IBM Agent Builder
6.3.5

User's Guide

Note

Before you use this information and the product it supports, read the information in “Notices” on page 373.

This edition applies to version 6.3.5 of IBM® Agent Builder and to all subsequent releases and modifications until otherwise indicated in new editions.

© Copyright International Business Machines Corporation 2010, 2021.
US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

Figures ix

Tables xiii

Chapter 1. Overview of Agent Builder 1
- Common Agent Builder procedures 1
- Data sources and data sets 2
- Monitoring multiple servers or instances of a server 4
- Testing, installing, and configuring an agent 4
- Operating system requirements 5
- Features specific to IBM Tivoli Monitoring 6

Chapter 2. Installing and starting Agent Builder 7
- Prerequisites for installing and running Agent Builder 7
  - Detailed system requirements for Agent Builder 7
- Installing Agent Builder 7
  - Using the installation wizard to install Agent Builder 7
  - Silent installation 9
- Starting Agent Builder 10
- Setting the default browser in Agent Builder 10
- Setting the default Time Stamping Authority in Agent Builder 10
- Uninstalling Agent Builder 11
- Silent uninstallation 11

Chapter 3. Creating an agent 13
- Naming and configuring the agent 13
- Defining initial data sources 15
  - Selecting key attributes 15

Chapter 4. Using the Agent Editor to modify the agent 17
- Default operating systems 18
- Self-Describing Agent 18
- Environment variables 19
  - List of environment variables 19
- Watchdog information 29
- Cognos information 30
- Generate Agent wizard link 30
- The **Data Source Definition** page 31
  - Copying data sources by using the **Data Source Definition** page 31
- **Runtime Configuration Information** page 32
- Agent XML Editor page 32
- Saving your edits and changes 32
- Committing a version of the agent 33
- Setting a new version number for your agent 34
- Changing the product code 34

Chapter 5. Editing data source and attribute properties 35
- Creating, modifying, and deleting attributes 36
  - Creating attributes 37

Copying attributes.... 37
Editing attributes.... 37
Creating derived attributes.... 37
Editing derived attributes.... 39
Removing attributes.... 39
Fields and options for defining attributes.... 40
Attribute types.... 41
Numeric aspects of attributes.... 42
Specifying an enumeration for an attribute.... 44
Specifying severity for an attribute used as a status indicator.... 44
Filtering attribute groups.... 45
Formula Editor.... 45
Changing the Formula Editor component view.... 46
Component types.... 46
Formula Editor common options.... 48
Formula Editor - Formula errors.... 49
Formula operators and functions.... 50
Specifying operating systems.... 56
Configuring and Tuning data collection.... 56
Data types.... 57

Chapter 6. Defining and testing data sources.... 63
Monitoring a process.... 64
Defining connections for process browsing.... 67
Monitoring a Windows service.... 67
Defining connections for service browsing.... 68
Monitoring data from Windows Management Instrumentation (WMI).... 69
Testing WMI attribute groups.... 71
Monitoring a Windows Performance Monitor (Perfmon).... 71
Testing Perfmon attribute groups.... 73
Monitoring data from a Simple Network Management Protocol (SNMP) server.... 73
SNMP MIB errors.... 76
SNMP MIB Parsing options.... 76
Testing SNMP attribute groups.... 77
Monitoring events from Simple Network Management Protocol event senders.... 78
SNMP Event Configuration properties.... 80
Testing SNMP event attribute groups.... 82
Monitoring Java Management Extensions (JMX) MBeans.... 83
JMX configuration.... 91
JMX notifications.... 93
JMX monitors.... 93
Specific fields for Java Management Extensions (JMX) MBeans.... 98
Testing JMX attribute groups.... 101
Monitoring data from a Common Information Model (CIM).... 102
CIM configuration.... 103
Testing CIM attribute groups.... 104
Monitoring a log file.... 104
Log file parsing and separators.... 112
Testing log file attribute groups.... 113
Monitoring an AIX Binary Log.... 115
Monitoring a Windows Event Log.... 116
Filtering by event type.... 117
Filtering by event source.... 118
Filtering by event identifier.... 118
Monitoring a command return code.... 118
Editing a command file definition.... 122
Monitor output from a script.... 122

Collecting script data from a remote system... 123
Script parsing and separators... 123
Steps for monitoring output from a script... 125
Monitoring data from Java Database Connectivity (JDBC)... 128
JDBC configuration... 131
Stored procedures... 133
Testing JDBC attribute groups... 134
Monitoring system availability by using Ping... 135
Configuration files... 136
Testing Ping attribute groups... 137
Monitoring HTTP availability and response time... 137
HTTP tables... 138
Monitoring a URL... 142
Monitor `https://` URLs... 143
Proxy server... 143
HTTP configuration... 143
Testing HTTP attribute groups... 145
Monitoring data from a SOAP or other HTTP data source... 145
XML representation of JSON data... 149
Specific fields for SOAP attributes... 150
SOAP configuration... 152
Testing SOAP attribute groups... 153
Monitoring data by using a socket... 154
Sending socket information to the agent... 156
Encoding of socket data... 159
Socket errors... 160
Socket configuration... 161
Remote socket port connection... 162
Sample script for socket... 162
Testing socket attribute groups... 163
Use the Java API to monitor data... 164
Running the Java application... 166
Generated sample Java application... 167
Java API configuration... 175
Testing Java application attribute groups... 176

Chapter 7. Creating data sets from existing sources... 177
Joining two attribute groups... 177
Manipulating attributes in joined attribute groups... 181
Joined attributes... 181
Creating a filtered attribute group... 182

Chapter 8. Creating a navigator group... 185

Chapter 9. Using subnodes... 187
Creating subnodes... 192
Subnode configuration... 193
Configuring a subnode... 194
Subnode configuration overrides... 195
Advanced subnode configuration... 195
Configuring a subnode from the command line... 197
Subnode configuration example... 197
Subnodes and Windows data sources... 203
Subnodes and Script data sources... 203

Chapter 10. Customizing agent configuration... 205
Changing configuration properties by using the Agent Editor... 207

Configuring a Windows remote connection..... 207
Creating a user with Windows Management Instrumentation (WMI) permissions.....208
Configuring a Secure Shell (SSH) remote connection..... 210

Chapter 11. Creating workspaces, Take Action commands, and situations.....213
Creating situations, Take Action commands, and queries..... 213
Creating workspaces..... 213

Chapter 12. Preparing the agent for Cloud APM..... 219

Chapter 13. Preparing the agent for Cloud Pak for Multicloud Management..... 223
Defining resources..... 223
Building resource relationships.....224

Chapter 14. Data Definition Designer.....227

Chapter 15. Testing your agent in Agent Builder.....229
Attribute group testing..... 229
Attribute group testing - preferences..... 231
Attribute group testing - configuration..... 231
Full agent testing..... 232
Test Environment variables..... 236

Chapter 16. Installing your agent into a monitoring infrastructure for testing and use..... 239
Installing an agent..... 239
Installing an agent locally..... 239
Creating the agent package..... 241
Installing the package in an IBM Tivoli Monitoring environment..... 242
Configuring and starting the agent in an IBM Tivoli Monitoring environment..... 243
Installing and using an agent in an IBM Cloud Application Performance Management environment..... 244
Agent post-generation and installation results..... 246
Uninstalling an agent..... 254
Removing a Tivoli Monitoring agent by using the Tivoli Enterprise Portal.....255
Removing a Tivoli Monitoring agent without using the Tivoli Enterprise Portal..... 255
Clearing a Tivoli Monitoring agent from the Tivoli Enterprise Portal..... 255
Uninstalling an IBM Cloud Application Performance Management agent..... 256

Chapter 17. Importing application support files.....257
Exporting and importing files for Tivoli Enterprise Monitoring Agents..... 257
Exporting and importing files for Tivoli System Monitor Agents..... 258

Chapter 18. Event filtering and summarization..... 259
Controlling duplicate events.....259
Viewing event filtering and summarization in the Tivoli Enterprise Portal..... 260

Chapter 19. Troubleshooting and support..... 267

Appendix A. Sharing project files..... 269
Share a Solution Installer Project..... 269

Appendix B. Command-line options..... 271
Command - **`generatelocal`**.....272
Command - **`generatemappingfile`**.....272

Command - **generatezip**..... 273

Appendix C. Attributes reference..... 275
Availability node..... 275
Performance Object Status node..... 280
Thread Pool Status attribute group..... 285
Event log attribute node..... 289
Log File Summary..... 291
AIX Binary Log attribute group..... 293
Monitor and Notification attribute groups..... 296
Counter Notifications..... 297
Gauge Notifications..... 300
Registered Monitors..... 302
String Notifications..... 304
SNMP Event attribute groups..... 306
JMX Event attribute groups..... 307
Ping attribute group..... 309
HTTP attribute groups..... 312
Discovery attribute groups..... 317
Take Action Status attribute group..... 318
Log File Status attribute group..... 322
Log File RegEx Statistics attribute group..... 325

Appendix D. Creating application support extensions for existing agents..... 331
Creating an Application Support Extension project..... 331
Adding support files to a project..... 331
Generating the Application Support Extension installation image..... 332
Installing your Application Support Extension..... 333
Converting a **Solution Install Project** to an Application Support Extension project..... 333

Appendix E. Cognos data model generation..... 335
Prerequisites to generating a Cognos data model..... 335
Tivoli Data Warehouse..... 335
Tivoli Common Reporting..... 338
Framework Manager..... 339
Creating reports..... 339
Populating the ManagedSystem Table..... 345
Exporting reports and data models from Tivoli Common Reporting..... 348
Importing reports into Agent Builder..... 349
Installing reports from an agent package into Tivoli Common Reporting..... 350

Appendix F. ICU regular expressions..... 351

Appendix G. Creating Non-agent file bundles..... 357
Remote Deploy Bundle Editor..... 357
Adding commands to the bundle..... 358
Adding prerequisites to the bundle..... 359
Adding files to the bundle..... 359
Generating the bundle..... 359
Creating deployable bundles for Tivoli Netcool/OMNIbus probes..... 360

Appendix H. Dynamic file name support..... 361

Appendix I. SNMP trap configuration..... 365

Appendix J. Take Action commands reference..... 369

SSHEXEC action....369

Accessibility features....371

Notices....373
Trademarks....374

Figures

1. Process Monitor page example....66

2. Runtime Configuration page.... 81

3. Test Event Settings window that shows collected SNMP event data.... 83

4. Data Collection Status window.... 83

5. JMX connection properties.... 85

6. Java Management Extensions (JMX) Browser window....87

7. JMX Agent-Wide Options window 90

8. Add Filter example 1....110

9. Add Filter example 2....111

10. Example attribute value output when Agent parses a simple log file data row....113

11. Example attribute value output when Agent parses a complex log file data row....113

12. Parse Log window that shows parsed log file attribute values....114

13. Example attribute value output when Agent parses complex script output....124

14. SOAP Browser window.... 147

15. SOAP Browser window 148

16. SOAP Browser window.... 149

17. Sample agent structure.... 167

18. Attribute Group Information pageAttribute Group Information window....180

19. Locating source attribute information....182

20. Subnodes in the Navigator tree.... 188

21. Subnodes monitoring different systems.... 189

22. Subnode types in Navigator tree.... 190

23. Monitoring multiple subnode instances of the same subnode type.... 191

24. Example: data collection in a subnode..... 192

25. SNMP Version 1 Properties expanded..... 196

26. Configuration property definitions in the Agent Builder..... 198

27. Top section with agent-level configuration for the Agent Cfg property..... 199

28. Main section with the agent-wide default value for the Overridable Cfg property..... 200

29. Example Subnode section page with no subnode 201

30. Example Subnode section page with two subnode instances defined..... 202

31. Setting the sysadmin user ID..... 214

32. Setting the sysadmin user ID (continued)..... 215

33. Setting the sysadmin user ID (continued)..... 216

34. Setting workspace properties..... 217

35. Setting workspace properties (continued)..... 218

36. Test Agent section of the Agent Editor, Agent Information page..... 233

37. Agent Test view with example subnode and navigator group highlighted..... 234

38. Agent Test perspective..... 235

39. The Attribute Group Test view that shows more information (Performance Object Status) about data collections for the Managed_URLs and Managed_Nodes attribute groups..... 236

40. Manage Tivoli Enterprise Monitoring Services window..... 250

41. Nodes for attribute groups in the new agent..... 251

42. Availability node..... 252

43. Performance Object Status node..... 253

44. Event log node..... 254

45. Historical view and cache view when event filtering or summarization is not enabled..... 261

46. Historical view and cache view when Only send summary events is selected..... 262

47. Historical view and cache view when Send all events is selected..... 263

48. Historical view and cache view when Send first event is selected..... 264

49. Historical view and cache view when Event threshold is selected..... 265

50. Selecting agent project file..... 341

51. Selecting Publish Packages..... 342

52. Selecting Common Reporting..... 343

53. Selecting Report Studio..... 344

54. Report Studio..... 345

55. The Content Administration tab..... 348

56. The Content Administration tab with agent package listed..... 349

57. Examples of configuration record types 2 and 3..... 366

Tables

1. Quick-reference information for creating agents 1

2. Quick-reference information for other functions 2

3. Environment variable descriptions including their default values and valid value ranges 20

4. Fields for editing data sources 35

5. Fields and options for defining attributes 40

6. Numeric attribute options 42

7. Valid format parameters for StringToTivoliTimestamp 52

8. StringToTivoliTimestamp examples 53

9. Fields on the Process Monitor page 64

10. SNMP Events configuration properties 81

11. Filter options 130

12. Supported SQL data types for use with a monitoring agent 131

13. Network Management configuration properties 137

14. HTML elements searched for objects to monitor 139

15. HTTP Attribute Information - Managed URLs 140

16. HTTP Attribute Information - URL Objects 141

17. URLs file entries 142

18. URL Monitoring configuration properties 144

19. Proxy Server configuration properties 144

20. Java configuration properties 144

21. SOAP Attribute Information 150

22. HTTP Server configuration properties 152

23. File types for supplemental files 155

24. Sample error code....158

25. Characters to encode in attribute values....159

26. Performance Object Status values....160

27. Socket configuration property....161

28. File types for supplemental files....165

29. Java trace level options....169

30. The data types of attribute fields and their IBM Tivoli Monitoring attribute type equivalents....169

31. Internal error codes for the agent....173

32. Changes to an agent that require modifications to the Java source....173

33. Java configuration properties....175

34. source attribute group one (single row)....178

35. source attribute group 2 (single row)....178

36. Resulting join....178

37. source attribute group one (single row)....178

38. source attribute group two (more than one row)....178

39. Resulting join....178

40. source attribute group one (more than 1 row)....179

41. source attribute group 2 (more than 1 row)....179

42. Resulting join (joining on Attribute3 and Attribute7)....179

43. Environment variables....236

44. Command quick-reference table....271

45. Required arguments....346

46. Optional arguments....346

47. Regular expression metacharacters....351

48. Regular expression operators....352

49. Replacement text characters..... 354

50. Flag options.....355

51. Predefined Variables for Commands.....358

52. Categories supported by the SNMP Data Provider..... 366

53. Severities supported by the SNMP Data Provider 367

54. Statuses supported by the SNMP Data Provider 367

55. Source IDs supported by the SNMP Data Provider.....368

Chapter 1. Overview of Agent Builder

You can use IBM Agent Builder to create and modify custom agents that extend the monitoring capabilities of an IBM Tivoli® Monitoring, IBM Cloud Application Performance Management, IBM Cloud Pak for Multicloud Management environment. A custom agent uses either of these environments to monitor any type of in-house or customized software.

Agent Builder is based on Eclipse, an open source integrated development environment.

Agent Builder includes the following features for the Tivoli Monitoring and Cloud APM environments:

Define and modify agents
You can create and modify agents. The agents collect and analyze data about the state and performance of different resources, such as disks, memory, processor, or applications, and provide this data to the monitoring environment.

Test and prepare agents for deployment
You can test an agent within Agent Builder, collecting data on the host where Agent Builder runs (in some cases you can collect information from a different host too). You can package the agent for easy distribution and deployment.

The following additional features are available for Tivoli Monitoring:

Custom workspaces, situations and Take Action commands
You can use Agent Builder to package additional workspaces, situations and Take Action commands as application support extensions with a new or existing agent running in the Tivoli Monitoring environment

Report data models
You can use Agent Builder to generate a Cognos® data model which you can use to build Tivoli Common Reporting reports. These reports can be packaged as part of your agent image.

Common Agent Builder procedures

The following table lists the main procedures that you can complete with Agent Builder.

You can use Agent Builder to create agents for the IBM Tivoli Monitoring and IBM Cloud Application Performance Management environments. You can also use it to create application support extensions for the Tivoli Monitoring environment. Application support extensions are created by creating workspaces and situations to enhance one or more existing agents.

Before you use Agent Builder, you must install it. For instructions, see Chapter 2, “Installing and starting Agent Builder,” on page 7.

To create, test, and use an agent, complete the procedures in the following table in the order that they are listed.

Table 1. Quick-reference information for creating agents

Goal	Refer to
Create an agent by using the **Agent** wizard.	• Chapter 3, “Creating an agent,” on page 13
Create data sources and attributes for your agent. **Important:** For a Cloud APM environment, a summary dashboard can display up to approximately five attributes; one of the attributes must denote overall agent or subnode status.	• Chapter 5, “Editing data source and attribute properties,” on page 35

© Copyright IBM Corp. 2010, 2021

Table 1. Quick-reference information for creating agents (continued)	
Goal	**Refer to**
For the Tivoli Monitoring environment, create workspaces and situations for your agent. • Running at least Tivoli Monitoring Version 6.1 Fix Pack 1 • Setting the Tivoli Universal Agent solution version back to "00" • Setting the value for "AppTag"	• Chapter 11, “Creating workspaces, Take Action commands, and situations,” on page 213 • Chapter 17, “Importing application support files,” on page 257
For the Cloud APM environment, create resource definitions and dashboards for your agent.	• Chapter 12, “Preparing the agent for Cloud APM,” on page 219
For the Tivoli Monitoring environment, create Cognos data models for reports for your agent.	• Appendix E, “Cognos data model generation,” on page 335
Test and debug your created agent, ensuring the availability of monitoring information.	• Chapter 15, “Testing your agent in Agent Builder,” on page 229 • Appendix B, “Command-line options,” on page 271 • Chapter 4, “Using the Agent Editor to modify the agent,” on page 17.
Generate an installation package and install the agent on the monitored host.	• “Installing an agent” on page 239
Remove an agent that you created with the Agent Builder.	• “Uninstalling an agent” on page 254

You can also use Agent Builder for packaging custom workspaces, situations, and Take Action commands as application support extensions for existing agents. These functions are available only for the Tivoli Monitoring environment:

Table 2. Quick-reference information for other functions	
Goal	**Refer to**
Create custom workspaces, situations, and Take Action commands.	• Chapter 11, “Creating workspaces, Take Action commands, and situations,” on page 213
Package your application support extension.	• Appendix D, “Creating application support extensions for existing agents,” on page 331
Build custom bundles.	• Appendix G, “Creating Non-agent file bundles,” on page 357

Data sources and data sets

An agent can monitor information from one or several data sources. It presents the information to the monitoring infrastructure as attributes, which are organized into data sets.

When you create an agent, you must define a *data source* for it. You can add more data sources. The data source defines how the agent gathers the monitoring information.

You can use Agent Builder to create agents that use data sources monitoring information from the following *data providers*:

- Process and service availability
- Network system availability (using ICMP ping)
- Command return codes
- Script output
- The Windows Event Log
- Windows Management Instrumentation (WMI)
- Windows Performance Monitor (Perfmon)
- Simple Network Management Protocol (SNMP)
- SNMP Events
- Hypertext Transfer Protocol (HTTP) availability and response time
- SOAP or other HTTP data source
- Java™ Database Connectivity (JDBC)
- Java application programming interface (API)
- Java Management Extensions (JMX)
- Common Information Model (CIM)
- Log files
- AIX® binary logs
- Socket

You can also use other development tools to create custom monitoring applications that pass information to the agent through log, script output, and Java API data sources.

When you add a data source, Agent Builder adds the corresponding *data set* to the agent. The data set organizes the information that is presented to the monitoring environment. In IBM Tivoli Monitoring, a data set is known as an *attribute group*.

A data set can consist of several *attributes*, which are values that the data source provides. Each time the monitoring environment queries the agent, it fetches values from data sources and returns then as attributes in data sets.

Some data sources can return several *rows* of attribute values in the same query, for example, if the data source monitors several services at once.

Most data sources present information as one data set. SNMP and JMX data sources can, depending on the configuration, provide diverse sets of information. When you add an SNMP or JMX data source, Agent Builder creates multiple data sets to accommodate this information.

You can edit the data sets to filter the data and to create additional *derived* attributes, that is, attributes that are calculated from existing attributes using a formula. You can also join data sets, creating a new data set with information from two or more data sets. In this way, users can view combined information from different data sources.

In IBM Tivoli Monitoring, you can view all attribute content. You can also create workspaces that present information from all agent data sets in a customized view. You can use IBM Tivoli Monitoring to create situations that are triggered when any attribute reaches a certain value. A situation can issue an alert and to call a system command.

In IBM Cloud Application Performance Management, you must define a *summary* dashboard for the agent, selecting up to five attributes that are visible in the dashboard. You can also define a *detail* dashboard that displays information from any data sets as tables. You can create thresholds that are triggered when any attribute reaches a certain value; you are not required to add this attribute to the dashboard. A threshold can issue alerts.

In IBM Cloud Pak for Multicloud Management you define resources for the agent, select attributes to be displayed, and identify important metrics to chart.

Monitoring multiple servers or instances of a server

An agent can monitor multiple servers, including multiple instances of the same server. There are two ways of creating such agents: multiple instances of an agent and subnodes within an agent.

Multiple instances are a standard way to monitor application servers that can have a number of similar instances on the same host. Many standard agents in IBM Tivoli Monitoring and IBM Cloud Application Performance Management support multiple instances.

With *multiple instances*, you install an agent on a monitored hosts and then configure one or several instances, setting a name for every instance. Configure an instance of the agent for each instance of the server that you want to monitor. Each instance is a separate identical copy of the agent, and it can be started and stopped separately.

You can also define one or several types of *subnode* within an agent. Each type must correspond to a different type of resource that an agent can monitor. A subnode type contains data sources and data sets; you can also define data sources and data sets at agent level, outside any subnode. When you install the agent on a host, you can configure the required number of subnodes of each type; for every subnode type, you can set the number of subnodes independently. For IBM Cloud Application Performance Management, you can create a dashboard for the agent and a separate dashboard for each subnode.

Subnodes require different configuration steps on the monitored host. Also, to reconfigure, add or remove a subnode. you must stop and restart the entire agent; an instance can be reconfigured, added, or removed without affecting other instances. However, subnodes have a number of advantages:

- With subnodes, you can monitor a large amount of server instances while consuming less resources. As a guideline, the number of agent instances of a specific type supported on a single system is 10. But an agent can monitor up to 100 local or remote servers using subnodes.
- One agent can include subnode types for a few different kinds of servers. On the monitored system, you can configure any number of subnodes of each type. You can use this feature to conserve resources further.
- An agent with subnodes can supply system-wide data on the agent level.

You can define both multiple instances and subnodes for the same agent. In this case, each instance can include a number of subnodes. You can stop and restart each instance independently of other instances; all subnodes in an instance are stopped and restarted together.

Testing, installing, and configuring an agent

You can create an installation package for an agent and then install it on any number of monitored hosts. For some data sources, you need to set configuration values for collecting data.

After defining data sources and attributes for an agent, you can test it by running it within Agent Builder. You can test a single data set (attribute group) or the full agent.

To test the agent more extensively and to use it, you can create an installation image. This image provides scripts for installing and configuring the agent on any monitored host.

Tip: Before installing the agent, ensure that the operating system agent for your monitoring environment (IBM Tivoli Monitoring, IBM Cloud Application Performance Management, or IBM Cloud Pak for Multicloud Management) is installed on the host.

After installing the agent, you might need to configure it. If the agent supports multiple instances, you must configure the agent to create at least one instance.

Some data sources require additional configuration values; for example, for the SNMP data source, you must configure the IP address of the host that you monitor using the SNMP protocol. Use the configuration script, which is deployed by the installation package, to set these values.

Alternatively, you can set these values in Agent Builder before creating the installation image. In this case, you do not have to set them again on the monitored hosts.

Tip: The help files for your custom agent might not display in Help Contents after the Cloud APM server is upgraded. To display the help files, complete these steps:

1. Download the latest version of IBM Agent Builder.
2. Re-create your custom agent. Make sure to assign a higher version number, fix pack, or patch level in the Agent Information page.
3. Install your custom agent on the monitored host.
4. From the Cloud APM console, click **Help** > **Help Contents** from the navigation bar. Your custom agent help is displayed.

Operating system requirements

Agents that are created by Agent Builder are supported on various operating systems, depending on the monitoring environment and on the settings you select when creating the agent.

In a Tivoli Monitoring environment, agents that are created by Agent Builder can support the following operating systems:

- AIX
- HP-UX
- Linux®
- Solaris
- Windows

The agents support the same operating system versions as the OS agents. For details, access the Software Product Compatibility Reports website. Search for the `Tivioli Monitoring` product name and select the `OS Agents & TEMA (Tivoli Enterprise Monitoring Agent)` component check box.

In an IBM Cloud Application Performance Management environment, agents that are created by Agent Builder can support the following operating systems:

- AIX
- Linux
- Windows

The agents support the same versions as the OS agents. For details, use the links in the Component reports section of System requirements (APM Developer Center).

To run your monitoring agent in an Tivoli Monitoring environment, install the appropriate operating system agent on every monitored system where your agent runs.

To run your monitoring agent in an IBM Cloud Application Performance Management environment, install any of the agents shipped with IBM Cloud Application Performance Management on every monitored system where your agent runs.

Note: Agent Builder browsers operate on the data sources and information accessible from the system on which the Agent Builder is run. Ensure that you run Agent Builder on either of the following types of systems:

- A system that runs on the same level as the operating system and monitored applications for which you are developing the agent
- A system that connects to another system that runs on the same level as the operating system and monitored applications for which you are developing the agent

Features specific to IBM Tivoli Monitoring

Agent Builder provides several features that apply only to IBM Tivoli Monitoring.

You can use navigator groups to organize the data that the agent displays in the IBM Tivoli Monitoring navigator views and workspaces. A navigator group combines the data from several attribute groups (data sets) into a single view, while hiding the original separate data sets from the user.

You can use Tivoli Enterprise Portal to create workspaces, situations, and Take Action commands for your agent. You can then use Agent Builder to save the workspaces, situations, and Take Action commands as application support files and bundle them with the agent. Moreover, Agent Builder can also import workspaces, situations, and Take Action commands for other agents and create custom application support files for them.

Agent Builder can generate a Cognos data model for the agent. Use the data model to import agent information into the Cognos Framework Manager, a part of IBM Tivoli Common Reporting, for report creation.

Chapter 2. Installing and starting Agent Builder

Before you install IBM Agent Builder, ensure that your system meets the prerequisites. Then use the installation wizard or the silent installation procedure to install Agent Builder.

Tip: For information about installing or modifying an *agent*, see "Installing an agent" on page 239.

Prerequisites for installing and running Agent Builder

To install and run Agent Builder, your system must meet certain requirements.

To install the Agent Builder, ensure that you have:

- A system with a minimum of 1 GB of free disk space. Agents that you develop will require additional disk space.
- A supported operating system. Agent Builder can run on the following operating systems:
  - Windows Windows
  - Linux Linux (x86 64-bit only)
- Linux If you are using the Linux operating system, you must install the `libstdc++.so.5` library. You can install the following packages that provide this library:
  - On Red Hat Enterprise Linux, `compat-libstdc++-33`
  - On SUSE Enterprise Linux, `libstdc++-33`

Windows On a Windows system, you must be able to run Agent Builder as a user with Administrator permissions. These permissions ensure that Agent Builder has an environment consistent with the agents that are developed with it.

Linux On a Linux system, you can run Agent Builder as root or as an ordinary user. However, if you run it as an ordinary user, testing of agents will be limited and in some cases might not be available.

Detailed system requirements for Agent Builder

Use the Software Product Compatibility Reports to view the detailed system requirements for Agent Builder.

Access the Software Product Compatibility Reports website. Search for the `IBM Agent Builder` product name.

Installing Agent Builder

You can use the installation wizard or the silent installation procedure to install Agent Builder.

Tip: Before you install Agent Builder, uninstall any previous versions. For more information about uninstalling, see ("Uninstalling Agent Builder" on page 11). None of your existing agent information is lost when you uninstall.

Using the installation wizard to install Agent Builder

You can use the installation wizard to install IBM Agent Builder.

Before you begin

Ensure that your system meets the prerequisites. For information about prerequisites, see "Prerequisites for installing and running Agent Builder" on page 7

© Copyright IBM Corp. 2010, 2021

Procedure

1. If you are not signed in to IBM Marketplace, sign in with your IBMid and password and go to **Products and services**.

   The **Products and services** page is available to active subscribers. If you have any issues, go to the Cloud Application Performance Management Forum or to Marketplace support.
2. Download the Agent Builder installation archive file:
   a) In the Cloud APM subscription box, click **Manage** > **Downloads**.
   b) Select **Multi-Platform** as the operating system.
   c) Select the IBM Agent Builder package.
   d) Click **Download** and save `IBM_Agent_Builder_Install.tar` to your system.
3. Extract the installation archive file.
4. Use the following command in the extracted image directory to start the installation:
   - Windows `setup.bat`
   - Linux UNIX `./setup.sh`

   Important: Run the installation program with the same user ID that you intend to run the Agent Builder with.
5. When the **IBM Agent Builder** window opens, select your language, and click **OK**.
6. On the **Introduction** page, click **Next**.
7. On the **Software License Agreement** page, click **I accept the terms in the license agreement**, and click **Next**.
8. On the **Choose Install Folder** page, click one of the following options:
   - **Next** to install Agent Builder to the directory specified in the **Where Would You Like to Install?** field.
   - **Restore Default Folder** to install the Agent Builder in a default directory.
   - **Choose** to select a different directory.

   Note: The directory name that you choose must not contain the following characters:

   !
   #
   %
   ;

   If it includes any of these characters, Agent Builder might not start.
9. On the **Pre-Installation Summary** page, click **Install**.
10. On the **Installing IBM Agent Builder** page, wait for the **Install Complete** page to open, then click **Done**.

Results

Windows After the Agent Builder is installed, an option is added to the Start menu and an Agent Builder icon is added to your desktop. The installation log files are in *install_dir*\IBM_Agent_Builder_InstallLog.xml.

Linux UNIX After the Agent Builder is installed, the Agent Builder executable file is named *Install_Location*/agentbuilder. The installation log files are in *install_dir*/IBM_Agent_Builder_InstallLog.xml.

Silent installation

You can install Agent Builder by using a silent installation method. This method does not require a graphical environment and can be easily replicated on several hosts.

About this task

The silent installation options file, `installer.properties`, is included in the installation image at the root of the installation directory. You must modify this file to meet your needs, and then run the silent installer. You can copy this file to other hosts and quickly install Agent Builder on all of them.

Procedure

1. If you are not signed in to IBM Marketplace, sign in with your IBMid and password and go to **Products and services**.

