNMP traps received by the local host will trigger asynchronous methods in the TIBCO Hawk agent (for example, `onHawkTrap()`). When the SNMP trap handler is not running, SNMP traps are ignored by the microagent.

## snmp:startTrapHandler

*Method*

Starts the TIBCO Hawk SNMP trap handler.

**Type** SYNC, IMPACT_ACTION

**Arguments** None

**Returns** None

**Usage Notes** This method starts adapter trap handler. When the SNMP trap handler is running, SNMP traps received by the local host will trigger asynchronous methods in the TIBCO Hawk agent (for example. `onHawkTrap()`). When the SNMP trap handler is not running, SNMP traps are ignored by the microagent.

# snmp:stopTrapHandler

*Method*

Stops the TIBCO Hawk SNMP trap handler.

**Type** SYNC, IMPACT_ACTION

**Arguments** None

**Returns** None

**Usage Notes** This method stops the TIBCO Hawk SNMP trap handler. When the SNMP trap handler is running, SNMP traps received by the local host will trigger asynchronous methods in the TIBCO Hawk agent (for example. `onHawkTrap()`). When the SNMP trap handler is not running, SNMP traps are ignored by the microagent.

# snmp:getTrapStatistics

*Method*

Returns statistics regarding the SNMP traps processed.

**Type** SYNC, IMPACT_INFO

**Arguments** None

**Returns**

| Name | Type | Description |
|---|---|---|
| `Traps Sent` | Integer | The count of SNMP traps sent. |
| `Agent Alive Traps Sent` | Integer | The count of Agent Alive SNMP traps actually sent by the alert publisher. |
| `Agent Dead Traps Sent` | Integer | The count of Agent Dead SNMP traps actually sent by the alert publisher. |
| `Alert Traps Sent` | Integer | The count of Alert SNMP traps actually sent by the alert publisher. |
| `Alert Clear Traps Sent` | Integer | The count of Alert Clear SNMP traps actually sent by the alert publisher. |
| `PDU Queue Size` | Integer | The current size of the SNMP trap handler PDU queue. |
| `PDUs Queued` | Integer | The count of PDU's queued. |
| `PDUs Processed` | Integer | The count of PDU's processed. |
| `Traps received` | Integer | The count of all SNMP traps received by the trap handler. |
| `V1 Traps Received` | Integer | The count of SNMP Version 1 traps received by the trap handler. |
| `V2 Traps Received` | Integer | The count of SNMP Version 2 traps received by the trap handler. |
| `Non-Trap PDUs Received` | Integer | The count of non-trap PDU's received by the trap handler. |

**Usage Notes** Use this method to monitor the SNMP trap activity. The appropriate counters are reset to zero when the trap handler or trap publisher are stopped.

## snmp:isPublisherRunning

*Method*

Checks if the TIBCO Hawk SNMP trap publisher is currently running.

**Type** SYNC, IMPACT_INFO

**Arguments** None

**Returns**

| Name | Type | Description |
|---|---|---|
| `IsRunning` | Boolean | A return value of TRUE means the publisher is currently running. A value of FALSE means the publisher is not currently running. |

**Usage Notes** This method returns a boolean value indicating whether the TIBCO Hawk SNMP trap publisher is currently running or not. Which TIBCO Hawk events are published can be configured using the methods `getAlertFilters()` and `setAlertFilters()`. The currently active TIBCO Hawk alerts can be queried using the method `getActiveAlerts()`.

## snmp:startPublisher

*Method*

Starts the TIBCO Hawk SNMP trap publisher.

**Type** SYNC, IMPACT_ACTION

**Arguments** None

**Returns** None

**Usage Notes** This method starts the TIBCO Hawk SNMP trap publisher. Which TIBCO Hawk events are published can be configured using the methods `getAlertFilters()` and `setAlertFilters()`. The currently active TIBCO Hawk alerts can be queried using the method `getActiveAlerts()`.

## snmp:stopPublisher

*Method*

Stops the TIBCO Hawk SNMP trap publisher.

**Type** SYNC, IMPACT_ACTION

**Arguments** None

**Returns** None

**Usage Notes** Which TIBCO Hawk events are published can be configured using the methods `getAlertFilters()` and `setAlertFilters()`. The currently active TIBCO Hawk alerts can be queried using the method `getActiveAlerts()`.

# snmp:getActiveAlerts

*Method*

Returns the active alerts for the specified alert type, agent host name, and agent host DNS name. Select `ALL` to get alerts for all types, all host names, and/or all host DNS names, respectively.

**Type** SYNC, IMPACT_INFO

**Arguments**

| Name | Type | Description |
|---|---|---|
| `Alert Type` | String | The type of TIBCO Hawk alerts to be retrieved. Only active alerts of the specified type will be returned. Select `ALL` to retrieve all active alerts regardless of type. |
| `Host Name` | String | The host name of the TIBCO Hawk agent for which active alerts are to be returned. Select `ALL` to retrieve active alerts for all TIBCO Hawk agents regardless of host name. |
| `Host DNS Name` | String | The host DNS name of the TIBCO Hawk agent(s) for which active alerts are to be returned. Select `ALL` to retrieve active alerts for all TIBCO Hawk agents regardless of host DNS name. |

**Returns**

| Name | Type | Description |
|---|---|---|
| `Rulebase` | String | The name of the rulebase which generated this alert. |
| `Time Created` | String | The time the alert was created. |
| `Host IP` | String | The IP address of the TIBCO Hawk agent where this alert originated. |
| `Host State` | String | The highest alert state currently pending for the TIBCO Hawk agent where this alert originated. The alert states are `AGENT DEAD`, `ALERT HIGH`, `ALERT LOW`, `WARNING`, and `OK`. |
| `Host DNS Name` | String | The host DNS name for the TIBCO Hawk agent where this alert originated. |
| `Start Time` | String | The time the TIBCO Hawk agent where this alert originated was started. |

| Name (Cont'd) | Type (Cont'd) | Description (Cont'd) |
|---|---|---|
| `Index` | String | An identifier for the alert which is unique across all TIBCO Hawk agents. It is the concatenation of the originating TIBCO Hawk agent's host name and DNS name along with the alert ID. The alert ID alone is not unique across all TIBCO Hawk agents. |
| `Rulebase` | String | The name of the rulebase which generated this alert. |
| `Time Created` | String | The time the alert was created. |
| `Host IP` | String | The IP address of the TIBCO Hawk agent where this alert originated. |
| `Host State` | String | The highest alert state currently pending for the TIBCO Hawk agent where this alert originated. The alert states are `AGENT DEAD, ALERT HIGH, ALERT LOW, WARNING`, and `OK`. |

**Usage Notes** This method returns the list of active alerts maintained by the trap publisher. The trap publisher must be running when using this method or no data is returned.

## snmp:getAlertFilters

*Method*

This method gets the current alert filter settings for the TIBCO Hawk alert publisher. These filters control which alerts are published. Alerts can be filtered according to alert severity or alert source, or both. An alert source can be either a TIBCO Hawk agent or a rulebase, or both.

