TIBCO Hawk®

Concepts

Version 6.2.5 | November 2024

Copyright © 1996-2024. Cloud Software Group, Inc. All Rights Reserved.

Contents

Contents **2**

Distributed Application Monitoring **6**

Distributed Application Monitoring Challenges 6

Monitoring Requirements 7

TIBCO Hawk Monitoring System 7

Architecture 8

TIBCO Hawk Agent 9

TIBCO Hawk Microagent (HMA) 9

TIBCO Hawk Console 10

TIBCO Hawk Admin Agent 10

TIBCO Hawk Application Management Interface (AMI) 11

TIBCO Hawk Plug-in 11

TIBCO Hawk Adapter 11

TIBCO Hawk Event Service 11

TIBCO Hawk Message Transport 12

How it Works 13

TCP Transport for TIBCO Hawk 13

TCP Transport for TIBCO Hawk Architecture 13

Network Partition Strategies 16

Key Features 17

Scalability 18

Location Transparency and Fault Tolerance 18

Advanced Monitoring Logic 19

Flexibility 19

About TIBCO Hawk Agents **20**

Overview 20

Microagents 20
View a Microagent 22
Invoke a Microagent Method 22
Subscribe to a Microagent Method 22
Alert Messages 23
Suspend Alert Messages 23
Clear Alert Messages 23

TIBCO Hawk Console Features 25
Accessible by Using REST APIs 25
Multiple Domain Management by Using Proxy Domains 25
User Management 28
Security 28

Monitoring with Rulebases 29
Rulebases and Rules 29
Binding a Rule 30
Data Source 31
Tests 31
Compound Tests 31
Advanced Test Features 32
Actions 32
Advanced Action Features 32
Save a Rulebase 32

Advanced Rulebase Features 34
Rulebase Map 34
Agent Groups 35
Include a Rulebase 35
Add Commands to a Rulebase 36
Reference Variables in a Rulebase 37
Override a Rule 37

Posted Conditions 38
Testing for Non-Zero Values 38
Counting Results 39
Schedules and Period Groups 39
Automate Rulebase Management 40

Performing Group Operations 42
Group Operations 42
Interrogate a Microagents 43
Network Queries 44
Network Actions 44

Manage your Configuration 46
Configuration Modes 46

The TIBCO Hawk Event Service 47
Overview 47
AMI Instrumentation 47
Persistence of TIBCO Hawk Events using JDBC 48
Fault Tolerance 48
Interpreting Event Service Data Files 49
Executing Commands on Loss of Agent 51
Using the Event Service for Integration 52

Planning your Monitoring Strategy 53
Overview 53
Define Problem Areas 53
Define Information Requirements 54
Identify Corrective Actions 54
Map Information Requirements 54
Map Action Requirements 55
Refine Corrective Actions 55
Translate Requirements 55

Transform to Rule Elements 56
Plan Rulebases 57
Formulate Rules 57
Build and Test Rules 58
Refine Rules 58
Use Alternate Methods 58

TIBCO Documentation and Support Services 60

Legal and Third-Party Notices 62

Distributed Application Monitoring

TIBCO Hawk is a tool for monitoring and managing distributed applications and operating systems. TIBCO Hawk uses TIBCO Messaging software for communication and inherits many of its benefits. These benefits include a flexible architecture, enterprise-wide scalability, and location transparent product components that are simple to configure.

This chapter discusses the application monitoring challenges and how TIBCO Hawk meets these requirements. It provides an overview of TIBCO Hawk, the architecture and features of TIBCO Hawk components.

- Distributed Application Monitoring Challenges
- TIBCO Hawk Monitoring System
- Architecture
- How it Works
- TCP Transport for TIBCO Hawk
- Key Features

Distributed Application Monitoring Challenges

Computer systems today drive many aspects of business, controlling services and processes, transmitting information and identifying patterns and decisions. Increasingly the world is automated - yet who and what monitors the computers doing this work? How can we check their performance from the multitude of system parameters and program outputs?

System monitoring solutions require a number of capabilities to effectively aid the administrators of these systems. The operational alerts they create can also be used themselves as useful information in business applications that might track outputs from computer based measuring systems for example.

Monitoring Requirements

There are several requirements for operational systems monitoring. These include:

- Minimization of side-effects: Monitoring software should have negligible resource footprints, and should have a minimal effect on the performance of observed systems.
- Minimization of administration costs: Monitoring software should not be a maintenance burden, but a reliable beneficial tool, for the operations staff.
- Maximization of coverage: Monitoring software should monitor all the aspects of modern systems than are of interest to operations staff.

Most monitoring systems follow the *keep it simple* mantra for implementing monitoring. They might collect data via logs into a centralized server where scripts can execute against the monitor data – traditionally log file records – to determine required actions. However, TIBCO Hawk's designers anticipated some additional requirements:

- Ability to handle distributed systems and very large networks of computers: with the transition from mainframe to client-server to distributed computing throughout the 1980s to 1990s and 2000s, more fault-tolerant monitoring systems were envisaged that would not require administration of centralized servers and specialized fault-tolerance mechanisms
- Ability to grow to new monitoring requirements with minimal system intervention: both new type of measurement and new source of measurements would be required as new types of hardware and software were/are deployed over time.

TIBCO Hawk Monitoring System

TIBCO Hawk is an event-based monitoring system built around the concept of a distributed, autonomous smart agent that operates on each managed machine in the network.

The main objectives are:

- Deliver real-time visibility and control of distributed enterprise applications
- Utilize rules to assure system availability and performance

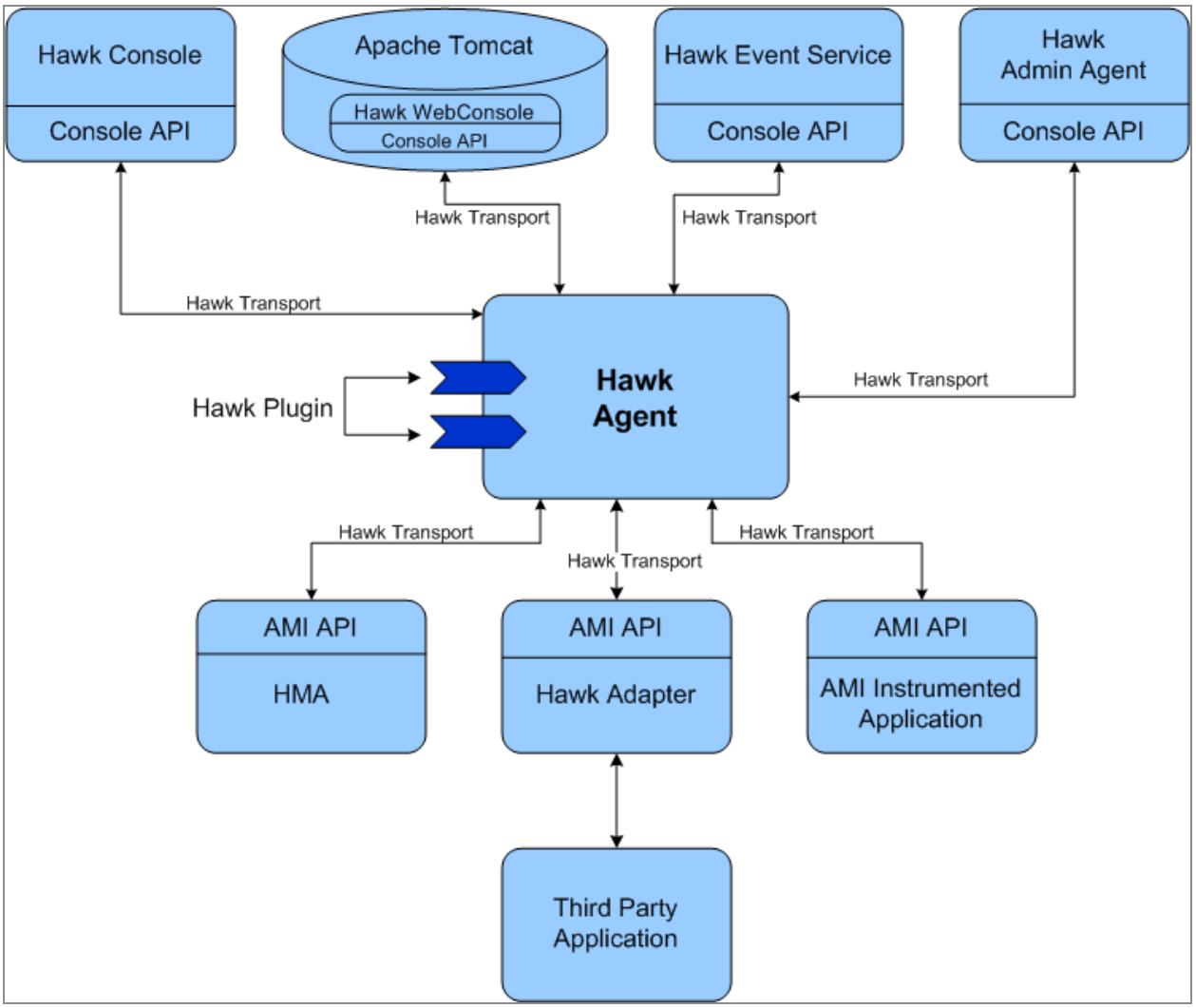
The software is designed specifically for monitoring distributed systems, so there is no centralized console or frequent polling across the network. With this structure, TIBCO Hawk

is able to scale to multi-thousand node global networks without the use of hierarchical managers and has the flexibility to allow individual managed entities to be added or modified without the need to re-configure or re-start any other parts of the system.

Architecture

The following diagram gives an overview of TIBCO Hawk system architecture.

Figure 1: TIBCO Hawk System Architecture



TIBCO Hawk consists of the following components:

- TIBCO Hawk Agent
- TIBCO Hawk MicroAgent (HMA)
- TIBCO Hawk Console
- TIBCO Hawk Admin Agent
- TIBCO Hawk Application Management Interface (AMI)
- TIBCO Hawk Console API
- TIBCO Hawk Plug-in
- TIBCO Hawk Adapter
- TIBCO Hawk Event Service
- TIBCO Hawk Message Transport

TIBCO Hawk Agent

A TIBCO Hawk Agent is an autonomous process that should be installed on each computer to monitor systems and applications on that computer. The Hawk Agents operate autonomously and use sets of rules, called “Rulebases”. These rules help Hawk agents to configure system management, status reporting, and automation tasks. For more information, refer to About TIBCO Hawk Agents.

TIBCO Hawk Microagent (HMA)

The term *microagent* is a generic term used to refer to the Hawk managed set of methods exposed by an application. This is done by instrumenting the application using Hawk AMI API or by using a Hawk plug-in or an adapter. See the following sections for a short description of each of them. The MicroAgent is covered in more detail later on in this book.