   The **Products and services** page is available to active subscribers. If you have any issues, go to the Cloud Application Performance Management Forum or to Marketplace support.
2. Download the Agent Builder installation archive file:
   a) In the Cloud APM subscription box, click **Manage** > **Downloads**.
   b) Select **Multi-Platform** as the operating system.
   c) Select the IBM Agent Builder package.
   d) Click **Download** and save `IBM_Agent_Builder_Install.tar` to your system.
3. Extract the installation archive file.
4. Create a copy of the `installer.properties` file, which is located in the installation image directory.
5. Edit the new file to suit your needs. An example of the contents of this file is:

```
# ------------------------------------------------------------------
# IBM Agent Builder
#
# (C) Copyright IBM Corporation 2009.  All rights reserved.
#
# Sample response file for silent install
#
# To use this file, use the following command:
#
# Windows:
#    setup.bat -i silent -f <path>\installer.properties
#
# Linux or AIX:
#    setup.sh -i silent -f <path>/installer.properties
#
# Where
#    <path> is a fully-quailfied path to the installer.properties
#    file (including the drive letter or UNC path name on Windows).
#    <path> cannot contain spaces.
# ------------------------------------------------------------------


# ------------------------------------------------------------------
# This property indicates that the license has been accepted
# ------------------------------------------------------------------
# LICENSE_ACCEPTED=FALSE

# ------------------------------------------------------------------
# This property specifies the install directory
#
# On Windows, the default is:
#     C:\\Program Files (x86)\\IBM\\AgentBuilder
#
# On Linux, the default is:
#     /opt/ibm/AgentBuilder
# ------------------------------------------------------------------
#USER_INSTALL_DIR=C:\\Program Files (x86)\\IBM\\AgentBuilder
#USER_INSTALL_DIR=/opt/ibm/AgentBuilder
```

6. Start the silent installation by running the following command in the extracted installation image directory:

Windows `setup.bat -i silent -f path/installer.properties`

Linux UNIX `./setup.sh -i silent -f path/installer.properties`

Where *path* is the fully-qualified path to the `installer.properties` file (including the drive letter or UNC path name on Windows). The path cannot contain spaces.

Starting Agent Builder

After installing Agent Builder, you can start it.

Procedure

- Start the Agent Builder by using one of the following methods
  - Windows On Windows systems:
    - From a command-line type: *Install_Location*\agentbuilder.exe.
    - Select **Start** > **All Programs** > **IBM** > **Agent Builder**.
    - Click the **Agent Builder desktop icon**.
  - Linux On Linux systems, start the following executable file: *INSTALL_DIR*/agentbuilder

Note: When you run the Agent Builder, it prompts you for the location of your workspace directory. The files that create your agents are saved in that directory. You can designate any directory as your workspace.

Setting the default browser in Agent Builder

Linux On Linux systems, you might need to set the Agent Builder default browser so that help panels are displayed.

Procedure

1. Select **Window** > **Preferences** to open the **Preferences** window.
2. Select and expand the **General** node.
3. Select **Web Browser**.
4. Select **Use external web browser**.
5. Select the browser that you want to use.
6. Optional: To add a web browser, complete the following steps
   a) Click **New**.
   b) In the **Name** field, enter a descriptive name for the browser.
   c) In the **Location** field, enter the full path to the browser executable file .
   d) Click **OK**.
7. Click **OK**.

Setting the default Time Stamping Authority in Agent Builder

You can set the Time Stamping Authority for JAR files in the Agent Builder **Preferences** window. If the default Time Stamping Authority signing certificate expires, by setting a new authority, you can continue to verify JAR files.

Procedure

1. Select **Window** > **Preferences** to open the **Preferences** window.
2. Select and expand the **IBM Agent Builder** node.

3. Select **Jar Signing**.
4. Select **Add time stamp to signed JAR files**.
5. Enter the URL of the Time Stamping Authority.
6. Click **OK**.

Uninstalling Agent Builder

Depending on your operating system, you can use different procedures to uninstall Agent Builder.

Procedure

- Linux
  On Linux systems, run the following command:
  a) *INSTALL_DIR*/uninstall/uninstaller

     where *INSTALL_DIR* is the name of the directory where Agent Builder is installed.
- Windows
  On Windows 7, Windows Server 2008 R2, and later versions of Windows, complete the following steps:
  a) Open Windows Programs and Features by selecting **Start** > **Control Panel** > **Programs** > **Programs and Features**.
  b) Select **IBM Agent Builder** from the list of installed programs.
  c) Click **Uninstall/Change**.
  d) Click **Uninstall** on the **Uninstall IBM Agent Builder** page.
  e) Click **Done** on the **Uninstall Complete** page.

  Tip: On Windows 7 and Windows Server 2008 R2, you can also go to the **Windows Programs and Features** window by selecting **Start** > **Computer** > **Uninstall or change a program**. Then, continue from step "2" on page 11.
- Windows
  On other Windows systems, complete the following steps:
  a) From the Windows Control Panel, select **Add/Remove Programs**.
  b) Click **IBM Agent Builder**.
  c) Click **Change/Remove**.
- On all operating systems, you can also use the silent uninstallation method. Start the silent uninstallation by running the following command:
  - Windows On Windows systems, *INSTALL_DIR*/uninstall/uninstaller.exe -i silent
  - Linux On Linux systems, *INSTALL_DIR*/uninstall/uninstaller -i silent

Silent uninstallation

You can use the silent uninstallation method to uninstall.

Procedure

- Start the silent uninstallation by running the following command:

```
INSTALL_DIR/uninstall/uninstaller[.exe] -i silent
```

Chapter 3. Creating an agent

To start creating an agent in Agent Builder, use the new agent wizard. With this wizard you can set the basic agent configuration and create one data source. You can then work on the agent in Agent Builder to add more data sources and other options, including subnodes and navigator groups.

Naming and configuring the agent

Use the **Agent** wizard to name your agent, set its version, supported operating systems, and other configuration settings.

Procedure

1. Use one of the following ways to start the new agent wizard:
   a) Click the **Create New Agent** icon on the toolbar.
   b) From the Main menu, select **File** > **New** > **Agent**.
   c) From the Main menu, select **File** > **New** > **Other**. In the **Select a Wizard** page, double-click the **Agent Builder** folder, then double-click **Agent**.

   The **Agent** wizard opens.
2. Click **Next**.
3. In the **New Agent Project** page, set the name of the project in the **Project name** field. Agent Builder uses this name for the folder that contains the agent files. You can optionally change the following settings:
   - If you want to store the agent files in a different location, clear **Use default location** and click **Browse** to select the new directory in the **Location** field.
   - You can change how the Eclipse Navigator View displays resources by adding them to various working sets. For more information, see the Eclipse help. To add the agent to Eclipse working sets, select **Add project to working sets** and click the **Select** button to add the sets to the **Working sets** field.
4. Click **Next**.
5. In the **General Information** page, configure the following settings:
   - Type the copyright statement that you want to use for your new agents in the **Copyright** field. This statement must meet your legal requirements for copyrights. This copyright statement is inserted into all files that are generated for the agent; you can edit it later.
   - Select the operating systems for which you want your agent to be built.

     Important: If you want to run a full test of the agent inside Agent Builder (for instructions, see "Full agent testing" on page 232), ensure that:

     – If you are running Agent Builder on Windows, the 32-bit version of the operating system is installed.
     – If you are running Agent Builder on Linux, the 64-bit version of the operating system is installed.

     Important: In some rare cases, you might need to install your agent on a 64-bit system where only a 32-bit operating system agent is installed. In this case, ensure that the 64-bit version of the operating system is not selected and the 32-bit version is selected.

     Important: 64-bit Windows Server 2003 R2 and earlier Windows systems are not supported by the agents created using Agent Builder.
6. Click **Next**.

© Copyright IBM Corp. 2010, 2021

7. In the **Agent Information** page, configure the following settings:

   - Set the service name for the agent in the **Service name** field. The name is displayed in the **Manage Tivoli Monitoring Services** window in an IBM Tivoli Monitoring environment and in the **Manage Monitoring Services** utility and Threshold editor in an IBM Cloud Application Performance Management. On Windows systems, it is also the name of the Windows service that runs the agent. The full service name always starts with `Monitoring Agent for`. You enter the remaining part of the name, which normally describes the service that this agent monitors. The name can contain letters, numbers, spaces, and underscores.
   - Set a three-character product code for the agent in the **Product code** field. A product code is required for both IBM Tivoli Monitoring and IBM Cloud Application Performance Management. A range of product codes is reserved for use with the Agent Builder. The permitted values are `K00-K99`, `K{0-2}{A-Z}`, and `K{4-9}{A-Z}`.

     Important: These values are for internal use only and are not intended for agents that are to be shared or sold outside your organization. If you are creating an agent to be shared with others, you must send a note to toolkit@us.ibm.com to reserve a product code. The request for a product code must include a description of the agent to be built. A product code is then assigned, registered, and returned to you. When you receive the three-letter product code, you are told how to enable the Agent Builder to use the assigned product code.
   - Set a string that uniquely identifies the organization that develops the agent in the **Company identifier** field (`IBM` is reserved). You can take it from the URL of your company; for example, if the company website is `mycompany.com`, use the text `mycompany`.
   - Set a string that uniquely identifies the agent in the **Agent identifier** field. By default, Agent Builder sets the Agent identifier to be the same as the Product code.

     Important: The combined length of the **Agent identifier** field and the **Company identifier** field cannot exceed 11 characters.
   - Set the agent version in the **Version** field. The agent version contains three digits in the format *V*`.`*R*`.`*R*, where:

     V = Version
     R = Release
     R = Release

     For displaying in the monitoring environment, the *V*`.`*R*`.`*R* value is converted into the following format: `0V.RR.00.00`

     Tip: In the agent editor, a **patch level** field is available. The **patch level** field can be used when you release a fix for an agent, without updating the version.
   - If you want your agent to support multiple instances, select the **Support multiple instances of this agent** check box. You can use multiple instances of an agent to monitor several instances of an application on the same host, or to use an agent installed on one host to monitor several software servers on different hosts. When you install an agent that support multiple instances, you can create and configure as many instances as necessary.

What to do next

Click **Next** to define an initial data source for your agent. For more information, see "Defining initial data sources" on page 15

Defining initial data sources

When creating an agent, define the initial data that the agent is to monitor. You can add more data sources later in the agent editor.

About this task

Define the data sources that your new agent is to monitor by using the **Agent Initial Data Source** page. For detailed instructions about creating data sources from various data providers, see Chapter 6, "Defining and testing data sources," on page 63.

Procedure

1. On the **Agent Initial Data Source** page, select one of the **Monitoring Data Categories** and one of the **Data Sources**.
2. Click **Next** The wizard guides you through the process of defining and configuring any of the data collection types that you specify.

   Tip: You can use this wizard to define a data source or to add a subnode or navigator group for organizing the agent. For more information about subnodes, see Chapter 9, "Using subnodes," on page 187. For more information about navigator groups, which are used only for IBM Tivoli Monitoring, see Chapter 8, "Creating a navigator group," on page 185.
3. If you defined a new data source that might return more than one data row, you are prompted to select key attributes. For more information, see ("Selecting key attributes" on page 15).
4. After you define the first data source, the **Data Source Definition** window displays. To add another data source, select the agent, or a subnode or navigator group if one is present, and click the **Add to Selected** button.
5. To finish defining data sources, click **Finish**. Aent Builder creates the new agent and opens it in the agent editor.

Selecting key attributes

When an attribute group returns more than one data row, you must select key attributes.

About this task

When an attribute group can return more than one data row, each row represents an entity that is being monitored. Each time monitored data is sampled, the monitoring environment matches a row to the entity that is being monitored and to previous samples for that entity. This matching is done with key attributes. One or more attributes in the attribute group can be identified as key attributes. These key attributes, when taken together, distinguish one monitored entity from another. The key attributes do not change from one sample to the next for the same monitored entity.

Rate and delta attributes are calculated by comparing the current sample to the previous sample. Identical key attributes ensure that the agent is comparing values for the same monitored entity. Similarly, the summarization and pruning agent summarizes samples that have identical key attributes. In addition, any attribute that is set as a key attribute can also be used as a "Display item" in a situation.

You specify the details about your new data source in the **Agent Initial Data Source** page. If the selected data source might return multiple data rows, Agent Builder can sometimes detect the key attributes. Otherwise, it prompts you to select key attributes.

Procedure

- On the **Select key attributes** page, take one of the following steps:
  - Click one or more attributes from the list that are the key attributes for this entity. To select more than one attribute, hold down the Ctrl key.

- If this attribute group returns only one row, select **Produces a single data row**. If this option is selected, no key attributes are necessary because only one monitored entity is ever reported in this attribute group.

Chapter 4. Using the Agent Editor to modify the agent

Use the Agent Editor to change, save, and commit a version of your agent.

You can create a new agent in Agent Builder; for more information, see Chapter 3, “Creating an agent,” on page 13. After creating an agent, you can modify it using the Agent Editor.

To open an agent that you created in Agent Builder in the Agent Editor, in the **Project Explorer** pane, find the name of the agent and expand it. Under the name of the agent, double-click **Agent Definition**. Alternatively, double-click the `itm_toolkit_agent.xml` filename.

The Agent Editor is a multi-page Eclipse editor that you can use to modify the properties of an existing agent. Each page in the editor corresponds to a specific function of the agent.

The list of available pages is shown in the Outline view under the **Agent Definition** node. You can easily switch to another page by clicking a node in the Outline view. If the Outline view is missing, or hidden behind another view, you can reset the Agent Definition perspective. Reset the perspective by selecting **Window** > **Reset Perspective**. Alternatively, right-click the **Agent Definition** tab and select **Reset** from the menu.

Note: For detailed information and procedures for creating an agent, see Chapter 3, “Creating an agent,” on page 13.

The following pages are included in the Agent Editor:

- “Agent Information page” on page 17
- Data Source Definition page
- Runtime Configuration Information page
- Agent XML Editor page (itm_toolkit_agent.xml)

Note: When you view an Editor page, you can also switch to another page by clicking the tab for the page. Some pages show tabs only when they are selected in the Outline view. You can force a page to have a tab even when it is not selected. To force a page to have a tab, click the pin icon so that the pin in the icon points toward the page.

Agent Information page

The **Agent Information** page is the main page of the Agent Editor.

The **Agent Information** page contains the following information:

- General agent information, including the agent service name and the product code. You can click **Advanced** to set different names for different use, but this setting is normally not needed.
- Agent Content information
  - **Default Operating Systems** link
  - **Self-Describing Agent** link
  - **Environment Variables** link
  - **Watchdog Information** link
  - **Cognos Information** link
  - **Data sources** link
  - **Runtime Configuration** link
  - **Resources** link
  - **Dashboards** link
- **Test Agent** link
- **Generate Agent Wizard** link

© Copyright IBM Corp. 2010, 2021

- **Commit Agent Version** link

Configuring the time for transient error messages

Agent Editor wizards sometimes display transient error messages. A message is displayed for a short time (by default, 3 seconds) in the header of the wizard. You can configure the duration for which these messages are displayed. To change this setting:

1. Select **Window** > **Preferences** from the Agent Builder menu bar. The **Preferences** window opens.
2. Select **Agent Builder**.
3. Set the **Time (seconds) that transient error message are displayed** setting.
4. Click **OK**.

Default operating systems

Use the **Default Operating Systems** page to change the operating systems for which your agent is built.

Procedure

- To open the **Default Operating Systems** page, click **Default Operating Systems** in the **Agent Content** section of the **Agent Information** page or the **Default Operating Systems** node in the Outline View.
- In the **Default Operating Systems** page, select the operating systems that your agent must support.

  When you generate an installation package for the agent, Agent Builder adds files for the selected operating systems to the package. Data sources that you add to your agent that are not specific to the Windows operating system are available on any of the selected operating systems. The operating systems on which any specific data source is available can be changed from this default selection. To change the Operating Systems available for a specific data source, use the **Operating Systems** pane of the **Data Source Definition** page. If default operating systems are not selected, operating systems must be selected for each specific data source on the **Data Source Definition** page.

  Important: If you want to run a full test of the agent inside Agent Builder (for instructions, see "Full agent testing" on page 232), ensure that:

  – If you are running Agent Builder on Windows, the 32-bit version of the operating system is installed.
  – If you are running Agent Builder on Linux, the 64-bit version of the operating system is installed.

  Important: In some rare cases, you might need to install your agent on a 64-bit system where only a 32-bit operating system agent is installed. In this case, ensure that the 64-but version of the operating system is not selected and the 32-bit version is selected.

Self-Describing Agent

For the IBM Tivoli Monitoring environment, use the **Self-Describing Agent** page to specify whether the agent's support files are bundled with the agent. For the IBM Cloud Application Performance Management environment, you must leave Self-Describing Agent enabled.

Procedure

- To open the **Self-Describing Agent** page, click **Self-Describing Agent** in the **Agent Content** section of the **Agent Information** page or the **Self-Describing Agent** node in the Outline View.

  Self-description is enabled by default for all new agents that are created with Agent Builder 6.2.3 or later. If the agent is for the IBM Cloud Application Performance Management environment, self-description must be enabled.

  When self-description is enabled for an agent, application support packages are included in the agent image. The inclusion enables the agent to seed the support files for the Tivoli Enterprise Monitoring Server, Tivoli Enterprise Portal Server, the Tivoli Enterprise Portal Browser. For more information about

self-describing agents, see the *IBM Tivoli Monitoring Installation and Setup Guide* and the *IBM Tivoli Monitoring Administrator's Guide*. In an IBM Cloud Application Performance Management environment, self-description enables the agent to seed support files onto the Cloud APM server; the seeding is a required step in the environment.

Note: In a IBM Tivoli Monitoring environment, you must have Tivoli Monitoring version 6.2.3 or later installed for the self-describing agent feature to work, and self-description must be enabled in Tivoli Monitoring. By default self-description is turned off in Tivoli Monitoring.

Note: Selecting the **Enable self-description for this agent** check box does not prevent your agent from working on previous versions of Tivoli Monitoring.

Environment variables

Use the **Environment Variables** page to view and modify environment variables that are available to your agent while it is running.

Before you begin

For more information about the **Agent Editor** and **Agent Information** page, see Chapter 4, "Using the Agent Editor to modify the agent," on page 17.

About this task

The environment variables can be defined by you, for access inside a script, or predefined variables that cause the agent to behave in a certain way. See "List of environment variables" on page 19 for a list of predefined variables.

Procedure

1. To open the **Environment Variables** page, click **Environment Variables** in the **Agent Content** section of the **Agent Information** page. Alternatively, click **Environment Variables** node in the **Outline** view.
2. In the **Environment Variables** page, click **Add** to add a new variable. Alternatively, to edit an existing variable, select it and click **Edit**.
3. In the **Environment Variable Information** window, set the following values:
   - In the **Name** field, type a variable name or select a predefined name from the list.
   - In the **Value** field, type a value for the variable if you want to set a variable for the agent. If you do not enter a value, the agent propagates a value for the existing variable.
   - In the **Description** field, type a description of the variable, or keep the existing description of a predefined variable.

   a) Click **OK**.

   The new variable is listed in the table on the **Agent Information** page.

List of environment variables

Use environment variables to control the behavior of the agent at run time.

Environment variables can be built into the agent by using the **Environment Variables** page. On Windows systems, environment variables are defined in the agent K*XX*ENV file. On UNIX and Linux systems, these variables can be defined in the agent `$CANDLEHOME/config/XX.ini` file, where *XX* is the two-character product code. The agent must be restarted for the new settings to take effect.

Note: Environment variables are not set correctly on a remote system that runs C Shell. Use a different shell if you want to use environment variables.

Table 3. Environment variable descriptions including their default values and valid value ranges

Environment variable	Default value	Valid values	Description
CDP_DP_REFRESH_INTERVAL	60 if subnodes are defined, otherwise not set	Any positive integer (such as 3600 for a 1-hour interval)	The interval, in seconds, at which attribute groups are updated in the background. If this variable is not set or is set to 0, background updates are disabled. Use this variable to to tune behavior so that the data is fresh enough without causing undue load on the application from which data is being collected. If a thread pool is configured (see variable CDP_DP_THREAD_POOL_SIZE), then the attribute groups can be refreshed in parallel. If there is no thread pool, the updates happen serially, which can take a long time. Logically equivalent to a thread pool size of 1.
CDP_*ATTRIBUTE_GROUP*_ REFRESH_INTERVAL	Not Applicable	Any positive integer (such as 600 for a 10-minute interval	Override the interval, in seconds, at which a particular attribute group is updated in the background when the agent is updating data in the background because CDP_DP_REFRESH_INTERVAL was set. This variable works in the same way as CDP_DP_REFRESH_INTERVAL except it targets only the specified attribute group. The attribute group name in the variable name must be in uppercase, even if the actual attribute group name is not. If the CDP_DP_REFRESH_INTERVAL environment variable has not been set, the attribute group override does not take effect. You can simulate background collection for a subset of attribute groups by using a large value for CDP_DP_REFRESH_INTERVAL, such as 86400 for once a day.

Table 3. Environment variable descriptions including their default values and valid value ranges (continued)

Environment variable	Default value	Valid values	Description
CDP_DP_THREAD_POOL_SIZE	15 if subnodes are defined, otherwise not set	Any non-negative integer	The number of threads that are created to run background data collections at an interval that is defined by CDP_DP_REFRESH_INTERVAL. If this variable is not set or is set to 0, there is no thread pool. If CDP_DP_THREAD_POOL_SIZE is set to a value greater than 1 and CDP_DP_REFRESH_INTERVAL is set to 0, te value of CDP_DP_THREAD_POOL_SIZE is ignored and data collection happens on demand. The Thread Pool Status attribute group shows how the thread pool is running. Use the Thread Pool Status to adjust the thread pool size and refresh interval for best results. By default, the query for this attribute group is not displayed on the agent Navigator tree. You might not remember to include the query in a custom workspace for the agent. However, you can easily view it by assigning the Thread Pool Status query to a base agent level workspace view.
CDP_DP_CACHE_TTL	55	Any integer greater than or equal to 1.	Data that is collected for an attribute group is cached for this number of seconds. Multiple requests for the same data in this time interval receive a cached copy of the data. This value applies to all attribute groups in the agent.
CDP_*ATTRIBUTE_GROUP*_CACHE_TTL	Value of CDP_DP_CACHE_TTL	Any integer greater than or equal to 1.	Data that is collected for the particular specified attribute group is cached for this number of seconds. Multiple requests for the same data in this time interval receive a cached copy of the data. This value overrides CDP_DP_CACHE_TTL for the specified group. The attribute group name in the variable name must be in uppercase, even if the actual attribute group name is not.

Table 3. Environment variable descriptions including their default values and valid value ranges (continued)

Environment variable	Default value	Valid values	Description
CDP_DP_IMPATIENT_ COLLECTOR_TIMEOUT	5 if subnodes are defined, otherwise not set	Any positive integer	The number of seconds to wait for a data collection before a timeout and cached data is returned, even if the cached data is stale. (Cached data is stale if older than CDP_DP_CACHE_TTL seconds). If this variable is not set, the agent waits until the data collection completes. The wait at times can make the Tivoli Enterprise Portal timeout and give up waiting. If no thread pool is configured, this variable is ignored and data collection is done synchronously.
CDP_JDBC_MAX_ROWS	1000	Any positive integer	The maximum number of rows of data that the JDBC data provider returns. A result set that contains more than this number of rows is processed only up to this maximum value. Queries can be developed to prevent too much data from being returned to IBM Tivoli Monitoring.
CDP_NT_EVENT_LOG_GET_ALL _ENTRIES_FIRST_TIME	NO	YES, NO	If set to YES, the agent sends an event for every event in the Windows event log. If set to NO, only new events in the Windows event log are sent.
CDP_NT_EVENT_LOG_CACHE _TIMEOUT	3600	Any integer greater than or equal to 300.	The number of seconds Windows Event log events are cached by the agent. All cached events are returned when the event log attribute group is queried. **Note:** This variable is no longer used. Use the CDP_PURE_EVENT_CACHE_SIZE variable.
CDP_PURE_EVENT_CACHE_SIZE	100	Any positive integer greater than or equal to 1.	Maximum number of events to cache for a log file data source that is configured to process new records, for the Windows Event Log attribute group. And also for JMX monitors and notifications. Each new record in the log causes an event to be sent. This environment variable defines how many events are remembered in a cache by the agent. The cached values are returned when the attribute group is queried.
CDP_DP_ACTION_TIMEOUT	20 seconds	Any positive integer greater than or equal to 1.	The number of seconds to wait for a Take Action that is being handled by the agent to complete.

Table 3. Environment variable descriptions including their default values and valid value ranges (continued)

Environment variable	Default value	Valid values	Description
CDP_DP_SCRIPT_TIMEOUT	30 seconds	Any positive integer greater than or equal to 10.	The number of seconds to wait for the program started by a script-based attribute group to complete.
CDP_DP_PING_TIMEOUT	30 seconds	Any positive integer greater than or equal to 10.	The number of seconds to wait for the program started by a command return code to complete. **Note:** This variable is not related to the ICMP ping data provider.
CDP_SNMP_MAX_RETRIES	2	Any positive integer	The number of times to try sending the SNMP request again. The total number of requests that are sent to the SNMP agent is this value plus one if no responses are received.
CDP_SNMP_RESPONSE_TIMEOUT	2 seconds	Any positive integer	The number of seconds to wait for each SNMP request to timeout. Each row in an attribute group is a separate request. This timeout value is the number of seconds to wait for a response before you try again. The total timeout for a single row of data is (CDP_SNMP_MAX_RETRIES + 1) * CDP_SNMP_RESPONSE_TIMEOUT. The total default timeout value is (2+1) * 2 = 6 seconds.
CDP_DP_HOSTNAME	Name of the first installed network interface	An IP address or host name	Sets the preferred host name (network interface) on a multiple interface system. Use this environment variable if the agent binds its listening ports to a non-default network interface address. Used by the SNMP data provider. For Socket data sources, this variable applies if CDP_DP_ALLOW_REMOTE is also set.
CDP_SNMP_ALLOW_DECREASING_OIDS	NO	YES, NO	If set to YES, the SNMP data providers do not check whether returned OIDs are increasing. Set to YES with caution because the monitored agent might have problems that this check would normally catch.
KUMP_DP_COPY_MODE_SAMPLE_INTERVAL	60	Wait time in seconds	For a log file data provider, specifies how long to wait before it rereads the contents of a file when the agent is defined to **Process all records when the file is sampled**. The time is specified in seconds.

Table 3. Environment variable descriptions including their default values and valid value ranges (continued)

Environment variable	Default value	Valid values	Description
KUMP_MAXPROCESS	100%	5-100%	For a log file data provider, specifies the maximum processor usage to use to process file data. Values range from 5 to 100 percent. The default is 100 percent.
KUMP_DP_SAMPLE_FACTOR	5	Any non-negative integer	For a log file data provider, sets the sampling factor when you select **Process all records when the file is sampled** on the Agent Builder. This wait time ensures that patterns that span multiple records are written before scans are logged for the pattern.
KUMP_DP_EVENT	5	Any non-negative integer	For a log file data provider, sets the sampling frequency for Event data, in seconds.
KUMP_DP_FILE_EXIST_WAIT	YES	YES, NO	For a log file data provider, specifies that the file monitoring thread continues to run if it detects that the monitored file is absent or empty. The thread waits for the file to exist, rechecks every few seconds, and starts or restarts monitoring when the file becomes available.
KUMP_DP_FILE_SWITCH_ CHECK_INTERVAL	600	Any non-negative integer	The frequency in seconds that the log file Data Provider searches for a different monitoring file to switch to when dynamic file name support is enabled.
KUMP_DP_FILE_ROW_ PAUSE_INCREMENT	None	Any non-negative integer	For a log file data provider, specifies how many file records are read before the file monitoring thread pauses. The pause is so that previous updates can be processed. Use this environment variable only if the monitored file receives high-volume bursts of new records and you are concerned that some record updates might be lost.
CDP_COLLECTION_TIMEOUT	60 seconds	Any positive integer	The number of seconds that the agent waits for a response from a data collector that was started in another process. JMX, JDBC, HTTP, and SOAP data collectors are examples.
CDP_SSH_TEMP_DIRECTORY	. (period)	Any valid path string on the remote system	For an SSH enabled Script data provider, specifies a location on the remote system. The script files that are provided with the agent are to be uploaded to this location. A relative lo\cation is relative to the user's home directory. The default of . (period) denotes the user's home directory.

Table 3. Environment variable descriptions including their default values and valid value ranges (continued)

Environment variable	Default value	Valid values	Description
CDP_SSH_DEL_COMMAND	rm -Rf	Any valid delete command string on the remote system	For an SSH enabled Script data provider, specifies the command to start to delete the uploaded script files that are provided with the agent.
CDP_SNMP_SEND_DELAY_ FACTOR	0 milliseconds	Any positive integer	The initial SNMP send is delayed from 0 to the number of milliseconds specified. This variable is only enabled if the thread pool is also enabled. The delay does not apply to all sends, only to the first send made by an attribute group. This variable is useful if the device that is being monitored can sometimes fail to respond correctly if it receives multiple requests at the same time.
CDP_ICMP_PING_REFRESH_ INTERVAL	60 seconds	Any integer greater than or equal to 1	The systems in a device list file are pinged at this interval. If the pings use too much time, there is always a delay of at least CDP_PING_MIN_INTERVAL_DELAY seconds before the pings begin again. Data is refreshed no more frequently than this setting. Data can be refreshed less frequently based on the number of entries in the device list file and the time it takes to receive responses.
CDP_ICMP_PING_MIN_ INTERVAL_DELAY	30 seconds	Any integer greater than or equal to 1 and less than the CDP Ping refresh interval	After the devices in a device list file are pinged, the next ping refresh interval does not begin until at least this number of seconds elapses.
CDP_ICMP_PING_BURST	10	Any integer greater than or equal to 0	The number of pings that are sent before the agents pauses for the amount of time that is specified by the CDP_ICMP_PING_BURST_DELAY variable. A value of 0 disables this function.
CDP_ICMP_PING_BURST_DELAY	10	Any integer greater than or equal to 0	The amount of time in milliseconds to wait after a set number of pings are sent as defined by the CDP_ICMP_PING_BURST variable. A value of 0 disables this function.

Table 3. Environment variable descriptions including their default values and valid value ranges (continued)

Environment variable	Default value	Valid values	Description
CDP_ICMP_PING_TIMEOUT	2000 milliseconds	Any integer greater than or equal to 1	The number of milliseconds to wait for a ping response. This setting applies to each ping attempt that is made. Ping attempts are made 3 times for each host. If no response is received from any of the 3 attempts, the total time waited for a reply is CDP_ICMP_PING_TIMEOUT multiplied by 3. By default, this value is 6000 milliseconds. Changing the value for CDP_ICMP_PING_TIMEOUT causes the default TIMEOUT enumeration for the Current Response Time attribute to no longer apply. Change the TIMEOUT enumeration to the new value of CDP_ICMP_PING_TIMEOUT multiplied by 3.
CDP_JDBC_CONNECTIONLESS	false	true, false	If set to true, JDBC connections are closed after each data collection attempt. That is, all attribute groups attempt to create their own connection each time data is collected. Connections are not reused if this variable is enabled. If set to false, one connection to the database is made and that connection is shared among the attribute groups.
CDP_SSH_EXCLUDED_ ENVIRONMENT_VARIABLES	None	A comma-separated list of environment variable names	For an SSH enabled Script data provider, specifies the set of local environment variables that must not be set in the environment of the remote system.