**Type** SYNC, IMPACT_INFO

**Arguments** None

**Returns**

| Name | Type | Description |
|---|---|---|
| `Agent Alive` | String | The value of the current filtering in effect for agent alive alerts. A value of `ON` indicates that agent alive alerts are being published and that of `OFF` indicates that agent alive alerts are not being published. |
| `Agent Dead` | String | The value of the current filtering in effect for agent dead alerts. A value of `ON` indicates that agent dead alerts are being published and that of `OFF` indicates that agent dead alerts are not being published. |
| `Alert High` | String | The value of the current filtering in effect for high severity alerts. A value of `ON` indicates that high severity alerts are being published and that of `OFF` indicates that high severity alerts are not being published. |
| Alert Medium | String | This return value specifies the current filtering in effect for medium severity alerts. ON indicates that medium severity alerts are being published. OFF indicates that medium severity alerts are not being published. |
| `Alert Low` | String | The value of the current filtering in effect for low severity alerts. A value of `ON` indicates that low severity alerts are being published and that of `OFF` indicates that low severity alerts are not being published. |
| `Information` | String | The value of the current filtering in effect for information alerts. A value of `ON` indicates that information alerts are being published and that of `OFF` indicates that information alerts are not being published. |

| Name (Cont'd) | Type (Cont'd) | Description (Cont'd) |
|---|---|---|
| `Rulebase` | String | The Perl5 pattern being used to filter alerts by rulebase name. Only alerts generated by rulebases with names that match the specified Perl5 pattern will be published. ALL indicates that alerts for all rulebases should be published. This filter is not applied to agent alive and agent dead alerts. |
| `Host Name` | String | The Perl5 pattern being used to filter alerts by the TIBCO Hawk host name. Only alerts originating from TIBCO Hawk agents with host names that match the specified Perl5 pattern will be published. ALL indicates that alerts for all TIBCO Hawk agent host names should be published. |
| `Host DNS Name` | String | The Perl5 pattern being used to filter alerts by the TIBCO Hawk host DNS name. Only alerts originating from TIBCO Hawk agents with host DNS names that match the specified Perl5 pattern will be published. ALL indicates that alerts for all TIBCO Hawk agent host DNS names should be published. |

# snmp:setAlertFilters

*Method*

This method sets the current alert filter settings for the TIBCO Hawk alert publisher. These filters control which alerts are published. Alerts can be filtered according to alert severity or alert source, or both. An alert source can be a TIBCO Hawk agent or a rulebase, or both.

**Type** SYNC, IMPACT_ACTION

**Arguments**

| Name | Type | Description |
|---|---|---|
| `Agent Alive` | String | Controls the filtering of agent alive alerts. Select ON to enable publishing of agent alive alerts. Select OFF to disable publishing of agent alive alerts. Select CURRENT to retain current filter setting (no change). |
| `Agent Dead` | String | Controls the filtering of agent dead alerts. Select ON to enable publishing of agent dead alerts. Select OFF to disable publishing of agent dead alerts. Select CURRENT to retain current filter setting (no change). |
| `Alert High` | String | Controls the filtering of high severity alerts. Select ON to enable publishing of high severity alerts. Select OFF to disable publishing of high severity alerts. Select CURRENT to retain current filter setting (no change). |
| Alert Medium | String | Controls the filtering of medium severity alerts. Select ON to enable publishing of medium severity alerts. Select OFF to disable publishing of medium severity alerts. Select CURRENT to retain current filter setting (no change). |
| `Alert Low` | String | Controls the filtering of low severity alerts. Select ON to enable publishing of low severity alerts. Select OFF to disable publishing of low severity alerts. Select CURRENT to retain current filter setting (no change). |
| `Information` | String | Controls the filtering of information alerts. Select ON to enable publishing of information alerts. Select OFF to disable publishing of information alerts. Select CURRENT to retain current filter setting (no change). |

| Name (Cont'd) | Type (Cont'd) | Description (Cont'd) |
|---|---|---|
| `Rulebase` | String | The Perl5 pattern used to filter alerts by rulebase name.<br>Only alerts generated by rulebases with names that match the specified Perl5 pattern will be published.<br>ALL indicates that alerts for all rulebases should be published. This filter is not applied to agent alive and agent dead alerts. |
| `Host Name` | String | The Perl5 pattern used to filter alerts by TIBCO Hawk agent host name. Alerts originating from the TIBCO Hawk agent with host names that match the specified Perl5 pattern will be published. ALL indicates that alerts for all TIBCO Hawk agent host names should be published. |
| `Host DNS Name` | String | The Perl5 pattern used to filter alerts by TIBCO Hawk agent host DNS name. Alerts originating from the TIBCO Hawk agents with host DNS names that match the specified Perl5 pattern will be published. ALL indicates that alerts for all TIBCO Hawk agent host DNS names should be published. |

**Returns** None

## snmp:getEventStatistics

*Method*

Returns statistics related to the monitored TIBCO Hawk events.

**Type** SYNC, IMPACT_INFO

**Arguments** None

**Returns**

| Name | Type | Description |
|---|---|---|
| `Event Queue SIze` | Integer | The current size of the TIBCO Hawk event queue. |
| `Events Queued` | Integer | The count of the TIBCO Hawk events queued. |
| `Events Processed` | Integer | The count of the TIBCO Hawk events processed. |
| `Alerts Processed` | Integer | The count of the TIBCO Hawk alerts processed. |

# snmp:sendHawkUserTrap

This section only describes the TIBCO Hawk user trap. For the procedure on how to send a user-defined SNMP trap, see Using the MIB Browser, page 23.

*Method*

Send a `hawkUserTrap` to the specified target. This trap allows you to specify an integer ID and descriptive text.

**Type** SYNC, IMPACT_ACTION_INFO

**Arguments**

| Name | Type | Description |
|---|---|---|
| `Host` | String | The host name or IP address of the target host for this SNMP request. |
| `Community` | String | The SNMP community to be used for this SNMP request. |
| `Port` | Integer | The port number on the target host for this SNMP request. |
| `SNMP Version` | String | The SNMP version to be used for this SNMP request. |
| `hawkUserTrapID` | Integer | The user specified ID for the SNMP trap being sent. This allows the user to uniquely identify the trap. It is not the same as the object ID used in the SNMP trap. |
| `hawkUserTrapText` | String | The user specified description string for the SNMP trap being sent.<br>This string should not be greater than 62 KBytes. |

**Returns** None

## snmp:onHawkUserTrap

*Method*

Returns `hawkUserTrap` SNMP traps received by this host.

**Type** Asynchronous, IMPACT_ACTION_INFO

**Arguments**

| Name | Type | Description |
|---|---|---|
| `Peer` | String | The IP address of the SNMP peer who sent this SNMP trap. |
| `Agent` | String | The IP address of the SNMP agent which generated this SNMP trap. |
| `Enterprise` | String | The SNMP enterprise ID associated with this SNMP trap. |
| `Community` | String | The SNMP community associated with this SNMP trap. |
| `Type` | Integer | The SNMP generic trap type associated with this SNMP trap. |
| `Specific` | Integer | The SNMP specific trap type associated with this SNMP trap. |
| `Time` | String | The timestamp associated with this SNMP trap. |
| `hawkUserTrapID` | Integer | The user specified ID associated with this SNMP trap. This allows the user to uniquely identify the trap. |
| `hawkUserTrapText` | String | The user specified description string associated with this SNMP trap.<br>This string should not be greater than 62 KBytes. |

**Returns** None

**Usage Notes** The trap Handler must be started in order to receive any traps. See the snmp:startTrapHandler method for additional information.

## snmp:onUnknownTrap

*Method*

Returns any unknown SNMP traps received by this host.

**Type** Asynchronous, IMPACT_INFO

**Arguments** None

**Returns**

| Name | Type | Description |
|---|---|---|
| `Peer` | String | The IP address of the SNMP peer who sent this SNMP trap. |
| `Agent` | String | The IP address of the SNMP agent which generated this SNMP trap. |
| `Enterprise` | String | The SNMP enterprise ID associated with this SNMP trap. |
| `Community` | String | The SNMP community associated with this SNMP trap. |
| `Type` | Integer | The SNMP generic trap type associated with this SNMP trap. |
| `Specific` | Integer | The SNMP specific trap type associated with this SNMP trap. |
| `Time` | String | The timestamp associated with this SNMP trap. |

**Usage Notes** An unknown trap is any trap for which a specific microagent method has not been defined.

The trap handler must be started in order to receive any traps. See the snmp:startTrapHandler method for additional information.

# snmp:getReleaseVersion

*Method*

Returns version information about the current release of this microagent.

**Type** SYNC, IMPACT_INFO

**Arguments** None

**Returns**

| Name | Type | Description |
|---|---|---|
| `Name` | String | The name of the application which implements this microagent. |
| `Version` | String | The release version. |
| `Date` | String | The release date. |
| `Major` | Integer | The release version major number. |
| `Minor` | Integer | The release version minor number. |
| `Update` | Integer | The release version update number. |

## snmp:setTraceLevel

*Method*

Sets the trace level setting.