TIBCO Hawk Microagent or HMA is a standalone partner process to the Hawk Agent and provides various methods to the Hawk agent to instrument and monitor the host operating system and applications. The HMA is an application that uses the Hawk AMI application programming interface (API) for instrumentation. The AMI API details are provided in the section below.

Like Hawk Agent, Hawk Microagent should be installed on each computer you wish to monitor.

TIBCO Hawk Console

Hawk Console is a REST-based web application that provides a central view of all the distributed components interacting within the TIBCO Hawk system.

Hawk Console enables you to:

- Monitor the overall health of the system infrastructure across the network in the form of color-coded heat maps.
- Administer rulebases and rules, and specify appropriate actions, if the rules are triggered.
- Register and monitor multiple Hawk domains from hybrid deployments such as Amazon Web Services (AWS) deployment, on-premises deployment, and Kubernetes cluster deployment.

For details, see TIBCO Hawk Console Features.

Note As TIBCO Hawk WebConsole is deprecated in Hawk 6.1, TIBCO recommends that you use Hawk Console for monitoring and management.

TIBCO Hawk Admin Agent

The TIBCO Enterprise Administrator (TEA) is a central administration console that provides multiple products to be configured and administered. Hawk provides an agent for TEA (the TIBCO Hawk Admin Agent) which, when registered with the TEA server, allows you to monitor and do the Hawk administration from within the TEA web user interface. Using the Admin agent, you can get a consolidated view of all the distributed infrastructure components interacting within the TIBCO Hawk system. Admin Agent provides pictorial details of each of the components and allows basic monitoring and management via alerts and rules.

TIBCO Hawk Application Management Interface (AMI)

TIBCO Hawk Application Management Interface (AMI) is a set of API that allows developer community to extend and enhance instrumentation of various infrastructure components in the network by plugging into the Hawk system and making their applications manageable via the Hawk Agent. For more information, see the *TIBCO Hawk Programmer's Guide*.

TIBCO Hawk Plug-in

TIBCO Hawk plug-ins are Java components that reside and run inside the process space of a Hawk Agent. They are used to connect to a third party application using its specific protocols and expose them as microagents to Hawk, thereby enabling them to be managed by Hawk. TIBCO Hawk installation provides TIBCO Hawk EMS Plug-in and Hawk JVM Plug-in. For more information, refer to *TIBCO Hawk Plug-in Reference Guide*.

TIBCO Hawk Adapter

TIBCO Hawk Adapter is similar to the Hawk Plug-in in terms of interfacing a 3rd party application as a microagent, except that it runs outside the process space of the Hawk Agent. It runs as a standalone process that adapts the application specific management and monitoring protocols to the Hawk System by using the Hawk AMI API.

TIBCO Hawk Event Service

TIBCO Hawk Event Service is a Console API-based application that records activities of TIBCO Hawk Agents. The Event Service logs all TIBCO Hawk system events such as agent activation and expiration, alerts, clears, and Microagent and rulebase changes. It writes entries on these activities to data files which can be accessed by external applications. It can also execute a user-supplied script to notify system administrators of an expired agent. When communication with an agent is lost, the Event Service can invoke a user-provided script. Alerts and notifications can be recorded to log files or a database.

Typically, a single active instance of the TIBCO Hawk Event Service runs on the network. You may run additional instances as standby for the primary instance. For more information, refer to The TIBCO Hawk Event Service.

TIBCO Hawk Message Transport

TIBCO Hawk provides a choice of message and event communication mechanisms for inter process communications between various Hawk components.

You can choose one of the following as the primary message transport mechanisms to communicate between Hawk Agents and Console-API based console application such has Hawk Console, Hawk WebConsole (deprecated in Hawk 6.1), or Hawk Display (deprecated in Hawk 5.0).

- **TCP Transport for TIBCO Hawk** (Default message transport mechanism for Hawk and distributed as part of the standard Hawk installation)

  TCP Transport for TIBCO Hawk is a TCP based transport for Hawk components using the Akka clustering designs. For more information on TCP Transport for TIBCO Hawk, see TCP Transport for TIBCO Hawk.

- **TIBCO Rendezvous** (Need to install separately)

  TIBCO Rendezvous allows distributed applications to exchange data across a network. TIBCO Rendezvous provides software applications robust support for network data transport and network data representation. It is supported on many hardware and software platforms. Hence it is possible to communicate seamlessly among various application programs running on a variety of computers architecture and operating systems in a network. Please refer to TIBCO Rendezvous documentation for details.

- **TIBCO Enterprise Messaging Service** (Need to install separately)

  TIBCO Enterprise Messaging Service (EMS) is based on Java Message Service (JMS), the messaging specification of the J2EE (Java Platform Enterprise Edition) architecture. It provides a standardized interface for enabling communications between J2EE-compliant applications, Enterprise Java Beans, and various application servers.

  TIBCO EMS product is a high-performance implementation of JMS specifications and employs a store-and-forward architecture. It supports both queue-based and publish/subscribe messaging, local messaging transactions (in which multiple messages may be sent or consumed as an atomic unit of work), message selectors, and much more. Refer to TIBCO EMS Documentation for details.

How it Works

TIBCO Hawk *agents* perform the work required to monitor applications and systems. An agent can have a rulebase, rulebase map, or schedules object.

An agent runs on each node on the network and monitors local conditions. Each agent uses collections of locally loaded rules organized into rulebases to apply monitoring logic. A rulebase tells an agent how to monitor particular application or system resources and what actions to take when specific conditions are detected. TIBCO Hawk includes prebuilt rulebases that monitor basic system level parameters, and administrators can build additional rulebases using editors in TIBCO Hawk Console. Rulebases can be selectively loaded to an agent or group of agents on a temporary or permanent basis.

The TIBCO Hawk Console application shows alert messages generated by rulebases and presents them in an organized view. Alerts are color-coded to indicate the severity of a reported problem. Clicking on a alert message displays the error message along with a recent history of problems on the node. TIBCO Hawk Console does not store centralized monitoring intelligence; it simply offers a view of events on your distributed systems.

TCP Transport for TIBCO Hawk

TCP Transport for TIBCO Hawk is a TCP-based transport for Hawk components that use Akka clustering designs.

For more information on the Akka clustering, refer to the Akka documentation at http://akka.io/docs/.

Also, 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.

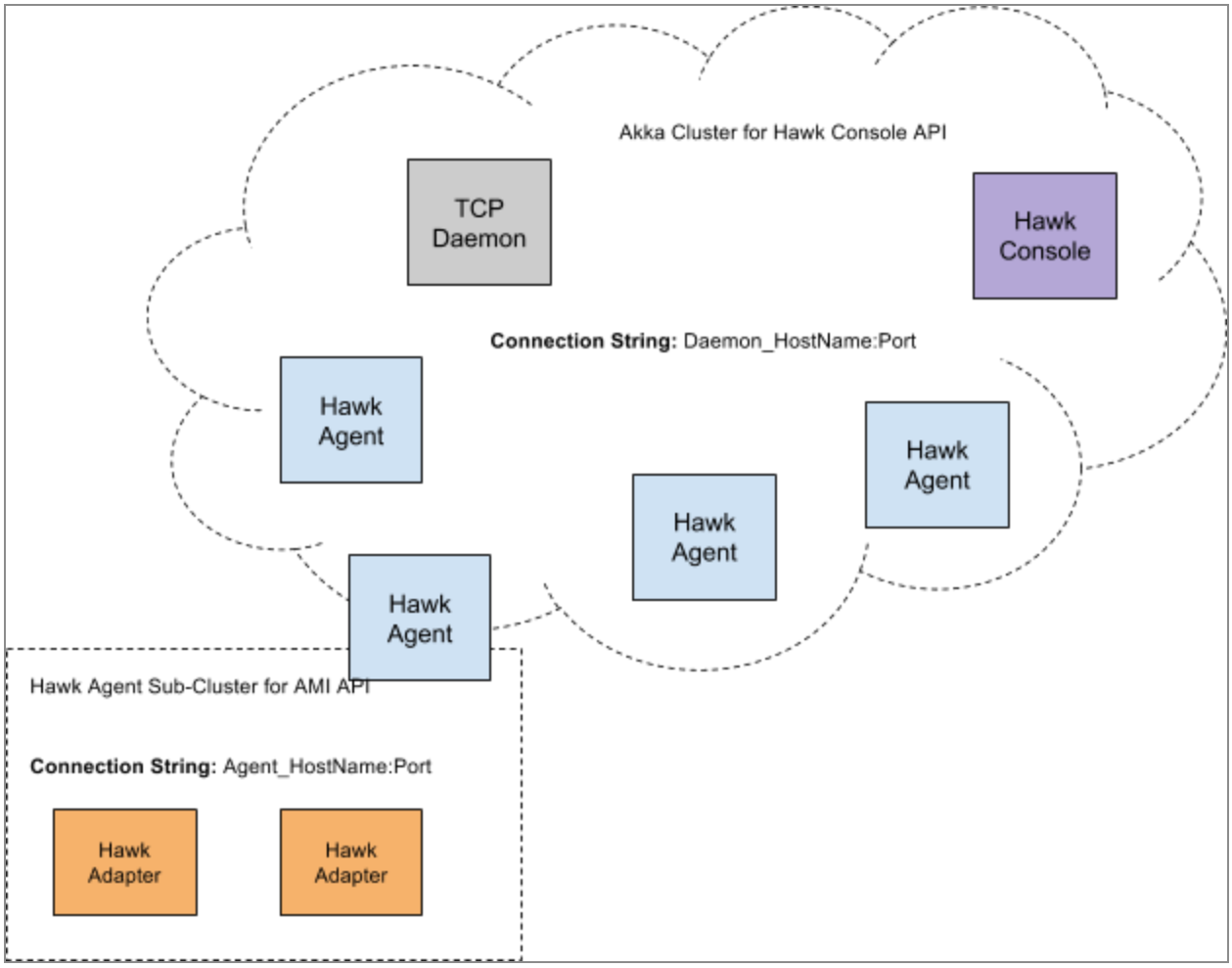
TCP Transport for TIBCO Hawk Architecture

The following are the key components of the TCP Transport for TIBCO Hawk:

- Cluster Manager
- Console and Hawk agent main cluster

- Hawk agent and AMI subcluster
- TCP Transport and TIBCO Rendezvous Bridge for Hawk Microagent (HMA)

Figure 2: The TCP Transport for TIBCO Hawk Architecture



Cluster Manager

The Cluster Manager is seed node for the TCP Transport Cluster. This is also the initial point of contact in the cluster. To use the TCP Transport for TIBCO Hawk, start the Cluster Manager process before starting any Hawk components. For using the fault tolerance, you can start multiple Cluster Managers.

Console and Hawk Agent Main Cluster

All the Hawk Agents, Hawk Consoles, and Hawk TCP daemons form one TCP Transport Cluster.