Table 3. Environment variable descriptions including their default values and valid value ranges (continued)

Environment variable	Default value	Valid values	Description
CDP_DP_EVENT_LOG_MAX_BACKLOG_TIME	0 seconds	0, 1, or any integer greater than 1	If set to 0, and CDP_DP_EVENT_LOG_MAX_BACKLOG_EVENTS is not set to 1 or a greater integer, does not process events that are generated while the agent is shut down. 0 is the default. If set to 1, and CDP_DP_EVENT_LOG_MAX_BACKLOG_EVENTS is not set to an integer greater than 1, processes all events that are generated while the agent is shut down. If set greater than 1, and CDP_DP_EVENT_LOG_MAX_BACKLOG_EVENTS is not set greater than 1, processes events that are generated within that value in seconds of the current computer time. For example, if the value is set to 300, at startup, the agent processes all events that are generated within 300 seconds of the current time. Where a value greater than 1 is entered for both CDP_DP_EVENT_LOG_MAX_BACKLOG_TIME and CDP_DP_EVENT_LOG_MAX_BACKLOG_EVENTS variables, either that time interval of events or that number of events is processed. Which variable is chosen depends on which is matched first.
CDP_DP_EVENT_LOG_*Windows_Event_Log*_MAX_BACKLOG_TIME	0 seconds (Do not process missed events while the agent is shut down)	0, 1, or any integer greater than 1	If set to

Table 3. Environment variable descriptions including their default values and valid value ranges (continued)

Environment variable	Default value	Valid values	Description
CDP_DP_EVENT_LOG_ MAX_BACKLOG_EVENTS	0 events	0, 1, or any integer greater than 1	If set to 0, and CDP_DP_EVENT_LOG_MAX_BACKLOG _TIME variable is not set to 1 or a greater integer, does not process events that are generated while the agent is shut down. 0 is the default. If set to 1, and the CDP_DP_EVENT_LOG_MAX_BACKLOG _TIME variable is not set to an integer greater than 1, processes all events that are generated while the agent is shut down. If set greater than 1, and CDP_DP_EVENT_LOG_MAX_BACKLOG _TIME is not set greater than 1, processes at most that number of events that are generated while the agent is shut down. For example, if the value is set to 200, then at startup of the agent the 200 events that generated directly before startup are processed. Where a value greater than 1 is entered for both CDP_DP_EVENT_LOG_MAX_BACKLOG _EVENTS and CDP_DP_EVENT_LOG_MAX_BACKLOG _TIME, either that number of events or that time interval of events is processed. Which variable is chosen depends on which is matched first.
CDP_DP_EVENT_LOG_ *Windows_Event_Log*_MAX_BACK LOG_ EVENTS	0 events (Do not process missed events while the agent is shut down)	0 or any integer greater than or equal to 1	If set to
CDP_HTTP_READ_TIMEOUT	10	Any positive integer	The number of seconds to wait for a reply to the HTTP request.
CDP_JAT_THREAD_POOL_SIZE	15	Any positive integer	The number of threads that are used by the Java providers for handling data collection requests. JMX, JDBC, HTTP, and SOAP data providers are the providers that can benefit from this thread pool.
CDP_HTML_OBJECTS_THREAD_ POOL_SIZE	10	Any positive integer	The number of threads that are used to download page objects that are found in URLs monitored with the HTTP data provider.

Table 3. Environment variable descriptions including their default values and valid value ranges (continued)			
Environment variable	**Default value**	**Valid values**	**Description**
CDP_HTTP_SOAP_MAX_ROWS	500	Any positive integer	The maximum number of rows that are returned by the HTTP SOAP data provider.
CDP_DP_ALLOW_REMOTE	NO	NO, YES	If set to Yes, the agent allows remote socket connections. If set to No, the agent allows only socket connections from the local host. No is the default.
CDP_DP_INITIAL_COLLECTION_DELAY	varies	Any positive integer	The number of seconds, after the agent starts, until the thread pool begins its scheduled data collections.

Watchdog information

Use the **Watchdog Information** page to specify configuration information for the Agent Watchdog.

About this task

To open the **Watchdog Information** page, click **Watchdog Information** in the **Agent Content** section of the **Agent Information** page. You can also select the **Watchdog Information** node in the Outline View.

You can specify the following configuration information for the Agent Watchdog:

- **Monitor this agent by default**

  Select this check box to put the agent under management by Agent Management Services when the agent is installed. The agent is monitored for unhealthy behavior or abnormal termination and is restarted by a watchdog.

- **Check frequency (seconds)**

  How often the watchdog checks the agent process for unhealthy behavior or abnormal termination. The default is every 180 seconds.

- **Maximum number of restarts**

  Number of times the Watchdog restarts the agent because of unhealthy behavior or abnormal termination in a 24-hour period before it alerts the administrator of the problem. The period starts at midnight each day. So, the first period from when the agent is started might be "short."

  A restart occurs if the agent goes down for any reason. The Watchdog also stops and restarts the agent if the agent becomes unresponsive or unhealthy, for example. if the memory threshold is crossed. The default is four restarts in a 24-hour period, where the period is measured from midnight to 11:59 p.m. At midnight, the daily restart count for the agent returns to 0 automatically.

- **Memory Threshold Information**

  Size of the agent process (in megabytes) to which the agent can grow before its watchdog deems it unhealthy. There is a separate value for Windows, Linux, and UNIX. If the agent process grows beyond the threshold, the watchdog stops the process and restarts it. There are no defaults for these properties. If no value is specified, the Watchdog does not monitor the process size. The metric uses the working set size on Windows, and the user memory on UNIX and Linux.

If the Watchdog stops the agent, and the maximum number of restarts is reached, the Watchdog sends an alert that the agent exceeded its restart count, and stops doing auto-restarts. The Watchdog still reports whether the agent is up or down assuming it is started in another manner such as through the Tivoli Enterprise Portal.

You must manually restart the agent by using the AMS Start Agent Take Action command so the restart count does not get reset.

The count gets reset in one of the following ways (the Watchdog continues to work and report status, but does not do auto-restarts):

- The clock strikes midnight.
- The user uses the AMS Start Agent Take Action command, which has an input parameter called **resetRestartCount**. If you enter a value of 1 (meaning "true"or "yes"), the daily restart count resets back to 0.

For more information, see the following sections in the *IBM Tivoli Monitoring Administrator's Guide*:

- For Tivoli System Monitor Agents

  Configuring Agent Management Services on Tivoli System Monitor Agents
- For Tivoli Enterprise Monitoring Agents

  Installing and configuring Tivoli Agent Management Services

Cognos information

Use the **Cognos Information** page to specify the information that is used when a Cognos data model is generated for your agent. This information is used only for the IBM Tivoli Monitoring environment.

Procedure

1. To open the **Cognos Information** page, click **Cognos Information** in the **Agent Content** section of the **Agent Information** page or the **Cognos Information** node in the Outline View.
2. In the **Data Source** field, enter the name of the data source that connects Tivoli Common Reporting to the IBM Tivoli Data Warehouse.

   The default value is `TDW`.
3. In the **Schema** field, enter the name of the database schema that is used for the Tivoli Data Warehouse, which is used to fully qualify table names in Cognos reports.

   The default value is `ITMUSER`. This value can be changed in Framework Manager when the generated Cognos model is loaded into Framework Manager.

   The **Add this attribute group to a reporting category** check box in the **Data Source Definition** page determines where in the Cognos model the attribute group is placed. If not selected, the attribute group is placed in the extended attributes folder in the Cognos model. If selected, the attribute group is placed in the selected subfolder (availability or performance) in the Key Metrics folder. For more information about the data source fields, see Table 4 on page 35.

What to do next

You can use the Cognos data model to create Tivoli Common Reporting reports for your agent, see Appendix E, “Cognos data model generation,” on page 335.

Generate Agent wizard link

When you finish creating or editing the new agent, use the Generate Agent wizard to prepare the installation.

Procedure

- When you finish creating or editing the new agent, on the **Agent Editor Agent Information** page, click the **Generate Agent Wizard** link.

  With the Generate Agent wizard, you can:

  – Generate the agent files with a Tivoli Monitoring installation on the local system. For instructions, see “Installing an agent locally” on page 239.

– Create a package so the agent can be installed on other systems. For instructions, see "Creating the agent package" on page 241.

The Data Source Definition page

Use the **Data Source Definition** page to manipulate data sources.

About this task

The **Data Source Definition** page lists the data sources that are configured for the agent. When you select a data source or attribute in the tree, the page is updated to display the properties for the selected object. Use the fields to modify the properties for the data source or attribute selected.

Note: For detailed instructions about creating data sources from various data providers, see Chapter 6, "Defining and testing data sources," on page 63.

Procedure

- To open the **Data Source Definition** page, click **Data Sources** in the **Agent Content** section of the **Agent Information** page or the **Data Sources** node in the **Outline** view.
- You can add more data sources by clicking **Add to Selected** or right-clicking in the navigation tree and selecting one of the options.
- You can remove data sources and attributes by right-clicking on them and selecting **Remove**.
- You can add, modify, and remove attributes. For instructions, see Chapter 5, "Editing data source and attribute properties," on page 35

Copying data sources by using the Data Source Definition page

Use the **Data Source Definition** page to copy data sources.

Before you begin

Go to the **Data Source Definition** page. For more information, see "The Data Source Definition page" on page 31

About this task

Data sources that result in attribute groups can be copied to the clipboard and pasted back to this agent or another agent. Data sources that do not result in attribute groups are Availability and Windows Event Log data sources.

Procedure

1. Select the attribute groups that you want to copy.
2. Cut or copy the attribute group by using one of the following methods:
   - Click **Edit** > **Cut** > **Edit** > **Copy** from the menu bar.
   - Right-click one of the selected items and click **Cut** or **Copy** from the menu.
   - Use one of the operating system or Eclipse key strokes that calls the cut or copy action. For example, on Windows systems, pressing **Ctrl-C** calls the copy action.

   To remove data sources from their existing location and place them in the clipboard, use **Cut**. To leave data sources in place and copy them to the clipboard, use **Copy**.
3. Select the parent of an attribute group (the agent, a subnode, or a navigator group) or select an existing attribute group.
4. Paste the selection by using one of the following choices:
   - Select **Edit** > **Paste** from the menu bar.

- Right-click the node where you want to paste the selection in the tree, and click **Paste** on the menu.
- Use one of the operating system or Eclipse key strokes that calls the paste action. For example, on Windows systems, pressing **Ctrl-V** calls the paste action.

Results

The attribute groups from the clipboard are placed in the selected parent. Alternatively, if an attribute group is selected, the attribute groups are placed in the parent of the selected attribute group.

If there is a name conflict with another attribute group while pasting, the pasted attribute group name is changed slightly to avoid the conflict.

Runtime Configuration Information page

The **Runtime Configuration Information** page displays the configurable variables in the agent. You can set values for the variables when you install the agent on a monitored host.

These values are made available to command return codes and scripts through the environment. To open the **Runtime Configuration Information** page, click **Runtime Configuration** in the **Agent Content** section of the **Agent Information** page or the **Runtime Configuration** node in the Outline View. The Agent Builder automatically constructs the name of the environment variable from the product code and the label.

You can add and change the configuration properties and provide default values by using the **Runtime Configuration Information** page.

Agent XML Editor page

The **Agent XML Editor** page displays the XML for the agent definition.

The agent definition XML includes the information that is displayed in all other parts of Agent Builder. If you change the XML, the information displayed in Agent Builder reflects the change.

Attention: Do not make any changes in the XML. Such changes can cause errors that might prevent you from generating the agent or negatively affect the functioning of the agent.

Saving your edits and changes

Changes that you make with the editor are not stored until you save them.

Procedure

- Perform a save in one of the following ways:
  - Select **File** > **Save**, selecting the save (diskette) icon.
  - Press **Ctrl+S**

  When you save, a validation occurs to ensure that the information is complete. If problems occur, information about the error is displayed in the Eclipse **Problems** view. If this view is not visible, select **Window** > **Show View** > **Problems**. If you attempt to generate an agent that has errors, an error message is displayed.

  Note: You must correct all errors and save the changes before you can generate and install the agent.

Committing a version of the agent

Commit your agent when you are certain you are finished developing this version of the agent and you are ready to deliver it.

About this task

IBM Tivoli Monitoring systems require that new versions of an agent include all of the information that is contained in the previous versions of that agent that were used in the monitoring environment. Including all information from previous versions is necessary so workspaces, situations, and queries continue to work if the new agent is installed on some monitored hosts, but the old one remains on the others.

After you complete developing and testing an agent, you must commit the agent as the final version for a certain version number. Agent Builder ensures that no information is removed after you commit the agent. Subsequent builds of the agent have a new version number.

There is a limit of 1024 versions.

Remember: If you make changes to an agent that is to be tested and run in an IBM Cloud Application Performance Management environment, you must change the agent version.

Procedure

1. Open the **Agent Editor** window, **Agent Information** page.
2. In the **Commit Agent Version** area, click **commit this level**.
3. Back up the committed agent or check it into your version control system.

What to do next

After you commit an agent, any additional changes to the agent are part of a new version. You must enter the new version number before the additional changes can be saved. Any changes to the new version must not break compatibility with previous versions of the agent.

After you commit the agent, you cannot complete these actions on objects that existed before the agent was committed:

- Delete attributes from an attribute group.
- Delete attribute groups.
- Reorder existing attributes in an attribute group.
- Reorganize existing attribute groups (by using Navigator items).
- Move attribute groups or navigator groups into or out of subnodes.
- Rename attribute groups.
- Rename attributes.
- Change data types of existing attributes.
- Change a subnode name or type if it contains an attribute group that existed before the agent was committed.
- Change a company identifier or agent identifier for the agent.
- Change the product code of the agent. For more information, see (“Changing the product code” on page 34).

You can complete the following actions after you commit the agent:

- Add new attributes to existing attribute groups.
- Add new attribute groups.
- Reorder new attributes.
- Organize new attribute groups by using navigator items.

- Create new subnode types.
- Add new queries.
- Add new situations.
- Add new workspaces.

Setting a new version number for your agent

To save changes to a committed agent, you must enter a new version number.

Procedure

1. Open the **Agent Editor** window, **Agent Information** page.
2. Enter a version, fix path, or patch level that is higher that current level after the Version prompt.
3. Make the edits your agent.

   Tip: If you commit an agent and forget to change the agent version, you are prompted for the new version when you save any of your changes.

Changing the product code

If you change the product code, you have an agent that is incompatible with any previous version of the agent. Any records of previous commit actions are lost and you are developing a new agent.

Any files, situations, Take Action commands, or workspaces that you exported from IBM Tivoli Monitoring and imported into the agent are deleted from the agent.

If you try to change the product code of an agent that was committed, Agent Builder displays a warning and asks if you want to continue.

When you click **Yes** in the **Agent Product Code** window you are warned that the contents of the agent support files are no longer valid. You are also warned that the files will be removed next time the agent is saved.

Chapter 5. Editing data source and attribute properties

When you add data sources to your agent, Agent Builder creates corresponding data sets. You can edit the data sets and attributes in them to provide the necessary monitoring information.

Procedure

To edit or remove information from a data set (attribute group):

1. In the **Agent Content** area of the **Agent Information** page click **Data Sources**.

   The **Data Source Definition** page opens.
2. Select the data set (attribute group).

   The attribute group information area of the page is updated to display the properties for the selected data set.

   Note: Alternatively, if you are on the last page of the **Agent** wizard, you can double-click the data source to open the **Attribute Group Information** window. This window has the same information as the attribute group information area of the **Data Source Definition** page.

   (Table 4 on page 35) describes the field information that is applicable to all of the data sources. Use the fields to modify the properties for the data source or attribute selected.

Table 4. Fields for editing data sources

Field name	Description	Acceptable values and examples
Attribute group name	Name of the data source as it is displayed in the Tivoli Enterprise Portal or in the IBM Cloud Application Performance Management console	Acceptable values: Descriptive string less than or equal to 32 characters long. It must be unique within the agent. The first character must be a letter and remaining characters can be letters, numbers, or underscores. An underscore is displayed as a space. Do not use spaces or special characters.
Help text	Help text for the data source	Acceptable values: String up 256 characters long.
Produces a single data row	The data source returns 1 row of data. Editable in all sampled data sources.	Example: If you are monitoring physical system memory, choose a single row. A system typically manages all of its memory in a single pool; so only one row of data can be returned.

© Copyright IBM Corp. 2010, 2021

Table 4. Fields for editing data sources (continued)		
Field name	**Description**	**Acceptable values and examples**
Can produce more than one data row	The data source can return any number of rows of data. Editable in all sampled data sources.	Example: If you are monitoring disk drives, choose multiple rows because there can be more than one disk in a system. For keys, choose the attributes that distinguish a disk from another. For a disk, the key attribute is disk number, drive letter, volume label, or whatever is appropriate in your environment.
Produces Events	The data source returns event-based data, 1 row of data per event.	Example: An SNMP event-based data source sends notifications (traps) as performance thresholds are crossed. **Note:** Not all data sources can produce events.
Add this attribute group to a reporting category	The category in the generated Cognos model to which the attributes in this attribute group are assigned.	Select the check box to place the attribute group in the selected subfolder (Availability or Performance) in the Key Metrics folder. If the check box is not selected, the attribute group is placed in the **Extended Metrics** folder in the Cognos data model.
Metric Category	The category to which the attributes in this attribute group are assigned.	Select either **Performance** or **Availability**.

Note:

a. The **Produce a single data row** and **Can produce more than one data row** fields do not affect data for an event data source.

b. For more about sampled and event data types, see (“Data types” on page 57).

c. For information about the fields for a specific data source, see the relevant data provider information in Chapter 6, “Defining and testing data sources,” on page 63.

Creating, modifying, and deleting attributes

You can create, modify, or delete attributes in a data set (attribute group).

To work with attributes, open the **Data Source Definition** page. For more information, see “The Data Source Definition page” on page 31.

Creating attributes

You can add new attributes to a data set.

Procedure

1. Right-click the data source and select **Add Attribute** on the menu.
   The **Attribute Information** page is displayed.
   Note: The page that is displayed depends on the data source for the attribute.
2. Specify your choices for the new attribute on the **Attribute Information** page.
   See "Fields and options for defining attributes" on page 40 for information about the fields and options.
3. To add more attributes, select **Add additional attributes** and click **Next**.
4. When finished adding attributes, click **Finish**.

Copying attributes

You can copy attributes from the **Data Source Definition** page.

Procedure

1. In the Agent Editor, **Data Source Definition** page, right-click the attribute that you want to copy, and click **Copy Attribute**.
2. In the **Copy Attribute** window, type the name of the new attribute in the **Name** field, and click **OK**.

Editing attributes

You can edit and change attribute information by using the **Data Source Definition** page.

Procedure

1. Select the attribute that you want to edit.
   The **Attribute Information** pane of the page is updated to show the properties for the selected attribute.
2. Specify your choices for the new attribute information.
   Note: On the last page of the **Agent** wizard (the **Data Source Definition** page), you can double-click the attribute to open the **Attribute Information** window. That window contains the same information as the Attribute Information pane of the **Data Source Definition** page.

Creating derived attributes

You can create an attribute that derives its value from other attributes instead of directly from the data source.

About this task

In the derived attribute, you can perform operations on the values of the source attributes. For example, you can perform basic arithmetic operations on numeric attributes or string concatenation on string attributes.

The basic expression syntax that is used for derived expressions contains functions. These functions provide a more complicated manipulation of data that includes short-term aggregation, conversion from string to integer, and accessing configuration properties and environment variables. In addition, an editor helps you visualize the expression as it is being built.

Procedure

1. On the **Data Source Definition** page, right-click the data source and click **Add Attribute**.
2. On the **Attribute Information** page, type an Attribute name and Help text.
3. Select **Derived from other attribute values**.
4. In the **Formula** field, type the formula text or click **Edit** to enter the formula with a graphical editor.

   See “Formula operators and functions” on page 50 for information about the operators and functions that can be used in the formula.

   Note: When you click **Edit**, the Formula Editor opens. See “Editing derived attributes” on page 39 for information about editing derived attributes.
5. Optional: Select or clear the **Interval specific calculations** check box to determine which two attribute sample values are used when the function is calculated.

   Use this option when your formula uses the `rate` or `delta` functions. For more information about **Interval specific calculations**, see “Interval specific calculations” on page 38. For more information about `rate` and `delta` functions, see “Formula operators and functions” on page 50.
6. In the **Attribute type** area, click the type of attribute.
7. Click **OK**.

   The **Data Source Definition** page is displayed again with the data source listed in it as before.
8. Click **Finish**.

   Important: If you create a derived attribute that references another derived attribute, ensure that the referenced attribute is listed earlier than the new attribute. If an attribute references another derived attribute that is located later in the list, the agent is unable to display the value for this attribute. If you create such an attribute, Agent Builder displays a warning.

Interval specific calculations

You can choose **Interval specific calculations** when you define a derived attribute that is based on the `rate` or `delta` functions.

You select **Interval specific calculations** on the **Derived Attribute Details** tab of the **Attribute Information** page. For more information, see “Creating derived attributes” on page 37.

When you use the **Interval specific calculations** selection, it is important to understand the concept of a delta or difference between attribute values. The delta is the difference between the most recent value of the attribute and a previous value of the attribute. The delta is returned directly by the `delta` function and is used by the `rate` function to calculate a result.

The `delta` or `rate` function must always have the `last` function as its only argument. The `last` function specifies which values of an attribute are used to determine the delta. If **Interval specific calculations** is not selected, the previous value that is used is always the second-most-recent value. If **Interval specific calculations** is selected, the previous value that is used is the value whose age (relative to the most recent value) is equal to the collection interval of the requester.

For example, suppose `CDP_DP_REFRESH_INTERVAL` is set to 120 seconds and `attribute A` has the following sampled values:

Time	Sampled value
current	`2800`
2 minutes (120 seconds) ago	`2600`
4 minutes (240 seconds) ago	`2499`
6 minutes (360 seconds) ago	`1500`
8 minutes (480 seconds) ago	`1200`
10 minutes (600 seconds) ago	`1000`

When **Interval specific calculations** is not selected, the `delta` function always returns `200`, the difference between the two most recent values, `2800 - 2600`. The same value is returned whether the value is displayed on the Tivoli Enterprise Portal or in the IBM Cloud Application Performance Management console, used in a situation, or a historical collection.

When **Interval specific calculations** is selected, the `delta` function returns a value that depends on the collection interval of the requester.

If a derived attribute with the `delta` function is used in a situation with a 4-minute collection interval, the value that is returned by the `delta` function is `301`, the difference between the most recent value and the value obtained 4 minutes before that, `2800 - 2499`.

If a derived attribute with the `rate` function is used in a situation with a 10-minute (600-second) collection interval, the value that is returned by the `rate` function is 3, the difference between the most recent value and the value obtained 10 minutes before that, divided by the number of seconds in the interval `(2800 - 1000) / 600`.

Note: The Tivoli Enterprise Portal has no inherent collection interval, so delta and rate calculations for Tivoli Enterprise Portal requests always use the most recent and second most recent attribute values, the same result whether **Interval specific calculations** is selected or not.

For `delta` or `rate` to work correctly with **Interval specific calculations**,

- The agent must collect data periodically in the background, and not on demand (`CDP_DP_THREAD_POOL_SIZE` must be greater than 0).
- Every situation or historical collection interval in which the attribute is used must be a multiple of the background refresh interval (`CDP_DP_REFRESH_INTERVAL`).
- The count (the second argument of the last function) must be large enough to accommodate the largest collection interval from a situation or historical collection. For example, if the agent must support 10-minute (600 second) historical collection and `CDP_DP_REFRESH_INTERVAL` is 120 seconds, the count must be at least `6, 1+(600 / 120)`. A count value of 6 ensures that the `last` function returns the newest sample and samples up to 600 seconds old.

Note: If these conditions are not met, input values are likely invalid and a result of `0` is returned.

Editing derived attributes

Use the Formula Editor to edit derived attributes.

The Formula Editor is available on the **Attribute Information** page for a derived attribute, as described in “Creating derived attributes” on page 37. For more information about the Formula Editor, see “Formula Editor” on page 45

Removing attributes

You can remove one or several attributes from a data set using the **Data Source Definition** page.

Procedure

- To remove an attribute or attributes, right-click the attribute or attributes and select **Remove** from the menu that is displayed.

  Note: You cannot remove an attribute that is used by a derived attribute. You must first remove the reference by the derived attribute to the attribute you are removing.

Fields and options for defining attributes

Description of the field information and options for the **Attribute Information** page that are applicable to all of the data sources

For information about the specific field information for each of the data sources, see the relevant documentation for each data source.

Table 5. Fields and options for defining attributes

Field names/options	Description	Acceptable values
Attribute name	Name of the attribute as it is displayed in the Tivoli Enterprise Portal or in the IBM Cloud Application Performance Management console	String with the following characters: • A-Z • _ • a-z • 0-9 **Note:** The name must start with A-Z or a-z. The attribute name has a limit of 63 characters and the attribute group name has a limit of 63 characters
Help text	Help text for the attribute	String
Hidden - can only be used in derived attribute	If selected, the attribute is not displayed in the Tivoli Enterprise Portal or in the IBM Cloud Application Performance Management console. See note in the last row.	Not applicable
Derived from other attribute values	Attribute value is to be calculated from values of other attributes	Not applicable
Key Attribute	Attribute is a key in the table. Check whether this attribute helps to uniquely define the object that is being reported on. If the data is warehoused and summarized, the key attributes are used to roll up data in the summary tables.	This option is not available for Perfmon attributes.
Attribute Information pane	The contents of this tab depend on the type of data source to which this attribute belongs. See information in the chapter for the data source you want to monitor for more details. For a derived attribute, In the **Formula** field, enter a formula to calculate the value of the attribute that is based on other attributes or constants. You can type the formula in the **Formula** field or click **Edit** to use the graphical formula editor. See (“Formula Editor” on page 45).	

Table 5. Fields and options for defining attributes (continued)		
Field names/options	**Description**	**Acceptable values**
Attribute type	Describes how the attribute is displayed in the Tivoli Enterprise Portal or in the IBM Cloud Application Performance Management console. There are 3 types: • String • Numeric • Time stamp "Attribute types" on page 41 contains more information about the attribute types.	Table 6 on page 42 contains descriptions of the numeric attribute type values.
Enumerations	Can be a numeric with scale zero or string value.	Add your enumerations to the table by using the procedure in ("Specifying an enumeration for an attribute" on page 44). The enumeration name is displayed in the Tivoli Enterprise Portal or in the IBM Cloud Application Performance Management console when the corresponding Value is received in the attribute from the agent. This attribute is used for a set of specific values with identified meanings (for example, 1=UP, 2=DOWN).
Note: In cases where the attribute is used in calculations with other attributes, there are reasons not to display the base value. For instance, a number that represents a byte count wraps so quickly that it is of little use.		

Attribute types

There are three attribute types

The three types of attributes are:

- String
- Numeric
- Time stamp

String attributes

When you select **String**, use the **Maximum size** field to specify the maximum length of the string in bytes. The default size is 64 bytes.

A string value can contain any UTF-8 character. The maximum size is the total length of the buffer that is allocated to contain the string in bytes. Some non-ASCII UTF-8 characters take more than 1 byte, so you

must account for this space when you select a maximum size. Data aggregation in the warehouse displays the latest value that is collected during the period.

Numeric

When you specify **Numeric**, you can set a number of options. See Table 6 on page 42 for information about these options.

Time stamp

A Time stamp attribute is a string attribute with a format that conforms to the `CYYMMDDHHMMSSmmm` format (where C=1 for the 21st century). All 16 characters must be used for scripts or socket clients. When displayed in the Tivoli Enterprise Portal or in the IBM Cloud Application Performance Management console, a time stamp attribute type is displayed in the correct format for the locale.

When you use the browse feature for WMI, the Agent Builder automatically marks attributes whose CIM type is `CIM_DATETIME` as time stamps. The data provider automatically converts WMI attributes to this format.

Numeric aspects of attributes

Descriptions of the size, purpose, scale, and range aspects of attributes.

When you specify a numeric attribute, you must specify the size, purpose, scale, and range of the attribute. For more information, see (Table 6 on page 42).

Table 6. Numeric attribute options

Numeric aspects	Options and fields	Description
Size	**32 bits** **64 bits**	The value of 32-bit numbers can range from `-2147483648` to `2147483647` (roughly `-2,000,000,000` to `2,000,000,000`). The value of 64-bit numbers can range from `-9223372036854775808` to `9223372036854775807` (roughly $-9x10^{18}$ to $9x10^{18}$)

Table 6. Numeric attribute options (continued)

Numeric aspects	Options and fields	Description
Purpose	**Gauge**	Integer values where the raw values returned are larger or smaller than previous values. Negative values are supported. This type is the default type for integers. Data aggregation in the warehouse produces minimum, maximum, and average values.
	Counter	A positive integer value that contains raw values that generally increase over time. Data aggregation in the warehouse displays the total, high, low, and latest delta values. In the following example of Delta-based calculations, detailed data values in one hour are 9, 15, 12, 20, 22, and delta-based processing has the following rules: • If the current value is greater than or equal to the previous value, the output equals the previous value minus the current value • If the current value is less than the previous value, the output equals the current value • Because 15 is greater than 9, the output equals 6 • Because 12 is less than 15, the output equals 12 • Because 20 is greater than 12, the output equals 8 • Because 22 is greater than 20, the output equals 2 • The TOT_ value is 28, which is the total of outputs • The LOW_ value is 2, which is the lowest of outputs • The HI_ value is 12, which is the highest of outputs
	Property	A property of the object that does not frequently change. Data aggregation in the warehouse displays the latest value that is collected during the period.
	Delta	An integer value that represents the difference between the current value and the previous value for this attribute. Because this attribute is represented as a gauge in the warehouse, data aggregation in the warehouse produces minimum, maximum, and average values.
	Percent change	An integer value that represents the percent change between the current value and the previous value. This type is calculated as: `((new -old)*100)/old`. Because this type is represented as a gauge in the warehouse, data aggregation in the warehouse produces minimum, maximum, and average values.
	Rate of change	An integer value that represents the difference between the current value and the previous value, which is divided by the number of seconds between the samples. It converts a value (such as bytes) to the value per second (bytes per second). Because this type is represented as a gauge in the warehouse, data aggregation in the warehouse produces minimum, maximum, and average values.

Table 6. Numeric attribute options (continued)		
Numeric aspects	**Options and fields**	**Description**
Scale	**Decimal adjustment**	Scale determines how many decimal places are in the number. Each decimal place reduces the range that is mentioned earlier by a factor of 10. For example, a decimal adjustment of 2 shows two decimal places, and in a 32-bit number the allowable range becomes -21474836.48 to 21474836.47. When a non-zero decimal adjustment is specified, the number is manipulated internally as a floating point number. Therefore, the precision of large 64-bit numbers might be reduced.
Range	**Minimum** **Maximum**	Range gives the expected range of the value. If no minimum or maximum ranges are given, the maximum values that are described earlier are used. The range is used to produce a more useful initial view in some graphical Tivoli Monitoring workspace views.
Units		Unit of measurement for a numeric attribute.

Specifying an enumeration for an attribute

Specify a value enumeration by using the **Attribute Information** page.

About this task

Specifying an enumeration for an attribute involves a short procedure. When a value is encountered that has a defined enumeration, the enumeration name is displayed in the Tivoli Enterprise Portal or in the IBM Cloud Application Performance Management console instead of the value.

Procedure

1. In the **Attribute Information** page **Attribute type** area, click **Numeric**.
2. In the **Enumerations** area, click an enumeration, and click **Add**.
   The **Enumeration Definition** window is displayed.
3. Type the name and value of the enumeration in the fields in the window.
4. Click **OK**.
   You can then add more enumerations.

Specifying severity for an attribute used as a status indicator

In an IBM Cloud Application Performance Management environment, a summary dashboard must display a status. You must use an attribute to provide the status value. For this attribute, you must specify values that denote specific status severity.

About this task

The attribute that is used for status indication must be numeric. Select this attribute in the **Dashboard Setup** wizard; for instructions about using this wizard, see Chapter 12, “Preparing the agent for Cloud APM,” on page 219.