**Type** SYNC, IMPACT_ACTION_INFO

**Arguments**

| Name | Type | Description |
|---|---|---|
| `Information` | String | This level enables tracing of informational messages. This level can be enabled at all times. |
| `Warning` | String | This level enables tracing of warning messages. This level can be enabled at all times. |
| `Error` | String | This level enables tracing of error messages. This level can be enabled at all times. |
| `Debug` | String | This level increases the detail of information in trace output to aid in investigation of problems. This option is for troubleshooting purposes only, and under normal circumstances should not be enabled. |
| `AMI` | String | This trace level enables low level tracing of AMI operations. This level aids the investigation of problems related to AMI and should not be enabled under normal circumstances. |
| `SNMP` | String | This trace level enables low level tracing of SNMP operations. This level aids the investigation of problems related to SNMP and should not be enabled under normal circumstances. |
| `Publisher` | String | This trace level enables TIBCO Hawk alert publisher tracing. This level aids the investigation of problems related to the publisher and should not be enabled under normal circumstances. |

**Returns**

| Name | Type | Description |
|---|---|---|
| `Information` | String | The current information trace setting. |
| `Warning` | String | The current warning trace setting. |

| Name (Cont'd) | Type (Cont'd) | Description (Cont'd) |
|---|---|---|
| `Error` | String | The current error trace setting. |
| `Debug` | String | The current debug trace setting. |
| `AMI` | String | The current debug AMI trace setting. |
| `SNMP` | String | The current SNMP trace setting. |
| `Publisher` | String | The current TIBCO Hawk alert publisher trace setting. |

**Usage Notes** Each trace level can be set to On (enabled), Off (disabled), or Current (leave current setting as is).

## snmp:getTraceLevel

*Method*

Returns the current trace level settings.

**Type** SYNC, IMPACT_INFO

**Arguments** None

**Returns**

| Name | Type | Description |
|---|---|---|
| `Information` | String | The information trace setting. |
| `Warning` | String | The warning trace setting. |
| `Error` | String | The error trace setting. |
| `Debug` | String | The debug trace setting. |
| `AMI` | String | The debug AMI trace setting. |
| `SNMP` | String | The SNMP trace setting. |
| `Publisher` | String | The TIBCO Hawk alert publisher trace setting. |

# snmp:setTraceParameters

*Method*

Modifies the current trace configuration.

**Type** SYNC, IMPACT_INFO

**Arguments**

| Name | Type | Description |
|---|---|---|
| `Directory` | String | The information trace setting. |
| `File Name` | String | The warning trace setting. |
| `Max File Size` | String | Trace file maximum size before it is rolled over in Kilobytes |
| `Max Trace File` | Integer | Maximum number of trace file to keep in the trace directory. Use string `Current` for keeping the current setting for string parameters and use integer value `0` for keeping the current setting for integer parameters |

**Returns**

| Name | Type | Description |
|---|---|---|
| `Directory` | String | The current information trace setting. |
| `File Name` | String | The current warning trace setting. |
| `Max File Size` | String | Trace file maximum size before it is rolled over in Kilobytes. |
| `Max Trace File` | Integer | Maximum number of trace file to keep in the trace directory. Use string `Current` for keeping the current setting for string parameters and use integer value `0` for keeping the current setting for integer parameters |

## snmp:getTraceParameters

*Method*

Returns the current trace configuration.

**Type** SYNC, IMPACT_ACTION

**Arguments** None

**Returns**

| Name | Type | Description |
|---|---|---|
| `Directory` | String | The information trace setting. |
| `File Name` | String | The warning trace setting. |
| `Max File Size` | String | Trace file maximum size before it is rolled over in Kilobytes |
| `Max Trace File` | Integer | Maximum number of trace file to keep in the trace directory. Use string `Current` for keeping the current setting for string parameters and use integer value `0` for keeping the current setting for integer parameters |

## snmp:setSnmpVersionTrapPublisher

Sets the SNMP version used by the trap publisher.

**Type** SYNC, IMPACT_ACTION

**Arguments**

| Name | Type | Description |
|---|---|---|
| `SNMP Version` | String | The SNMP version to be used by the trap publisher. |

**Returns** None

## snmp:_onUnsolicitedMsg

*Method*

Subscribing to this method returns any unsolicited notifications that are sent from the managed application corresponding to this microagent. Invoking this method returns the last such message that was received (if any).

**Type** Asynchronous, IMPACT_INFO

**Arguments** None

**Returns**

| Name | Type | Description |
|---|---|---|
| `TYPE` | String | The type of this trap. |
| `TEXT` | String | The descriptive text for this trap. |
| `ID` | Integer | The ID for this trap. |
| `INBOX` | String | The inbox address of the adapter AMI interface, used to send the message and identify its source. |

# Chapter 4 Using Methods in a Rulebase

This chapter describes several sample implementations of default and custom adapter methods. Adapter methods can be used as a data source or an action in a TIBCO Hawk rulebase.

## Topics

# Monitoring Printer Activity

This section uses the custom microagent, `HP_Printer`, and its microagent descriptor file created in the previous chapter to create a rulebase which will monitor a printer for a ’paper jam’ condition. By following the steps in this section, you generate a TIBCO Hawk alert and view the alert in TIBCO Hawk WebConsole.

To follow this example, the adapter must be installed and running on a machine where a TIBCO Hawk agent is also running. To use the new microagent, include the microagent descriptor file in the `-xml_file` parameter of the adapter configuration file `(tibhawksnmp.cfg)`, then restart the adapter.

## Building a Rule for Data Capture

These steps demonstrate how to create a TIBCO Hawk rulebase for monitoring and managing the adapter using TIBCO Hawk WebConsole. The rulebase contains a simple rule that uses the `HP_PRINTERS:getgdStatusEntry` method as a data source. A test of the `gdStatusPaperJam` element of this data source checks for a true condition. If the TIBCO Hawk agent where the adapter is running detects the true condition of the `gdStatusPaperJam` element, the agent generates an alert message.

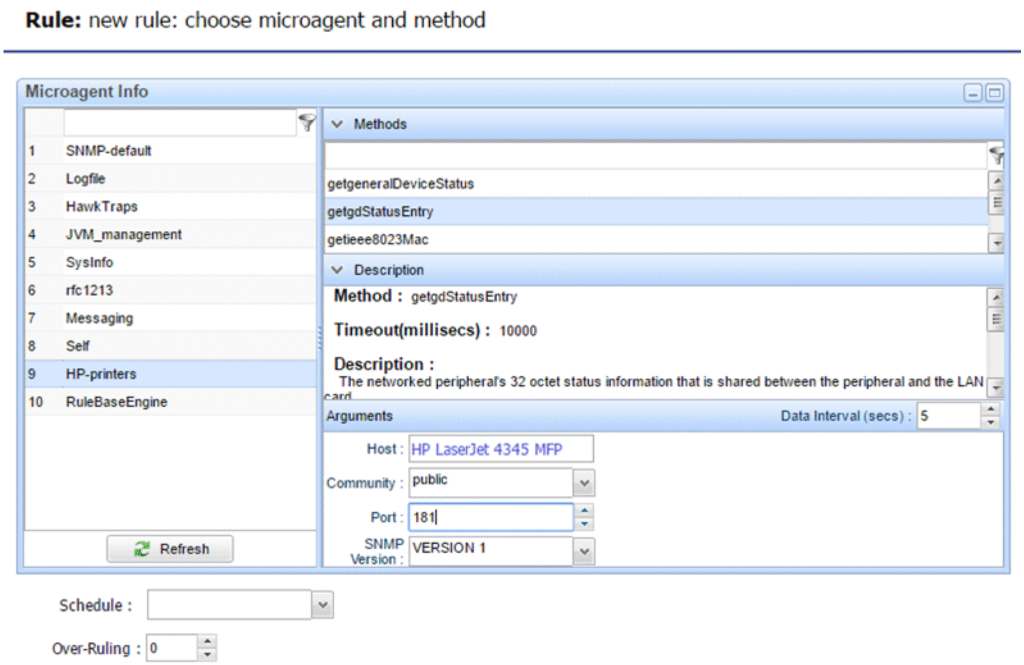
To create a new rulebase in TIBCO Hawk WebConsole:

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 `PaperJam` in the **Rulebase Name** field.
4. Enter the other fields, as required.
5. Click **Apply Changes** to save the current changes.