To form a cluster, all Hawk components and the daemon process are configured with daemon process as the seed node. The daemon process connects to itself and forms a cluster. All other Hawk components (Hawk agents and Hawk console) joins the cluster by connecting to the daemon process. It is best practice to have an odd number of daemon processes (such as 1, 3, 5, and so on). After the daemon process is started all other Hawk components can be started in any order. All agents and consoles in the cluster should have the same domain name.The configuration parameter for connecting to the daemon process is

```
tcp_session self_ip:port daemon_ip:port
```

where,

> *self_ip:port* is the socket address of the Hawk component (Hawk agents, Hawk Console, or Cluster Manager) that is joining the cluster.
>
> *daemon_ip:port* is the socket address of the Cluster Manager acting as a seed node for the TCP Transport Cluster. For the seed node daemon, *self_ip:port* and *daemon_ip:port* is same.

For fault tolerance, multiple seed nodes can be specified as comma separated list:

```
tcp_session self_ip:port daemon1_ip:port1,daemon2_ip:port2,...
```

Start the first seed node in the comma-separated list first and then other seed nodes. The cluster is not ready till the first node is started.

The Hawk Console subscribes to TCP Transport Cluster membership events. The membership ensures that the console gets notified whenever a new Hawk Agent is *up* or an existing one is *down*.

Hawk Agent and AMI Subcluster

The Hawk agent and its AMI applications form a separate cluster. For this cluster, the Hawk agent becomes the daemon process. The Hawk agent is the seed node through which all the AMI applications join the cluster. The configuration parameter for the Hawk agent's AMI component is:

```
ami_tcp_session self_ip:port
```

where,

self_ip:port is the socket address of the Hawk agent acting as the Cluster Manager for the cluster

For the AMI application, the configuration parameter for connection to the Hawk agent in the cluster is:

```
tcp_session=self_ip:port agent_ip:ami_tcp_session_port
```

where,

self_ip:port is the socket address for the AMI application

agent_ip:ami_tcp_session_port is socket address of the Hawk agent AMI component for the TCP Transport Cluster. This is the same socket address that was used in the *self_ip:port* attribute of the `ami_tcp_session` parameter in the Hawk agent of the TCP Transport Cluster.

TCP Transport and TIBCO Rendezvous Bridge for Hawk Microagent (HMA)

The TCP Transport and TIBCO Rendezvous Bridge is required for using Hawk Microagent (HMA) with TCP Transport for TIBCO Hawk. Since the TIBCO Rendezvous transport support Hawk C/C++ AMI API, a bridge is required to use HMA in the TCP Transport Cluster. In this bridge, the TIBCO Rendezvous transport is used for HMA and the TCP Transport for TIBCO Hawk for all other AMI applications. The AMI specific transport implementation of TCP Transport for TIBCO Hawk handles both the sessions.

Network Partition Strategies

In case of network failure or an unreachable node, the TCP Transport for TIBCO Hawk might create a network partition and marks the node down based on predefined partition strategies. In network partitioning, the transport keeps the best partition alive and shuts down the other partition. The TCP Transport for TIBCO Hawk handle the network partitions based on the following strategies:

- Quorum based strategy
- Majority based strategy

Note Despite of any strategy, do not add a new Cluster Manager to an already running cluster.

Quorum-Based Strategy

In the Quorum-based strategy of network partitioning, the network partition has to maintain a minimum number of TCP daemons in the TCP Transport Cluster to be operational. This minimum number of daemons is termed as *quorum size*. In case of network failure, the cluster with the required quorum size remains operational. The TCP Transport Cluster shuts down the other partition and marks any unreachable node in the other partition as down. For example, there are five Cluster Manager in the cluster with quorum size set to three. If the network partition occurs with one partition having three Cluster Managers and other having two then the partition with two Cluster Managers shuts down.

quorum size = (number of daemons / 2) + 1

Majority-Based Strategy

In the majority based strategy, if the network partition occurs then the partition which has the majority of nodes survives and the other partition shuts down. For example, if there are five nodes in the TCP Transport Cluster and network partition occurs with three and two nodes, then the partition having two nodes shuts itself down. The advantage of the majority based strategy over the Quorum-based strategy is that cluster can gradually downsize without outage. In case, network splits with equal partitions than one which has the oldest node survives.

Key Features

As the complexity and size of a distributed network increases, most tools that manage these networks also increase in size and complexity. Many of these tools use a centralized approach for gathering information and performing tasks, making heavy use of polling and point-to-point messaging. The result can be excessive network bandwidth, bottlenecks, and compromised reliability and efficiency of the network. TIBCO Hawk addresses these problems by providing a monitoring solution that is scalable, location transparent, reliable, and flexible.

Scalability

A scalable network management tool seeks to minimize network traffic so the least amount of bandwidth is used. Also, to be truly scalable, the bandwidth consumed should not grow proportionately as a distributed network grows.

TIBCO Hawk agents monitor conditions on their local machines and send alerts over the network only when problems are detected, an approach that has major scalability advantages over a central, dedicated console server. With centralized server-based architectures, monitoring data is collected from remote nodes via network polling or point to point messages to the console generated by simple agents on the devices. After gathering or receiving this information, the console server then makes centralized monitoring decisions. The network bandwidth consumed by polling-based systems grows proportionately with the number of monitored nodes, as does resource utilization on the console machine. Eventually, additional console servers must be employed to scale the system. Using the distributed event-driven monitoring architecture employed by the TIBCO Hawk system, network bandwidth and system resources are conserved by decentralizing and distributing the monitoring load. Another advantage of this method is that alerts are generated only when a problem exists, so under normal conditions minimal network bandwidth is used.

Location Transparency and Fault Tolerance

A TIBCO Hawk agent runs on each node, where it collects information, applies monitoring logic, and carries out event notification and corrective actions. TIBCO Hawk agents are autonomous because they monitor and perform management tasks independently of TIBCO Hawk Console. Fault-tolerance is built in because agents perform tasks independently of any other agents or user interface. If one or more agents on the network ceases to function, other agents continue to monitor local data and perform tasks.

Note

On the contrary, in the console server model, if the server becomes disabled, all monitoring stops until another console server can be brought on line. Additionally, as all connections are point-to-point, all remote agents must be manually reconfigured to connect to the new server and all polled connections need to be rediscovered.

Advanced Monitoring Logic

TIBCO Hawk provides flexibility in performing monitoring tasks. The rules or policies you create can be very simple or complex. Using advanced features, you can implement solutions like the following:

- Automatically restart a failed process.
- Create a series of escalating actions to respond to a continuing or deteriorating problem.
- Clear an alert when certain criteria are met.
- Selectively ignore an intermittent condition and respond only after it persists.
- Take an action when an alert clears, such as sending a stand-down notice to a backup systems administrator who is on call.
- Switch rules automatically to support different monitoring setups.
- Create thermostat-like controls to respond to fluctuating conditions.
- Automatically alter monitoring behavior based on user defined schedules.

Flexibility

A distributed network typically must support multiple platforms and enterprise applications. In such a heterogeneous environment, many different combinations of objects require monitoring and many variations of routine tasks require automation. The TIBCO Hawk environment is flexible enough to support these variations, both internally and through components designed to communicate with external application environments.

Administrators can tailor the TIBCO Hawk environment by building custom rules and by automatically executing commands and custom scripts. Programmers can extend TIBCO Hawk by instrumenting applications using the AMI protocol, or by using AMI to build gateways between agents and non-instrumented applications. The Console API also provides a comprehensive interface for communicating with TIBCO Hawk agents, allowing programmers to build customized display applications or sophisticated, multi-level decision-making programs that can draw data across multiple agents. The Configuration Object API allows you to build custom rulebases, and can also build custom rulebase editor.

About TIBCO Hawk Agents

This chapter discusses how a TIBCO Hawk agent uses microagents to gather and receive information. It also describes the alert messages generated by an agent.

- Overview
- Microagents
- View a Microagent
- Invoke a Microagent Method
- Subscribe to a Microagent Method
- Alert Messages

Overview

To monitor a system or application on the network using TIBCO Hawk, you install and run a TIBCO Hawk agent on the host machine. The agent is a process that monitors activity on a particular machine by querying microagents.

Microagents

The agent interfaces the managed objects on its local machine using microagents. Microagents communicates with managed objects such as operating system subsystems, agent components, log files, event logs or applications. Each microagent exposes a set of methods to the agent that the agent uses to collect information and take action.

A set of default microagents is included when you install TIBCO Hawk software. The complete list varies by operating system, but the following standard microagents are installed on all platforms:

- The `Self` microagent gathers version information for the local agent, the security policy in effect, and available microagents. It also activates or deactivates the collection of diagnostics for technical support.

- The `SysInfo` microagent gathers basic information on the operating system, hardware architecture, computer name and IP address of the agent machine.
- The `Process` microagent counts instances of a process on the agent machine and gathers process usage statistics.
- The `FileStat` microagent gathers information on a disk file or files on the agent machine.
- The `Rendezvous` microagent gathers information on TIBCO Rendezvous daemon activity on an agent machine and on TIBCO Rendezvous licensing. This micro agent is available only when Hawk is configured to use Rendezvous as a message transport mechanism between Agent and HMA.
- The `RuleBaseEngine` microagent gathers information about the rulebases and takes actions on rulebases that affect the TIBCO Hawk agent.
- The `TcpClusterStatus` microagent provides methods to monitor the health of the TCP transport cluster and TCP daemons.
- The `TcpMessaging` microagent provides methods to send and receive messages using the TCP Transport for TIBCO Hawk.
- The `Logfile` microagent responds asynchronously each time a new line is added to a log file you specify. You can create tests that search for specific strings in a log file line.
- The `Custom` microagent executes any command-line executable that can run on the current agent machine. The microagent can also return text or numeric results from the executable.

  This microagent is typically used to give a TIBCO Hawk agent access to information it cannot obtain from other microagents. It is a useful tool for extending agent capabilities.

For detailed information about these microagents with their methods, arguments and results, as well as descriptions of platform-specific microagents, see the *TIBCO Hawk Microagent Reference*.

Microagent methods can be accessed interactively from TIBCO Hawk Console or through rulebases downloaded to the agent. Monitoring with Rulebases, talks about how to use a method as the data source of a rule.

You can also access methods programmatically using the Console API. For more information, see the *TIBCO Hawk Programmer's Guide*.

View a Microagent

When you start up an instance of TIBCO Hawk Console, it automatically discovers machines running TIBCO Hawk agents on your network depending on the Hawk message transport mechanism you have configured.

Each agent has a set of default microagents that are loaded when the agent is started. If you install and start an adapter or gateway, or instrument an application with AMI, microagents for these objects are dynamically added to the agent. In TIBCO Hawk Console, you can view microagents and their methods for any discovered TIBCO Hawk agents. For detailed information about how to get the list of microagent for an agent, see the *TIBCO Hawk Console User's Guide*.