You can specify values for the attribute that correspond to the Normal, Warning, and Critical severity. Any other value denotes an "Unknown" severity status; you can also define some values as "Not defined" explicitly, and the "Unknown" status user interfaces displayed for these values.

Procedure

1. Select the attribute that you want to edit.

   The Attribute Information pane of the page is updated to show the properties for the selected attribute.
2. In the Attribute Information pane, click the **Severity** tab.
3. Select the necessary severity (Normal, Warning, Critical, and Not defined) and click **Edit**.
4. Select **Range** or **Single number**, enter the range of values or the single numeric value, and click **Ok**.
5. Optional: If you need to add another value for the same severity, for example; both 2 and 25 denote warning, click **Add**, select the severity, enter the value, and click **OK**.

Filtering attribute groups

You can create a filter to limit the data that is returned from an attribute group that returns sampled data.

Before you begin

If the attribute group exists, open the **Data Source Definition** page. For more information, see "The Data Source Definition page" on page 31.

If you want to create an attribute group, follow the steps in "Defining initial data sources" on page 15and click **Advanced** in the initial data source information page.

Procedure

1. Use one of the following steps to begin creating the filter:
   - If you are creating an attribute group, click **Advanced** in the initial data source information page.
   - If the attribute group exists, select the attribute group in the **Data Source Definition** page and click **Advanced** in the **Data Source Definition** page.
2. In the **Advanced Data Source Properties** page, enter a selection formula. The selection formula that you enter must evaluate to a Boolean result, true, or false.

   In the **Advanced Data Source Properties** page, you can click **Edit** to enter or modify the formula by using the Formula Editor. For more information about the Formula Editor, see "Formula Editor" on page 45
3. When you finish entering the filter selection formula, click **OK** until you return to the **Data Source Definition** page.

   When the filter is created, the agent uses the filter to evaluate each row of data. When the filter evaluates to *true* for a row of data, the data is sent to IBM Tivoli Monitoring or IBM Cloud Application Performance Management. When the filter evaluates to *false*, the row of data is not sent and is discarded.

What to do next

You can validate that the filter is working as intended by using the test function for the attribute group. For more information about attribute group testing, see "Attribute group testing" on page 229

Formula Editor

Use the Formula Editor to create and change formulas in Agent Builder.

The Formula Editor, which is a graphical tool, is displayed when you do one of the following tasks:

1. Creating or editing derived attributes, see "Creating derived attributes" on page 37 and "Editing derived attributes" on page 39
2. Creating Filtered Attribute groups, see "Creating a filtered attribute group" on page 182
3. Filtering data from attribute groups, see "Filtering attribute groups" on page 45

Attention:

- When you create derived attributes, the formula that you create must result in a data type that matches the type of the attribute. For example, if the derived attribute type is a number, the formula you create must evaluate to a numeric result.
- When you create filtered attribute groups or filter data from attribute groups, the formula that you create must result in a Boolean value, "true" or "false".

Note: In the following views, the Formula Editor is shown creating formulae for derived attributes. The views are identical when you use the Formula Editor with filtered attribute groups or to filter data from attribute groups. The views show the heading **Derived Formula Editor** or **Filter Formula Editor** depending on use.

When the Formula Editor is displayed, the current formula is loaded into the editor. If a formula does not exist, you can enter one by typing directly into the formula space in the **Formula Editor** window. Alternatively you can click **Insert** to begin entering a formula by using the editor menu options. The editor contains two views of the formula in the default window, and an option for a third view:

Component view (default)
The components of the edited formula are shown in the **operand** areas and **Operator** field. The operator and its two operands can be edited by using the selection menus.

Formula view (default)
The complete formula is in the formula field in the window. You can edit the formula by typing in this box.

Formula hierarchy tree view (option)
The formula hierarchy tree is displayed by selecting the **Show formula hierarchy** check box. The state of the check box is remembered in subsequent invocations of the Formula Editor.

Changing the Formula Editor component view

Change the component view in the Formula Editor.

About this task

The component that is shown in the component view can be changed in the following ways:

Procedure

- Move the cursor in the formula text.
- Select a different node in the formula hierarchy tree.
- Select **Up one Level** or one of the Edit buttons.

Component types

You can use the Formula Editor to edit the current component and any operands or function arguments of that component. Some components can appear differently in the Formula Editor when selected.

Formula Editor Attribute component

Use the attribute component in the Formula Editor to select and manipulate attributes in formulae.

About this task

You can select an attribute from a list of attributes for the attribute group in the component view of the Formula Editor.

Procedure

1. To work with a specific attribute, select that attribute from the list and click **Edit**

The **Edit the Selected Attribute** window is displayed.

2. You can manipulate the selected attribute in the following ways:
   - You can replace the attribute with a string or number by selecting **String** or **Number**. The attribute list is replaced by an entry field and the contents are no longer compared to the list of valid attribute names.
   - You can replace the attribute with a function by clicking **Function**. Parentheses are added after the name and the list now contains valid function names to choose from.
   - You can type an attribute name instead of selecting one. Typing a name is useful if you did not yet define all of the attributes in this attribute group.
     - A warning is displayed if there is no attribute with the name that was entered.
     - An error is displayed if characters are entered that cannot be part of an attribute name.
     - The **OK** button is disabled until the warning or error is corrected.
   - Attributes are not filtered based on type. If an attribute (or any value) of the wrong type is selected or entered, a warning message is displayed.

Formula Editor Literal components

Use the string and number components in the Formula Editor to manipulate literals in formulae.

About this task

A literal is any value that is entered directly in the formula that does not come from an attribute value or from a function. A literal value can be either a string or a number.

Procedure

- You can replace a literal string or number with an attribute by clicking **Attribute**. A valid attribute name must be selected or entered without quotation marks.
- You can replace a literal string or number with a function by clicking **Function**. Parentheses are added after the name and the selection list contains valid function names to choose from.
  - A warning is displayed if a number is entered where a string is expected or vice versa.
  - If **Number** is selected, an error is displayed if the content of the field is not a number. **OK** is disabled until the error is corrected.

Formula Editor Operator component

Use the operator component in the Formula Editor manipulate operators in formulae.

About this task

An operator component shows an operator and its operands.

Procedure

- In the Formula Editor component view select the operator from the **Operator** list, between the two operands. The (%) operator multiplies the first operand by 100, and then divides by the second operand.
- Select the operator (`+ - * / or %`).
  - The **Left operand** section of the page is before the operator.
  - The **Right operand** section is after the operator.
  - Simple operands (attributes and literals) can be edited without having to change the selected component to the operand as described in "Formula Editor Attribute component" on page 46 and "Formula Editor Literal components" on page 47.

- Complex operands, which consist of other operators or functions, can be edited by clicking **Edit**. This action highlights the operand component instead of the entire operator.

Formula Editor Conditional expression component

The conditional expression component shows a condition, a value to return if the condition is true, and a value to return if the condition is false.

- The expression in the **Condition** section must evaluate to true or false. Operators `(==)`, `(!=)`, `(<)`, `(<=)`, `(>)`, `(>=)`, `(&&)`, `(||)`, `(!)` are available to form expressions that return true or false.
- Simple operands (attributes and literals) can be edited without having to change the selected component to the operand as described in "Formula Editor Attribute component" on page 46 and "Formula Editor Literal components" on page 47.
- Complex operands, which consist of other operators or functions, can be edited by clicking Edit. This action highlights the operand component instead of the entire conditional expression.
- See "Formula Editor common options" on page 48 for information about using the following options: **Insert**, **Remove**, **Up one Level**, and **Edit**.

Related concepts

"Formula Editor" on page 45
Use the Formula Editor to create and change formulas in Agent Builder.

Formula Editor Function component

Use the function component in the Formula Editor to select and manipulate function components in formulae.

About this task

The function component shows the function and its arguments.

Procedure

- To work with the functions Select the **Function name** from the list in the Formula Editor.
  - The description of the selected function is shown after the function.
  - **Function argument** sections are shown after the function name. The appropriate number of arguments for the selected function are shown. A description specific to the function selected is shown.
  - Simple arguments (attributes and literals) can be edited without having to change the selected component to the operand as described in "Formula Editor Attribute component" on page 46 and "Formula Editor Literal components" on page 47.
  - Complex arguments, which consist of operators or other functions, can be edited by clicking **Edit**. This action highlights the argument component instead of the entire function.
- For functions that take a variable number of arguments, add arguments by clicking **Insert** or remove arguments by clicking **Remove** in addition to the actions described in "Formula Editor common options" on page 48.
- For the `getenv` function, a configuration property can be chosen by clicking **Insert**. If you select the Configuration property choice, the **Configuration Properties** window is displayed.

Formula Editor common options

You can use some options in all views in the Formula Editor

The Formula Editor common options are:

- **Insert**
- **Remove**

- **Up one Level**
- **Edit**

Insert

Insert inserts an operator or a function before the component. The component is demoted to one of the operator operands or one of the function arguments. For example, if you click **Insert** before the `sqrt(attr2)` function, you are asked what you want to insert and the following choices are displayed:

- **An operator with sqrt(attr2) as one of the operator's operands**
- **A function with sqrt(attr2) as the function's first argument**
- **A conditional expression with sqrt(attr2) as the true or false values**

If you click **Insert** before the `getenv` function, you are asked what you want to insert and the following choices are displayed:

- **Configuration property**: use this option to retrieve the value of a configuration property that you have set up for the agent, or else of any environment variable (for example, `JAVA_HOME`) on the host running the agent.
- **An operator with attr2 as one of the operator's operands**
- **A function with attr2 as the function's first argument**
- **A conditional expression attr2 as the true or false values**

Remove

Remove is available only for operators and functions, and is the inverse of **Insert**. When you click **Remove**, you are asked what is to replace the removed operator or function. For example, **Remove** before the `sqrt(attr2)` function shows the following choices:

- **The current argument 1, attr2**
- **A new string, number, or attribute reference**

Select **A new string, number, or attribute reference** to discard the entire tree after the point that is being removed and replace it with a new attribute or literal value.

Click **The current argument** to promote the selected operand or argument to replace the removed operator or function. You can click subsequent choices if there are more arguments or operands. Any other operands or arguments are discarded.

Up one Level

Click **Up one Level** to move up in the tree.

Edit

Click **Edit**, before a complex operand or argument, to make it the component to be edited.

Click **Up One Level** after you click **Edit** to restore the current component to what it was before you clicked **Edit**.

Formula Editor - Formula errors

Correcting formula errors in the **Formula Editor**

The component view is different when there is no formula or the entered formula cannot be parsed. It does not display a formula tree. Instead, it displays an error message.

You can correct a formula with parsing errors by typing directly in the formula field, or by replacing it with a new formula by clicking **Insert**. In this case, **Insert** presents the following choices:

- **An attribute**

- **A string**
- **A number**
- **An operator**
- **A conditional expression**
- **A function**

Related concepts

“Formula Editor” on page 45
Use the Formula Editor to create and change formulas in Agent Builder.

Formula operators and functions

A reference (including examples) of formula operators and functions that are used in the formula editor.

A derived attribute value is the result of evaluating an expression that is based on constants and other attribute values in the same data source. The expression grammar is the normal mathematical expression - operand operator operand with parentheses used for grouping. Numeric attributes can be combined with other numeric attributes or constants by using the normal mathematical operators: `+ - * /,` and %, which multiplies the **Left operand** by 100 and divides by the **Right operand**. String attributes can be combined with other string attributes or constants with +. You can also use the following described functions. Functions are entered in the format: `function_name(argument_1, argument_2, argument_3)`.

An attribute is represented by its name (the same name you see in the **Data Sources** Information tree). Integer constants are specified as numbers. String constants are surrounded by quotation marks.

You can use the following functions in a formula:

abs
Returns the absolute value of a number

atof
Converts a string to a floating point value

atoi
Converts a string to an integer value. It operates in the same way the normal **C atoi** works: it stops at the first non-decimal character.

average
Returns a single value that is the average of a set of values. The set of values comes from the arguments of the function. Several individual values can be given (for example attribute names or constants), each in a separate argument. Alternatively the last function can be the only argument to this function (to calculate the average of the most recent values of an attribute).

Examples of this function in use are:

```
average (Attr_A, AttrB, Attr_C)
```

```
average (last (Attr_A, 10))
```

ceiling
Returns the least integer that is not less than the argument.

For example, where `attribute_a = 12.4`, `ceiling(attribute_a)` returns the value 13. And, where `attribute_a = -12.4`, `ceiling(attribute_a)` returns the value -12.

delta
The difference between the most recent value of an attribute and a previously collected value of that attribute. The single argument to delta must be the `last` function, which obtains the current and previous values of an attribute. A normal use might look like:

```
delta (last(OtherAttribute, 2))
```

For more information about which attribute values from the last function are used to calculate the delta, see “Interval specific calculations” on page 38. This function is applicable only for derived attributes, not for attribute group filters.

floor
Returns the greatest integer that is not greater than the argument.

For example, where `attribute_a = 12.4`, `floor(attribute_a)` returns the value 12. And, where `attribute_a = -12.4`, `floor(attribute_a)` returns the value -13.

getenv
Returns the value of the provided environment or "configuration variable".

ipAddressToName
Converts an IP address to a host name. This function requires one argument, an IP address string in dotted decimal notation. If the address cannot be resolved, then the IP address is returned.

itoa
Converts an integer into a string. This function is most useful when you want to concatenate a numeric value onto a string. The derived string + function takes only two string arguments.

last
Returns a list of values for use by the `min`, `max`, `average`, `stddev`, `rate` and `delta` functions. It takes two arguments: the attribute to collect and the number of values to use in the calculation. If the required attribute is an integral value in a string attribute, the first argument can contain the `atoi` function, such as `atoi(numericalStringAttribute)`. The second argument must be a number. It can either be hardcoded as a constant or it can be the result of an `atoi(getenv("ENV_VAR"))` expression. It cannot reference an attribute value.

Examples of this function in use are:

```
average (last (Attr_A, 10))
```

```
last (Attribute_A, ${K01_NUM_COLLECTIONS}))
```

Restriction: You can use the `last` function only once in a specific formula.

matches
Returns a Boolean, true, or false, indicating whether a regular expression matches a value. It takes two arguments, string source and a regular expression whose result the string is compared to. This function is useful for filtering attribute groups.

max
Returns a single value that is the maximum of a set of values. The set of values comes from the arguments of the function. Several individual values can be given (for example attribute names or constants), each in a separate argument. Alternatively the last function can be the only argument to this function (to calculate the maximum of the most recent values of an attribute).

min
Returns a single value that is the minimum of a set of values. The set of values comes from the arguments of the function. Several individual values can be given (for example attribute names or constants), each in a separate argument. Alternatively the last function can be the only argument to this function (to calculate the minimum of the most recent values of an attribute).

nameToIpAddress
Converts a host name to an IP address. This function requires one argument, a host name string. If the address cannot be resolved, then the host name is returned.

NetWareTimeToTivoliTimestamp
Converts a Novell NetWare hexadecimal time value to a Tivoli Monitoring time stamp. This function requires one argument, a special NetWare hexadecimal time value. The attribute type is timestamp.

rate
The rate of change (per second) between the most recent value of an attribute and a previously collected value of that attribute. The single argument to rate must be the `last` function, which obtains the current and previous values of an attribute. A normal use might look like:

```
rate (last(OtherAttribute, 2))
```

For more information about which attribute values from the last function are used to calculate the rate, see "Interval specific calculations" on page 38. This function is applicable only for derived attributes, not for attribute group filters.

replaceFirst
Replaces the first occurrence of a substring that matches a regular expression with a replacement string. This function takes three arguments. First: the input string. Second: the regular expression which is used to match a substring in the input string. Third: the replacement string. See (Appendix F, "ICU regular expressions," on page 351) for details on the regular expressions and substitution values that are allowed in the replacement string.

replaceAll
Replaces all occurrences of substrings that match a regular expression with a replacement string. This function takes three arguments. First: the input string. Second: the regular expression which is used to match a substring in the input string. Third: the replacement string. See (Appendix F, "ICU regular expressions," on page 351) for details on the regular expressions and substitution values that are allowed in the replacement string.

round
Mathematically Rounds the number to the nearest whole number.

sqrt
Returns the square-root of a number

stddev
Returns a single value that is the standard deviation of a set of values. The set of values comes from the arguments of the function. Several individual values can be given (for example attribute names or constants), each in a separate argument. Alternatively the last function can be the only argument to this function (to calculate the standard deviation of the most recent values of an attribute).

StringToTivoliTimestamp
Converts a date and time string to a Tivoli Monitoring time stamp. This function requires two arguments. The first argument is a free-form string representation of the time stamp. The second argument is a format string that identifies how to parse the free-form string representation of a time stamp. (Table 7 on page 52) describes the valid format parameters. The attribute type is timestamp.

Table 7. Valid format parameters for StringToTivoliTimestamp

Symbol	Meaning	Format	Example
y	Year	yy yyyy	*96* *1996*
M	Month **Note:** Only English month strings are supported.	M or MM MMM MMMM	*09* *Sept* *September*
d	day	d dd	*2* *02*

Table 7. Valid format parameters for StringToTivoliTimestamp (continued)			
Symbol	**Meaning**	**Format**	**Example**
E	Day of week **Note:** Only English day-of-week strings are supported.	EE EEE EEEE	*Sa* *Sat* *Saturday*
h	Hour in AM or PM (1-12)	hh	*07*
H	Hour in day (0-23)	HH	*00*
m	Minute in hour	mm	*04*
s	Second in minute	ss	*05*
S	Millisecond	S SS SSS	*2* *24* *245*
a	AM or PM marker	a or aa	*am*
Any other ASCII character	skip this character	- (hyphen) (space) / (forward slash) : (colon) * (asterisk) , (comma)	

Table 8 on page 53provides examples of string representations of time stamps and the format strings that are used to parse them.

Table 8. StringToTivoliTimestamp examples. A table listing and explaining a few examples of string representations of time stamps.	
String representation of the time stamp	**Format string**
96.07.10 at 15:08:56	`yy.MM.dd ** HH:mm:ss`
Wed, August 10, 2010 12:08 pm	`EEE, MMMM dd, yyyy hh:mm a`
Thu 21/01/2010 14:10:33.17	`EEE dd/MM/yyyy HH:mm:ss.SS`

sum
Returns a single value that is the sum of a set of values. The set of values comes from the arguments of the function. Several individual values can be given (for example attribute names or constants), each in a separate argument. Alternatively the last function can be the only argument to this function (to calculate the sum of the most recent values of an attribute).

TivoliLogTimeToTivoliTimestamp
Converts a Tivoli log file time stamp to a Tivoli Monitoring time stamp. This function requires one argument, the string time stamp from a Tivoli log file. The attribute type is timestamp.

tokenize
One token of a tokenized string. This function requires three arguments. The first argument is a string to be split into tokens. The second argument gives one or more characters in the string that separate one token from another. Any occurrence of any of the characters from this argument is used to identify and separate tokens in the first argument. The third argument is the index of the token to return as a result of this function. The first token is index 0, the second token is index 1, and so on. This argument can also be the string LAST to return the last token.

UTCtoGMT
Converts Coordinated Universal Time to a GMT Tivoli Monitoring time stamp. This function requires one argument, the integer time_t value. The attribute type is timestamp.

UTCtoLocalTime
Converts Coordinated Universal Time to a local Tivoli Monitoring time stamp. This function requires one argument, the integer time_t value. The attribute type is timestamp.

The following functions take no arguments and return a number.

count
Keeps a counter that starts at 1 the first time it is called, and increments by 1 each subsequent time it is called. If you use it in an expression that also uses `last`, it matches the number of elements that are stored by `last()`, but only until `last()` reaches its maximum. At that point, `last()` starts deleting the oldest value for each new one, thus staying at the same number of total values, while `count()` keeps increasing forever.

cumulativeSum
Returns the sum of argument values of duplicate events that are represented by a flow control summary event. Or returns the argument if it is a single event from a data source. It takes a single numeric argument. This function applies only to event attribute groups with `event filtering` and summarization turned on.

eventThreshold
Returns the threshold value that is configured for the attribute group which generated the event. A number, with three enumerations:

- `SEND_ALL (-3)`
- `SEND_FIRST (-2)`
- `SEND_NONE (-1)`

The number in parentheses is the raw value. However, the Agent Builder defines the enumerations so by default the text version is visible on the Tivoli Enterprise Portal or in the IBM Cloud Application Performance Management console. If you specify an actual numeric threshold and not one of the three pre-defined choices, that number is returned by this function. The value is an integer > 0. This function applies only to event attribute groups with `event filtering` and summarization turned on.

isSummaryEvent
Returns 0 if it is a single event from a data source, or 1 if the event is a flow control summary event. The displayed values are Event and Summary Event if you use the default attribute for the function. If you create the attribute manually, the displayed values are 0 and 1, unless you define the names as enumerations. This function applies only to event attribute groups with `event filtering` and summarization turned on.

occurrenceCount
The number of matching events that are represented by a flow control summary event, or 1 if it is a single event from a data source. (A flow control summary event includes the first event). This function applies only to event attribute groups with `event filtering` and summarization turned on.

summaryInterval
Returns the summary interval that is configured for the attribute group which generated the event, in seconds. This function applies only to event attribute groups with `event filtering` and summarization turned on.

Examples

Examples of the use of formula operators and functions to created derived and filtered attributes

Example 1 - Derived Attributes

If you have a data source that defines the following attribute type:

Name	String
xBytes	Numeric
yBytes	Numeric
Virtual_Size	Numeric

You can define:

- An attribute `totalBytes` to be the sum of `xBytes` and `yBytes`. You enter the formula `xBytes + yBytes`.
- An attribute `yPercent` to be a percentage of the total bytes, which is `yBytes`, can be defined as `yBytes % (xBytes + yBytes)` or `yBytes % totalBytes`.

Example 2 - Derived Attributes

This formula returns the maximum of the recently collected values for the `Virtual_Size`attribute. The number of samples that are collected is the value of the configuration variable, *K4P_COLLECTIONS_PER_HISTORY_INTERVAL* (accessed through `getenv`), converted to a number (through `atoi`):

```
max(last(Virtual_Size,atoi(getenv("K4P_COLLECTIONS_PER_HISTORY_INTERVAL"))))
```

Example 3 - Derived Attributes

This formula returns the square-root of the sum of the squares of the `xBytes` and `yBytes` attribute values:

```
sqrt(xBytes * xBytes + yBtyes * yBytes)
```

Example 4 - Derived Attributes

This formula returns the average of the `xBytes` attribute from the 20 most recent samples of the attribute group. If fewer than 20 samples are collected since the agent was started, it returns the average of the `xBytes` attribute from all samples:

```
average(last(xBytes,20))
```

Example 5 - Filtered Attributes

You have a data source that returns:

```
Name    Type  Size  Used  Free
Memory  MEM      8     4     4
Disk1   DISK   300   200   100
Disk2   DISK   500   100   400
```

You are only interested in the disk usage. The solution is to create a filter to limit the data that is returned. To limit the returned data, you create a simple filter that returns a Boolean, true, or false value, as follows

Disk Filter:

```
Type=="DISK"
```

Now when the filter Type=="DISK" is true, the attribute group returns only disk usage data, for example:

```
Name    Type   Size  Used  Free
Disk1   DISK   300   200   100
Disk2   DISK   500   100   400
```

Example 6 - Filtered Attributes

You have a data source that returns:

```
Name   Size  Used  Free
Memory    8     4     4
Disk1   300   200   100
Disk2   500   100   400
```

The data that is returned is similar to the previous example, however, there is not a Type attribute present this time. Here you can use the matches function to find any data rows with a name attribute value that matches "Disk" followed by a number.

Disk Filter:

```
matches(Name, "Disk[0-9]*")
```

Now when the filter matches the string "Disk" followed by a number in attribute Name, only the disk usage data rows are returned:

```
Name   Size  Used  Free
Disk1   300   200   100
Disk2   500   100   400
```

Specifying operating systems

When you define data sources that are not available on all operating systems that the agent supports, you must specify the operating systems where the data source runs.

About this task

By default, the data source provides data on all of the operating systems that are defined at the agent level, as desrcibed in “Default operating systems” on page 18. You can change the operating systems for each data source.

Procedure

1. To open the Operating Systems section, click **Operating Systems** in the **Data Source Information** page when you add a data source.
2. Select the operating systems on which the data source is to operate.

   Select individual operating systems, all operating systems, all operating systems of a specific type, or the agent default operating systems.

Configuring and Tuning data collection

When an Agent Builder agent is created, you can configure and tune its data collection to achieve the best results.

How you configure and tune your agent can be different for different Agent Builder agents and even between attribute groups in a single agent. Agent Builder agents can include two types of data and they support two basic methods of data collection for the most common type of data.

Data types

An agent collects two types of data:

1. Most Tivoli Monitoring attribute groups represent snapshots of data. Someone asks for the data and it is returned. Agents use this type of data to represent configuration, performance, status, and other information where a one time collection of a set of data makes sense. This data is called *sampled data*.
2. Some Tivoli Monitoring data represents events. In this case, an event happens and the agent must forward data to Tivoli Monitoring. Examples of events are SNMP Traps, Windows Event Log entries, and new records that are written to a log file. For simplicity, these types of data are grouped and referred to as *event data*.

Sampled data

When sampled data is required, a request is sent to the agent for a specific attribute group. The request might be initiated by clicking a workspace in the Tivoli Enterprise Portal. Other things that might initiate a request are a situation that is running, a data collection for the Warehouse, or a SOAP request. When the agent receives the request, the agent returns the current data for that attribute group. Tivoli Enterprise Portal requests target a specific attribute group in a particular Managed System Name (MSN). Situations and historical requests are more interesting, especially in an agent which includes subnodes. When a situation needs data for an attribute group in a subnode, the agent receives one request with a list of the targeted subnodes. The agent must respond with all the data for the requested attribute group for all of the subnodes before Tivoli Monitoring can work on the next request.

The most straightforward way for an agent to satisfy a request is to collect data every time it receives a request from Tivoli Monitoring. Agent Builder agents do not collect data every time. Data is not collected every time because it often takes time or uses resources to collect data. And in many cases the same data is requested many times in a short period. For example, a user might define several situations that run at the same interval on an attribute group and the situations can signal several different conditions. Each of these situations results in a request to the agent, but you might prefer each of the situations to see the same data. It is likely that as each situation sees the same data, more consistent results are obtained, minimizing the demand for system resources by the monitoring agent.

The agent developer can configure agents to optimize data collection by choosing to run the collection in one of the following two modes:

1. **On-demand collection**: The agent collects data when it receives a request and returns that data.
2. **Scheduled collection**: The agent runs data collection in the background on scheduled intervals and returns the most recently collected data when it receives a request.

The agent uses a short-term cache in both of these modes. If another request for data is received while the cache is valid, the agent returns data from the cache without collecting new data for each request. Using data from the cache solves the problem that is caused by multiple concurrent situations (and other types of) requests. The amount of time the data remains valid, the scheduled collection interval, the number of threads that are used for collection and whether the agent runs in on demand or scheduled mode are all defined by environment variables. Using the environment variables, you can tune each agent for the best operation in its environment.

See the following examples that illustrate how the agent works in both modes:

- Agent 1 (*on-demand* collection): A simple agent that collects a small amount of data that is normally accessed only by situations or on an infrequent basis in the Tivoli Enterprise Portal. Data collection is reasonably fast, but it can use up computing and networking resources. This agent is normally defined to run on demand. If no situations are running or no one clicks the Tivoli Enterprise Portal, the agent does nothing. When data is needed, it is collected and returned. The data is placed into the short-term cache so that further requests at about the same time return the same data. This type of collection is likely the most efficient way for this agent to run because it collects data only when someone actually needs it.
- Agent 2 (*scheduled* collection): A complex agent that includes subnodes and collects data from multiple copies of the monitored resource. Many copies of the resource can be managed by one agent. It is

normal to run situations on the data on a relatively frequent basis to monitor the status and performance of the monitored resource. This agent is defined to run a *scheduled*collection. One reason for running a *scheduled* collection is the way that situations are evaluated by Tivoli Monitoring agents. Because situations are running on the attribute groups in the subnodes, the agent receives one request for the data from all of the subnodes simultaneously. The agent cannot respond to other requests until all of the data is returned for a situation. If the agent collected all of the data when the request arrived, the agent would freeze when you click one of its workspaces in theTivoli Enterprise Portal. To avoid freezing the agent, the agent builder automatically defines all subnode agents to run as scheduled collection. The agent developer tunes the number of threads and refresh interval to collect the data at a reasonable interval for the data type. For example, the refresh interval can be one time a minute, or one time every 5 minutes.

Environment variables

An agent determines which mode to use and how the scheduled data collection runs based on the values of a set of environment variables. These environment variables can be set in the definition of the agent on the **Environment Variables** panel. Each environment variable is listed in the menu along with the default values. The environment variables can also be set or modified for an installed agent by editing the agent's environment (`env`) file on Windows or initialization (`ini`) file on UNIX. The environment variables that control data collections for sampled attribute groups are:

- `CDP_DP_CACHE_TTL=<validity period for the cached data - default value 55 seconds>`
- `CDP_DP_THREAD_POOL_SIZE=<number of threads to use for concurrent collection - default value 15 for subnode agents>`
- `CDP_DP_REFRESH_INTERVAL=<number of seconds between collections - default value 60 seconds for subnode agents>`
- `CDP_DP_IMPATIENT_COLLECTOR_TIMEOUT=<amount of time to wait for new data after validity period expires - default value 5 seconds>`

The most important of these variables are `CDP_DP_CACHE_TTL, CDP_DP_REFRESH_INTERVAL`, and `CDP_DP_THREAD_POOL_SIZE`.

If `CDP_DP_THREAD_POOL_SIZE` has a value greater than or equal to 1 or the agent includes subnodes, the agent operates in *scheduled* collection mode. If `CDP_DP_THREAD_POOL_SIZE` is not set or is 0, the agent runs in *on-demand* collection mode.

If the agent is running in *scheduled* mode, then the agent automatically collects all attribute groups every `CDP_DP_REFRESH_INTERVAL` seconds. It uses a set of background threads to do the collection. The number of threads is set by using `CDP_DP_THREAD_POOL_SIZE`. The correct value for the `CDP_DP_THREAD_POOL_SIZE` varies based on what the agent is doing. For example:

- If the agent is collecting data from remote systems by using SNMP, it is best to have `CDP_DP_THREAD_POOL_SIZE` similar to the number of remote systems monitored. By setting the pool size similar to the number of monitored remote systems, the agent collects data in parallel, but limits the concurrent load on the remote systems. SNMP daemons tend to throw away requests when they get busy. Discarding requests forces the agent into a try-again mode and it ends up taking more time and more resources to collect the data.
- If the agent includes a number of attribute groups that take a long time to collect, use enough threads so that long data collections can run in parallel. You can probably add a few more for the rest of the attribute groups. Use threads in this way if the target resource can handle it. Examples of when attribute groups can take a long time to collect are if the script runs for a long time, or a JDBC query takes a long time.

Running an agent with a larger thread pool causes the agent to use more memory (primarily for the stack that is allocated for each thread). It does not however increase the processor usage of the process or increase the actual working set size of the process noticeably. The agent is more efficient with the correct thread pool size for the workload. The thread pool size can be tuned to provide the wanted behavior for a particular agent in a particular environment.

When data is collected, it is placed in the internal cache. This cache is used to satisfy further requests until new data is collected. The validity period for the cache is controlled by CDP_DP_CACHE_TTL. By default the validity period is set to 55 seconds. When an agent is running in scheduled mode, it is best to set the validity period to the same value as CDP_DP_REFRESH_INTERVAL. Set it slightly larger if data collection can take a long time. When set the validity period in this way, the data is considered valid until its next scheduled collection.