6. Click **Apply & Add Rule** to save the changes and proceed to the rule building process.

### Create a new rule

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




1. Select the `HP_PRINTERS` microagent. The Methods panel displays the methods for the microagent.
2. Select the `getgdStatusEntry` method.

   This method returns the general device parameters of the printer. It take a host name as an argument. This is a synchronous method. When the rule is active, the agent subscribes to this method and receives data every 60 seconds, by default.
3. In the **Host** field, specify the host name of an HP laser printer.

   This argument is the name of the printer that we want to monitor for a 'paper jam' condition.
4. Click **Apply Changes** to save the current changes.

5. Click **Apply and Add Test** to save the changes and proceed to defining the test process.

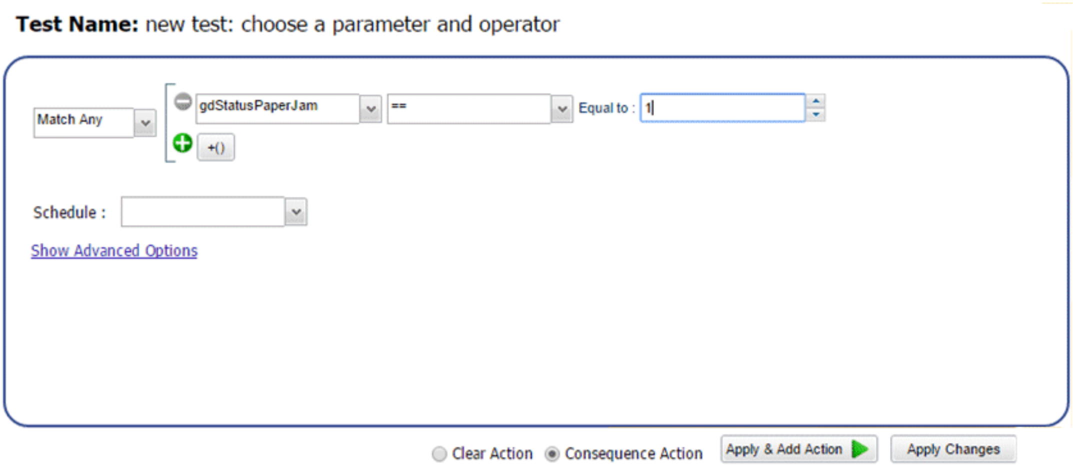
The rule is now configured to use the `getgdStatusEntry()` method of the `HP_PRINTERS` microagent.

## Applying Test Logic

Build test expression by specifying a test parameter and test operator in the Test screen.

To create a new test for the current rule:

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




2. You can specify if you want to match all, match any, or match none.
3. Specify the test expression as `gdStatusPaperJam` equal to (==) 1.
4. Click **Apply Changes** to save the test expression.
5. Click **Apply and Add Action** to save the changes and proceed to defining the action process.

## Generating an Alert Message

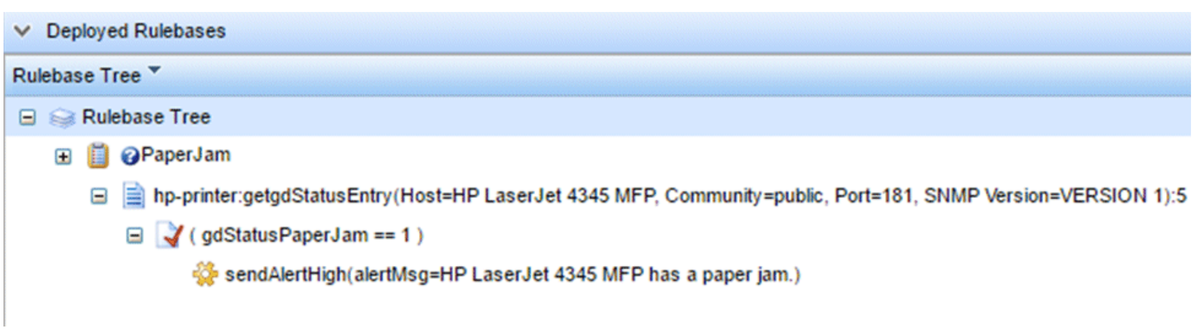
1. Go to the Action screen.


2. By default, the action type is **Alert** and the **Alert Level** is **High**.
3. In the **Alert Message** field, specify the following message:

```
HP LaserJet 4345 MFP has a paper jam
```

4. Click **Save and Deploy** to save changes and deploy it to the agent.

   The **PaperJam** rulebase is included in the Rulebase tree.

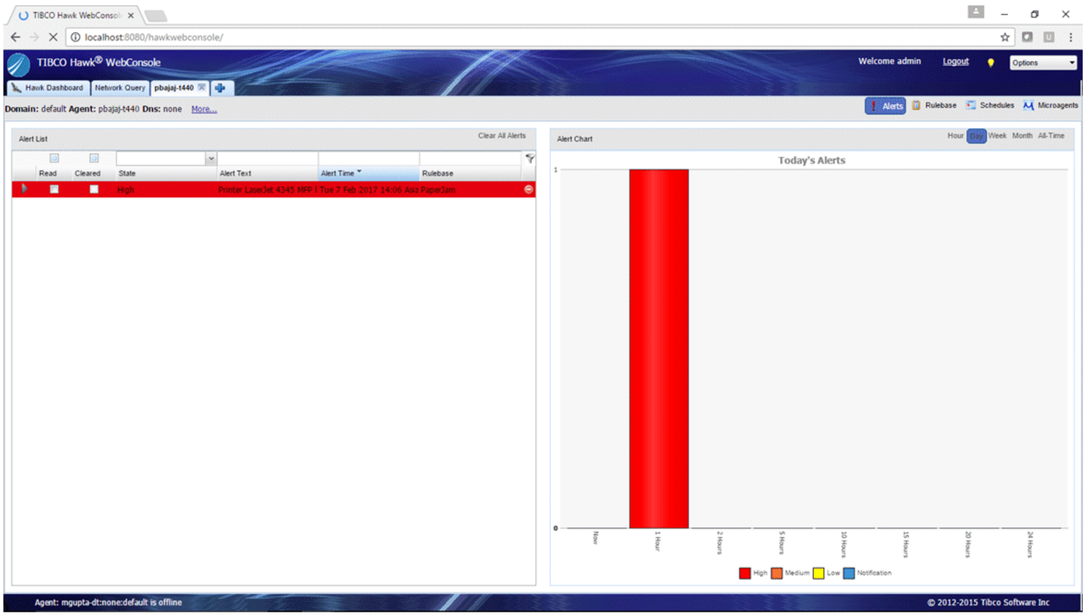



## Viewing the Alert Message in TIBCO Hawk WebConsole

To view the alert message generated by the **PaperJam** rulebase:

1. Click on the **Alerts** tab.

The Alert Display window shows the alert generated by the **PaperJam** rulebase:




2. Expand the alert to display detailed properties.

# Handling SNMP Traps

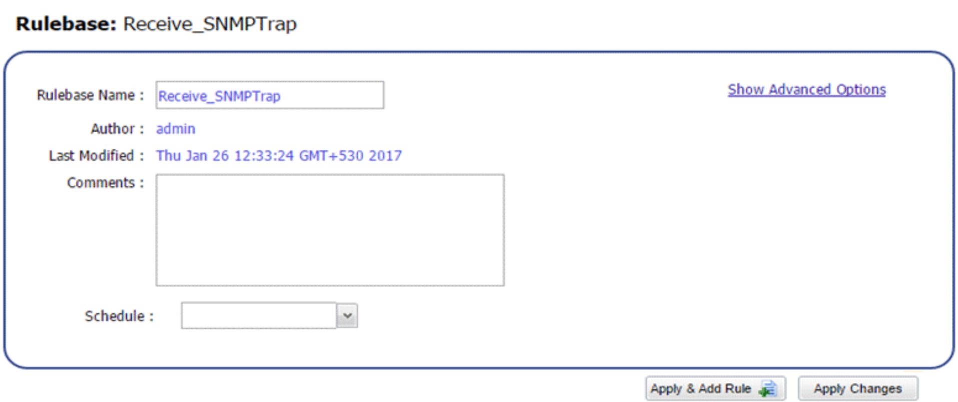
This section describes how to build a rule for capturing SNMP traps. The rulebase `Receive_SNMPTrap` contains a simple rule that uses the `snmp:onHawkUserTrap` method as a data source. When the TIBCO Hawk user trap is received, the adapter generates a TIBCO Hawk alert.