Invoke a Microagent Method

You can invoke a microagent method in TIBCO Hawk Console and immediately view the results. Invoking is useful when you want to test a method before using it in a rule, or to check a return value for troubleshooting purposes.

Microagent methods perform a wide variety of tasks, and are grouped into categories according to the impact that invoking the method has on a managed system. These categories are used by the TIBCO Hawk agent to display appropriate methods for the current context:

- `IMPACT_INFO` methods collect data.
- `IMPACT_ACTION` methods make some change to the microagent or the resource it represents.
- `IMPACT_ACTION_INFO` methods both collect data and make changes.

Microagent methods can be synchronous or asynchronous.

Subscribe to a Microagent Method

To view microagent method results over time, you can subscribe to the method in TIBCO Hawk Console and view results as they are returned. Subscribing provides continuous updates of method results and displays a moving window of the result history. Creating a subscription is useful when you want to test a range of return values before specifying boundaries in a rule, or to identify general patterns of activity.

Alert Messages

Alerts are messages an agent sends to TIBCO Hawk Console when a specified condition occurs. Alerts originate from rulebases that enforce your monitoring logic. In TIBCO Hawk Console, the colors of each agent and container icon summarize alert levels, and the Alert Display window shows alert details for a particular agent or all agents.

Suspend Alert Messages

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

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

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

Clear Alert Messages

Alerts are cleared when the alert condition, as defined by the rulebase, ceases to exist. Alerts generated by tests on synchronous data sources, which deliver data at fixed intervals, are cleared by default at the first test repetition when the condition no longer holds. Alerts generated by tests on asynchronous data sources, which deliver data when it becomes available, are cleared by default when the test does not evaluate to true for 15 minutes. You can modify this default behavior using advanced test and action properties in the rule. For more information, see the *TIBCO Hawk Console User's Guide*.

When you create an action that creates an alert, you may find it useful to include text that describes how the alert is cleared. For example, if an alert is raised when free disk space falls below 10% and is cleared only when disk space is above 15%, the alert text could be: "Free disk space is 11.6%; this alert will be cleared when the free disk space exceeds 15%. "

Active alerts are kept until cleared, and cleared alerts are purged by TIBCO Hawk Console when a buffer limit is reached. To view purged alerts, you can examine Event Service data files (if configured), where all alerts are written as they are raised and cleared. For more information on the TIBCO Hawk Event Service, see the *TIBCO Hawk Installation, Configuration, and Administration Guide*.

TIBCO Hawk Console Features

This chapter describes the features of TIBCO Hawk Console.

- Accessible by Using REST APIs
- Multiple Domain Management by Using Proxy Domains
- User Management
- Security

Accessible by Using REST APIs

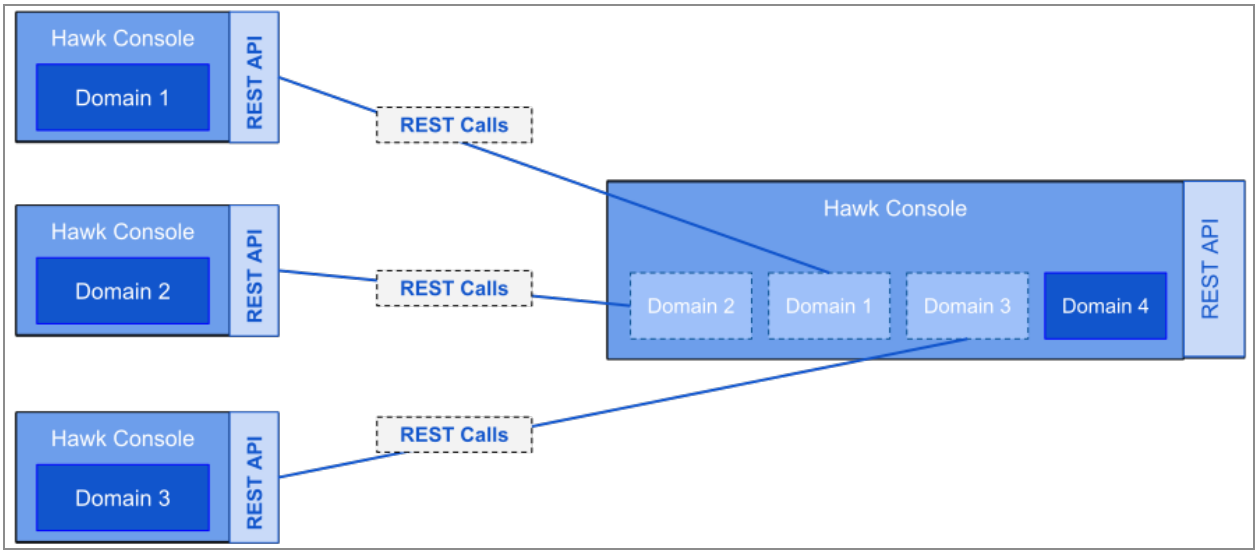
In addition to the web UI, you can use any REST client to access the Hawk Console features through the exposed REST APIs. After you start the Hawk Console, you can view these REST APIs by accessing the Swagger page URL `http://`*<Console_host_IP>:<Host_port>*`/HawkConsole/v1/docs`.

The Swagger page enables you to view the details of each API, such as model schema, required parameters, and response messages. Also, you can try out the APIs by providing the parameters to the API in the Swagger URL.

Multiple Domain Management by Using Proxy Domains

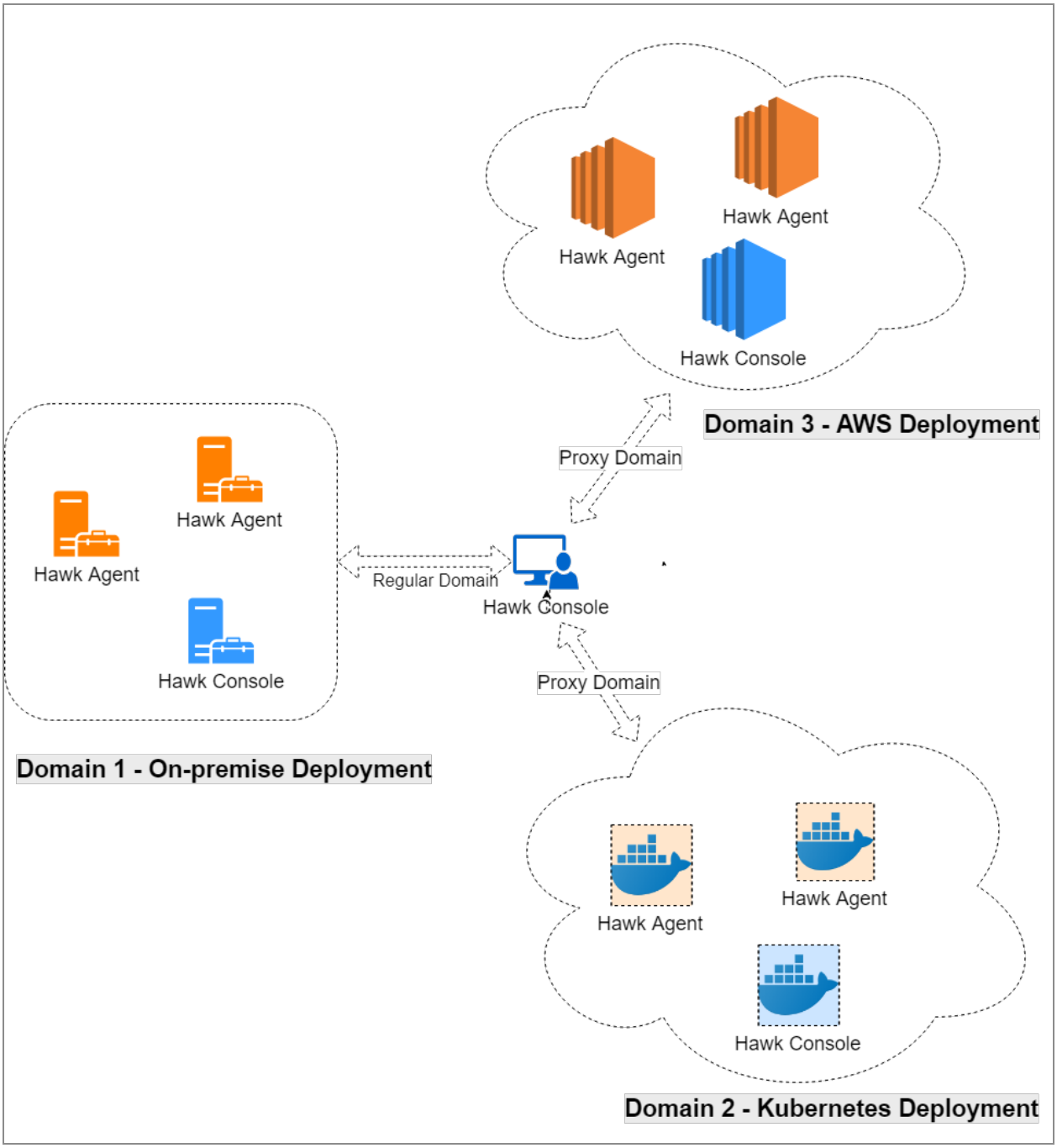
The Hawk Console can register domains from remote Hawk Consoles. Such domains are called *Proxy Domains*. The calls on the proxy domain are routed to the remote consoles using the REST API. See Hawk Console Multiple Domain Management.

Figure 3: Hawk Console Multiple Domain Management



The main advantage of the proxy domains is that it enables you to monitor Hawk domains across all deployments using a single interface. For example, if you have one Amazon Web Services (AWS) deployment, one on-premises deployment, and one Kubernetes cluster deployment, then all these deployments can be monitored using a single Hawk Console, see Sample Proxy Domain Connection. Each individual Hawk Console can still work independently. Also, when you use the proxy domains, the remote Hawk Console domains need not be in the same network.

Figure 4: Sample Proxy Domain Connection



You can register a proxy domain to your Hawk Console from the web interface by using the URL of that proxy domain. For details on how to register a domain to Hawk Console using web UI, see *TIBCO Hawk Console User's Guide*.

User Management

Hawk Console supports database and file-based user authentications. You can set the authentication mode using the Hawk Console configuration file (`hawkconsole.cfg`) located at *<CONFIG_HOME>*`/bin`.

For more details about Hawk Console configurations, see *TIBCO Hawk Installation, Configuration, and Administration Guide*.

Security

You can access Hawk Console over a secured channel by using SSL or TLS security protocols.

To enable the secured communication, uncomment and configure the following fields in the Hawk Console configuration file (`hawkconsole.cfg`):

- `-key_alias`
- `-key_password`
- `-key_store`
- `-key_store_password`
- `-protocol`
- `-ciphers`

For more details about Hawk Console configurations, see *TIBCO Hawk Installation, Configuration, and Administration Guide*.