The final variable is CDP_DP_IMPATIENT_COLLECTOR_TIMEOUT. This variable comes into play only when CDP_DP_CACHE_TTL expires before new data is collected. When the cache expires before new data is collected, the agent schedules another collection for the data immediately. It then waits for this collection to complete up to CDP_DP_IMPATIENT_COLLECTOR_TIMEOUT seconds. If the new collection completes, the cache is updated and fresh data is returned. If the new collection does not complete, the existing data is returned. The agent does not clear the cache when CDP_DP_CACHE_TTL completes to prevent a problem that is seen with the Universal Agent. The Universal Agent always clears its data cache when the validity period ends. If the Universal Agent clears its data cache before the next collection completes, it has an empty cache for that attribute group and returns no data until the collection completes. Returning no data becomes a problem when situations are running. Any situation that runs after the cache cleared but before the next collection completes sees no data and any of the situations that fire are cleared. The result is floods of events that fire and clear just because data collection is a little slow. The Agent Builder agents do not cause this problem. If the 'old' data causes a situation to fire generally the same data leaves that situation in the same state. After the next collection completes, the situation gets the new data and it either fires or clears based on valid data.

Attribute groups

Agent Builder agents include two attribute groups that you can use to inspect the operation of data collection and to tune the agent for your environment. The attribute groups are Performance Object Status and Thread Pool Status. When these attribute groups are used to tune data collection performance, the most useful data is:

- Performance Object Status, Average Collection Duration attribute. This attribute shows you how long each attribute group is taking to collect data. Often a small percentage of the attribute groups in an agent represents most of the processor usage or time that is used by the agent. You might be able to optimize the collection for one or more of these attribute groups. Or you can modify the collection interval for one or more groups, if you do not need some data to be as up-to-date as other data. For more information, see (“Examples and advanced tuning” on page 60).
- Performance Object Status, Intervals Skipped attribute. This attribute shows you how many times the agent tried to schedule a new collection for the attribute group and it found that the previous collection was still on the queue, waiting to be run, or already running. In a normally behaved agent this attribute value is zero for all attribute groups. If this number starts growing, you tune the data collection, by adding threads, lengthening the interval between collections, or optimizing the collection.
- Thread Pool Status, Thread Pool Avg Active Threads attribute. You can compare this value to the Thread Pool Size attribute group to see how well your thread pool is being used. Allocating a thread pool size of 100 threads when the average number of active threads is 5 is probably just wasting memory.
- Thread Pool Status, Thread Pool Avg Job wait and Thread Pool Avg Queue Length attributes. These attributes represent the time an average data collection spends waiting on the queue to be processed by a thread and the average number of collections on the queue. Because of the way this data is collected, even an idle system indicates that at least an average of one job is waiting on the queue. A larger number of waiting jobs or a large average wait time indicates that collections are being starved. You can consider adding threads, lengthening the interval between collections or optimizing the collection for one or more attribute groups.

Event data

Agent Builder agents can expose several types of event data. Some behavior is common for all event data. The agent receives each new event as a separate row of data. When a row of event data is received, it is sent immediately to Tivoli Monitoring for processing, and added to an internal cache in the agent.

Situations and historical collection are performed by Tivoli Monitoring when each row is sent to Tivoli Monitoring. The cache is used to satisfy Tivoli Enterprise Portal or SOAP requests for the data. The agent can use the cache to perform duplicate detection, filtering, and summarization if defined for the attribute group. The size of the event cache for each attribute group is set by CDP_PURE_EVENT_CACHE_SIZE. This cache contains the most recent CDP_PURE_EVENT_CACHE_SIZE events with the most recent event returned first. There are separate caches for each event attribute group. When the cache for an attribute group fills, the oldest event is dropped from the list.

The Agent Builder agent can expose events for:

- Windows Event Log entries
- SNMP Traps or Informs
- Records added to log files
- JMX MBean notifications
- JMX monitors
- Events from a Java API provider or socket provider.
- Joined attribute groups (where one of the data sources is an event data source)

These events are handled in the most appropriate way for each of the sources. SNMP Traps and Informs, JMX notifications and events from the Java API and socket providers are received asynchronously and forwarded to Tivoli Monitoring immediately. There is no requirement tune these collectors. The agent subscribes to receive Windows Event Log entries from the operating system by using the Windows Event Log API. If the agent is using the older Event Logging API, it polls the system for new events by using the thread pool settings. For joined attribute groups where one of the data sources is an event data source, there is no tuning to apply to the joined attribute group. Though the joined attribute group does benefit from any tuning applied to the event source group.

File monitoring is more complicated. The agent must monitor the existence of the files and when new records are added to the files. The agent can be configured to monitor files by using patterns for the file name or a static name. As the set of files that matches the patterns can change over time, the agent checks for new or changed files every KUMP_DP_FILE_SWITCH_CHECK_INTERVAL seconds. This global environment variable governs all file monitoring in an agent instance. When the agent determines the appropriate files to monitor, it must determine when the files change. On Windows systems, the agent uses Operating System APIs to listen for these changes. The agent is informed when the files are updated and processes them immediately. On UNIX systems, the agent checks for file changes every KUMP_DP_EVENT seconds. This global environment variable governs all file monitoring in an agent instance. When the agent notices that a file changed, it processes all of the new data in the file and then waits for the next change.

Examples and advanced tuning

Example

Environment variables that are used for more advanced tuning are defined at the agent level. You set the following variables one time and they apply to the all of the attribute groups in the agent:

- CDP_DP_CACHE_TTL
- CDP_DP_IMPATIENT_COLLECTOR_TIMEOUT
- KUMP_DP_FILE_SWITCH_CHECK_INTERVAL
- KUMP_DP_EVENT

You can make the following variables apply to individual attribute groups. They still have a global setting that applies to all other attribute groups in the agent:

- CDP_DP_REFRESH_INTERVAL
- CDP_PURE_EVENT_CACHE_SIZE

If you defined an agent to include the following six attribute groups:

- EventDataOne
- EventDataTwo
- EventDataThree
- SampledDataOne
- SampledDataTwo
- SampledDataThree

You might set the following default variables:

- `CDP_DP_CACHE_TTL=55`
- `CDP_DP_IMPATIENT_COLLECTOR_TIMEOUT=2`
- `CDP_DP_REFRESH_INTERVAL=60`
- `CDP_PURE_EVENT_CACHE_SIZE=100`

As a result, all of the attribute groups which contain sampled data (SampledDataOne, SampledDataTwo, and SampledDataThree) would be collected every 60 seconds. Each of the event attribute groups (EventDataOne, EventDataTwo, and EventDataThree) would store the last 100 events in their cache.

These settings might work perfectly, or there might be reasons that you must control the settings at a more granular level. For example, what if EventDataOne generally receives 10 times as many events as EventDataTwo and EventDataThree? To further complicate things, there really is a link between EventDataOne and EventDataTwo. When one event is received for EventDataTwo, there are always multiple events for EventDataOne and users want to correlate these events. There is not a single correct setting for the cache size. It would be nice to be able to have EventDataOne store a larger number of events and EventDataTwo store a smaller number. You can achieve this storage by setting `CDP_PURE_EVENT_CACHE_SIZE` to the size that makes sense for most of the event attribute groups, 100 seems good. Then, you can set `CDP_EVENTDATAONE_PURE_EVENT_CACHE_SIZE` to 1000. That way all of the corresponding events are visible in the Tivoli Enterprise Portal.

The same thing can be done with `CDP_DP_REFRESH_INTERVAL`. Set a default value that works for the largest number of attribute groups in the agent. Then set `CDP_`*`attribute group name`*`_REFRESH_INTERVAL` for the attribute groups which must be collected differently. To optimize collection, set the default `CDP_DP_REFRESH_INTERVAL` to match the `CDP_DP_CACHE_TTL` value. `CDP_DP_CACHE_TTL` is a global value so if set to a value less than a refresh interval, unexpected collections might occur.

Chapter 6. Defining and testing data sources

Agent Builder supports a number of data providers. You can create data sources from each data provider. The procedure for creating and testing data sources is different for each data provider.

For most data providers, when you create a data source, a data set (attribute group) is added to the agent. The data set contains the information that is gathered by this data source.

A data source with a Process, Windows service, or Program return code data provider uses the special Availability data set. Only one Availability data set can be created in an agent. It contains the information that is gathered by all data sources with a Process, Windows Service, or Program Return Code data provider in this agent.

All Windows log data sources in an agent or subnode place event information into one Event Log data set.

Setting up a data source for IBM Cloud Pak for Multicloud Management

In IBM Cloud Pak for Multicloud Management, you can use data from all data sets in the thresholds that you create. For data to be visible in the IBM Cloud Pak console, you must model the data as one or more resources.

These agent resources should group subsets of the data so that each resource represents a logical entity in the application, system, or network environment. Each resource can contain any subset of the information contained in any number of data sources. Each resource definition should include one data source with at least one attribute that can be used to identify the resource. If the data source is single-row, the agent creates one resource. If the data source is multi-row, the agent creates a resource for each unique set of values. A resource can include an event data source as additional data. All of the data selected when the resource is defined is displayed in a table in the IBM Cloud Pak console. You can choose to plot a subset of the data in a line graph by specifying a *units* value for the attribute.

For more information, see Chapter 13, “Preparing the agent for Cloud Pak for Multicloud Management,” on page 223.

Setting up a data source for IBM Cloud Application Performance Management

In Cloud APM, you can use data from all data sets in the Details dashboard and to set up thresholds using the threshold manager. If you want to use information from a data set in the summary dashboard for the agent or subnode, including the status indicator, as well as for resource information (service name, address, and port), the data set must produce only one row.

For most data providers, you can select **Produces a single data row** in the data set configuration. If the gathered information would include more than one row, you can click **Advanced** to set up a filter that ensures the correct row is produced (for instructions, see “Filtering attribute groups” on page 45). You can test your data source to ensure that the gathered information produces the row that you need.

For some data providers, the data set must produce multiple rows. Also, the process, Windows service, and command return code data sources place data into a single Availability data set, which produces multiple rows. In such cases, you must create a filtered data set that produces one row. For instructions about creating a filtered data set (attribute group), see “Creating a filtered attribute group” on page 182.

Some other data providers produce event data; a row is included for every new event. Do not use these data providers for summary or resource information in Cloud APM.

The following data providers must produce a data set with multiple rows:

- Process (uses the Availability data set)
- Windows service (uses the Availability data set)
- Program return code (uses the Availability data set)
- For some data types, SNMP and JMX

© Copyright IBM Corp. 2010, 2021

- Depending on the application, Socket and Java API

The following data providers produce event data:

- SNMP event
- Log file
- AIX binary log
- Windows event log
- Depending on the application, Socket and Java API

One of the attributes of the data set must provide a status value. Cloud APM uses this value for the overall status indicator. If the row does not include an attribute that can be used as a status indicator, you can create a derived attribute to calculate the status. You must configure the status severity values; for instructions, see "Specifying severity for an attribute used as a status indicator" on page 44.

Monitoring a process

You can define a data source that monitors a process or several processes which run on a server. The processes must run on the same host as the agent. For every process, the data source adds a row to the Availability data set.

Procedure

1. On the **Agent Initial Data Source** page or the **Data Source Location** page, click **A process** in the **Monitoring Data Categories** area.
2. In the **Data Sources** area, click **A process**.
3. Click **Next**.
4. On the **Process Monitor** page, in the **Process information** area, provide the display name and process name. You can type the process name manually or obtain it by clicking **Browse**. Clicking **Browse** shows a list of processes that are currently running on the local system or on a remote system.

   You can further discriminate processes by selecting the **Use argument match** and **Match full command line** options. For example, if multiple instances of the same processes are running on the system, one instance can be distinguished from another by using these options.

*Table 9. Fields on the **Process Monitor** page.* A table listing the fields in the **Process Monitor** page and their descriptions

Field name	Description	Acceptable values
Display name	Descriptive name for the component of the application that is implemented by the process as it is shown in the Tivoli Enterprise Portal or in the IBM Cloud Application Performance Management console	Descriptive string
Process name	Name of the process that is being monitored	Valid executable file name
Use argument match	Select if you want to match on the process arguments.	On or Off

*Table 9. Fields on the **Process Monitor** page.* A table listing the fields in the **Process Monitor** page and their descriptions *(continued)*

Field name	Description	Acceptable values
Argument	Argument string on which to match. Argument matching looks for the provided string as a substring of the arguments. Matching is successful if you provide any part of the arguments as the input string.	String
Match full command line	Specify the entire name of the executable file that might include the path	On or Off
Command line	Matches the provided string against the fully qualified command name that is used to start the process. Command arguments are not included. Fully qualified means the path to the command must be included.	String
Operating systems	Select the operating systems on which this process runs	Any selection

5. If you click **Browse**, the **Process Browser** window opens. This window initially contains detailed information about each process on the Agent Builder system. The information includes the ID, the process name, and the full command line for the process. Select one or more processes or work with the list in the **Process Browser** window by using one or more of the following actions:
   a) To sort the list of processes, click the column heading.
   b) To refresh the information in the window, click the **Refresh** (lightening bolt) icon.
   c) To search for specific processes, click the **Search** (binoculars) icon.
      You can enter a search phrase and select options section to search by process identifier, name, and command line.
   d) To view processes on a different system, select a previously defined system from the **Connection Name** list. Or click **Add** to enter the system information for a new system.
      For more information, see "Defining connections for process browsing" on page 67. You can load processes from more than one system at a time, and switch between connections while processes are loading for one or more connections.
      Note: When you browse remote systems, the command-line details are available only when you browse through a Tivoli Enterprise Portal Server.

In the following example, after you select `svchost.exe`, it is shown in the **Process name** field on the **Process Monitor** page (Figure 1 on page 66).

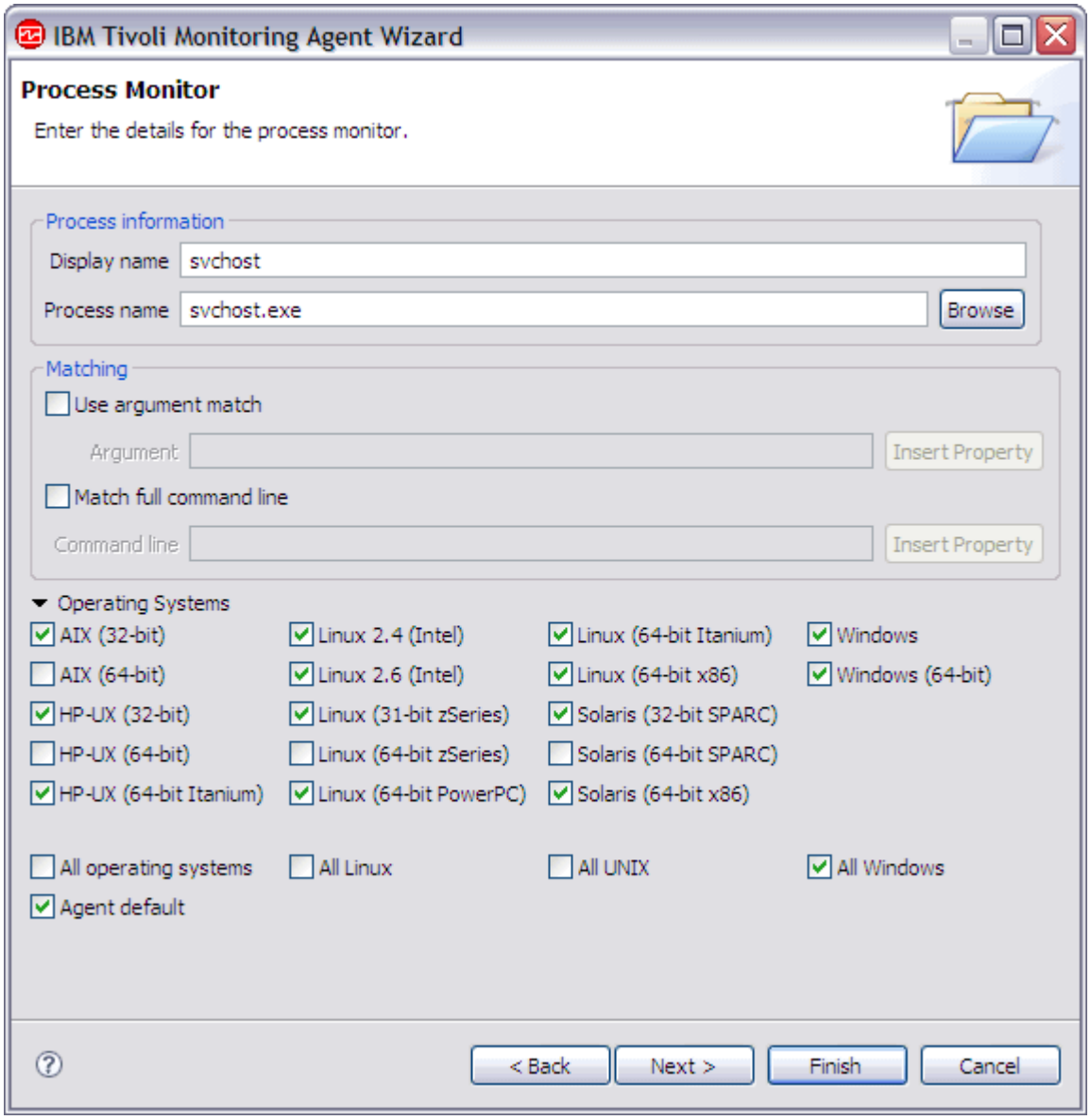


Figure 1. Process Monitor page example

6. Complete the **Process Monitor** page by using the information in (Table 9 on page 64).

   Note: If the process you described in this monitor is applicable to only some of the operating systems that your application runs on, you might want to create one or more process monitors with the same display name to cover the other operating systems. Add the process monitors one at a time. Ensure that the display name is the same for each monitor, but that the process name can be found on the operating systems that are selected.

7. Do one of the following steps:

   - If you are using the **Agent** wizard, click **Next**.
   - Click **Finish** to save the data source and open the Agent Editor.

What to do next

If you want to use the data from this data source in the summary dashboard for IBM Cloud Application Performance Management, you must create a filtered data set (attribute group) based on the Availability data set and configure it as providing a single row. Use the NAME field to select the row for your process.

You can use the Status field for status; DOWN means that the process is not running, while UP means it is running. In the new filtered attribute group, select the Status field and specify the severity values for it.

If several copies of the process are running, several rows with this process name are present in the Availability data set, and all of then include the UP status. Your filtered data set must be configured to return one row, so any of these rows might be returned, but the Status value is valid in any case.

For instructions, see:

- "Creating a filtered attribute group" on page 182
- "Specifying severity for an attribute used as a status indicator" on page 44
- Chapter 12, "Preparing the agent for Cloud APM," on page 219

Defining connections for process browsing

When you define a process data source, you can view and select processes from other systems. However, when the agent runs, it monitors processes that run on the same system as the agent.

About this task

You must have credentials for the other systems or they must be monitored by a Tivoli Monitoring operating system agent.

Procedure

1. To define a connection, click **Add** in the **Process Browser** window.

   You can select either a connection type (Secure Shell (SSH), Windows, or Tivoli Enterprise Portal Server Managed System) or select an existing connection to use as a template.

   To add a Managed System connection, you require a Tivoli Enterprise Server host name, Tivoli Monitoring user name, and password. You also require the managed system name of the remote connection. When a managed system is selected, the table lists the process on the remote system.

   Note: The OS agent must be running on the system you are attempting to browse. The agent must also be connected to a running Tivoli Enterprise Monitoring Server and Tivoli Enterprise Portal Server.

   To add Secure Shell (SSH) or Windows connections, you require a host name, user name, and password.
2. When you add a connection, you can select the connection from the **Connection Name** list in the **Process Browser** window.

   If all the fields required to make the connection are not saved (for example, the password), the **Connection Properties** window for that connection opens. Enter the missing information. For Tivoli Enterprise Portal Server Managed System connections, you must connect to the Tivoli Enterprise Portal Server before you can enter a managed system.
3. Enter your user name and password, and then click the **Refresh** (lightening bolt) icon to connect before you select the managed system.

What to do next

To delete a connection, select the connection and click **Edit** to open the **Connection Properties** window. Select the **Remove this connection** check box and click **OK**.

Monitoring a Windows service

You can define a data source that monitors a service or several services which run on a Windows system. The services must run on the same host as the agent. For every service, the data source adds a row to the Availability data set.

Procedure

1. On the **Agent Initial Data Source** page or the **Data Source Location** page, click **A process** in the **Monitoring Data Categories** area.

2. In the **Data Sources** area, click **A Windows service**.
3. Click **Next**.
4. On the **Service Monitor** page, in the **Display name** field, type a description. In the **Service name** field, provide the name of the service application. You can type it manually or click **Browse** to view a list of services that are currently running on the local system or on a remote system.

   If you click **Browse**, the **Service Browser** window opens. This window initially contains detailed information about each service on the Agent Builder system. The information includes the service name, the display name, the state, and the description for the service.

   Note: Local services are not shown when Agent Builder is not running on a Windows system. A remote Windows system must be defined or selected, see ("Defining connections for service browsing" on page 68).

   Note: The service description is not available when you are browsing through the Tivoli Enterprise Portal Server or from a UNIX or Linux system.
5. Select one or more services or do one or more of the following steps to work with the list in the **Service Browser** window:
   - To sort the list of services, click the column heading.
   - To refresh the information in the window, click the **Refresh** (lightening bolt) icon.
   - To search for a service, click the **Search** (binoculars) icon to open the **Service Search** window. You can search by the service name, display name, and description.
   - To view services on a different system, select a previously defined system from the **Connection Name** list or click **Add** to enter the system information. For more information, see ("Defining connections for service browsing" on page 68). You can load services from more than one system at a time, and switch between connections while services are loading for one or more connections.
6. After selecting or entering the name of the service, complete one of the following steps:
   - If you are using the **Agent** wizard, click **Next**.
   - Click **Finish** to save the data source and open the Agent Editor.

What to do next

If you want to use the data from this data source in the summary dashboard for IBM Cloud Application Performance Management, you must create a filtered data set (attribute group) based on the Availability data set and configure it as providing a single row. Use the NAME field to select the row for your process.

In the new filtered attribute group, select the Functionality_Test_Status field and specify the severity values for it.

For instructions, see:

- "Creating a filtered attribute group" on page 182
- "Specifying severity for an attribute used as a status indicator" on page 44
- Chapter 12, "Preparing the agent for Cloud APM," on page 219

Defining connections for service browsing

In addition to selecting services from the system where Agent Builder is running, you can select services from other Windows systems.

About this task

To select services from other Windows systems, you define a connection to the remote system. You must have credentials for the systems or they must be monitored by a Tivoli Monitoring operating system agent.

Procedure

1. To define a connection, click **Add** in the **Service Browser** window.

   The **Select Connection Type** window opens. To add a Managed System connection, you require a Tivoli Enterprise Server host name, Tivoli Monitoring user name and password, and the managed system name. When a managed system is selected, the table lists the service on the remote system.

   Note: The OS agent must be running on the system you are attempting to browse and also connected to a running Tivoli Enterprise Monitoring Server and Tivoli Enterprise Portal Server.

   You require a host name, user name, and password to add a Windows connection.
2. Select a connection type (Windows, or Tivoli Enterprise Portal Server Managed System) or select an existing connection to use as a template.

   The **Connection Properties** window opens.
3. Complete the Connection Properties.
4. Click **Finish**
5. When you add a connection, you can select the connection from the **Connection Name** list in the **Service Browser** window.

   If the fields necessary to make the connection are not saved (for example, the password), the**Connection Properties** window opens and you can enter the missing information.

   a) For Tivoli Enterprise Portal Server Managed System connections, you must connect to the Tivoli Enterprise Portal Server before you can enter a managed system. Enter your user name and password, and then click the **Refresh** (lightening bolt) icon to connect before you select the managed system.
6. To delete a connection, follow these steps:

   a) Select the connection in the **Service Browser** window.

   b) Click **Edit** to open the **Connection Properties** window.

   c) Select the **Remove this connection** check box.

   d) Click **OK**.

Monitoring data from Windows Management Instrumentation (WMI)

You can define a data source to collect data from Windows Management Instrumentation (WMI) on the system where the agent runs or on a remote system. A data source monitors a single WMI class and places all values from this class into the data set that it produces. If the class provides several instances, the data set has multiple rows; you can filter by instance name to ensure the data set has one row.

Before you begin

If your agent collects data from a remote system by using Windows Management Instrumentation (WMI), it requires permissions to access WMI data on the remote system. The agent can access WMI data on a remote system when you provide credentials of an account with permissions to access WMI data on the system. The Administrator account has the required permissions. In the procedure that follows you can either provide the Administrator credentials or the credentials of another user with the required permissions. For more information about creating a user account with permissions to browse WMI data, see “Creating a user with Windows Management Instrumentation (WMI) permissions” on page 208.

To collect metrics through the Windows APIs, the agent must be hosted on a Windows operating system. Remote registry administration must be enabled on the remote systems.

Procedure

1. On the **Agent Initial Data Source** page or the **Data Source Location** page, click **Data from a server** in the **Monitoring Data Categories** area.

2. In the **Data Sources** area, click **WMI**.
3. Click **Next**.
4. On the **Windows Management Instrumentation (WMI) Information** page, complete one of the following steps:
   - Type a name for the WMI namespace and a name for the WMI class name in the fields. Then go to step "9" on page 70
   - Click **Browse** to see all of the WMI classes on the system.

     To browse a remote system, select a system from the list (if one is defined). Alternatively click **Add** to add the host name of a Windows system. Provide the credentials of a user account with permissions to access WMI data on the remote system, or provide Administrator credentials for the remote system. The page is updated with the information for the remote system. Browsing is available only when the Agent Builder is run on a Windows system, and can browse only Windows systems.

5. Click the plus sign (+) next to a class to expand the class and show the attributes.
6. From the list, select the class with its associated attributes that you want to specify, and click **OK**.

   Note: You can click the **Search** (binoculars) icon to find your selection in the list. Type a phrase in the **Search phrase** field; specify your preference by clicking either the **Search by name**, **Search by class description**, or **Search by class properties** fields; and click **OK**. If you find the item for which you are searching, select it and click **OK**.

   The **WMI Information** page of the wizard opens again, showing the selected WMI class information.
7. Optional: You can test this attribute group by clicking **Test**. For more information about testing, see "Testing WMI attribute groups" on page 71
8. Optional: You can create a filter to limit the data that is returned by this attribute group by clicking **Advanced**. For more information about filtering data from an attribute group, see "Filtering attribute groups" on page 45
9. Click **Next**.

   Note: If you typed the WMI Namespace and WMI class name manually you are brought to the **Attribute Information** page, where you can complete attribute information. On the **Attribute Information** page, you can select **Add additional attributes** if you want to add more attributes. Click **Finish** to complete.
10. On the **Select key attributes** page, select key attributes or indicate that this data source produces only one data row. For more information, see ("Selecting key attributes" on page 15).
11. Do one of the following steps:
    - If you are using the **Agent** wizard, click **Next**.
    - Click **Finish** to save the data source and open the Agent Editor.
12. You can add attributes and supply the information for them. For more information, see "Creating attributes" on page 37.

    In addition to fields that are applicable to all data sources (Table 5 on page 40), the **Attribute Information** page for the WMI data source has the following field:

    Metric name
    Property name from the class you want to collect
13. If you want to set global options for the data source, click **Global Options**.

    Select the **Include remote Windows configuration properties** check box if you want to include this option, and click **OK**.

    For information about Windows remote connection configuration for Windows data sources, see ("Configuring a Windows remote connection" on page 207).

Testing WMI attribute groups

If you are running Agent Builder on a Windows system, you can test a WMI attribute group within Agent Builder.

Procedure

1. You can start the Testing procedure in the following ways:
   - During agent creation click **Test** on the **WMI Information** page.
   - After agent creation, select an attribute group on the Agent Editor **Data Source Definition** page and click **Test**. For more information about the Agent Editor, see Chapter 4, "Using the Agent Editor to modify the agent," on page 17.

   After you click **Test** in one of the previous two steps, the **WMI Test** window is displayed.
2. Optional: Before you start testing, you can set environment variables and configuration properties. For more information, see "Attribute group testing" on page 229).
3. Click **Start Agent**.

   A window indicates that the Agent is starting.
4. To simulate a monitoring environment request for agent data, click **Collect Data**.

   The agent queries WMI for data. The **WMI Test** window collects and displays any data in the agent's cache since it was last started.
5. Optional: Click **Check Results** if the returned data is not as you expected.

   The **Data Collection Status** window opens and shows you more information about the data. The data that is collected and displayed by the **Data Collection Status** window is described in ("Performance Object Status node" on page 280).
6. Stop the agent by clicking **Stop Agent**.
7. Click **OK** or **Cancel** to exit the **WMI Test** window. Clicking **OK** saves any changes that you made.

Related concepts

"Testing your agent in Agent Builder" on page 229
After you use Agent Builder to create an agent, you can test the agent in Agent Builder.

Monitoring a Windows Performance Monitor (Perfmon)

You can define a data source to collect data from Windows Performance Monitor (Perfmon). A data source monitors a Perfmon object. The counters in the object are placed in attributes in the resulting data set. If the class provides several instances, the data set has multiple rows; you can filter by instance name to ensure the data set has one row.

Procedure

1. On the **Agent Initial Data Source** page or the **Data Source Location** page, click **Data from a server** in the **Monitoring Data Categories** area.
2. In the **Data Sources** area, click **Perfmon**.
3. Click **Next**.
4. On the **Perfmon Information** page, complete one of the following steps:
   - Type the name of the object in the **Object Name** field, and click **Next** to define the first attribute in the attribute group.

     Note: If you type the name for the Windows Performance Monitor object, it must be the English name.
   - Click **Browse** to view the list of Perfmon objects.

     When the Performance Monitor (Perfmon) Object Browser window initially opens, the window populates with the information from the local system. To browse a remote system, select a system

from the list (if one is defined), or click **Add** to add the host name of a Windows system. Provide an Administrator ID and password. The window updates with the information for the remote system. Browsing is available only when Agent Builder is running on a Windows system, and can browse only Windows systems. For example, you cannot add the host name of a Linux or Solaris system to do a remote browse.

– When you click an object name, the available counters in that object are shown in the window.

- To sort the Windows Performance Monitor objects or counters, click the column heading.
- To refresh the information in the window, click **Refresh**.
- To search for specific objects or counters click the **Search** (binoculars) icon to open the **Performance Monitor Search** window. You can search object names, counter names, or both. The search operation does a substring match and is not case-sensitive.
- Select an object and click **OK**.
- The **Perfmon Information** page opens with the name of the selected object in the **Object Name** field.

- If you want to set global options for the data source, click **Global Options**

  Select the **Include remote Windows configuration properties** check box if you want to include this option, and click **OK**.

  For information about Windows remote connection configuration for Windows data sources, see ("Configuring a Windows remote connection" on page 207).

5. If the Windows Performance Monitor object selected returns multiple instances and you want to filter the results that are based on the instance name:
   a) Select the **Filter by Perfmon Instance Name** check box on the **Perfmon Information** page.
   b) In the **Perfmon Instance Name** field, type the name of the instance to be filtered, or click **Browse** to list the instances available.
   c) To browse a remote system, either select one from the list, or click **Add** to add the host name of a Windows system. After you select a host, provide an Administrator ID and password. The table is updated with the list of instances on the remote system.

   Note: You can also filter by attribute group, see step "9" on page 72
6. If the selected Windows Performance Monitor Object is to return multiple instances, and you want the instance name to be returned, select **Return Instance Name** on the **Perfmon Information** page.