To follow the example, the adapter must be installed and running on a machine where a TIBCO Hawk agent is also running. The TIBCO Hawk agent service must be started, and TIBCO Hawk SNMP Trap Handler must also be running.

## Building a Rule for Data Capture

To create a new rulebase in TIBCO Hawk WebConsole:

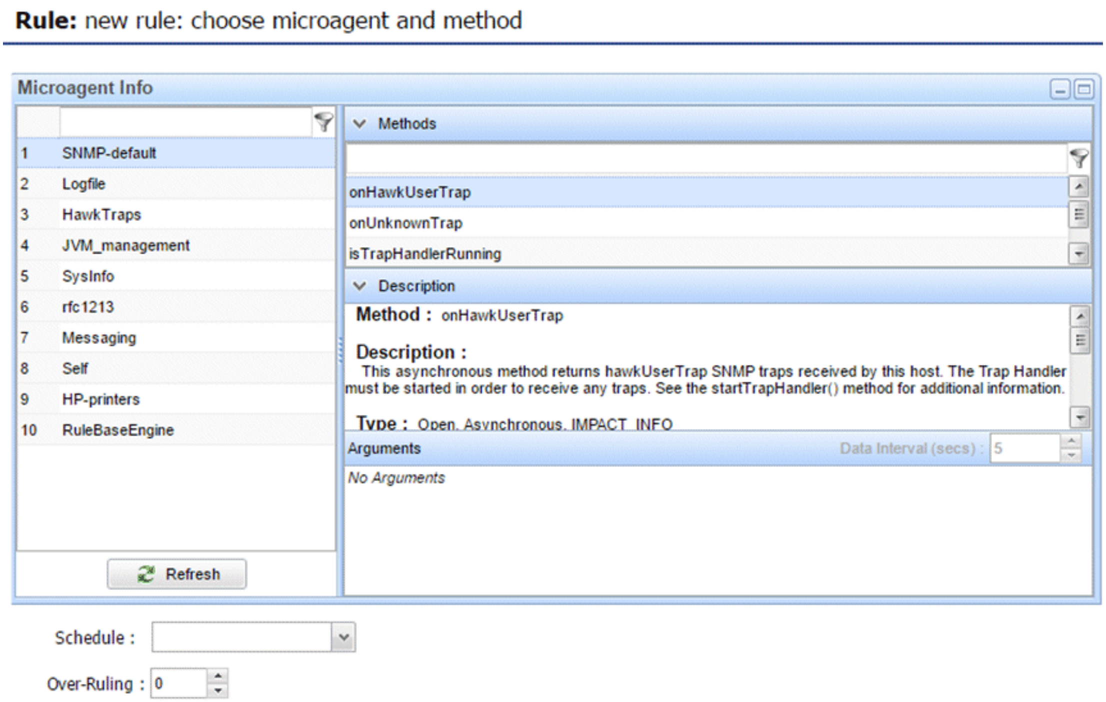
1. Click an agent listed in the **Agents** portlet on the Hawk Dashboard.
2. Click the **Rulebase** tab. The Rulebase screen lists the rulebases loaded by the selected agent and provides a toolbar for performing rulebase operations.
3. Click the **Create new Rulebase** icon from the toolbar to create a new rulebase. A new rulebase is created with the default name "New-Rulebase". Type **Receive_SNMPTrap** in the **Rulebase Name** field.




4. Enter the other fields, as required.
5. Click **Apply Changes** to save the current changes.
6. Click **Apply & Add Rule** to save the changes and proceed to the rule building process.

### Create a new rule

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




2. Select the **SNMP-default** microagent. The Methods panel displays the methods for the microagent display.

3. Select the **onHawkUserTrap** 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.

4. Click **Apply Changes** to save the current changes.

5. Click **Apply and Add Test** to save the changes and proceed to defining the test process.

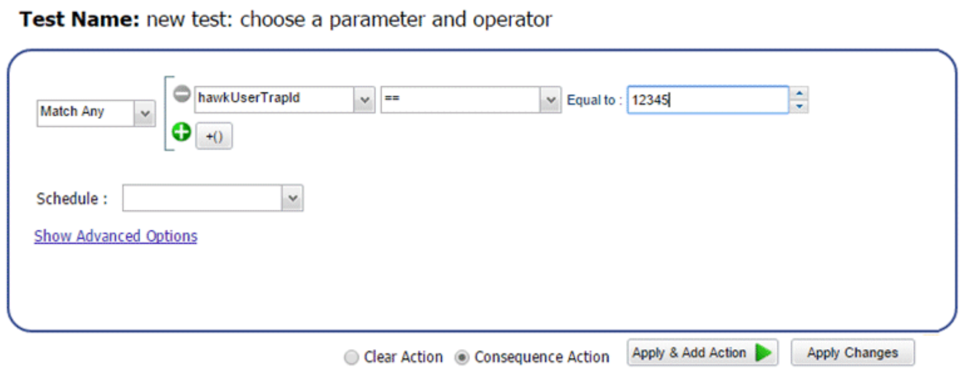
   The rule is now configured to use the **onHawkUserTrap()** method of the **SNMP** microagent.

## Applying Test Logic

Build a test expression by specifying a test parameter and test operator in the Test screen. The following test checks the **hawkUserTrapId** of the incoming **SNMP** trap on the machine where the adapter is running.

To create a new test in the current rule:

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




2. You can specify if you want to match all, match any, or match none.
3. Specify the test expression as `hawkUserTrapId` equal to (==) 12345.
4. Click **Apply Changes** to save the test expression.
5. Click **Apply and Add Action** to save the changes and proceed to defining the action process.

## Generating a TIBCO Hawk Alert

To generate an alert message on receiving the SNMP Trap:

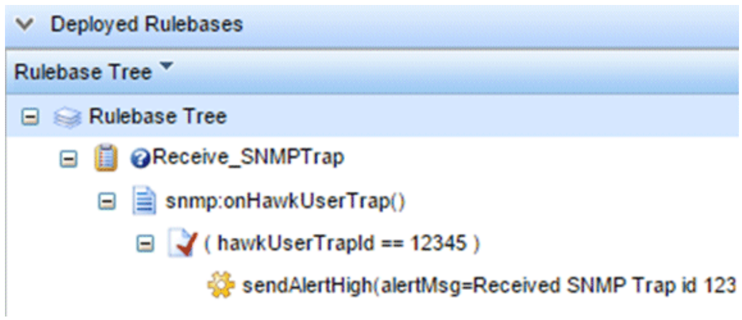
1. Go to the **Action** screen.

2. By default, the action type is **Alert** and the **Alert Level** is **High**.
3. In the **Alert Message** field, specify the following message:

   `Received SNMP Trap Id 12345`

4. Click **Save and Deploy** to save changes and deploy it to the agent.

   The **Receive_SNMPTrap** rulebase is included in the Rulebase tree.



# Sending SNMP Traps

By following the steps in this section, you send an SNMP trap when the TIBCO Hawk agent detects that the number of processes exceed 50. The generated SNMP trap will have a `hawkUserTrapId` of 12345.