Monitoring with Rulebases

This chapter introduces the concept of a *rulebase*, a configuration object that allows the TIBCO Hawk agent to manage systems and applications on the network.

- Rulebases and Rules
- Binding a Rule
- Tests
- Actions
- Save a Rulebase

Rulebases and Rules

TIBCO Hawk Agent can use several rules that contain monitoring criteria. Multiple rules can be consolidated in a collection called rulebase. In other words, a rulebase is a collection of one or more rules, whereas a rule is a user defined monitoring policy. It specifies a data source in the form of a microagent method, one or more tests that check for conditions, and one or more actions to perform if the test condition evaluates to true. Rules can monitor parameters of an operating system, application or other managed object and perform tasks.

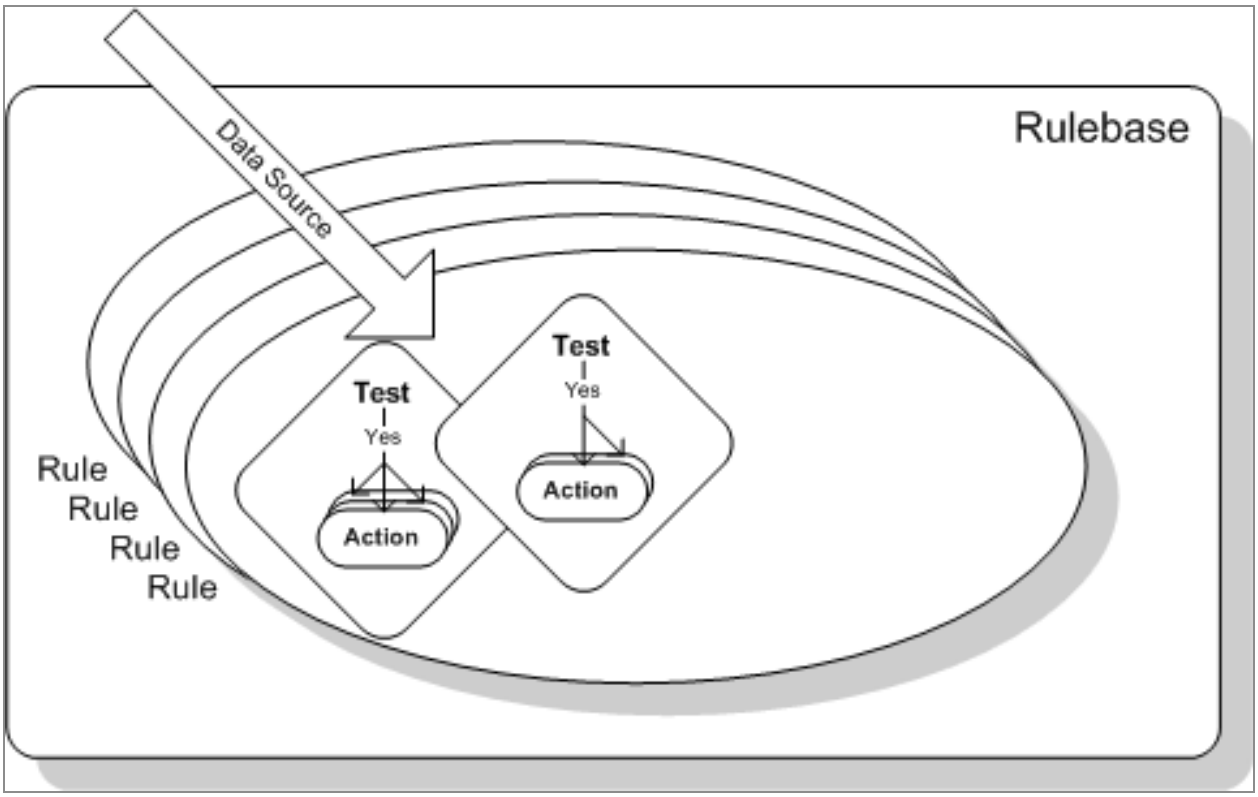
TIBCO Hawk installation provides several such default rulebases with specialized rules. You can create additional rulebases with specialized rules. Using these rulebases, you can perform general monitoring tasks and test for very specific conditions. Escalations can be applied to actions to allow for a staged response to a condition. An agent can be configured to initially take corrective actions itself, but if the condition still persists, then call for outside help from administrators. A rulebase can be deployed on a single Hawk Agent, on a group of Hawk Agents.

TIBCO Hawk Console provides an easy-to-use WYSIWYG rulebase editor that allows construction of simple and complex rules and rulebases. No scripting is required and an administrator does not need to learn special rule syntax. To simplify rule administration, rules in a rulebase should be related. Multiple rules in the same rulebase can monitor a particular application or system function. For example, an application rulebase could include one rule for raising a medium-level alert if disk space or CPU usage exceeds certain

thresholds. Another rule could initiate a high-level alert and send a text or email message to the system administrator if the application process terminates.

Rulebases and associated objects have a hierarchical structure. Every rulebase contains rules which are made up of data sources, tests, and actions. The following diagram illustrates the relationships between objects in a rulebase.

Figure 5: Relationship Between Objects in a Rulebase



Binding a Rule

When you create a rule, you specify the monitoring logic and package it for a TIBCO Hawk agent. The agent can apply the rule again and again without intervention. If a problem occurs, the agent can solve it by taking corrective action, or notify you that the problem requires attention, or both, depending on how the rule is designed and implemented.

Rules consist of data sources, tests, and actions. Data sources are microagent methods that periodically collect or asynchronously return information to an agent. One or more tests are applied to the resulting data set. When a particular test evaluates to true, one or more actions can be triggered.

Figure 6: Flow of Rules and Tests

Although a rule uses a single data source as input, you can use a posted condition to indirectly use multiple data sources in a test. For more information, see Posted Conditions.

Data Source

The data source for a rule is its source of input data, and is always a method of a microagent. When a rule is active, the TIBCO Hawk agent subscribes to the specified method and passes method results to the test.

Tests

The data source of a rule provides information about some condition on a managed node. After information is received, one or more tests are applied to evaluate it. Each sample of data from the data source is distributed to all tests in the rule. Each test uses the data to compute a true or false value which is used in determining when to trigger actions.

Note	A test is true only when the entire test evaluates to true, not just the test expression.

Evaluation intervals for a test depend on the type of data source used by the rule. Synchronous data sources return results to the agent at regularly scheduled intervals, where an asynchronous data source transmits data whenever it becomes available.

Test transitions, or changes between values of successive tests, can be one of four types: false to true, true to true, true to false, and false to false. By default, tests on synchronous data sources become true whenever the test expression is true, then become false the first time the test is false. Tests on asynchronous data sources, however, become false by default only after 15 minutes have passed since the last true evaluation.

Compound Tests

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

expressions and insert operators in the compound test editor.

Advanced Test Features

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

Actions

Each test has one or more related actions. An action is the consequence of evaluation of a rule, such as an alert message or a custom script. Whenever the rule receives information from its data source, tests are evaluated. If a test evaluates to true, the related actions are triggered. Once triggered, actions are performed unless advanced options delay or prevent the action.

Advanced Action Features

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

Save a Rulebase

Once the creation of the rulebase is complete, there are several options for you to consider. The option you choose depend on the agent configuration mode.

- **Deploy a rulebase on the agent** - The agent will immediately start performing the monitoring tasks defined in the rulebase. Agents running in Automatic Configuration mode will also store the rulebase in their auto-configuration directory for automatic loading upon startup. Agents running in Manual Configuration mode will not store the rulebase anywhere, and will only perform the monitoring tasks defined in the rulebase for the life of the currently running process.
- **Send the rulebase to a set of agents** - The set of agents will immediately start performing the monitoring tasks defined in the rulebase. Agents running in

Automatic Configuration mode will also store the rulebase in their auto-configuration directory for automatic loading upon startup. Agents running in Manual Configuration mode will not store the rulebase anywhere, and will only perform the monitoring tasks defined in the rulebase for the life of the currently running process.

Advanced Rulebase Features

This chapter describes some advanced TIBCO Hawk rulebase features.

- Rulebase Map
- Agent Groups
- Include a Rulebase
- Add Commands to a Rulebase
- Reference Variables in a Rulebase
- Override a Rule
- Posted Conditions
- Schedules and Period Groups
- Automate Rulebase Management

Rulebase Map

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

To efficiently manage agent configuration, an entire enterprise should use the same rulebase map.

Note

- Rulebase maps are supported only in Hawk Display (deprecated in Hawk 5.0) and Hawk WebConsole (deprecated in Hawk 6.1).
- Rulebase maps are supported only when running in Manual Configuration mode. For more information, see Configuration Modes.

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

Agent Groups

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

Every rulebase map can have two types of groups: user-defined and automatic. Automatic groups, defined for you in TIBCO Hawk, consist of operating system groups and the ALL group. The ALL group includes every agent on your network. The set of operating system groups includes one group for each operating system represented in the ALL group. For example, a network where TIBCO Hawk agents run on Solaris and Microsoft Windows has Solaris and Microsoft Windows operating system groups, and an ALL group.

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

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

Include a Rulebase

A rulebase can specify other rulebases to automatically load when it is loaded by an agent. This feature is useful for maintaining modular rulebase sets. For example, if rulebase A should always be loaded with rulebase B, then rulebase B can have A as a member of its include list.

You can use include lists to load rulebases when an agent starts or after it is running. However, after making updates, you should always manually distribute included rulebases using the `Send To` feature. An agent loads an included rulebase only if it is not already loaded. For example, an agent loads rulebase B and included rulebase A when it starts. If

you modify both rulebases, then distribute the updated rulebase B to other agents, the new version of rulebase A is not reloaded because this rulebase already exists on the agent. For more information, see *TIBCO Hawk WebConsole User's Guide*.

Similarly, included rulebases remain loaded on the agent when the parent rulebase is removed. To remove included rulebases, either delete them manually using TIBCO Hawk WebConsole or invoke the `RuleBaseEngine:unloadRuleBase()` method. For more information, see the *TIBCO Hawk Microagent Reference*.

Note

Include lists are supported only when running in Manual Configuration mode. For more information, see Configuration Modes.

Add Commands to a Rulebase

You can add commands or scripts to a rulebase, then execute the command from the **Get Commands** menu for an agent from Hawk WebConsole. When you select **Get Commands** for an agent, a list of all commands from all rulebases currently loaded on the agent displays. The command is executed using the `Custom:execute()` microagent method on that agent.

This feature allows you to organize scripts and utilities for individual nodes, and to execute them from your TIBCO Hawk environment. You can execute commands by selecting a menu option in TIBCO Hawk WebConsole, so starting new windows and manually typing command syntax are not required. Since each node with a TIBCO Hawk agent can have a customized set of scripts or other executables supporting the applications running on that node, adding commands to a rulebase can make these utilities accessible to users without requiring them to manually log on. This is especially useful for nodes that exist outside a firewall.