   Checking this option adds an attribute to the data source that is not shown in the list of attributes. This attribute contains the instance name.

   Note: If you browsed for the selected object, and that object is defined as having multiple instances, this check box is selected automatically.
7. If you did not check the option to return the instance name, the Select key attributes page opens. On the Select key attributes page, select key attributes or indicate that this data source produces only one data row. For more information, see ("Selecting key attributes" on page 15).
8. Optional: You can test this attribute group by clicking **Test**. For more information about testing, see "Testing Perfmon attribute groups" on page 73
9. Optional: You can create a filter to limit the data that is returned by this attribute group by clicking **Advanced**.

   For more information about filtering data from an attribute group, see step "Filtering attribute groups" on page 45

   Note: You can also filter by instance name, see "5" on page 72
10. Do one of the following steps:

    - If you are using the New Agent wizard, click **Next**.
    - Click **Finish** to save the data source and open the Agent Editor.

The **Agent Editor Data Source Definition** page shows a list that contains the object and information about the object.

11. You can add attributes and supply the information for them. For more information, see ("Creating attributes" on page 37).

    In addition to the fields applicable to all data sources, the **Perfmon Attribute Information** page for the data source has the following field:

    Metric name
    Name of the counter for the specific object.

What to do next

For information about Windows remote connection configuration for Perfmon data sources, see "Configuring a Windows remote connection" on page 207.

Testing Perfmon attribute groups

If you are running Agent Builder on a Windows system, you can test the Perfmon attribute group that you created.

Procedure

1. You can start the Testing procedure in the following ways:
   - During agent creation click **Test** on the **Perfmon Information** page.
   - After agent creation, select an attribute group on the Agent Editor **Data Source Definition** page and click **Test**. For more information about the Agent Editor, see Chapter 4, "Using the Agent Editor to modify the agent," on page 17.

   After you click **Test** in one of the previous two steps, the **Perfmon Test** window is displayed.
2. Optional: Before you start testing, you can set environment variables and configuration properties. For more information, see "Attribute group testing" on page 229.
3. Click **Start Agent**. A window indicates that the Agent is starting.
4. To simulate a request from the monitoring environment for agent data, click **Collect Data**.

   The agent queries Performance Monitor for data. The **Perfmon Test** window collects and shows any data in the agent's cache since it was last started.

   Note: You might not see useful data for all attributes until you click **Collect Data** a second time. The reason is that some Performance Monitor attributes return delta values, and a previous value is required to calculate a delta value.
5. Optional: Click **Check Results** if the returned data is not as you expected.

   The **Data Collection Status** window opens and shows you more information about the data. The data that is collected and shown by the **Data Collection Status** window is described in "Performance Object Status node" on page 280.
6. Stop the agent by clicking **Stop Agent**.
7. Click **OK** or **Cancel** to exit the **Perfmon Test** window. Clicking **OK** saves any changes that you made.

Related concepts

"Testing your agent in Agent Builder" on page 229
After you use Agent Builder to create an agent, you can test the agent in Agent Builder.

Monitoring data from a Simple Network Management Protocol (SNMP) server

You can define a data source to monitor an SNMP server. A data source monitors all data from a single SNMP object identifier (OID) and a single host. if you select an element of the OID registration tree under

which other objects are registered, a data set is created for each distinct set of scalar or table values. If an object returns scalar data, the data set has a single row. If an object returns tabular data, the data set has multiple rows.

About this task

Simple Network Management Protocol V1, V2C (note that the version is V2C and not just V2), and V3 are supported by agents.

Procedure

1. On the **Agent Initial Data Source** page or the **Data Source Location** page, click **Data from a server** in the **Monitoring Data Categories** area.
2. In the **Data Sources** area, click **SNMP**.
3. Click **Next**.
4. On the Simple Network Management Protocol (SNMP) Information page, type the display name or click **Browse** to see all of the objects on the system.

   After you define the data source, you can add an attribute. The OIDs for these attributes can be long and difficult to type correctly. Using the Browse option is an easy way to input the correct OID.

   Note: The browser does not browse the live system, it reads definitions, Management Information Bases (MIBs).

   Note: Clicking the **Refresh** icon clears the in-memory version of the parsed MIB files and reparses the files in the workspace cache. The cache is in the following location: *workspace_directory* `\.metadata\.plugins\ com.ibm.tivoli.monitoring.agentkit\mibs`

   Where:

   workspace_directory
   Identifies the workspace directory that you specified when you initially ran the Agent Builder, see ("Starting Agent Builder" on page 10).

   a) If the MIB that defines the wanted object is not loaded, click **Manage Custom MIBs** to open the Manage Custom MIBs dialog.

   b) Click **Add** to browse to the MIB file to add. To delete a MIB from the cache, select it and click **Remove**.

   c) Click **OK** to update the cache.

   If there are any errors when the MIBs are parsed, the Manage Custom MIBs dialog remains open. This dialog gives you the opportunity to add or remove MIBs to eliminate the errors.

   Clicking **Cancel** returns the MIB cache to the state it was in when the dialog was opened.

   Agent Builder includes a set of MIBs:

   - `hostmib.mib`
   - `rfc1213.mib`
   - `rfc1243.mib`
   - `rfc1253.mib`
   - `rfc1271.mib`
   - `rfc1286.mib`
   - `rfc1289.mib`
   - `rfc1315.mib`
   - `rfc1316.mib`
   - `rfc1381.mib`
   - `rfc1382.mib`

- rfc1443.mib
- rfc1461.mib
- rfc1471.mib
- rfc1493.mib
- rfc1512.mib
- rfc1513.mib
- rfc1516.mib
- rfc1525.mib
- rfc1573a.mib
- rfc1595.mib
- rfc1650.mib
- rfc1657.mib
- rfc1659.mib
- rfc1666.mib
- rfc1695.mib
- rfc1747.mib
- rfc1748.mib
- rfc1757.mib
- rfc1903.mib
- rfc1907.mib
- rfc2011.mib
- rfc2021.mib
- rfc2024.mib
- rfc2051.mib
- rfc2127.mib
- rfc2128.mib
- rfc2155.mib
- rfc2206.mib
- rfc2213.mib
- rfc2232.mib
- rfc2233.mib
- rfc2238.mib
- rfc2239.mib
- rfc2320.mib
- rfc3411.mib

All of these MIBs are standard, IETF defined MIBs. The MIBs are included because they represent common definitions that can be useful in monitoring. Also, many of the MIBs are necessary so that custom MIBs can resolve the symbols that they import.

d) Select an object from the list.

Click the plus sign (+) next to an object to expand and show the levels.

e) From the list, select the object that you want to specify and click **OK**.

The new data source is then listed on the **Data Source Definition** page.

Note: If you select an object that defines other objects (objects that are nested underneath the first object), all of these objects are turned into data sources. If you select a high-level object, many data sources are added.

5. On the **Simple Network Management Protocol Information** page, select the operating systems.
6. Optional: You can test the data source or sources by clicking **Test** on the **Simple Network Management Protocol Information** page.

   For more information about testing, see "Testing SNMP attribute groups" on page 77
7. Optional: You can create a filter to limit the data that is returned by this attribute group by clicking **Advanced**. For more information about filtering data from an attribute group, see "Filtering attribute groups" on page 45
8. Click **Next**.
9. On the **Attribute Information** page, specify the information for the attribute.
10. Do one of the following steps:
    - If you are using the New Agent wizard, click **Next**.
    - Click **Finish** to save the data source and open the Agent Editor.
11. For more information about adding attributes and supplying the information for them, see "Creating attributes" on page 37.

    In addition to fields that are applicable to all data sources, the **Attribute Information** page for the SNMP data source has the following fields:

    Metric name
    Arbitrary string

    Object identifier
    Full OID that is registered to the object, not including index values

What to do next

You can use the runtime configuration of the agent to set the monitored host.

To enable Agent Builder to generate 64-bit data types and to handle the maximum value for 32-bit unsigned MIB properties, see "SNMP MIB Parsing options" on page 76.

SNMP MIB errors

Dealing with errors in SNMP MIBs.

It is not unusual to find errors when you are adding SNMP MIBs. Click **Details>>** in the **Agent Builder Error** window to see what the MIB error is.

One of the most common errors is missing definitions that are defined in other MIBs. You can import several MIBs simultaneously to resolve this problem, or you can incrementally add MIBs until all of the missing definitions are resolved. Agent Builder can use any definitions that are resolved. So you can choose to ignore an error that affects only that part of the MIB that you do not plan to use. The order of the MIBs does not matter because they are all loaded, and then the references are resolved.

SNMP MIB Parsing options

Set your preferences for SNMP MIB parsing

Procedure

1. In the Agent Builder, select **Window** > **Preferences** to open the **Preferences** window.
2. In the navigation pane, expand **IBM Agent Builder**.
3. Click **MIB Parsing** to open the **MIB Parsing** window.

The MIB parser that is used by Agent Builder uses the grammar that is defined by ASN.1 to parse the MIBs. Some MIBs do not follow the grammar correctly. The parser can relax certain rules to accommodate the most common errors. By relaxing these rules, you can parse non-conforming MIBs.

Allow types to start with lowercase letters
Allows types that people write in MIBs, such as values

Allow numeric named numbers
Allows numbers that start with uppercase letters

Allow underscore in value name
Allows underscore characters

Allow values to begin with uppercase letters
Allows values that start with uppercase letters

Ignore duplicate MIBs
Turns off warning for duplicate MIB modules

4. Optional: Selecting the **Create 64-bit attributes for 32 bit unsigned MIB properties** check box, enables the Agent Builder to generate 64-bit data types to handle the maximum value for 32-bit unsigned MIB properties. Selecting this option does not change any existing agent field definitions. You must browse to the MIB file to create new data sources for these properties.
5. When you are finished editing the preferences, click **OK**.

Testing SNMP attribute groups

You can test the SNMP attribute group that you created within Agent Builder.

Procedure

1. You can start the Testing procedure in the following ways:
   - During agent creation click **Test** on the **Simple Network Management Protocol Information** page.

     Note:

     If the SNMP object selected contains more than one attribute group, you are prompted to select the attribute group to test.
   - After agent creation, select an attribute group on the **Agent Editor Data Source Definition** page and click **Test**. For more information about the Agent Editor, see Chapter 4, “Using the Agent Editor to modify the agent,” on page 17

   After you click **Test** in one of the previous two steps, the SNMP Test settings window opens.
2. Select an existing connection from **Connection name** or click **Add** and you are prompted to select a connection type. Alternatively select an existing connection to use as a template, by using the **Create Connection Wizard**
3. After you select a connection type or an existing connection, click **Next** to complete the SNMP connection properties. When complete click **Finish** to return to the SNMP Test settings window.
4. Optional: Before you start testing, you can set environment variables and configuration properties. For more information, see (“Attribute group testing” on page 229).
5. Click **Start Agent**. A window indicates that the Agent is starting.
6. To simulate a request from Tivoli Enterprise Portal or SOAP for agent data, click **Collect Data**. The agent queries the configured SNMP connection for data.
7. The **Test Settings** window collects and shows any data in the agent's cache since it was last started.
8. Optional: Click **Check Results** if the returned data is not as you expected.

   The **Data Collection Status** window opens and shows you more information about the data. The data that is collected and shown by the **Data Collection Status** window is described in “Performance Object Status node” on page 280
9. Stop the agent by clicking **Stop Agent**.

10. Click **OK** or **Cancel** to exit the **Test Settings** window. Clicking **OK** saves any changes that you made.

Related concepts

"Testing your agent in Agent Builder" on page 229
After you use Agent Builder to create an agent, you can test the agent in Agent Builder.

Monitoring events from Simple Network Management Protocol event senders

You can define a data source to collect data from SNMP Trap and Inform events. You must set the port in the agent runtime configuration and configure the servers to send event to the agent host on this port. All the monitored events are placed as rows in a data set.

About this task

Simple Network Management Protocol (SNMP) V1, V2C (note that this version name is V2C and not just V2), and V3 are supported by agents. SNMP Traps and Informs can be received and processed by the agent. Data that is received by this provider is passed to the monitoring environment as events.

For more information about the attribute groups for SNMP events, see ("SNMP Event attribute groups" on page 306).

Procedure

1. On the **Agent Initial Data Source** page or the **Data Source Location** page, click **Data from a server** in the **Monitoring Data Categories** area.
2. In the **Data Sources** area, click **SNMP Events**.
3. Click **Next**.
4. In the **Simple Network Management Protocol Event Information** window, do one of the following steps:
   - Click **All Events** to create an attribute group that sends an event for any received SNMP event.
   - Click **Generic Events** to create an attribute group that sends an event for any received generic SNMP event that matches any of the selected generic event types.
   - Click **Custom Events** to create one or more attribute groups that send events for enterprise-specific SNMP events. Click **Browse** to choose the events to be monitored.

     In the **Simple Network Management Protocol (SNMP) Management Information Base (MIB) Browser** window, the events in the selection pane are organized by the MIB module in which they were defined. Expand an SNMP object to show the events in that MIB module. In the list, click the object that you want to specify and click **OK**.

     Select the **Include attributes that show information defined in the trap configuration file** check box if you have a trap configuration file that contains static data for your traps. For more information about the SNMP trap configuration file, see (Appendix I, "SNMP trap configuration," on page 365).

     Select the **Include variable binding (VarBind) data attribute** check box if you want to include an attribute with all of the variable binding (VarBind) data that is received in the trap protocol data unit (PDU). For more information about this attribute, see the attribute definition ("SNMP Event attribute groups" on page 306).

     Note:

     a. The browser does not browse the live system; it reads definitions and, Management Information Bases (MIBs). The list of MIBs included with Agent Builder is defined in "Monitoring data from a Simple Network Management Protocol (SNMP) server" on page 73. MIBs loaded by either SNMP data provider are available in both.

b. If you select a MIB module or individual events, all the events in that module are converted to separate data sources. One attribute is added for each of the variables that are defined in the event. If you want all the events for the selected modules or traps to arrive in a single event source, select the **Collect events in a single attribute group** check box. If you select individual traps and the **Collect events in a single attribute group** flag is selected, one attribute is added for each of the variables that are defined in each of the events (duplicate variables are ignored). If you select a module, variable attributes are not added.

c. If you want to type your own filter, use the following syntax:

The value of the OID (object identifier) element is used to determine which traps to process for this attribute group.

- **Trap matching:** The OID attribute of the global_snmp_event_settings_for_group element can be a comma-delimited list of tokens. A single token has the following syntax:

```
[enterpriseOID][-specificType]
```

- **Example:** "1.2.3.5.1.4,1.2.3.4.5.6.7.8.9-0" The first token matches any trap with an enterprise OID of 1.2.3.5.1.4. The second token matches any trap with an enterprise of 1.2.3.4.5.6.7.8.9 and specific of 0. Because the tokens are listed together in one attribute group, an event received that matches either is processed by that attribute group.

d. Every event that is received is processed only by the first attribute group that matches the received event. Subnode attribute groups are processed first, and then the base attribute groups are processed. The agent developer must ensure that the groups are defined in a way so that events are received in the expected attribute group.

5. In the **SNMP Event Information** window, select the **Subnode Host matching** check box to match events to subnodes. If the SNMP event attribute group is part of a subnode, you can select the **Subnode Host Matching** check box to control whether the event must come from the SNMP agent that is monitored.

   For example: You have an agent to monitor routers, where each subnode instance represents a specific router. You develop an agent to collect data from a router with the SNMP data collector. You also define an attribute group to receive SNMP events sent by that router. Each router instance includes the same data that is defined for the event filter. Therefore, you need another way to make sure that events from your router are shown in the attribute group for that router.

   When subnode host-matching is selected, an event that is sent by the router is compared to the host defined for the SNMP data collector. If the host in use by the SNMP data collector is the same host that sent the received event, the subnode instance processes the SNMP event. Otherwise, the event is passed to the next subnode instance. Address-matching applies only to subnodes. No address-matching is done by the SNMP event attribute groups in the base agent. For the address-matching to work, the subnode definition must contain at least one SNMP attribute group. The SNMP host that is used by SNMP for that subnode instance is the host that is used for matching.

   If the **Subnode Host Matching** check box is clear, your subnode instances do not do this extra comparison. You must allow the user to configure a different OID filter for each subnode in this case. Otherwise, you do not need to include SNMP event attribute groups in the subnode definition.

6. In the **SNMP Event Information** window, select the operating systems.
7. Optional: You can click **Test** in the **SNMP Event Information** window to start and test your agent.

   For more information, see "Testing SNMP event attribute groups" on page 82
8. Optional:

   In the **SNMP Event Information** window, click **Advanced** to select **Event Filtering and Summarization Options**. For more information, see Chapter 18, "Event filtering and summarization," on page 259.

   a) When you finish selecting **Event Filtering and Summarization Options**, return to the **SNMP Event Information** window. If you previously selected **Custom Events** in the **SNMP Event Information** window, click **Next**, to select key attributes, otherwise skip the next step.

b) On the Select key attributes page, click one or more key attributes for the attribute group, or click **Produces a single data row**.

9. Click **Next**, or click **Finish** if you are using the new agent wizard to save the agent and open the Agent Editor.
10.

What to do next

For information about adding further attributes, see (“Creating attributes” on page 37).

SNMP Event Configuration properties

Certain configuration properties are automatically created when an SNMP Event attribute group is added to the agent

After a data source is added, the configuration is displayed on the **Runtime Configuration Information** page of the Agent Editor. For example, Figure 2 on page 81 shows the configuration sections and some configuration properties that are automatically created when an SNMP Event attribute group is added to the agent.

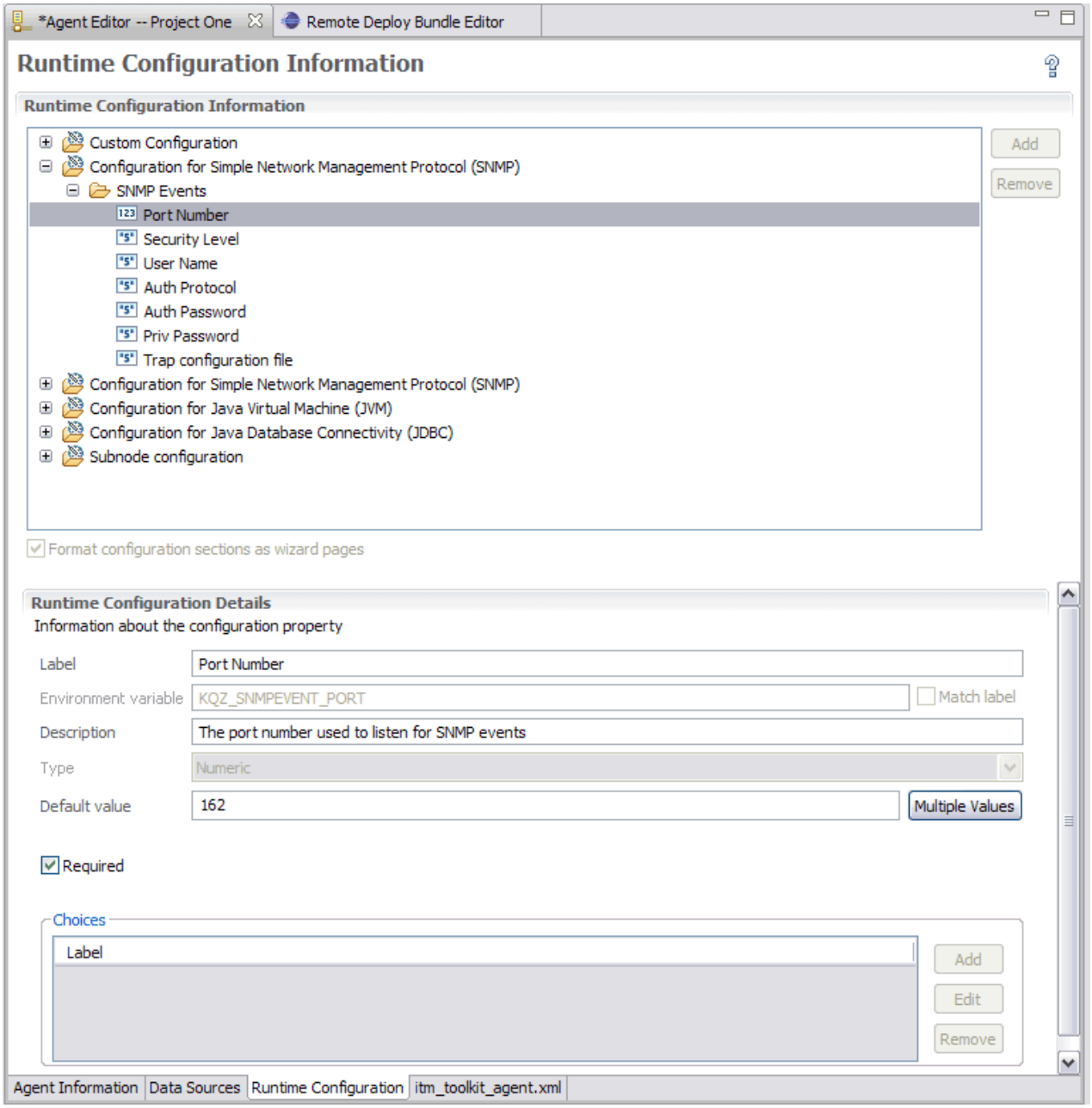


Figure 2. Runtime Configuration page

The labels, descriptions, and default values of predefined configuration properties can be changed, but variable names and types cannot be changed. The SNMP Events configuration section contains the following properties:

Table 10. SNMP Events configuration properties

Name	Valid values	Required	Description
Port Number	positive integer	Yes	Required port number that is used for listening to events
Security Level	noAuthNoPriv, authNoPriv, authPriv	No	SNMP V3 security level
User Name	String	No	SNMP V3 user name

Table 10. SNMP Events configuration properties (continued)			
Name	**Valid values**	**Required**	**Description**
Auth Protocol	MD5 or SHA	No	SNMP V3 authentication protocol
Auth Password	String	No	SNMP V3 authentication password
Priv Password	String	No	SNMP V3 privacy password
Trap configuration file	File name that includes the path	No	Location of the trap configuration file. If the file is not located by using this configuration property, an attempt is made to find a `trapcnfg` file in the agent bin directory.

No configuration is required for V1 or V2C events. All V1 or V2C events are processed regardless of the source or community name specified. The only supported privacy protocol is DES, so there is no option to specify the privacy protocol. The SNMP V3 configuration options are not required (each can be optionally specified). If you want to specify them, you must specify the appropriate values for the security level you select.

Testing SNMP event attribute groups

You can test the SNMP event attribute group that you created, within Agent Builder.

Before you begin

To test the SNMP event attribute group, use a test program, or application to generate SNMP events.

Procedure

1. You can start the Testing procedure in the following ways:
   - During agent creation click **Test** in the **SNMP Event Information** window.
   - After agent creation, select an attribute group on the Agent Editor **Data Source Definition** page and click **Test**. For more information about the Agent Editor, see Chapter 4, “Using the Agent Editor to modify the agent,” on page 17

   After you click **Test** in one of the previous two steps, the **Test Event Setting** window opens.
2. Optional: Before you start testing, you can set environment variables and configuration properties. For more information, see “Attribute group testing” on page 229. For more about SNMP Event configuration properties, see “SNMP Event Configuration properties” on page 80.
3. Click **Start Agent**. A window indicates that the Agent is starting.

   When the agent starts, it listens for SNMP events according to its configuration.

   Note: The agent that starts is a simplified version that includes the one attribute group you are testing.
4. To test your agent's data collection you generate SNMP events that match the agents configuration. You can do this using an application or an event generator.

   When the agent receives SNMP events that match its configuration, it adds the events to its internal cache.
5. To simulate a request from the monitoring environment for agent data, click **Collect Data**.

The **Test Event Settings** window collects and shows any events in the agent's cache since it was last started. An example data collection is shown in Figure 3 on page 83

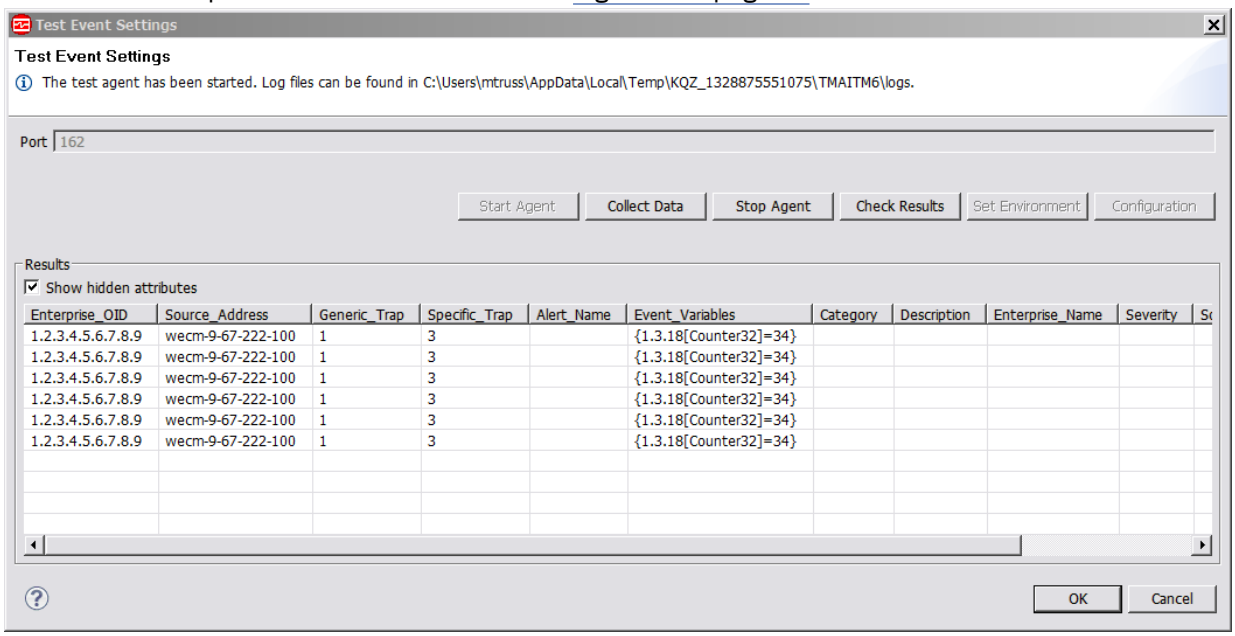


*Figure 3. **Test Event Settings** window that shows collected SNMP event data*

6. Optional: Click **Check Results** if the returned data is not as you expected.

   The **Data Collection Status** window opens and shows you more information about the data. An example is shown in (Figure 4 on page 83). The data that is collected and shown by the **Data Collection Status** window is described in "Performance Object Status node" on page 280

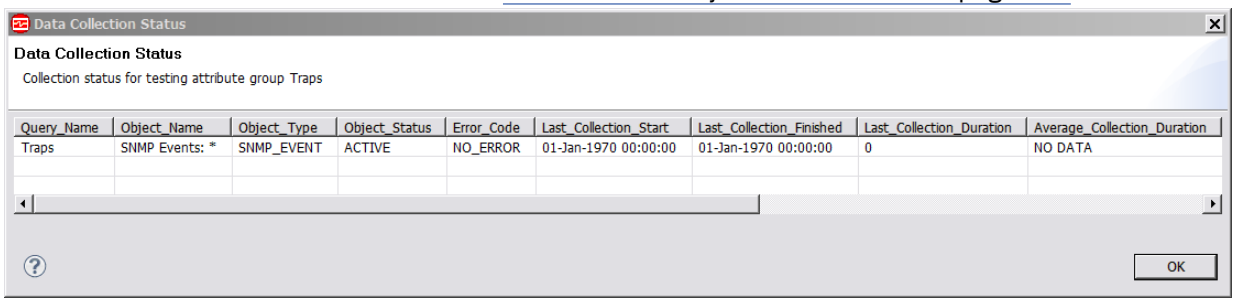


*Figure 4. **Data Collection Status** window*

7. Stop the agent by clicking **Stop Agent**.
8. Click **OK** or **Cancel** to exit the **Test Event Settings** window. Clicking **OK** saves any changes that you made.

Related concepts

"Testing your agent in Agent Builder" on page 229
After you use Agent Builder to create an agent, you can test the agent in Agent Builder.

Monitoring Java Management Extensions (JMX) MBeans

You can define a data source to collect data from JMX MBeans. Data from every monitored MBean is placed into a data set. Depending on the MBean, the data set can produce a single row or multiple rows.

About this task

Each JMX data source that you define must identify either a single MBean (single instance) or a certain type of MBean (multiple instances). You must know the Object Name of the MBean or an Object Name pattern for a MBean type that contains the data you want to collect. Use an Object Name pattern to identify only a set of similar MBeans. The set of MBeans that matches the pattern must all provide the

data that you want to see in the monitoring table. A typical Object Name pattern looks like `*:j2eeType=Servlet,*`. This Object Name Pattern matches all MBeans that have a j2eeType of servlet. You can expect any MBean matching that pattern to have a similar set of exposed attributes and operations that can be added to your data source. A data source that uses that pattern collects data from each MBean matching that pattern. The attributes that you define for this data source must be available for any MBean matching the Object Name pattern of the data source.

Java Version 5 or later is supported.

Procedure

1. On the **Agent Initial Data Source** page or the **Data Source Location** page, click **Data from a server** in the **Monitoring Data Categories** area.
2. In the **Data Sources** area, click **JMX**.
3. Click **Next**.
4. On the **JMX Information** page , click **Browse** to see all of the JMX MBeans on the MBean server.

   After you define the data source, you can use the browse function to pre-populate your attribute list. You can then add to, remove from, or modify the attributes that the browser inserted. The names for these attributes can be long and difficult to type correctly. Using the Browse option is an easy way to input the correct name.

   Note: You can manually create JMX data sources by specifying an Object Name and clicking **Next** without using the browser. Manually creating JMX data sources creates two data sources. An event data source that contains predefined attributes for JMX notifications is created. Also, a collection data source is defined containing one attribute that you must specify in the wizard.

   MBean pattern
   Shows the MBean pattern.

   Global JMX Options
   Shows the level of support.

   Support is provided for the following JMX servers:

   - Java 5 operating system MBean Server. Connection is made by using the JSR-160 connector. Notifications and monitors are supported.
   - WebSphere® Application Server, version 6 and later. Connectors are provided for both SOAP and RMI protocols. JMX Monitors are not supported because MBeans cannot be created by a remote agent.
   - WebSphere Community Edition and other Apache Geronimo-based application servers. Connection is made through standard JSR-160 connectors. JMX notifications and monitors are supported in versions 1.1 and later.
   - JBoss Application Server, version 4.0 and earlier.
   - JBoss Application Server, JSR-160 connection.
   - WebLogic Server, version 9 and newer. Connector is provided for T3 protocol.
5. The first time that you run the JMX browser, there are no items in the **MBean server** scroll down menu. To add connections, click the **Add**) button.

   Use the **Edit** button to modify or delete the connection that you already defined and selected in the scroll down menu. The connection definitions are stored in the workspace, so that, when you create a connection, it is remembered. Complete the following steps to create a connection. If you already have a connection, skip to the next step.

   a) To create a connection to an MBean Server, click **Add** to add a connection or to edit an existing connection.

      The **Java Management Extensions (JMX) Browser** window is shown when no connections are defined.

   b) After you click **Add** to add a connection, the **Select Connection Type** page opens.

c) Use the MBean Server Connection wizard to connect to an MBean server. The new connections that are listed on the page are selections that you can make to create connection. You can use the list of existing connections to create a new connection using an existing connection as a template. Select one of the new connection types and click **Next** to begin creating a connection.

d) After you select a connection type, you might be asked to select a more specific type of connection. Two templates that are based on the **Standard JMX Connections (JSR-160)** connection type are shown. Select the template that is most appropriate for your MBean server and click **Next**.