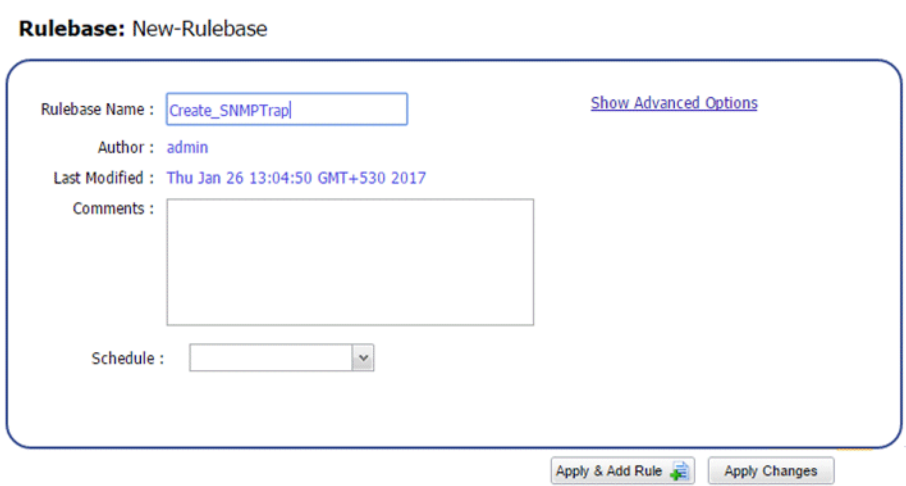
To follow the example, the adapter must be installed and running on a machine where a TIBCO Hawk agent is also running. The TIBCO Hawk agent service must be started, and TIBCO Hawk SNMP Trap Handler must also be running.

## Building a Rule for Data Capture

These steps demonstrate how to create TIBCO Hawk rulebase for monitoring and managing the adapter using TIBCO Hawk WebConsole. The rulebase contains a simple rule that uses the `Process:getInstanceCount()` method as a data source. A test on data returned by this data source checks if the number of processes equals or exceeds 50. If the TIBCO Hawk agent where the adapter is running detects the existence of this condition, the adapter generates a SNMP trap.

To create a new rulebase in TIBCO Hawk WebConsole:

1. Click an agent listed in the **Agents** portlet on the Hawk Dashboard.
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 **Create_SNMPTrap** in the **Rulebase Name** field.



4. Enter the other fields, as required.
5. Click **Apply Changes** to save the current changes.
6. Click **Apply & Add Rule** to save the changes and proceed to the rule building process.

### Create a new rule

1. On the Rule Editing screen, choose a microagent and then select the corresponding method.
2. Select the **Process** microagent. The Methods panel displays the methods for the microagent.
3. Select the **getInstanceCount** method.

   This method returns the count of the processes by process name. This is a synchronous method. When the rule is active, the agent subscribes to this method and receives data every 60 seconds, by default.
4. Specify a **TimeInterval** of 30 seconds.
5. Click **Apply Changes** to save the current changes.
6. Click **Apply and Add Test** to save the changes and proceed to defining the test process.

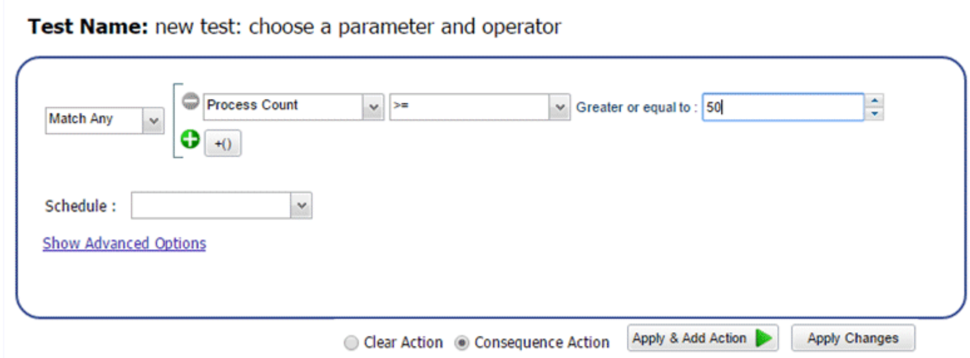
The rule is now configured to use the **getInstanceCount()** method of the **Process** microagent.

## Applying Test Logic

Build a test expression by specifying a test parameter and test operator in the Test screen. The following test checks the number of processes on the machine where the adapter is running.

To create a new test in the current rule:

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

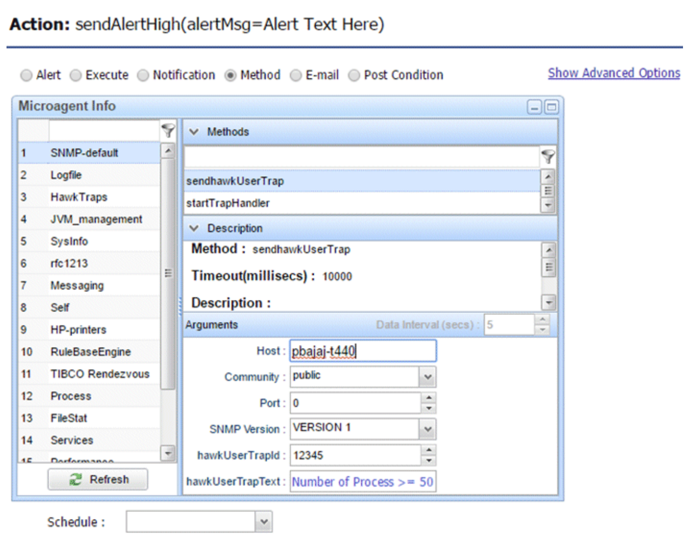



2. You can specify if you want to match all, match any, or match none.
3. Specify the test expression as Process Count greater or equal to (>=) 50.
4. Click **Apply Changes** to save the test expression.
5. Click **Apply and Add Action** to save the changes and proceed to defining the action process.

## Generating a SNMP Trap

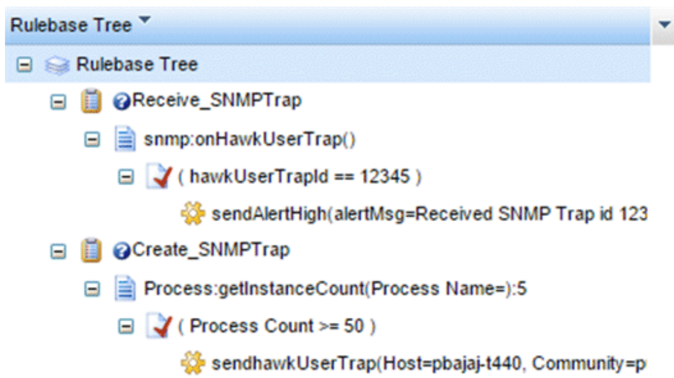
The following action executes the `SNMP-default:sendHawkUserTrap` method which send a SNMP Trap to the selected host.

1. Go to the **Action** screen.



2. Select **Method** as the action type.
3. Select the **SNMP-default** microagent and select the **sendHawkUserTrap** method.
4. In the Method Arguments, select the **Host** to which you want to send the trap, assign the trap a **hawkUserTrapId** and give a **hawkUserTrapText**.
5. Click **Save and Deploy** to save changes and deploy it to the agent.

   The **Create_SNMPTrap** rulebase is included in the Rulebase tree.




## Viewing the Results

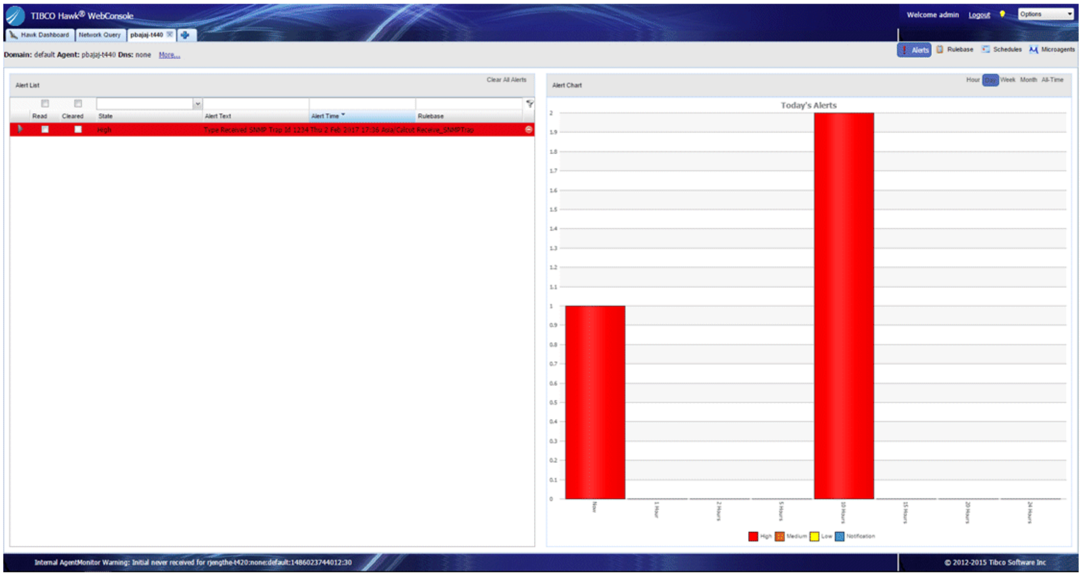
The section Sending SNMP Traps, page 83, describes steps to create a rulebase `Send_SNMPTrap` that generated a HawkUserTrap of id 12345 when the method `SNMP-dafault:sendhawkUserTrap()` returned a value that equaled or exceeded 50.