Any command can be added to a rulebase, but rulebase commands are typically related to the resources or activity the rulebase monitors. For example, if a rulebase is monitoring the `httpd` daemon, the commands could execute scripts such as: `start_httpd` or `kill_httpd`.

To add commands to a rulebase, first you specify the command syntax in a rulebase and save it on one or more agents. Then you can execute the command on an agent using TIBCO Hawk WebConsole.

Reference Variables in a Rulebase

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

Supported Types of Variables

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

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

Referencing these variables outside of a rulebase is not supported. For more information, see *TIBCO Hawk Console User's Guide*.

Override a Rule

When designing rules in a rulebase, you look for similar requirements among groups of agents. Rulebases can then be distributed to these groups to perform common tasks. If some agents in a group have slightly different rulebase requirements than others, you can use overriding to compensate for small discrepancies. Minimal configuration changes are required.

Overriding is a way to set precedence among similar rules. For example, a group of machines on a network all run the TIBCO Rendezvous daemon, `rvd`. You build and distribute a generic rulebase with a rule to check that the `rvd` process does not consume more than 2 MB of memory and monitors the various Host Status advisory messages for the sessions. However, two machines in this group are servers with more memory running large TIBCO Rendezvous applications. They can handle heavier loads than the other machines. For the server machines, instead of using a totally separate rulebase, you can preserve the common functionality of the Host Status advisory monitoring in the original rulebase, but add a second rulebase containing only one rule with a higher memory threshold that also has a higher precedence than the same rule in the original rulebase. The first rulebase is loaded on all agents, but you load the second only on the two servers.

Since the rule on the two servers has a higher precedence, the higher threshold rule is used and the lower threshold rule in the original rulebase is disabled.

Note	Rule names are automatically assigned and guaranteed unique in a rulebase. For rules to overlap, they must be defined in different rulebases loaded on the same agent.

Posted Conditions

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

Testing for Non-Zero Values

Posted conditions have integer values assigned internally by a TIBCO Hawk agent. When a rulebase is loaded, values are set to 0 until the action is performed. Performing the action creates the posted condition and gives it a non-zero value, signifying that the condition does exist and has not yet been cleared. Although the value is internal, a test in another rule can check the posted condition for existence or non-zero values. Posted conditions that can be used in a test are listed along with other parameters in the Test Builder dialog.

For example, a rule could check CPU utilization every 60 seconds. If the CPU is greater than 80% utilized when the test is evaluated, a Post Condition action named `SystemTooBusy` is performed and its integer value becomes non-zero. Another rule retrieving the process table every 60 seconds with a compound test to identify the processes consuming more that 50% of CPU when total usage exceeds 80% can be written as:

```
((%CPU>50) AND (SystemTooBusy>0))
```

If this test evaluates to true, an alert message that includes the PID for the process is generated. Results from multiple data sources are required to perform this monitoring task.

Counting Results

You can also use posted conditions to count the number of results returned by the microagent method. Some microagent methods return tabular data when invoked. If the method returns a table of results, an instance of the associated test is generated for every row of the table. The actual value of a posted condition is the number of associated test instances that evaluated to true. You can use this relationship to count the number of rows in the table that match some condition. A non-zero value means that at least one row in the table matched the test condition. The value of the posted condition is the exact number of rows that matched the test condition.

For example, a software virus spawns multiple copies of a common process such as a web browser. The virus uses the `SYSTEM` account to accomplish this, which is not the normal account for a web browser. To check for the presence of this virus on your network you could create several tests and use a posted condition. The first rule could retrieve the process table, and use a compound test to check for more than one copy of the browser process running as `SYSTEM`.

```
If ((name=="netscape") AND (user=="SYSTEM"))
```

When the test evaluates to true, a posted condition named `SystemBrowser` is created. The second rule tests the value of the posted condition. It does not require data from a microagent, but uses a token data source: the microagent method `Self:getUptime():10`. This method returns the number of seconds the agent has been running every 10 seconds. The actual data retrieved from the data source is discarded, but every 10 seconds this method triggers a test for `SystemBrowser>1`. When the test evaluates to true, a high-level alert is generated.

Schedules and Period Groups

A schedule is a configuration object that defines when a rulebase, rule, test or action is active. You define a schedule using TIBCO Hawk Console and save the schedule information in a file. Then you can send the schedule to one or more TIBCO Hawk agents, and apply the schedule to rulebase objects.

Schedules

In a schedule, you specify the times during which a particular monitoring activity is in effect by defining inclusion periods and exclusion periods. An inclusion period consists of specific times when an object can be active.

For example, you might have a rulebase that should be used only during business hours, so you create a schedule with an inclusion period of Mondays through Fridays, 9 AM to 5 PM. The schedule may also define exclusion periods, or times when the object must not be active, such as between 12:00 PM and 12:59 PM for lunch. A schedule is active (or "in-schedule") during inclusion periods only if no exclusion period applies. In this example, the schedule is in-schedule at 11:30 AM on Tuesday mornings. It is out-of-schedule an hour later at 12:30 PM and back in-schedule at 1 PM. The schedule is out-of-schedule at all times on Saturdays and Sundays.

Using schedules is described in the next section.

Period Groups

A period group defines a set of related dates or times. For example, a period group named Holiday may include all of the holidays within a year. Period groups are stored together with the schedules of an agent or repository in a schedule file.

Note	Period Groups are supported only in Hawk Display (deprecated in Hawk 5.0) and Hawk WebConsole (deprecated in Hawk 6.1).

Using period groups is described in *TIBCO Hawk WebConsole User's Guide*.

Rulebase Objects

Rulebase objects are related through a hierarchy. A schedule applied to one node in the hierarchy also affects all nested objects. For more information, see Rulebases and Rules.

Automate Rulebase Management

Using the following methods you can automate the volume of monitoring activity at any time:

- `RuleBaseEngine:loadRuleBase()`
- `RuleBaseEngine:loadRuleBaseFromFile()`
- `RuleBaseEngine:unloadRuleBase()`

These methods allow you to create management rulebases that operate above other rulebases. For detailed method descriptions, see the *TIBCO Hawk Microagent Reference* guide.

For example, an important application that requires monitoring runs only on demand. One rulebase monitors the application, then a management rulebase controls when the first rulebase is loaded and unloaded. The management rulebase contains one rule that uses the instance count for the application process as its data source. The test checks for at least one instance of the application process. If the process exists, a Method action loads the rulebase that monitors the application. When no instances of the application exist, this test evaluates to false and a clear action unloads the monitoring rulebase.

You could also create a rulebase (for example, `General`) and build routine rules to check on disk space, CPU utilization, collision rates on network interface cards, and so on. When a serious condition occurs, one of the rules in the general rulebase loads a rulebase (for example, `Extreme_Network_Analysis`). This rulebase tests more frequently, explains in more detail, and applies more severe consequences to the serious condition. This rulebase could be developed by a network expert and may start up detection tools to perform detailed analysis.

Running this extreme analysis rulebase even when no problem exists would generate an unnecessary amount of network traffic. However, during a crisis it provides the logic and information needed to resolve the problem. When the dangerous condition is over, the general health rulebase can unload the extreme analysis rulebase and return the monitoring level to normal.

Performing Group Operations

This chapter describes how to interrogate and perform tasks on multiple remote TIBCO Hawk agents using TIBCO Hawk WebConsole.

- Group Operations
- Interrogate a Microagents
- Network Queries
- Network Actions

Group Operations

Group operations allow you to detect and solve problems on the network from any location. Using Hawk Console, you can explicitly solicit information from multiple TIBCO HAWK agents, then take action based on the results. Group operations include:

- Network query — querying one or more agents on the network to determine where particular conditions exist.
- Network query with a test — applying a test to query results in the same manner as in a rulebase.
- Network action — performing an action on one or more agents on your network.

Note Network Queries and Network Actions are supported only in Hawk Display (deprecated in Hawk 5.0) and Hawk WebConsole (deprecated in Hawk 6.1).

Network query and action allow you to combine identical tasks you might perform on individual agents and implement them dynamically in a single operation. The query or action is performed by invoking microagent methods, using multiple instances of the same microagent distributed across the network.

Querying several agents can be a useful tool for determining thresholds when building tests in rules. You might first decide to test a potential threshold value through a query

before committing the rule to a rulebase. Similarly, using a network action you can experiment with the effects of a particular action. Network query and action are useful both as interactive tools, and for researching and testing how rules should be built.

The following table summarizes the different methods you can use to retrieve information and carry out tasks using TIBCO Hawk WebConsole.

Method to Retrieve Information and Perform Tasks

Network	Retrieving Information		Performing Tasks	
	Now	Over Time	Now	Over Time
On One Computer	Invoke a microagent method	Subscribe to a microagent method	Invoke a microagent method	Build a rulebase that takes an action at repeated intervals
On Multiple Computers	Perform a network query	Subscribe to microagent methods on multiple agents, or build a rulebase that sends alert messages, distribute it to other agents, and view all of the alerts at once	Perform a network action	Build a rulebase that takes an action at repeated intervals and distribute it to other agents

Interrogate a Microagents

When you interrogate microagents, you invoke a method interactively on any agent on your network and view the result locally in TIBCO Hawk Console. The microagent method can return information, or take an action on an agent or managed application, or both.

Network Queries

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

Note Network Queries are supported only in Hawk Display (deprecated in Hawk 5.0) and Hawk WebConsole (deprecated in Hawk 6.1).

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

- Tests are optional
- Only one test per query is allowed
- A test has no associated actions

For network queries that include a test, compound test syntax can extend its capabilities. For example, after querying a list of agents for a certain condition, you could apply a second condition to refine the list. Using compound test operators, you can specify complex logical criteria.

Network Actions

A network action is similar to a network query, except instead of gathering information you specify an action to perform on one or more remote agents. The action can be any task a microagent method of type `IMPACT_ACTION` or `IMPACT_ACTION_INFO` can perform. Examples of tasks you might perform as a network action include loading rulebases, starting and stopping processes, or executing custom scripts.

Note

- Network Actions are supported only in Hawk Display (deprecated in Hawk 5.0) and Hawk WebConsole (deprecated in Hawk 6.1).
- Since performing a single network action can have a widespread effect, use this feature with caution.

Manage your Configuration

This chapter describes TIBCO Hawk configuration modes.

- Configuration Modes

Configuration Modes

On your network, configuration objects such as schedules and rulebases are retrieved using either manual or automatic configuration. The mode you choose might depend on the number of TIBCO Hawk agents running on your network, and the number and complexity of configuration objects.

- With manual configuration, you manually configure which rulebases an agent loads by editing the rulebase map or adding them to the `-rulebases` configuration parameter. For more information, see Rulebase Map.

  At startup, the agent searches one or more directories or Repositories to find the specified configuration object. All changes are temporary, until you decide to make them permanent by saving them to a file or a Repository.