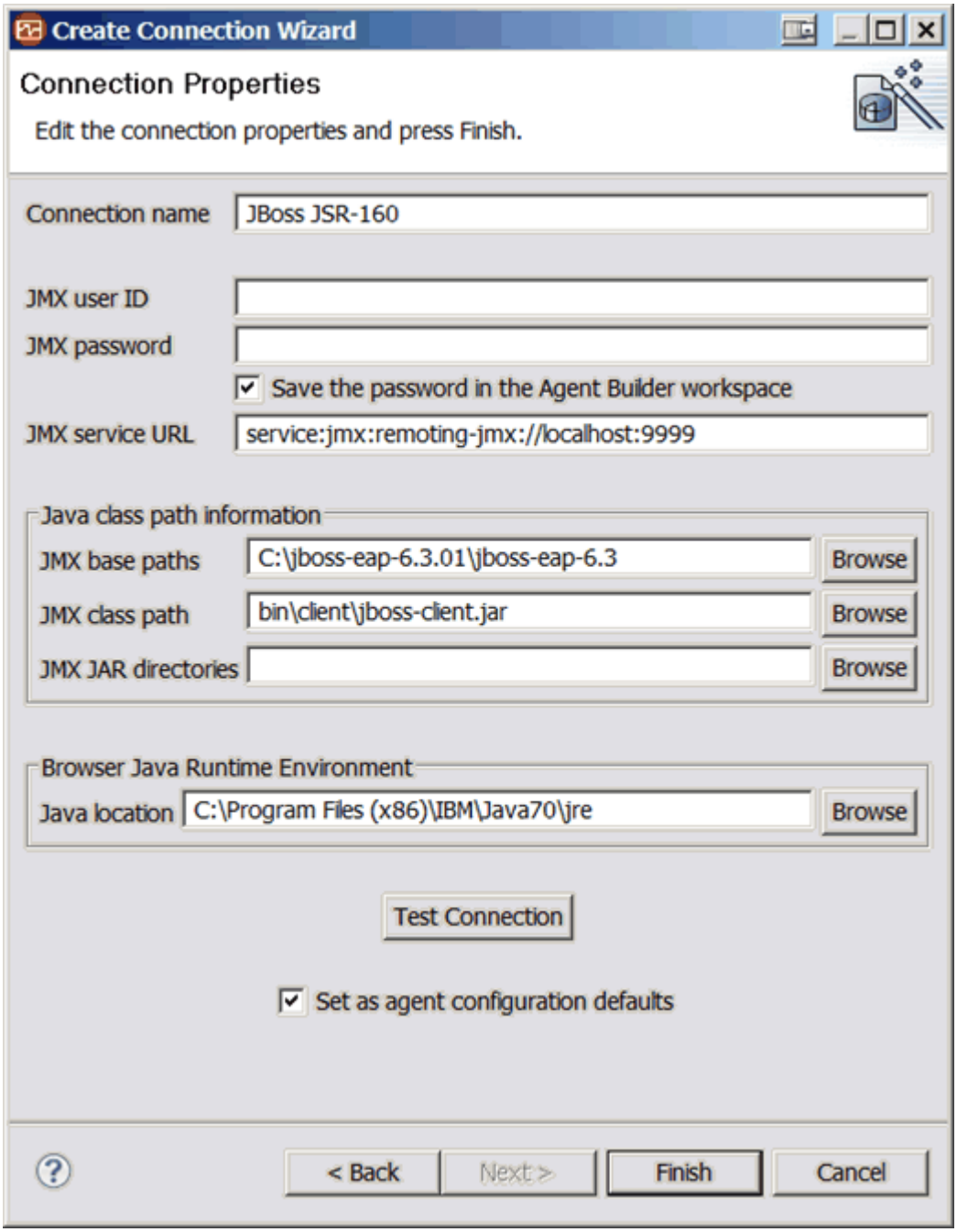


Figure 5. JMX connection properties

The **Connection Properties** page (Figure 5 on page 85) contains the details on how to connect to an MBean server. You must complete the page with details about your MBean server.

Important: If your data source connects to a remote WebSphere Application Server, ensure that WebSphere Application Server is also installed on the host that is running Agent Builder and set

the**Java location** setting to the Java runtime environment that the local WebSphere Application Server uses.

e) Select the **Save the password in the Agent Builder workspace** check box if you want to save the password for this connection.

f) Optional: Select **Set as agent configuration defaults** if you want the defaults for JMX to be copied from these connection properties.
For example, in Figure 5 on page 85 the default **JMX base path** is `C:\jboss-eap-6.3.01\jboss-eap-6.3`, the **JMX service URL** is `service:jmx:remoting-jmx://localhost:9999` and the **Java location** is `C:\Program Files\IBM\Java70\jre`

i) After you specify the properties that are required to connect, click **Test Connection** to ensure that the connection can be established. If the connection is not successful, correct the necessary properties.

ii) When the connection is successful, click **Finish** to return to the browser that uses the connection you configured.

The Java class path information in the **Connection Properties** page contains three fields. These fields must be completed as necessary to connect to an MBean server that requires Java classes that are not included in the Java runtime environment. Normally, the MBean server you want to connect to must be installed on the same system as the Agent Builder. In this case, specify the directory where the application that contains the MBean server was installed as the **JMX base paths** field. The **JMX Jar Directories** field then lists the directories relative to the Base Paths directory that contain the JAR files that are required to connect to the MBean server. The **JMX class path** field can be used to include specific JAR files. The JAR files that are listed in the **JMX JAR Directories** field are not required to be listed separately in the **JMX class path** field.

Any of the fields can contain more than one reference; separate the entries by a semicolon. These values are the same values that are required when you configure the agent. For more information, see (“JMX configuration” on page 91).

6. After you select a connection, the JMX Browser downloads information about the MBeans from the JMX server. This information is shown in the following four areas of the **JMX Browser** window (Figure 6 on page 87):

Directions for screens that begin with Java Management Extensions (JMX) Browser window to **Runtime Configuration** tab of the Agent Editor: From the **JMX Information** page, select **Browse**. From the browser (JMX Browser with no connection-selected), select **Add**. From the **JMX Connection Selection** page select **JBoss**, then select **Next**. From the **JMX Connection Properties** page, customize two Connection Properties: JBoss provider URL: `jnp://wapwin3.tivlab.raleigh.ibm.com:1099/` and **JBoss Jar Directories**: The full path to the directory that contains the following JAR files: `jbossall-client.jar`, `jboss-jmx.jar`, `jboss-jsr77-client.jar`, `jboss-management.jar`. Select **Finish**. This configuration sets up your JBoss connection so you can get similar screens as shown here.

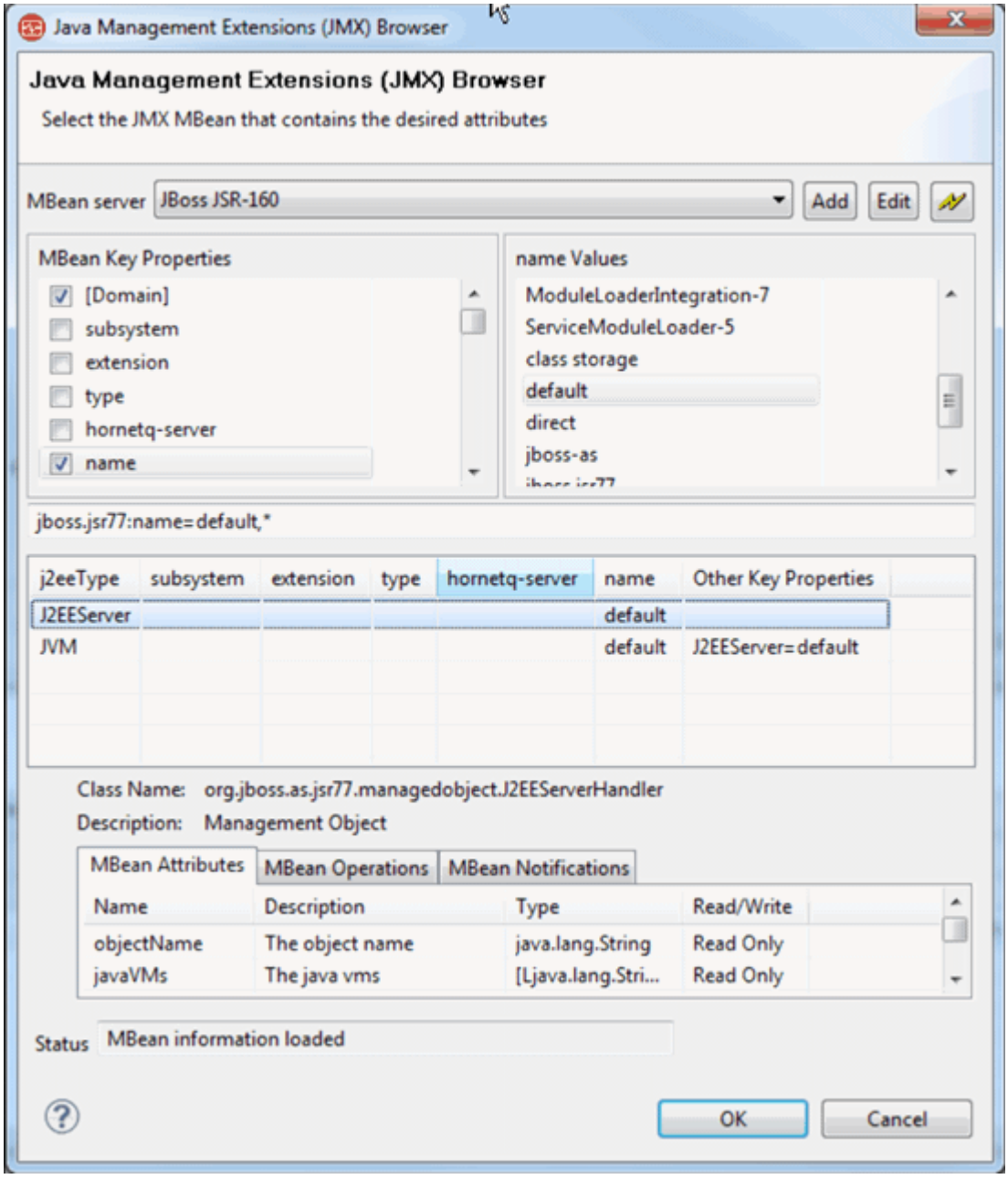


Figure 6. Java Management Extensions (JMX) Browser window

- **MBean Key Properties** area: This area is a collection of every unique Object Name key that is found from all the MBeans on the server. The **[Domain]** entry is special because it is not really a key. However, the **[Domain]** entry is treated as an implied key for the value of the MBean domain. Select an item from this list, and the MBeans that contain that key property are found. The list of values of the key property are shown in the **Selected Key Property Values** list. When you check a key property, it is included in the Object Name pattern for the data source.
- **Selected Key Property Values** area: This area shows the values of the currently selected MBean Key Property from all MBeans. Selecting one of these values checks the MBean key property. The selection also updates the Object Name Pattern shown in the message field with the MBean key property name and value.
- A table lists all MBeans matching the Object Name Pattern: As you select Key Properties and Values from the MBean Key Properties and Selected Key Property Values lists, you see the Object Name Pattern update. You also see the list of MBeans in this table change to reflect the list of MBeans that match the pattern you selected. If you have a pattern that is not matching any MBeans, you can

clear entries in the MBean Key Properties list. You clear entries by clicking the check box next to a key that is being used by your pattern and removing the check mark. Also, you can manually edit the pattern to find the MBeans you are looking for. The pattern *:* selects all MBeans.

You can use this table to browse the MBeans from the server and decide which ones contain the data you want to monitor. To help browse a potentially large number of MBeans, you can sort by any key attribute (from the menu or by clicking a column header). You can also show any key attribute in any column by selecting **Show Key Property** from the menu. When you see a key property value in the table that identifies MBeans you want to monitor, right-click on that value and choose **Select only MBeans with Key Property** from the menu.

- A table that contains details for a selected MBean: The JMX Browser shows you information about a single MBean. To see details for an MBean, you select the MBean from the table that shows the list of MBeans matching the current filter. The key information about the MBean is the list of Attributes, Operations, and Notifications it defines.

To create a data source from the JMX Browser, use the four panels that were described earlier to build an Object Name Pattern. Build the Object Name Pattern to match a set of MBeans that each contains the monitoring data you want to collect. For instance, if you wanted to monitor data from all of the ThreadPool MBeans, use the following steps:

a) Select **type** from the **MBean Key Properties** panel. Selecting **type** causes the values in the **Selected Key Properties Values** to be updated to list all unique values from the type key of any MBean.

b) Select **ThreadPool** from the list of values for the type key. After you select **ThreadPool**, the type key property name is selected in the **MBean Key Properties** panel and the Object Name Pattern is updated to *:type=ThreadPool,*. The list of MBeans is also updated to show only the MBeans that match this pattern.

c) Select one of the MBeans from the MBean list to see the attributes, operations, and notifications available for the MBean. If your MBean list contains more MBeans than you want to monitor, you must continue this procedure of selecting key properties and values. Continue until you have the Object Name Pattern that identifies the set of MBeans you want to monitor. You can also open a menu in the MBean list to update the Object Pattern with key property values shown in the table.

7. When the object name pattern is correct, select an MBean from the table.

   All attributes of the selected MBean are the initial attributes in the new JMX data source. Some attributes might not contain data. After the JMX data source is created, review the attributes and remove any that are not significant. If the selected MBean has no attributes, you are warned that the data source is created with no attributes. If the selected MBean contains notifications, an Event data source is also created to receive notifications from the MBeans.

   Important: For every MBean attribute, Agent Builder creates an attribute in the new data set. For a numeric MBean attribute, Agent Builder creates a numeric attribute. For any object types, including `String`, Agent Builder creates a string attribute containing a string representation of the value. If an object from an MBean attribute is of the `javax.management.openmbean.CompositeData` type, and the Agent Builder browser can read the object itself, it creates several attributes, one for each object embedded in the `CompositeData` object. To include values internal to an object other than a `CompositeData` object (fields or method return values), you need to create an attribute that has a more complex metric name, as described in "Specific fields for Java Management Extensions (JMX) MBeans" on page 98.

8. Click **Finish** in the filled on JMX Information page.

   Data sources are created based on the MBean that was selected in the previous step. If no MBean was selected, an attribute group with no attributes is created. A warning is shown, giving you a chance to select an MBean. The notification data source has the word, **Event**, at the beginning of the data source name to distinguish it from the data source that shows attributes.

9. To change other JMX options for the agent, click **Global JMX Options**. With these options, you can:

   a) Choose whether JMX monitors are supported by this agent. If you want JMX monitor attribute groups and Take Action commands to be created, select **Include JMX monitor attribute groups and take actions**

See the next section for a description of JMX monitors.

b) Select the types of MBean servers your agent connects to when it is deployed.

There are several vendor-specific types of servers that are listed, along with a generic JSR-160-Compliant Server for standards-based servers. You can select as many as needed, but you must select only server types that support the MBeans that are being monitored. You must select at least one. If you select more than one, at agent configuration time you are prompted to specify which type of server you want to connect to.

10. Click **OK** after you select the wanted option.
11. Optional: You can test this attribute group by clicking **Test**. For more information about testing, see ("Testing JMX attribute groups" on page 101)
12. Optional: You can create a filter to limit the data that is returned by this attribute group by clicking **Advanced**. For more information about filtering data from an attribute group, see "Filtering attribute groups" on page 45
13. Click **Next**.
14. On the **Select key attributes** page, select key attributes or indicate that this data source produces only one data row. For more information, see ("Selecting key attributes" on page 15).
15. Click **Next**.

    The **JMX Agent-Wide Options** window shows the types of application servers that the Agent Builder supports. If you previously selected **Set as agent configuration defaults** on the **Connection Properties** page, the type of application server that you browsed to is automatically selected.
16. In the **JMX Agent-Wide Options** window (Figure 7 on page 90), select any other types of application servers to which you want your agent to be able to connect.

    Note: In the example shown, choosing **JBoss Application Server JSR-160 connection** is the same as choosing **JSR-160-Compliant Server** except that different default values are supplied.

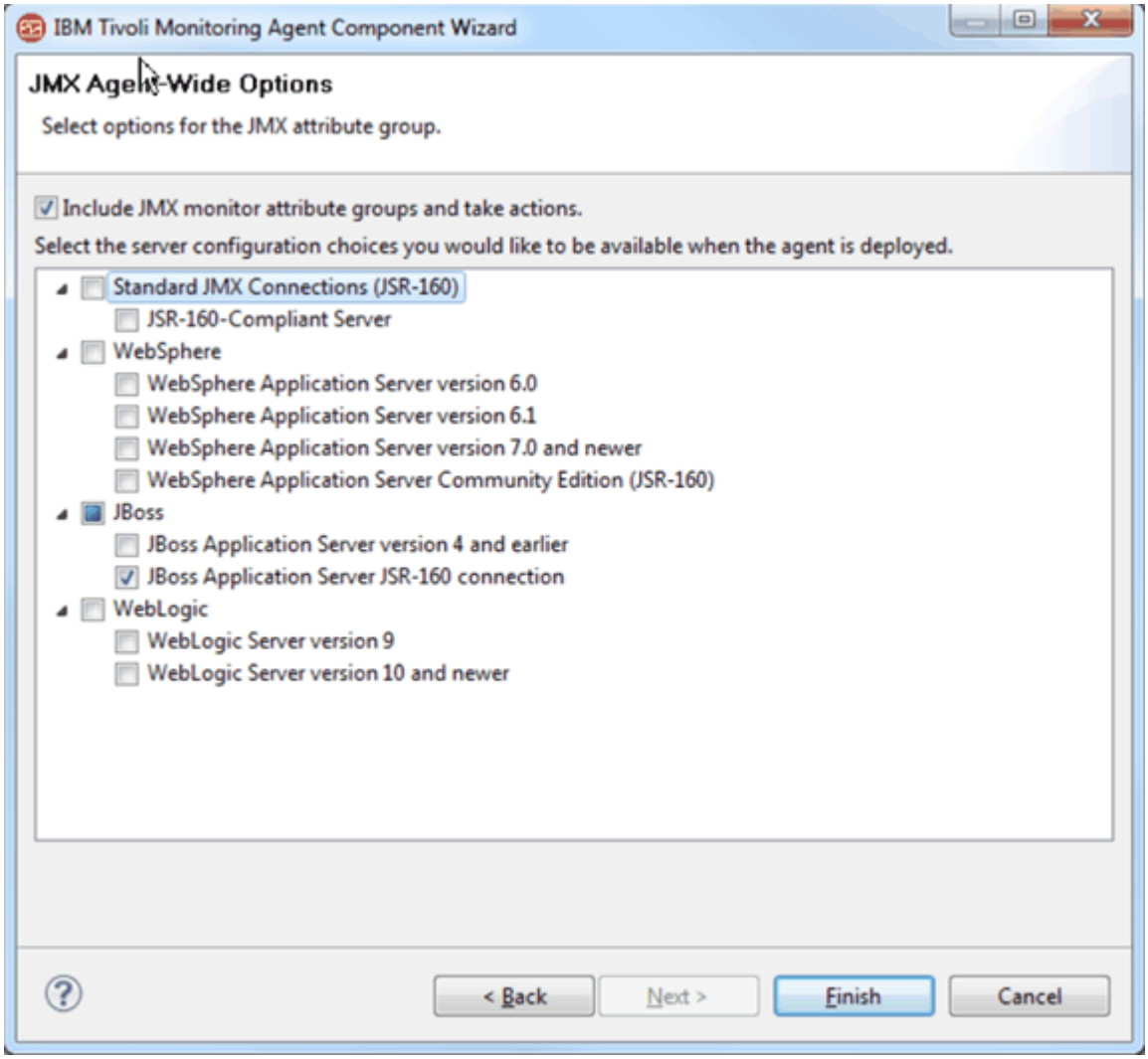


Figure 7. JMX Agent-Wide Options window

17. Do one of the following steps:
    - If you are using the New Agent wizard, click **Next**.
    - Click **Finish** to save the data source and open the Agent Editor.
18. If you want to change the types of application servers to which you can connect after the agent is created, click **Global JMX Options** in the **JMX Data Source Information** area.
19. In the **JMX Agent-Wide Options** window, change any selections that you want.
20. Click **OK**.
21. To view the configuration sections and properties that were automatically generated, click the **Runtime Configuration** tab of the Agent Editor.

    The default value of the `JBoss base paths` property has the value that was entered in the JMX browser.

What to do next

For more information about the attribute groups for JMX events, see "JMX Event attribute groups" on page 307,

JMX configuration

When you define a JMX data source in your agent, some configuration properties are created for you.

JMX runtime configuration is unique because it provides you with some control over how much configuration is displayed. The JMX client for the agent can connect to several different types of application servers. However, it is not necessary to support all of those types of application servers in any one agent. You can determine which types of application servers to support, and unnecessary configuration sections are not included in the agent.

In most cases, an agent is designed to monitor one JMX application server type. When you create the JMX data source, you can use the JMX Browser. When you use the JMX Browser, the JMX server configuration options that are used to browse the MBean server are added to your agent automatically. To change the types of application servers to which you can connect after the agent is created, click **Global JMX Options** in the **JMX Information** area. In the **JMX Agent-Wide Options** page, change any selections that you want.

You can design a generic agent that monitors more than one type of JMX application server. In this case, more than one JMX server configuration choice can be selected on the **JMX Agent-Wide Options** page. When more than one type of JMX connection is supported, the runtime configuration prompts you for the connection type that are used for that agent instance.

Note: An instance of an agent can connect only to one type of JMX application server. Subnodes can be used to connect to different JMX application servers of the same type within an agent instance. To connect to more than one type of JMX application server, you must configure at least one agent instance for each JMX application server type.

You can view, add, and change the configuration properties by using the Agent Editor. For instructions, see "Changing configuration properties by using the Agent Editor" on page 207. If a JMX data source is defined in a subnode, you are also able to specify Subnode Configuration Overrides. For instructions, see "Subnode configuration" on page 193.

If you define a JMX data source in your agent, the agent must use Java to connect to the JMX application server. Java configuration properties are added to the agent automatically.

The following Java configuration properties are specific to the agent runtime configuration:

Java Home
Fully qualified path that points to the Java installation directory

Configure the agent to use the same JVM that the application you are monitoring uses, particularly for the WebLogic Server and WebSphere Application Server.

JVM Arguments
Specifies an optional list of arguments to the Java virtual machine.

Trace Level
Defines the amount of information to write to the Java trace file. The default is to write-only Error data to the log file.

Note: Agent Builder does not require these properties because it uses its own JVM and logging, which is configured through the JLog plug-in.

If you define a JMX data source in your agent, the following required, common configuration fields are added to the agent automatically:

Connection
The type of connection to the MBean server

User ID
User ID that is used to authenticate with the MBean server.

Password
Password for the user ID.

Base paths
Directories that are searched for JAR files that are named in **Class path**, or directories that are named in **JAR directories**, that are not fully qualified. Directory names are separated by a semi-colon (`;`) on Windows, and by a semi-colon (`;`) or colon (`:`) on UNIX systems.

Class path
Explicitly named JAR files to be searched by the agent. Any that are not fully qualified are appended to each of the Base Paths until the JAR file is found.

JAR directories
Directories that are searched for JAR files. Directory names are separated by a semi-colon (`;`) on Windows, and by a semi-colon (`;`) or colon (`:`) on UNIX systems. The JAR files in these directories are not required to be explicitly identified; they are found because they are in one of these directories. Subdirectories of these directories are not searched. Any directory names that are not fully qualified are appended to each of the Base Paths until the directory is found.

Note: For remote monitoring, the JAR files and all of their dependent JAR files must be installed locally on the computer where the agent is running. These JAR files are the files that are required to connect to the application that is being monitored. These JAR files must be configured in **JAR directories**, and in **Base paths** and **Class path**. In addition, locally install a supported JVM for the application you are monitoring and specify the path in the **Java Home configuration** field.

Examples:

- For WebLogic 10, the class path is `server/lib/wlclient.jar;server/lib/wljmxclient.jar`. The base path points to the WebLogic application server directory where the `server/lib` directory is located.
- For WebSphere, the base path points to the location where the WebSphere Application Server is installed. Multiple base paths are listed in this example to provide a default for Windows and UNIX. The class path lists the JAR files relative to the base path. The relative value `lib` for the **JAR directories** field causes all JAR files in this directory under the base path to be loaded.
  - **Base paths**: `C:\Program Files\IBM\WebSphere\AppServer;/opt/IBM/WebSphere/AppServer`
  - **Class path**: `runtimes/com.ibm.ws.admin.client_6.1.0.jar;plugins/com.ibm.ws.security.crypto_6.1.0.jar`
  - **JAR directories**: `lib`

Depending on which JMX server types are selected in the JMX Agent-Wide Options page, some or all of the following configuration properties are added. Default values are provided by the Agent Builder, and can be modified:

JSR-160 Compliant Server connection-specific configuration properties:

JMX Service URL
JMX Services URL to connect to for monitoring.

WebSphere Application Server version 6.0 and later connection-specific configuration properties:

Host name
Host name of the system where the application server you are monitoring is located. For local monitoring, the name is the local system name. For remote monitoring, the name is the host name of the system where the application server is located.

Port
Port number to use on the host name to be monitored.

Connector protocol
Connector protocol to be used by the monitoring connection. RMI and SOAP are supported.

Profile name
Name of the profile to use for configuring the connection.

JBoss Application Server (non JSR-160) connection-specific configuration properties:

JNDI Name
JNDI Name that is used to look up the MBean server.

Provider URL
JMX Services provider URL to connect to for monitoring.

WebLogic Server connection-specific configuration properties:

Service URL
JMX Services provider URL to connect to for monitoring that includes the JNDI name.

Note: If WebSphere administrative security is enabled, you must make sure that client login prompts are disabled in the appropriate client connection properties files. For RMI connections, to prevent clients from prompting the user, you must modify the *com.ibm.CORBA.loginSource* property in the `sas.client.props` file in the profile properties directory of your WebSphere Application Server. For a SOAP connection, you must modify the *com.ibm.SOAP.loginSource property*in the `soap.client.props` file in the same directory. In both cases, the *loginSource* property must be set to not contain a value.

You can view, add, and change the configuration properties by using the Agent Editor. See ("Changing configuration properties by using the Agent Editor" on page 207). If a Windows data source is defined in a subnode, you can also specify Subnode Configuration Overrides. See "Subnode configuration" on page 193.

JMX notifications

In addition to providing monitoring data when requested, some MBeans also provide notifications.

A notification is an object that is generated by an MBean that is passed to registered listeners when an event occurs.

Agents that are built by the Agent Builder can define attribute groups that contain values from notifications rather than MBeans.

When the agent is started, a `notification listener` is registered with each MBean that matches the MBean pattern of the attribute group. The attribute group then displays one row per notification received. Each column contains one item of data from the notification. The data wanted from the notification is defined by a column value similar to the way column data is defined for MBeans.

For non-event based attribute groups, data is collected when needed. For event-based attribute groups, the agent maintains a cache of the last 100 events received. These events are used to respond to requests from the Tivoli Enterprise Portal. The events are forwarded immediately for analysis by situations and warehousing.

JMX monitors

In addition to providing monitoring data when requested, some MBeans also provide monitors.

The JMX Provider supports the ability for an agent to create JMX Monitors. A JMX Monitor is an MBean that the JMX agent creates on the JMX Server. It monitors the value of an attribute of another MBean and sends a notification when that value meets some criteria. Thresholds are defined that enable the Monitor to report on specific attribute values.

Not all application servers support the creation of monitors from a JMX client, which is true for current releases of WebSphere Application Server. JMX Monitors and Take Action commands can be included in your agent by selecting **Include JMX monitor attribute groups and take actions** under **Global JMX Options**.

Any MBean that reports on an attribute of another MBean can be considered a monitor. In practice, JMX defines three concrete monitor classes, which are the types of monitors that are created. The following concrete monitor types are created:

- String monitor – watches a string attribute, reports equality, or inequality of that string.

- Gauge monitor – watches a variable numeric attribute, reports up or down movement beyond threshold values.
- Counter monitor – watches an increasing numeric attribute, reports when it reaches a threshold value or increases by a certain amount.

The following attribute groups might be automatically added to the agent to collect or represent JMX Monitor notifications:

- `Registered Monitors`

  This attribute group displays all of the JMX Monitors that are added by the user.
- `Counter Notifications`

  This attribute group reports all notifications that are received from Counter Monitors.
- `Gauge Notifications`

  This attribute group reports all notification received from Gauge Monitors.
- `String Notifications`

  This attribute group reports all notifications that are received from String Monitors.

Take Action commands for JMX monitors

A monitor is created by running a Take Action command.

Three Take Action commands are defined, one to create each type of monitor, and a fourth Take Action is defined to delete an existing monitor. A 256-character limit applies to Take Action commands.

The monitor attribute groups are a part of every JMX agent that is built, including all agents that are built by the Agent Builder. The four Take Action commands are available to all agents, though they cannot be used unless it is a JMX agent.

JMX Add String Metric Watcher

Use this Take Action command to create a monitor to watch a string attribute.

Parameters

MBean pattern
All MBeans matching this pattern are monitored by this monitor.

Observed attribute
Name of the MBean string attribute that is being watched.

Notify match
True if a notification is to be sent when the monitored string matches a reference value, false if not (defaults to false).

Notify differ
True if a notification is to be sent when the monitored string does not match the reference value, false if not (defaults to true)

Reference value
String to compare with the observed attribute.

A default means that the argument is not specified.

Example: Request a notification when a service is stopped

```
STRING_METRIC_WATCHER [*:type=Service,*] [StateString] [true] [false] [Stopped]
```

Where:

`*:type=Service,*`
MBean pattern: Monitors any MBean with a key property named type whose value is Service.

`StateString`
Observed attribute: A string attribute that is common to all MBeans of `type=Service`.

true
Notify match: You want a notification to be sent to your agent when the `StateString` attribute matches your reference value of Stopped.

false
Notify differ: You do not want to be notified when the Service attribute does not match Stopped.

Stopped
Reference value: When the `StateString` attribute changes to the value Stopped, a notification is sent.

JMX Add Gauge Metric Watcher

Use this Take Action command to create a monitor to watch a gauge attribute.

Parameters

MBean pattern
All MBeans matching this pattern are monitored by this monitor.

Observed attribute
Name of the MBean string attribute that is being watched.

Difference mode
True if the value monitored is the difference between the actual current and previous values of the attribute. False if the value monitored is the actual current value of the attribute (defaults to false).

Notify high
True if a notification is to be sent when an increasing monitored value crosses the high threshold, false if not (defaults to true).

Notify low
True if a notification is to be sent when a decreasing monitored value crosses the low threshold, false if not (defaults to true).

High threshold
Value that the observed attribute is expected to stay under.

Low threshold
Value that the observed attribute is expected to stay over.

Example: Request a notification when free memory goes under 10 Mb

```
GAUGE_METRIC_WATCHER [ServerInfo] [FreeMemory] [false] [false] [true] [30000000] [10000000]
```

Where:

***:type=ServerInfo**
MBean pattern: Monitors any MBean whose name has a single key property named type whose value is ServerInfo.

FreeMemory
Observed attribute: Numeric attribute that fluctuates up or down, this one indicating the amount of free memory in the application server.

false
Difference mode: Monitors the actual attribute value, not the difference between one observation and another.

false
Notify high: Notification is not sent when free memory goes up.

true
Notify low: Notification is not sent when the free memory becomes too low.