The section Handling SNMP Traps, page 79, describes the steps to create a rulebase `Receive_SNMPTrap` which demonstrated the SNMP trap receiving capability of the adapter. On receiving the SNMP Trap, the rulebase generates a TIBCO Hawk alert.

To view the TIBCO Hawk alert generated when the rulebase `Receive_SNMPTrap` receives the SNMP trap sent by the rulebase `Send_SNMPTrap`:

1. Click on the **Alerts** tab.

The Alert Display window shows the alert generated by the `Create_SNMPTrap` rulebase:



2. Expand the alert to display detailed properties.

# Appendix A Product Document Type Definition (DTD)

This appendix describes the Document Type Definition (DTD) that defines syntax rules for the adapter configuration file.

## Topics

# Viewing the hawk.snmp.dtd File

The `hawk.snmp.dtd` file contains syntax rules for the adapter configuration files. During installation, this file is copied to the *HAWK_SNMP_HOME*`/samples` directory.

```
<!--
******************************************************************
* Module:  tibhawksnmp.dtd
* Purpose: This is the Document Type Declaration (DTD) for the
* TIBCO Hawk SNMP Adapter XML format. With this file included,
* the XML configuration file can be syntax checked using any
* "validating" XML parser.
*
******************************************************************
-->

<!ENTITY % string "CDATA">
<!ENTITY % valueIndex "(true|false)">
<!ENTITY % valueOperation
"(GET|GETTABLE|SET|SETTABLE|ONTRAP|SENDTRAP)">
<!ENTITY % valueType
"(INTEGER|COUNTER|STRING|IPADDRESS|PHYSADDRESS|OBJID|TIMETICKS|GAU
GE|COUNTER64|UINTEGER32|BITSTRING|OPAQUE|NSAP|NULLOBJ)">

<!ELEMENT TIBHAWK_SNMP (microagent+)>
<!ATTLIST TIBHAWK_SNMP
    dtd_version       %string;         #IMPLIED >

<!ELEMENT microagent (method+)>
<!ATTLIST microagent
    name              %string;         #REQUIRED
    display_name      %string;         #IMPLIED
    help              %string;         #IMPLIED >

<!ELEMENT method ( (inputParameter+)?, (outputFilter+)?,
(outputParameter+)? )>
<!ATTLIST method
    name              %string;         #REQUIRED
    operation         %valueOperation; #REQUIRED
    timeout           %string;         #IMPLIED
    help              %string;         #IMPLIED >

<!ELEMENT inputParameter ( (valueChoice+)? | (legalChoice+)? )>
<!ATTLIST inputParameter
    name              %string;         #REQUIRED
    type              %valueType;      #REQUIRED
    oid               %string;         #IMPLIED
    index             %valueIndex;     #IMPLIED
    help              %string;         #IMPLIED >

<!ELEMENT outputFilter ( (valueChoice+)? | (legalChoice+)? )>
```

```
<!ATTLIST outputFilter
    name                %string;            #REQUIRED
    type                %valueType;         #REQUIRED
    oid                 %string;            #IMPLIED
    index               %valueIndex;        #IMPLIED
    help                %string;            #IMPLIED >

<!ELEMENT valueChoice (#PCDATA)>
<!ATTLIST valueChoice
    value               %string;            #REQUIRED >

<!ELEMENT legalChoice (#PCDATA)>
<!ATTLIST legalChoice
    value               %string;            #REQUIRED
    label               %string;            #IMPLIED >

<!ELEMENT outputParameter (#PCDATA)>
<!ATTLIST outputParameter
    name                %string;            #REQUIRED
    type                %valueType;         #REQUIRED
    oid                 %string;            #IMPLIED
    index               %valueIndex;        #IMPLIED
    help                %string;            #IMPLIED >
```

# Appendix B Object Type Representations

This appendix lists the MIB object types defined by the SNMP protocol.

## Topics

# MIB Object Types

SNMP agents for different types of devices provide access to objects that are specific to the type of device. In order to enable the SNMP manager or management application to operate intelligently on the data available on the device, the manager needs to know the names and types of objects on the managed device. The following table lists the primitive object types defined by the SNMP protocol.

*Table 12 MIB Object Types*

| MIB Object Type | Description |
|---|---|
| STRING | Contains arbitrary byte strings. If the byte string contains only ASCII characters, SNMP managers display this value as a text string. Otherwise, SNMP managers display this type as a sequence of hexadecimal values, prefixed by the "hex:" keyword, and separated by colons. |
| COUNTER | Represents a numeric value that can only increase. Counters roll over at their maximum value, and can never be less than zero. |
| GAUGE | Represents a numeric value that can increase or decrease. |
| INTEGER | Represents a numeric value that can have positive or negative values. |
| TIMETICKS | Represents elapsed time. This time always has a resolution of one hundredth of a second. |
| OBJID | Represents the identifier of another SNMP object. If the named object is compiled into the MIB, the name is usually displayed as the name of the SNMP object. |
| IPADDRESS | Represents the IP address of a network device. This type of object is often displayed in type as an IP address in conventional dot-notation. |
| OPAQUE | Represents an arbitrary encoding. This data type is used to pass arbitrary information strings that do not conform to the data typing standards. |

# Appendix C Hawk Service Wrapper for Microsoft Windows

This appendix describes the TIBCO Hawk Service Wrapper for the Microsoft Windows environment.

## Topics

# Overview

The TIBCO Hawk SNMP Adapter for the Microsoft Windows platform is packaged as a Microsoft Windows service. The `tibhawksnmp.exe` program found in the bin directory of the TIBCO Hawk installation implements this Microsoft Windows service. The `tibhawksnmp.exe` program is a wrapper program that wraps the adapter Java application allowing it to run as a Microsoft Windows service. The `tibhawksnmp.exe` program can be used to perform the following functions:

- Install the adapter as a Microsoft Windows service
- Uninstall the adapter as a Microsoft Windows service
- Execute the adapter as a Microsoft Windows console application
- Execute the adapter as a Microsoft Windows service.

# Installing the Adapter as a Microsoft Windows Service

The `tibhawksnmp.exe` program can be installed as an instance of the adapter Microsoft Windows service. As part of the install command processing, the `tibhawksnmp.exe` program registers itself as the Microsoft Windows service executable. For this reason, the install command should be executed from the directory where the executable is present.

The syntax of the `tibhawksnmp.exe` program install command is:
`tibhawksnmp.exe --install` *<options>*

The following table describes the command lines options used:

*Table 13 Command-line Options for* `tibhawksnmp.exe`

| Option | Description |
|---|---|
| `-name` *servicename* | The given internal service name. The default is `TIBHawkSnmp`. |
| `-displayname` *displayname* | The given external service name. The default is adapter. |
| `-manual` | The service is configured for a manual start. |
| `-auto` | The service is configured for a auto start.<br>**Note:** Either `manual` or `auto` should be specified. If neither is specified, the default is `manual`. |
| `-account` *accountname* | The user account which runs the service. The default is the System account. |
| `-password` *password* | The account password. |
| `-params` ( $p_1$ $p_2$ ... $p_N$) | The startup parameters for the service. Parentheses are required to distinguish startup parameters from other install command line options and they must be delimited by blanks. |
| `-jvm11opts` ( $o_1$ $o_2$ ...$o_N$) | The Java 1.1 Virtual Machine options. Parentheses are required to distinguish JVM parameters from other install command line options and they must be delimited by blanks. |
| `-jvmopts` ( $o_1$ $o_2$ ...$o_N$) | The Java 2 Virtual Machine options. Parentheses are required to distinguish JVM parameters from other install command line options and they must be delimited by blanks. |

The `-name` and `-displayname` allow multiple instances of the adapter Microsoft Windows service to be installed. Each instance must have a unique -name and should have a unique `-displayname`.