- With automatic configuration, all changes applied to the agent are permanent. In this mode, you automatically specify rulebases for the agent to load at startup by saving and deleting rulebases from the auto-configuration directory.

You specify a configuration mode and other parameters when starting a TIBCO Hawk agent, and the agent searches the configuration source for configuration objects. A configuration source is one or more directories on the agent machine, or one or more Repository names on the network.

This section describes how configuration objects are stored and retrieved. For implementation details on Manual or Automatic Configuration modes, see *TIBCO Hawk Installation and Configuration*. For method descriptions, see the *TIBCO Hawk Microagent Reference*.

The TIBCO Hawk Event Service

This chapter describes the role of the TIBCO Hawk Event Service, a separate process that collects information about TIBCO Hawk agents.

- Overview
- Interpreting Event Service Data Files
- Executing Commands on Loss of Agent
- Using the Event Service for Integration

Overview

The TIBCO Hawk Event Service is a separate process that collects information about TIBCO Hawk agents. Because it runs independently from other TIBCO Hawk processes, the Event Service can detect and report the event if an agent process should fail.

The main tasks of the Event Service are:

- Record events reported by agents in text files or relational databases using JDBC
- Detect and respond to agent termination
- Asynchronously notify using AMI

For instructions on installing and starting the TIBCO Hawk Event Service, see *TIBCO Hawk Installation, Configuration, and Administration Guide*.

AMI Instrumentation

The Event Service is implemented as an AMI microagent, which allows users to be asynchronously notified on instances of agent activation, expiration, alerts generated and alerts cleared by Hawk Agents on a particular Hawk domain.

The microagent `COM.TIBCO.hawk.microagent.HawkEventService` exposes the following asynchronous methods:

- `_onUnsolicitedMsg`
- `onAgentAlive`
- `onAgentExpired`
- `onAlert`
- `onClear`
- `onMicroAgentChange`
- `onRulebaseChange`

For more information on these methods, see the TIBCO Hawk Microagent Reference.

Persistence of TIBCO Hawk Events using JDBC

All alerts generated and cleared by TIBCO Hawk Agents across the network, as well as agent activation and expiration events, are written to a relational database using JDBC. Data is stored in two separate tables, created automatically at startup (if they are not already present in the specified database):

- **HawkAgentInfo**. The events `onAgentAlive`, `onAgentExpired`, `onMicroAgentAdded`, `onMicroAgentRemoved`, `onRulebaseAdded` and `onRulebaseRemoved` add rows to this table.
- **HawkAlertClearInfo**. Events `onAlert` and `onClear` add rows to this table.

Fault Tolerance

You will normally run the TIBCO Hawk Event Service on a single system in a TIBCO Hawk managed network. However, multiple instances of the TIBCO Hawk Event Service can run on separate machines for fault tolerance.

For TIBCO Rendezvous transport, the Fault tolerance is implemented using TIBCO Rendezvous Fault Tolerance (TRFT). Each instance of a TIBCO Hawk Event Service process joins a fault tolerant group named `HawkEventService:`*hawkdomain*, where *hawkdomain* is the value of the `-hawk_domain` command line option.

To rank the members of a group, fault tolerance software sorts the members by weight. Weight is assigned using the `-ft` command line option. The member with the highest weight receives rank 1 (so it outranks all other members). When an instance fails, the next-

highest instance is activated and the member with the next highest weight receives rank 2; and so on. When two or more members have the same weight, fault tolerance software ranks them in a way that is opaque to programs.

Refer to TIBCO Rendezvous documentation for details regarding fault tolerance while using the TIBCO Rendezvous transport.

Interpreting Event Service Data Files

When running, the Event Service records:

- All alerts generated and cleared by TIBCO Hawk agents across the network, as well as changes in agent alert level (represented as icon colors in TIBCO Hawk Console)
- All instances of agent activation and expiration
- Add and remove operations for microagents and rulebases

It records these events in data files named `Event.dat`, located in the directory defined in the `-datadir` agent startup option. Data files contain event monitoring information, while separate log files named `Event.log` record the state of the Event Service itself.

The TIBCO Hawk Event Service creates rolling data files using the same mechanism as TIBCO Hawk log files. It also uses the same default values for file location, number and size.

Sample Alert Message

The following entry shows a sample alert message logged in an Event Service data file:

```
ALERT_RECEIVED : alert={ agent={ host-name=cricket, dns=none, host-
ip=123.123.123.123, network-ip=123.123.123.0 }, alert-id=58,
rulebase=Mail, alert-state=75, alert-text=%processor time  >= 25, time-
received=Thu Sep 24 09:03:56 EDT 1998 } ## Thu Sep 24 09:03:56 EDT 1998
##
```

The `dns` field has a value of `none` unless an agent domain is specified. The `alert-id` field is a numeric identifier for the alert that is unique across `host-name` and `dns` combinations. If a single condition generates multiple alerts, the alert messages have the same `alert-id` value.

Agent State Change Entries

Event Service data files contain the following types of entries for agent state changes:

```
AGENT_ALIVE : agent={ host-name=ultrahawk1, dns=none, host-
ip=160.101.246.16, network-ip=160.101.246.0 }, alert-state=75, ## Sat
Dec 18 11:40:36 EST 1999 ##
```

```
ALERT_CLEARED : agent={ host-name=protege, dns=none, host-
ip=123.123.123.123, network-ip=123.123.123.0 } alert-id=35, reason=test
evaluated to FALSE, ## Thu Sep 24 09:04:03 EDT 1998 ##
```

```
AGENT_STATE_CHANGE : agent={ host-name=cricket, dns=none, host-
ip=123.123.123.123, network-ip=123.123.123.0 }, new-alert-state=0, ##
Thu Sep 24 09:04:27 EDT 1998 ##
```

```
AGENT_REINITIALIZED : agent={ host-name=jaguar, dns=none, host-
ip=123.123.123.123, network-ip=123.123.123.0 }, alert-state=75, ## Thu
Sep 24 09:06:09 EDT 1998 ##
```

In agent entries, alert levels are represented numerically. To find the text equivalent, use the following table to translate the value of the `alert-state` field:

Numeric Representations of Alert Levels in Event.dat Files

Numeric Value	Alert Level
0	Notification
25	Low
50	Medium
75	High

Microagent and Rulebase Event Entries

Event Service data files contain the following types of entries for microagent and rulebase events:

```
MICROAGENT_ADDED : AgentID={ host-name=boxter, dns=none, host-
ip=160.101.246.18, network-ip=160.101.246.0 },
MicroAgentID=COM.TIBCO.hawk.hma.Network, at Sat Dec 18 11:50:38 EST 1999
```

```
MICROAGENT_REMOVE : AgentID={ host-name=boxter, dns=none, host-
ip=160.101.246.18, network-ip=160.101.246.0 },
MicroAgentID=COM.TIBCO.hawk.hma.FileSystem, at Sat Dec 18 11:50:36 EST
1999
```

```
RULEBASE_ADDED : AgentID={ host-name=pchawk1, dns=none, host-
ip=160.101.246.11, network-ip=160.101.246.0 }, rulebase=cricket,
state=75, at Sat Dec 18 11:50:36 EST 1999
```

```
RULEBASE_REMOVE : AgentID={ host-name=boxter, dns=none, host-
ip=160.101.246.18, network-ip=160.101.246.0 }, rulebase=System, at Sat
Dec 18 11:50:36 EST 1999
```

Executing Commands on Loss of Agent

If TIBCO Hawk agent process is terminated by user or abnormally due to loss of network communication, HAWK Event Service can be configured to execute a script or set of programmatic instructions. Such script could also send an e-mail or text message providing instant notification.

This configuration is facilitated in Hawk Event Service Configuration file (hawkevent.cfg) via "`-script`" parameter.

By default, this is not enabled. To execute a command script on loss of agent, you need to provide the fully-qualified name of an executable file in hawkevent.cfg. and enable it by uncommenting (removing "#" from the beginning of line) that parameter.

For example, `-script` <f*ully-qualified name of an executable file*>

An example of script file providing the notification message is as follows.

Hawk Event Service can provide the following arguments in this order:

1. Agent name (usually the name of the machine where the agent is installed)
2. IP address of the machine where the agent runs

For example, for an agent running on Microsoft Windows, a script named `sendpage.bat` might contain the following lines:

```
@echo off
```

```
rem Usage - sendpage.bat <agentname> <agent_ipaddress>
```

```
send_page.exe %1 %2
```

```
exit
```

You could also call a script that logs into the machine and checks for the existence of an agent process before sending notification.

Using the Event Service for Integration

In addition to storing a record of events, the TIBCO Hawk Event Service is a useful tool for exchanging information between TIBCO Hawk agents and other monitoring products. If an external application can read and parse log files, you can notify it of TIBCO Hawk events. Configure the application to read the Event Service data files and parse log entries, using pattern matching techniques to locate headings for particular event types.

This method of exchanging information is easier to manage and generates less network traffic than using SNMP traps or executing scripts on the remote machine. More significantly, if a TIBCO Hawk agent process terminates, the Event Service can notify the external application. Any mechanism that relies on the TIBCO Hawk agent to exchange data with external applications cannot provide this type of notification.

Planning your Monitoring Strategy

This chapter describes how to analyze situations on your network that could be monitored using TIBCO Hawk software. It also explains how to translate monitoring requirements into elements of a TIBCO Hawk rulebase.

- Overview
- Define Problem Areas
- Translate Requirements
- Formulate Rules

Overview

This chapter helps you identify the ways TIBCO Hawk software can help monitor and manage your distributed systems and applications. To create a monitoring strategy, you need to identify the potential sources of problems and analyze how they can be prevented. Most monitoring tasks can be automated, but you must identify the unique situations that occur on your network. If you are new to monitoring, this chapter will help you define a strategy. If you already have a monitoring strategy, reading this chapter will help you refine it.

Define Problem Areas

The first step in any monitoring strategy is to consider issues with resource contention or other problems that have occurred in the past. Make a list of questions you would like to answer about your network systems and applications, problems you would like to solve, and situations you would like to avoid. For example:

- Be notified when an internal application metric exposed through AMI is in a problem state.
- Be notified when critical processes fail or consume too many system resources, such as memory or CPU cycles.

- Be notified when disk space is low, or when it is decreasing at some critical rate.
- Gain more control over processes that slow down the system.
- Avoid multiple servers going down on the same day.
- Be notified when important messages appear in log files or event logs.

When the list is complete, assign a priority to each item. Decide which improvements provide the greatest benefit to your enterprise, then rank items in order of importance.

Define Information Requirements

For each item you choose to focus on, define the information that is required to answer the question, solve the problem, or avoid the situation. For example, to gain more control over processes that slow the system, you might track the number of processes are running on each server, or measure system response time.

Identify Corrective Actions

Identify actions that could fix or avoid the problems you defined. If the problem is slow servers, for example, a corrective action might be “Terminate redundant or low-priority processes.”

Map Information Requirements

The next step is to map information requirements to microagent methods. Evaluate TIBCO Hawk microagent methods and their results, both platform-independent and platform-specific. Consider methods that return data: `IMPACT_INFO` and `IMPACT_ACTION_INFO`. Often analogous methods for each operating system return similar information. If possible, match each information need to a microagent method result field.

A key requirement is that you specify one information source per item. You might also need to redefine questions or conditions to fit available options. For example, to solve the problem of the system running slowly, choose the most appropriate metric: CPU utilization, file reads per minute, or the number of simultaneously running processes. Usually one source provides the best answer. If multiple operating systems exist on your network, a separate mapping might be required for each operating system.

If a piece of information is not provided by default TIBCO Hawk microagents, look for alternate ways of providing the information to a TIBCO Hawk agent— through an AMI gateway to an application, or by retrieving data from custom scripts. If a third-party application is business-critical, consider instrumenting it using the TIBCO Hawk Application Management Interface (AMI).

Map Action Requirements

After information requirements are fulfilled by microagent methods, you map action requirements to microagent methods. Consider TIBCO Hawk microagent methods of type `IMPACT_ACTION` or `IMPACT_ACTION_INFO` to learn about available actions. If possible, map each required action to a microagent method.

If an action cannot be performed by TIBCO Hawk action methods, look for alternate ways to perform it. You can call custom executables or reach directly into an application using AMI (or by writing an AMI gateway to an application).

At this point, you can also define practical meanings for low, medium and high alert levels. For example, high-level alerts might be reserved for actions that require intervention by the system administrator, or for maintenance tasks that take more than 30 minutes to complete.

Refine Corrective Actions

Define timing and circumstances for corrective actions. Define when a question needs to be answered or a problem solved. How often and under what circumstances does a solution need to occur? What are the potential causes of the problem? How quickly does the information change? Does it build over hours, or happen all at once? How often does it happen? What are the symptoms? Is there a fault-tolerant backup for a critical application?

Define when a problem is considered resolved. Define a resolution for each problem: what signals that balance is restored? What is adequate? What is normal?

Translate Requirements

After monitoring requirements are defined, you can translate them into TIBCO Hawk terms. For each item, create a mapping table with a structure similar to the following table. To

find TIBCO Hawk equivalents, refer to the *TIBCO Hawk Methods Reference*, and consult task and action tables in Monitoring with Rulebases.

Mapping Table

Column	Sample Contents
Information to gather	The microagent and method to invoke, the script or command to execute, a manual procedure, or the AMI method to call. Include any arguments the method requires and any return values. Include all means of solving the problem.
How often to gather the information	Every sixty seconds, or whenever a log file entry is made, or whenever a message is received from a managed application through AMI.
Conditions that define the question, problem, or situation	When the disk space falls below ten percent, when the server load is twice normal, when the AMI message contains the word "error". If multiple circumstances apply, include them all.
Actions for TIBCO Hawk to take	Raising an alert, calling a microagent method, executing a script that pages the system administrator, triggering an email action, modifying an application through its management interface. Include all actions you want to take.
When resolution occurs	When the value returns inside a reasonable range, when the error message no longer exists (for how long?), when the instrumented application returns an acceptable value.

Transform to Rule Elements

Transform the table entries into TIBCO Hawk rule elements using the following equivalents:

- *Information* — a microagent, method and arguments in a rule's data source
- *How often* — a data delivery interval in a rule's data source

- *Conditions* — each test in a rule
- *Actions* — each test's actions
- *Resolution* — how a test is cleared (a clear test or other option)

Use the default rulebases supplied with the TIBCO Hawk software as examples. You can also copy these rulebases to use as the basis for new rules.

A data source includes a microagent method, its arguments, and a data delivery interval (for synchronous methods). The data source is used for all tests and actions in the rule. Since the first field in your table corresponds to a microagent method and the second field corresponds to a data delivery interval, all items in the list with the same entries in the first and second columns can be grouped together in the same rule.

Some items in the list might become tests in a common rule. For example, items that depend on the same "under what circumstances" information could be separate actions in the same test.

Plan Rulebases

Analyze your monitoring rules, tests and actions, then group them into sets based on function or location. For example, group together rules that:

- Monitor a particular application
- Perform a particular system function, such as process management
- Monitor separate operating systems into one set for each operating system

Following these guidelines is particularly important for taking advantage of Rulebase View. For more information, refer TIBCO Hawk Programmer's Guide.

Rules that remain after others are grouped into sets might belong to a base set for all agents. For identification purposes, rulebase names should reflect the functions they perform.

Formulate Rules

With requirements defined and translated into TIBCO Hawk equivalents, you can encapsulate monitoring logic in TIBCO Hawk rulebases.

Build and Test Rules

In TIBCO Hawk Console, create a new rulebase and then build each rule with a data source, tests and actions. If overlap exists between your requirements and default TIBCO Hawk rulebases, try copying and modifying an existing rulebase.

Consider simulating the situation in a test, such as starting several copies of a process and observing a TIBCO Hawk agent terminate all but one process. Use this technique to verify that the rule performs as planned.

Another way to test rules is to use network query and action and invoke methods that imitate the action of the rule.

Refine Rules

After rulebases are built and performing roughly as designed, you can fine-tune them as follows:

- Increase rule efficiency by considering ways to faster and better respond to monitored conditions. Advanced testing and action options might be more efficient. Consider using different metrics, or changing thresholds.
- Maximize the information content of messages by using variable substitution to add information from a data source to your alerts and notifications.
- Consider other monitoring options such as building your own scripts, or instrumenting an application with an AMI interface.
- Increase the automation of your monitoring effort as you see possibilities to automate tasks you now do yourself. For example, using escalation to automate a scaled response to a continuing problem can eliminate the need for intervention. Adding a true test counter to a test can reduce the number of false alarms.

Use Alternate Methods

You might already have developed scripts or executable programs that monitor critical functions. You can incorporate these programs into your TIBCO Hawk environment using the `execute`, `executeForNumber()` and `executeForString()` methods of the `Custom` microagent.

Examine your current monitoring solution to identify tasks that can be fully automated using TIBCO Hawk software. Some monitoring tasks performed by scripts may be easier to perform using TIBCO Hawk agents and rulebases. For example, if you use a script that checks for process existence on a UNIX system, the script probably executes the `ps` command every so often. Using the `Process` microagent is a more efficient way to do this because it does not have to start a new process each time the process table is checked.

TIBCO Documentation and Support Services

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

How to Access TIBCO Documentation

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

The Product Documentation website is updated frequently and is more current than any other documentation included with the product.

Product-Specific Documentation

The documentation for this product is available on the TIBCO Hawk® Documentation page.

How to Contact Support for TIBCO Products

You can contact the Support team in the following ways:

- To access the Support Knowledge Base and getting personalized content about products you are interested in, visit our product Support website.
- To create a Support case, you must have a valid maintenance or support contract with a Cloud Software Group entity. You also need a username and password to log in to the product Support website. If you do not have a username, 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 TIBCO Community.

Legal and Third-Party Notices

SOME CLOUD SOFTWARE GROUP, INC. (“CLOUD SG”) SOFTWARE AND CLOUD SERVICES EMBED, BUNDLE, OR OTHERWISE INCLUDE OTHER SOFTWARE, INCLUDING OTHER CLOUD SG SOFTWARE (COLLECTIVELY, “INCLUDED SOFTWARE”). USE OF INCLUDED SOFTWARE IS SOLELY TO ENABLE THE FUNCTIONALITY (OR PROVIDE LIMITED ADD-ON FUNCTIONALITY) OF THE LICENSED CLOUD SG SOFTWARE AND/OR CLOUD SERVICES. THE INCLUDED SOFTWARE IS NOT LICENSED TO BE USED OR ACCESSED BY ANY OTHER CLOUD SG SOFTWARE AND/OR CLOUD SERVICES OR FOR ANY OTHER PURPOSE.

USE OF CLOUD SG SOFTWARE AND CLOUD SERVICES IS SUBJECT TO THE TERMS AND CONDITIONS OF AN AGREEMENT FOUND IN EITHER A SEPARATELY EXECUTED AGREEMENT, OR, IF THERE IS NO SUCH SEPARATE AGREEMENT, THE CLICKWRAP END USER AGREEMENT WHICH IS DISPLAYED WHEN ACCESSING, DOWNLOADING, OR INSTALLING THE SOFTWARE OR CLOUD SERVICES (AND WHICH IS DUPLICATED IN THE LICENSE FILE) OR IF THERE IS NO SUCH LICENSE AGREEMENT OR CLICKWRAP END USER AGREEMENT, THE LICENSE(S) LOCATED IN THE “LICENSE” FILE(S) OF THE SOFTWARE. USE OF THIS DOCUMENT IS SUBJECT TO THOSE SAME TERMS AND CONDITIONS, AND YOUR USE HEREOF SHALL CONSTITUTE ACCEPTANCE OF AND AN AGREEMENT TO BE BOUND BY THE SAME.

This document is subject to U.S. and international copyright laws and treaties. No part of this document may be reproduced in any form without the written authorization of Cloud Software Group, Inc.

TIBCO, the TIBCO logo, the TIBCO O logo, Hawk, LogLogic, Rendezvous, Administrator, and BusinessWorks are either registered trademarks or trademarks of Cloud Software Group, Inc. in the United States and/or other countries.

All other product and company names and marks mentioned in this document are the property of their respective owners and are mentioned for identification purposes only. You acknowledge that all rights to these third party marks are the exclusive property of their respective owners. Please refer to Cloud SG’s Third Party Trademark Notices (https://www.cloud.com/legal) for more information.

This document includes fonts that are licensed under the SIL Open Font License, Version 1.1, which is available at: https://scripts.sil.org/OFL

Copyright (c) Paul D. Hunt, with Reserved Font Name Source Sans Pro and Source Code Pro.

Cloud SG software may be available on multiple operating systems. However, not all operating system platforms for a specific software version are released at the same time. See the “readme” file for the availability of a specific version of Cloud SG software on a specific operating system platform.

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

THIS DOCUMENT COULD INCLUDE TECHNICAL INACCURACIES OR TYPOGRAPHICAL ERRORS. CHANGES ARE PERIODICALLY ADDED TO THE INFORMATION HEREIN; THESE CHANGES WILL BE INCORPORATED IN NEW EDITIONS OF THIS DOCUMENT. CLOUD SG MAY MAKE IMPROVEMENTS AND/OR CHANGES IN THE PRODUCT(S), THE PROGRAM(S), AND/OR THE SERVICES DESCRIBED IN THIS DOCUMENT AT ANY TIME WITHOUT NOTICE.

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

This and other products of Cloud SG may be covered by registered patents. For details, please refer to the Virtual Patent Marking document located at https://www.cloud.com/legal.

Copyright © 1996-2024. Cloud Software Group, Inc. All Rights Reserved.