The `-params()` option allows for the specification of startup parameters for the adapter. These startup parameters are registered with the installation and become the default startup parameters for this instance of the adapter Microsoft Windows service. If startup parameters are specified when this service is actually started they will completely override these default values.

The `-jvm11opts()` option allows for the specification of Java 1.1 Virtual Machine (JVM) options to be passed to the JVM at execution time. This option supports all the Java 1.1 runtime options. These JVM options are platform specific. See the Java runtime executable (`jre.exe`) help for details on the options available on your platform.

The `-jvmopts()` option allows for the specification of Java 2 Virtual Machine (JVM) options to be passed to the JVM at execution time. This option supports all the Java 2 runtime options. These JVM options are platform specific. See the Java runtime executable (`java.exe`) help for details on the options available on your platform. This option is only required if you wish to specify Java 2 specific options.

The `tibhawksnmp.exe` program automatically detects the version of the JVM at execution time and uses the appropriate JVM options according to the following table:

*Table 14 JVM Command-line Options for* `tibhawksnmp.exe`

| JVM Options Specified | Running Under JVM 1.1 | Running Under JVM 1.2 or 1.3 |
|---|---|---|
| `None` | JVM defaults used | JVM defaults used |
| `-jvm11opts` only | `-jvm11opts` used | `-jvm11opts` used |
| `-jvmopts` only | JVM defaults used | `-jvmopts` used |
| `-jvm11opts` and `-jvmopts` | `-jvm11opts` used | `-jvmopts` used |

### Java Classpath

The `tibhawksnmp.exe` program automatically constructs the Java classpath for the adapter Microsoft Windows service. Assuming that the `tibhawksnmp.exe` program is located in directory *TIBCO_HOME*`\hawk\`*<version>*`\bin`, the constructed classpath will consist of the following in the specified order:

- *HAWK_SNMP_HOME*`\java`
- All `.jar` files located in the *HAWK_SNMP_HOME*`\java` directory.
- The Java Runtime default classpath.
- The CLASSPATH environment variable.

This classpath can be altered by specifying the `-cp` and `-classpath` options in the `-jvm11opts` and `-jvmopts`. The classpath specified in the `-cp` option is used as a prefix to the classpath described above. The classpath specified in the `-classpath` option replaces (overrides) the classpath described above.

There are additional classpath related properties that can be specified for the Java 2 platform. See the Java documentation for details. The `tibhawksnmp.exe` program passes the classpath described above, to the Java 2 virtual machine using the `-Djava.class.path` property. The user should not specify this property directly but use the `-cp` and `-classpath` options to alter the classpath.

## Uninstalling the Adapter as a Microsoft Windows Service

An instance of the adapter Microsoft Windows service can be removed using the `remove` command line option. The syntax of the command is:

`tibhawksnmp.exe --remove -name` <*servicename*>

where <*servicename*> is the internal service name used when the service was installed. If no <*servicename*> is specified, the default adapter service is removed.

The `--remove` command automatically stops the Microsoft Windows service if it is running before uninstalling it.

# Executing the Adapter as a Microsoft Windows Console Application

The adapter can be executed as a Microsoft Windows console application from the command line using the `tibhawksnmp.exe` program. Running the adapter as a console application is useful when diagnosing problems or when using it with a debugger. The command for executing the adapter is:

```
tibhawksnmp.exe --name <servicename> -xml_file <filename>
```

where <*servicename*> is the internal service name used when the service is installed, and <*filename*> is the complete pathname of the XML configuration file.

The command for specifying the `-help` option to display the command line syntax for the adapter is:

```
tibhawksnmp.exe -help/-h/-? <options>
```

The Microsoft Windows service for the adapter with the specified <*servicename*> must be installed before an adapter can be executed as a Microsoft Windows console application.

## Executing the Adapter as a Microsoft Windows Service

To start the service:

1. In Microsoft Windows Control Panel, open **Services**.
2. Click <*displayname*> **,** then click **Start.**

   The <*displayname*> is the external name given to this instance of the adapter when the service was installed.

The adapter service starts.

# Appendix D Perl5 Pattern Matching Syntax

This appendix describes the types of regular expressions you can use for the method arguments that allow Perl5 expressions.

## Topics

# Perl5 Pattern Matching Syntax

This section describes `Perl5` Pattern Matching syntax for pattern matching of test parameters. A regular expression has the form
`[m]/pattern/[i][m][s][x]`

The forward slashes surround the text pattern to match, and the letters in square brackets are optional modifiers. The use of whitespaces between the optional modifiers should be avoided.

> If an illegal Perl5 regular expression is used, an error dialog with the error message '`Invalid test expression`' is displayed.

The following table shows some of the basic elements of `Perl5` regular expressions.

*Table 15 Perl5 Pattern Match Regular Expression Elements*

| Element | Description | Example |
|---|---|---|
| . | matches any single character | `f.t` matches `fit` and `fat` |
| ^ | anchors the match to the start of a line or piece of text | `^i` matches `it` but not `ti` |
| $ | anchors the match to the end of a line or piece of text | `^$` matches only empty text |
| * | matches zero or more of the last character | `it*` matches `it` and `i` but not `ia` |
| + | matches one or more of the last character | `it+` matches `it` but not `ip` |
| ? | matches zero or one of the last character | `it?` matches `i` or `it` but not `itt` |
| \w | matches any alphanumeric character | `a\w` matches `at` but not `a+` |
| \W | matches any non- alphanumeric character | `a\W` matches `a+` but not `at` |
| [a-z] | matches a lower-case character within a range | `[a-d]` matches `c` but not `p` |
| [A-Z] | matches an upper-case character within a range | `[A-D]` matches `C` but not `c` or `P` |
| \d | matches any digit | `a\d` matches `a2` but not `at`<br>`/^\d\d/` matches any string that starts with 2 digits |
| \D | matches any non-digit | `a\D` matches `at` but not `a2` |

*Table 15 Perl5 Pattern Match Regular Expression Elements (Cont'd)*

| Element | Description | Example |
|---|---|---|
| [0-9] | matches any digit within a range | [1-4] matches 2 but not 5 |
| \n, \t | matches a new line character or tab character | one\two matches one <tab> two |
| \s | matches whitespace | (foo)\s matches 'foo ' but not 'foo'. |
| \S | matches non-whitespace | (foo)\s matches both 'foo ' and 'foo'. |
| \b | matches word boundaries | \b(frog) matches frog but not bullfrog. |
| \B | matches non-boundaries | \b(frog) matches both frog and bullfrog. |

The following table shows the available modifiers of a regular expression.

*Table 16 Regular Expression Modifiers*

| Modifier | Function |
|---|---|
| i | Do case-insensitive pattern matching. |
| m | Treat the string as multiple lines. |
| s | Treat the string as a single line. |
| x | Use extended regular expressions. |

The following table lists examples of Perl5 regular expressions to use when filtering alerts.

*Table 17 Examples*

| Description | Perl5 Regular Expression |
|---|---|
| Filter alerts generated only by a TIBCO Hawk rulebase named MediumHawkAlerts. | /^MediumHawkAlerts$/ |
| Filter alerts generated only by a TIBCO Hawk rulebase containing the characters 'Medium'. | /Medium/ |
| Filter alerts generated by TIBCO Hawk rulebases which begin with the characters 'Create' or 'New'. | /^Create\|^New/ |
| Filter alerts generated only by a TIBCO Hawk rulebase containing the characters 'hawk' or 'HAWK'. | /hawk/i |

# Index

## S



## T