30000000
High threshold: Even though you are not concerned with passing a high threshold, you need a reasonable high threshold value. A second low threshold notification does not occur until the attribute value hits or passes the high threshold.

10000000
Low threshold: Low threshold value that you want to be notified about.

JMX Add Counter Metric Watcher

Use this Take Action command to create a monitor to watch a counter attribute.

Parameters

MBean pattern
All MBeans matching this pattern are monitored by this monitor.

Observed attribute
Name of the MBean string attribute that is being watched.

Initial threshold
Value that the observed attribute is compared.

Offset
Value added to the threshold after the threshold is exceeded to create a changed threshold.

Modulus
Maximum value of counter after which it rolls over to 0.

Difference mode
True if the value monitored is the difference between the actual current and previous values of the attribute. False if the value monitored is the actual current value of the attribute (defaults to false). This mode effectively turns on rate-of-change monitoring.

Granularity period
Frequency with which measurements are taken (defaults to 20 seconds). Most important if difference mode is true

Example: Request a notification when any server has three or more errors

```
COUNTER_METRIC_WATCHER [*:j2eeType=Servlet,*] [errorCount] [3] [4] [] [diff] [gran]
```

Where:

`*:j2eeType=Servlet,*`
MBean pattern: Monitors any J2EE servlet MBean whose name has a single key property named type whose value is `ServerInfo`

`errorCount`
Observed attribute: Increasing numeric attribute, this one indicating the number of errors of the servlet.

3
Initial threshold: You want to be notified when `errorCount` meets or exceeds 3.

4
Offset: When you get a notification for three errors, 4 is to the previous threshold of 3 to make a new threshold of 7. A second notification will be sent after `errorCount` reaches 7; a third at 11; a fourth at 15, and so on. Zero or none is not valid because it expects the counter to always increase and not increasing the offset would not make sense for a counter.

Modulus:
errorCount has no architected maximum value, so use an unreasonably high value.

false
Difference mode: You are concerned with absolute error counts. Difference is true if you are interested in the rate that errorCount was increasing.

Granularity period: Not set, so take the default granularity period of 20 seconds. Granularity period is available for all monitor types. However, it is shown with a counter monitor so that a meaningful rate of change (with difference mode=true) can be determined.

JMX Delete Metric Watcher

Use this Take Action to delete a monitor.

Parameter

Number
Monitor number as shown in the REGISTERED_MONITORS table

Example: Delete monitor number 2

```
DELETE_WATCHER [2]
```

Where:

2=
Number of monitor to be deleted.

JMX operations

In addition to providing monitoring data when requested, some MBeans also provide operations.

Agents that have JMX data sources include the JMX_INVOKETake Action command that you can use to run JMX operations against the server you are monitoring.

Take Action command syntax

The action has the following syntax:

```
JMX_INVOKE [MBean pattern] [Operation name] [Argument 1] [Argument 2]
[Argument 3] [Argument 4]
```

Where:

MBean pattern
MBean query that selects the MBeans on which the operation runs. If the pattern matches more than one MBean, the operation runs on each of the matched MBeans.

Operation name
Name of the MBean operation to run.

Argument 1, Argument 2, Argument 3, Argument 4
Optional arguments that can be provided to the MBean operation. Arguments must be a simple data type such as a string or an integer.

The JMX invoke Take Action command returns success if the operation is successfully run. If the operation returns a value, that value is written to the JMX data provider log file.

Example: Start an operation to reset a counter

This action runs the resetPeakThreadCount operation on the Threading MBeans:

```
JMX_INVOKE [*:type=Threading,*] [resetPeakThreadCount][] [] [] []
```

Where:

:type=Threading,
MBean Pattern: This pattern matches all MBeans that have a type of Threading.

resetPeakThreadCount
Operation name: The operation that is run on every MBean that matches the pattern.

[] [] [] []
Argument 1, 2, 3, 4: The arguments are not needed for this operation. They are specified only to comply with the syntax of the action.

Example: Start an action with an argument

This action runs the getThreadCpuTime operation on the Threading MBeans. The result is logged to the JMX data provider trace file.

```
JMX_INVOKE [*:type=Threading,*] [getThreadCpuTime] [1] [] [] []
```

Where:

:type=Threading,
MBean Pattern: This pattern matches all MBeans that have a type of Threading

getThreadCpuTime
Operation name: The operation that is run on every MBean that matches the pattern.

1
Argument 1: The thread id that is being queried.

[] [] []
Argument 2, 3, 4: These arguments are not needed for this operation. They are specified as empty arguments to comply with the Take Action command syntax.

Running the JMX_INVOKE Take Action command

The agent developer cannot expect the user to run the JMX_INVOKE Take Action command. Instead, more actions must be developed that run the JMX_INVOKE Take Action. If possible in these actions, hide the details such as the operation name and the MBean pattern from the user.

Starting and stopping JMX monitors

JMX monitors are persistent across starts and stops of the agent and the JMX server.

If the agent detects that the JMX server was recycled, it reregisters the monitors. If the agent is recycled, monitors are reregistered. The monitor definitions are stored in a file that is named `default_`*`instanceName`*`.monitors` where *instanceName* is the agent instance name or default if it is a single instance agent. This file is in the following directory (note that *xx* denotes the two character product code):

- Windows systems: `TMAITM6/k`*`xx`*`/config`
- UNIX and Linux systems: *`architecture`*`/`*`xx`*`/config` (see “New files on your system” on page 246 for information about determining the architecture value)

If the agent is restarted, it uses the monitor definitions file to restore the monitors.

Specific fields for Java Management Extensions (JMX) MBeans

The syntax of the metric name for a JMX Attribute group must follow certain rules when specified on the **Attribute Information** window.

The syntax of the metric name for a JMX Attribute group consists of tokens that are separated by a period. The tokens form primary values and optionally secondary values:

- **Primary value**: a value that is obtained directly from the MBean or Notification in a specific row of the table. Primary values from an MBean are obtained either from an MBean attribute or from the invocation of an MBean operation (method call). Primary values from a Notification are obtained from a field or invocation of a method on the Notification object. Primary values can be primitive types, or can be Java objects.
- **Secondary value**: a value that is obtained by further processing a primary value or other secondary value. Secondary values are processed internally to the engine and do not involve calls to the JMX

server. If the primary (or other secondary value) is a Java object, a secondary value is the result of fetching a public field from that object. A secondary value can also be the result of a method call on that object. Such secondary values are obtained by using Java introspection of the primary (or other secondary) Java object. If the primary (or other secondary) value is a Java String in the form of an MBean name, the secondary value can be the domain. The secondary value can also be any of the properties that make up the MBean name.

The following syntax describes the format for the **Metric name** field:

```
Metric Name    =    PrimaryValue [ .SecondaryValue ]
PrimaryValue    =    Attribute.attributeName |
        Method.methodName |
        Domain |
        Property.propertyName |
        Field.fieldName |
        Name
SecondaryValue    =    Field.fieldName |
        Method.methodName |
        Domain |
        Property.propertyName |
        Explode |
        ElementCount
```

```
propertyName    =    the name of a key property in an MBean ObjectName
attributeName    =    the name of an MBean attribute
methodName    =    a zero-argument operation of an MBean or a zero-argument method
of a Notification or other Java object.
methodName(argument) = A single-argument operation of an MBean or a
single-argument method of a Notification or other Java object. The
argument will be passed to the method as a string.
fieldName    =    the name of a public instance variable in a Notification or
other Java object
notificationMethod    =    the name of a public zero-argument method of a
Notification object
```

By including only a primary value in the metric name definition, the data that is collected can be any of the following items:

- MBean domain
- MBean string value
- Key property from the MBean name
- Numeric or string attribute value on an MBean attribute (including the full name of another MBean). A numeric or string return value from an operation of a MBean.
- Value of a numeric or string public instance variable in a Notification object
- Numeric or string return value from an operation of a Notification.

By adding a secondary value to the definition of a metric, you can drill down into the primary value of a Java object. Also, you can start a public method or fetch a public instance variable.

By adding a secondary value to another secondary value in the definition of the metric, you can drill down into a secondary value object. You can continue as deeply as objects are nested inside an MBean or a Notification.

Tokens that make up primary and secondary values are either keywords or names. In most cases, a keyword token is followed by a name token. The following table shows some examples:

Metric name sample	Attribute group type	Description of the data returned
`Domain`	MBean	The domain portion of the MBean (the part before the colon).
`Name`	MBean	The full string representation of the MBean.
`Attribute.serverVendor`	MBean	MBean attribute serverVendor.

Metric name sample	Attribute group type	Description of the data returned
`Method.getHeapSize`	MBean	The value that is returned by the getHeapSize() on the MBean.
`Property.j2eeType`	MBean	The value of j2eeType is extracted from the MBean name.
`Field.Message`	Event (Notification)	The **Message** field in a notification.

The keywords `Attribute`, `Method`, and `Field` can return Java objects which contain other data. You can run operations on those objects by appending secondary value definitions. More examples:

Metric name sample	Attribute group type	Description of the data returned
Attribute.deployedObject.Method.getName	MBean	Takes the deployedObject attribute from the MBean and gets the result of the getName() method.
Attribute.eventProvider.Method.getException.Method.getDescription	MBean	Goes 3 deep: an attribute named eventProvider is presumed to be an object which has a getException() method. This method returns an object with a getDescription() method. That method is called and the return value is put in the column.
Attribute.HeapMemoryUsage.Method.get(used)	MBean	Takes the HeapMemoryUsage attribute from the MBean and gets the result of the get(String value) method. The string that is used is passed to the method as the argument. Only 1 argument can be provided and it must be a literal string value. Shows how you can collect data from an open MBean composite data structure.

`Domain` and `Property` can be used as keywords in secondary values if the previous value returned a String in the format of an MBean name. For example:

Metric name sample	Attribute group type	Description of the data returned
`Attribute.jdbcDriver.Property.name`	MBean	The attribute jdbcDriver returns an MBean name, and the key property, name, is extracted from the MBean name.
`Attribute.jdbcDriver.Domain`	MBean	The attribute jdbcDriver returns an MBean name, and the domain is extracted from the MBean name.

The `ElementCount` and `Explode` keywords run operations on arrays or collections of data.

- `ElementCount` – returns the number of elements in an array.
- `Explode` – explodes a row into several rows, one new row for each element of an array.

Examples of each of the keywords:

Metric name sample	Attribute group type	Description of the data returned
Attribute.deployedObjects.ElementCount	MBean	The MBean attribute deployedObjects is an array, and this column contains the number of elements in the array.
Attribute.deployedObjects.Explode.MBean.Property.j2eeType	MBean	Causes the table to have 1 row for each element in deployed objects. This column contains the j2eeType of the deployed Object.
Attribute.SystemProperties.Method.values.Explode.Method.get(key)	MBean	Causes you to get one new row for each entry in an open MBean tabular data structure. Each tabular data structure contains a composite data structure with an item named key, which is returned.

Testing JMX attribute groups

You can test the JMX attribute group that you created within Agent Builder.

Procedure

1. You can start the Testing procedure in the following ways:
   - During agent creation click **Test** on the **JMX Information** page.
   - After agent creation, select an attribute group on the Agent Editor **Data Source Definition** page and click **Test** . For more information about the Agent Editor, see Chapter 4, “Using the Agent Editor to modify the agent,” on page 17.

   After you click **Test** in one of the previous two steps, the **JMX Test** window is displayed
2. Select a connection from the list available under **Connection Name** or alternatively click **Add** to add a connection and follow the procedure that is detailed under “Monitoring Java Management Extensions (JMX) MBeans” on page 83.
3. Optional: Before you start testing, you can set environment variables, configuration properties, and Java information.

   For more information, see “Attribute group testing” on page 229. For more about JMX configuration, see “JMX configuration” on page 91.
4. Click **Start Agent**.

   A window indicates that the Agent is starting.
5. Click **Collect Data** to simulate a request from Tivoli Enterprise Portal or SOAP for agent data.

   The agent monitors the JMX Server for data. The **JMX Test** window collects and shows any data in the agent's cache since it was last started.
6. Optional: Click **Check Results** if the returned data is not as you expected.

   The **Data Collection Status** window opens and shows you more information about the data. The data that is collected and displayed by the Data collection Status window is described in “Performance Object Status node” on page 280
7. Stop the agent by clicking **Stop Agent**.
8. Click **OK** or **Cancel** to exit the **JMX Test** window. Clicking **OK** saves any changes that you made.

Related concepts

“Testing your agent in Agent Builder” on page 229

After you use Agent Builder to create an agent, you can test the agent in Agent Builder.

Monitoring data from a Common Information Model (CIM)

You can define a data source to receive data from a Common Information Model (CIM) data source. A data source monitors a single CIM class and places all values from this class into the data set that it produces. If the class provides several instances, the data set has multiple rows; you can filter by instance name to ensure the data set has one row.

About this task

This task describes the steps to configure a Common Information Model (CIM) data source.

Procedure

1. On the **Agent Initial Data Source** page or the **Data Source Location** page, click **Data from a server** in the **Monitoring Data Categories** area.
2. In the **Data Sources** area, click **CIM**.
3. Click **Next**.
4. On the **Common Information Model (CIM) Information** page **CIM information** area, make one of the following choices:
   - Complete the **Namespace** and **CIM class name** fields for the data that you want to collect.
   - Click **Browse** to browse a CIM repository on a specific system.

     The **Common Information Model (CIM) Class Browser** window is displayed. This browser connects to a CIM server and provides you with information about the classes that exist on that server.

     To browse a remote system, select a system from the **Hostname** list (if one is defined). Alternatively click **Add** to add the host name of the system where the CIM server is located.

     The syntax for specifying the host name is `http[s]://`*`hostname`*`:`*`port`*. If you provide the host name only, the Common Information Model (CIM) Class Browser connects by using a default URL of `http://`*`hostname`*`:5988`.

     If you provide a protocol without specifying a port, 5988 is used as the default for `http` or 5989 as the default for `https`.

     If you provide a port without specifying a protocol, `http` is used with the port provided.

     Provide a user ID and password for an account with read permission to the objects in the namespace that you want to browse. The window is updated with the information for the remote system.

     Agent Builder attempts to discover the namespaces available on the CIM Server. The discovered namespaces are displayed in the **Namespace** list. However, the Agent Builder might not be able to discover all namespaces that are available on the server. If you want to browse a namespace that is not listed in the **Namespace** list, click the plus (+) icon next to the **Namespace** list. Enter the name of the namespace in the field and click **OK**. If the namespace is present on the CIM server, the classes that are defined in the namespace are listed. The namespaces you type are saved and put into the **Namespace** list the next time you browse that particular CIM server.

     When you select a namespace from the **Namespace** list, the Agent Builder collects all of the class information for that particular namespace. Then, the Agent Builder caches this information so you can switch between namespaces quickly. If you want to force the Agent Builder to recollect the class information for a particular namespace, select the namespace and click **Connect**. Clicking **Connect** deletes any cached information, and causes the Agent Builder to recollect the class information.

You can click the **Search** (binoculars) icon to find your selection in the list. Type a phrase in the **Search phrase** field; specify your preference by clicking either the **Search by name** or **Search by class properties** fields; and click **OK**. If you find the item for which you are searching, select it and click **OK**.

5. On the Common Information Model (CIM) Information page, **Operating systems** area, select the operating systems on which the collection is to take place.
6. If you typed the Namespace and CIM class name in the **CIM information** area, do the following steps:
   a) Click **Next** to display the **Attribute Information** page and define the first attribute in the attribute group.
   b) Specify the information about the **Attribute Information** page, and click **Finish**.
7. If you browsed the CIM information, the Select key attributes page is displayed. On the Select key attributes page, select key attributes or indicate that this data source produces only one data row. For more information, see ("Selecting key attributes" on page 15).
8. If you browsed to the CIM information, click **Finish**.
9. Optional: You can test this attribute group by clicking **Test**. For more information about testing, see "Testing CIM attribute groups" on page 104
10. Optional: You can create a filter to limit the data that is returned by this attribute group by clicking **Advanced**. For more information about filtering data from an attribute group, see "Filtering attribute groups" on page 45
11. Do one of the following steps:
    a) If you are using the **Agent** wizard, click **Next**.
    b) Click **Finish** to save the data source and open the Agent Editor.

CIM configuration

Details about CIM configuration properties.

If you define a CIM data source in your agent, CIM configuration properties are added to the agent automatically. You can view, add, and change the configuration properties by using the Agent Editor. For instructions, see "Changing configuration properties by using the Agent Editor" on page 207). If a CIM data source is defined in a subnode, specify Subnode Configuration Overrides. For instructions, see "Subnode configuration" on page 193.

The following connection-specific configuration properties are on the CIM configuration page:

CIM Local or Remote
Local or remote authentication to the CIM server. Local/Remote Default value is Remote

CIM user ID
The user ID used to access the CIM server

CIM password
The password to access the CIM server

CIM host name
The host name to be accessed for CIM data

CIM over SSL
Use SSL for communication with the CIM server. The options are Yes and No. The default value is No.

CIM port number
The port number that is used for communication that is not secure.

CIM SSL port number
The port number that is used for secure communication. The default value is 5989. (The default value for Solaris 8 is normally different.)

Testing CIM attribute groups

You can test the CIM attribute group that you created, within Agent Builder.

Procedure

1. Start the Testing procedure in the following ways:
   - During agent creation click **Test** on the **CIM Information** page.
   - After agent creation, select an attribute group on the Agent Editor **Data Source Definition** page and click **Test**. For more information about the Agent Editor, see Chapter 4, "Using the Agent Editor to modify the agent," on page 17

   After you click **Test** in one of the previous two steps, the **Test Settings** window is displayed
2. Optional: Set environment variables and configuration properties before you start testing.

   For more information, see "Attribute group testing" on page 229.
3. Select or add a **Host name**.

   For more about adding a **Host name**, see "Monitoring data from a Common Information Model (CIM)" on page 102
4. Click **Start Agent**.

   A window opens indicating that the Agent is starting.
5. To simulate a request from Tivoli Enterprise Portal or SOAP for agent data, click **Collect Data**.

   The agent queries the CIM Server for data. The **Test Settings** window collects and shows any data in the agent's cache since it was last started.
6. Optional: Click **Check Results** if the returned data is not as you expected.

   The **Data Collection Status** window opens and shows you more information about the data. The data that is collected and displayed by the **Data Collection Status** window is described in "Performance Object Status node" on page 280
7. Stop the agent by clicking **Stop Agent**.
8. Click **OK** or **Cancel** to exit the **Test Settings** window. Clicking **OK** saves any changes that you made.

Related concepts

"Testing your agent in Agent Builder" on page 229
After you use Agent Builder to create an agent, you can test the agent in Agent Builder.

Monitoring a log file

You can define a data source to receive data from a text log file. The agent periodically parses the lines that are added the log file, and produces event information based on these lines. You can configure the way in which the agent parses the log into events. You can also configure the agent to filter and summarize the data. The resulting events are placed in a data set.

Before you begin

Note: The agent monitors log files that are in the same locale and code page that the agent runs in.

Procedure

1. On the **Agent Initial Data Source** page or the **Data Source Location** page, click **Logged Data** in the **Monitoring Data Categories** area.
2. In the **Data Sources** area, click **A Log File**.
3. Click **Next**.
4. On the **Log File Information** page, type the name of the log file you want to monitor in the **Log File Information** area.

   The file name must be fully qualified.

a) Optional: Part of the log file name can come from a runtime configuration property. To create a log file name, click **Insert Configuration Property** and select a configuration property.

b) Optional: The file can also be a dynamic file name. For more information, see (Appendix H, "Dynamic file name support," on page 361).

5. In the **Field Identification** area, click one of the following options:

Fixed number of characters
When selected, limits the number of characters.

With this option, each attribute is assigned the maximum number of characters it can hold from the log file. For example, if there are three attributes A, B, and C (in that order), and each attribute is a String of maximum length 20. Then, the first 20 bytes of the log record go into A, the second 20 into B, and the next 20 into C.

Tab separator
When selected, you can use tab separators.

Space separator
When selected, multiple concurrent spaces can be used as a single separator.

Separator Text
When selected, type in separator text.

Begin and End Text
When selected, type in both Begin and End text.

XML in element
When selected, type the name of the XML element to use as the record, or click **Browse** to define the element.

If you clicked **Browse**, the **XML Browser** window is displayed. If you use the browse function, the Agent Builder identifies all possible attributes of the record by looking at the child tags and their attributes.

Note: Unless you click **Advanced** and fill out the information in that window, the following assumptions are made about information that you complete:

- Only one log file at a time is monitored.
- Each line of the log file contains all the fields necessary to fill the attributes to be defined.

For more information about log file parsing and separators, see ("Log file parsing and separators" on page 112).

6. Optional: Click **Advanced** on the **Log File Information** page to do the following by using the **Advanced Data Source Properties** page:

- Monitor more than one file, or monitor files with different names on different operating systems or monitor files with names that match regular expressions.
- Draw a set of fields from more than one line in the log file.
- Choose **Event Filtering and Summarization Options**.
- Produce output summary information. This summary produces an additional attribute group at each interval. For more information about this attribute group, see "Log File Summary" on page 291. This function is deprecated by the options available in the Event Information tab.

a) To monitor more than one log file, click **Add** and type the name.

If more than one file is listed, a unique label must be entered for each file. The label can be displayed as an attribute to indicate which file generated the record. It must not contain spaces.

b) Optional: To select the operating systems on which each log file is to be monitored, follow these steps:

i) Click in the **Operating systems** column for the log file.

ii) Click **Edit**.

iii) In the **Operating Systems** window, select the operating systems.

iv) Click **OK** to save your changes and return to the **Advanced Data Source Properties** page.

c) Optional: Select **File names match regular expression** if the file name you are providing is a regular expression that is used to find the file instead of being a file name.

For more information, see Appendix F, "ICU regular expressions," on page 351. If you do not check this box, the name must be an actual file name. Alternatively it must be a pattern that follows the rules for file name patterns that are described in "Dynamic file name syntax" on page 361.

d) Optional: Select **One directory element matches regular expression** to match one subdirectory of the file name path with a regular expression.

You can select this option only if you also selected **File names match regular expression** in the previous step.

If regular expression meta characters are used in the path name, the meta characters can be used in only one subdirectory of the path. For example, you can specify `/var/log/[0-9\.]*/mylog.*` to have meta characters in one subdirectory. The `[0-9\.]*` results in matching any subdirectory of `/var/log` that consists solely of numbers and dots `(.)`. The `mylog.*` results in matching any file names in those`/var/log` subdirectories that begin with `mylog` and are followed by zero or more characters.

Because some operating systems use the backslash `(\)` as a directory separator it can be confused with a regular expression escape meta character. Because of this confusion forward slashes must always be used to indicate directories. For example, Windows files that are specified as `C:\temp\mylog.*` might mean the `\t` is a shorthand tab character. Therefore, always use forward slashes `(/)` on all operating systems for directory separators. The `C:/temp/mylog.*` example represents all files in the `C:/temp` directory that start with `mylog`.

e) In the **When multiple files match** list, select one of the following options:

- **The file with the highest numerical value in the file name**
- **The biggest file**
- **The most recently-updated file**
- **The most recently-created file**
- **All files that match**

Note: When you select **All files that match**, the agent identifies all files in the directory that match the dynamic file name pattern. The agent monitors updates to all of the files in parallel. Data from all files is intermingled during the data collection process. Its best to add an attribute by selecting **Log file name** in **Record Field Information** to correlate log messages to the log files that contain the log messages. Ensure that all files that match the dynamic file name pattern can be split into attributes in a consistent manner. If the log files selected cannot be coherently parsed, then its best to select **Entire record** in **Record Field Information** to define a single attribute. For more information about specifying **Record Field Information** for attributes, see step ("8" on page 108).

f) Choose how the file is processed.

With **Process all records when the file is sampled**, you can process all records in the entire file every time the defined sampling interval for the log monitor expires. The default interval is 60 seconds. This interval can be modified by using the *KUMP_DP_COPY_MODE_SAMPLE_INTERVAL* environment variable (specifying a value in seconds). The same records are reported every time unless they are removed from the file. With this selection, event data is not produced when new records are written to the file. With **Process new records appended to the file**, you can process new records that are appended to the file while the agent is running. An event record is produced for every record added to the file. If the file is replaced (first record changes in any way), the file is processed and an event is produced for each record in the file.

Note: If appending records to an XML log file, the append records must contain a complete set of elements that are defined within the XML element you selected as **Field Identification**.

g) If you chose to process new records that are appended to the file, you can also choose how new records are detected.

With **Detect new records when record count increases**, new records can be detected when the number of records in the file increases, whether the size of the file changes. This feature is useful when an entire log file is pre-allocated before any records are written to the file. This option can be selected for files that are not pre-allocated, but it is less efficient than monitoring the size of the file. With **Detect new records when the file size increases**, you can determine when a new entry is appended to a file in the typical way. There might be a brief delay in recognizing that a monitored file is replaced.

h) If you selected **Detect new records when the file size increases**, you can also choose how to process a file that exists when the monitoring agent starts.

Ignore existing records disables event production for any record in the file at the time agent starts. **Process ___ existing records from the file** specifies production of an event for a fixed number of records from the end of the file at the time the agent starts. **Process records not previously processed by the agent**: Specifies for restart data to be maintained by the monitoring agent so the agent knows which records were processed the last time that it ran. Events are produced for any records that are appended to the file since the last time the agent was running. This option involves a little extra processing each time a record is added to the file.

i) If you selected **Process records not previously processed by the agent**, you can choose what to do when the agent starts and apparently the existing file was replaced.

Process all records if the file has been replaced: If information about the monitored file and the restart data information do not match, events are produced for all records in the file. Examples of mismatches include: The file name is different, the file creation-time is different, the file-size decreased, the file last modification time is earlier than before. **Do not process records if the file has been replaced**: If the information about the monitored file and the restart data information do not match, disables processing of existing records in the file.

j) Click the **Record Identification** tab to interpret multiple lines in the log file as a single logical record.

Note: If you select **XML in element** as the field identification on the **Log File Information** page, the **Record Identification** tab does not display.

- **Single line** interprets each line as a single logical record.
- **Separator line** you can enter a sequence of characters that identifies a line that separates one record from another.

  Note: The separator line is not part of the previous or the next record.
- **Rule** identifies a maximum number of lines that make up a record and optionally a sequence of characters that indicate the beginning or end of a record. With **Rule**, you can specify the following properties:
  - **Maximum non-blank line** defines the maximum number of non-blank lines that can be processed by a rule.
  - **Type of rule**: Can be one of:
    - **No text comparison** (The Maximum lines per record indicates a single logical record).
    - **Identify the beginning of record** (Marks the start of the single logical record).
    - **Identify the end of record** (Marks the end of the single logical record).
  - **Offset**: Specifies the location within a line where the Comparison String must occur.
  - **Comparison Test**: Can either be **Equals**, requiring a character sequence match at the specific offset, or **Does not equal**, indicating a particular character sequence does not occur at the specific offset.
  - **Comparison String** defines the character sequence to be compared.
- **Regular Expression** identify a pattern that is used to indicate the beginning or end of a record. By using **Regular Expression**, you can specify the following properties:
  - **Comparison String** defines the character sequence to be matched.

    OR

– Beginning or end of record:
  - **Identify the beginning of record** marks the start of the single logical record.
  - **Identify the end of record** marks the end of the single logical record.

k) If you did select **Process all records when the file is sampled** earlier, click the **Filter Expression** tab. By clicking **Filter Expression** you can filter the data that is returned as rows based on the values of one or more attributes, configuration variables or both.

If you selected **Process new records appended to the file** earlier you cannot create a filter expression. For more information about filtering data from an attribute group, see (“Filtering attribute groups” on page 45).

l) If you selected **Process new records appended to the file** earlier, click the **Event Information** tab to select **Event Filtering and Summarization Options**.

For more information, see (Chapter 18, “Event filtering and summarization,” on page 259).

Note: The Summary tab can be present if the agent was created with an earlier version of Agent Builder. The summary tab is now deprecated by the Event Information tab

7. Optional: Click **Test Log File Settings** on the **Log File Information** page to start and test the data source. . Click **Test Log File Settings** after you select the options for the log source. When you test the log file data source and supply log file content, Agent Builder creates the attributes in the group automatically, based on the results of parsing the log. For more information about testing, see “Testing log file attribute groups” on page 113.
8. Use the following steps if you did not use the test function earlier and you typed the log file name in the **Log File Information** area of the **Log File Information** page:
   a) Click **Next** to display the **Attribute Information** page and define the first attribute in the attribute group.
   b) Specify the information, on the **Attribute Information** page, and click **Finish**.

   Note: When a log file attribute group is added to an agent at the default minimum Tivoli Monitoring version (6.2.1) or later, a `Log File Status` attribute group is included. For more information about the `Log File Status` attribute group, see (“Log File Status attribute group” on page 322).

   Along with the fields applicable to all data sources, the **Attribute Information** page for the log file data source has some additional fields in the **Record Field Information** area.

   The **Record Field Information** fields are:

   Next field
   Shows the next field after parsing, by using the delimiters from the attribute group (or special delimiters for this attribute from the Advanced dialog).

   Remainder of record
   Shows the rest of the record after previous attributes are parsed. This attribute is the last attribute, except for possibly the log file name or log file label.

   Entire record
   Shows the entire record, which can be the only attribute, except for possibly the log file name or log file label.

   Log file name
   Shows the name of the log file.

   Log file label
   Shows the label that is assigned to the file on the advanced panel.

   Note: Use the **Derived Attribute Details** tab only if you want a derived attribute, and not an attribute directly from the log file.
9. Click **Advanced** in the **Record Field Information** area to display the **Advanced Log File Attribute Information** page.
   a) In the **Attribute Filters** section, specify the criteria for data to be included or excluded.

Filtering attributes can enhance the performance of your solution by reducing the amount of data processed. Click one or more of the attribute filters:

- **Inclusive** indicates that the attribute filter set is an acceptance filter, meaning that if the filter succeeds, the record passes the filter, and is output.
- **Exclusive** indicates that the attribute filter set is a rejection filter, meaning that if the attribute filter succeeds, the record is rejected, and is not output.
- **Match all filters** indicates that all filters defined to the filter must match the attribute record in order for the filter to succeed.
- **Match any Filter** indicates that if any of the filters that are defined to the filter match the attribute record, the filter succeeds.

b) Use **Add**, **Edit**, and **Remove** to define the individual filters for an attribute filter set.

c) To add a filter, follow these steps:

i) Click **Add**, and complete the options in the **Add Filter** window as follows:

a) The **Filter criteria** section defines the base characteristics of the filter, including the following properties:

- **Starting offset** defines the position in the attribute string where the comparison is to begin.
- **Comparison string** defines the pattern string against which the attribute is defined.

  Type a string, pattern, or regular expression that is used by the agent to filter the data read from the file. The records that match the filter pattern are eliminated from the records that are returned to the monitoring environment, or are the only records returned. The result depends on whether you choose for the filter to be inclusive or exclusive.
- **Match entire value** checks for an exact occurrence of the comparison string in the attribute string. Checking starts from the starting offset position.
- **Match any part of value** checks for the comparison string anywhere in the attribute string. Checking starts from the starting offset position.

b) **The comparison string is a regular expression** indicates that the comparison string is a regular expression pattern that can be applied against the attribute string.

Regular expression-filtering support is provided by using the International Components for Unicode (ICU) libraries to check whether the attribute value examined matches the specified pattern.

To effectively use regular expression support, you must be familiar with the specifics of how ICU implements regular-expressions. This implementation is not identical to how regular expression support is implemented in `Perl`, `grep`, `sed`, Java regular expressions, and other implementations. See Appendix F, "ICU regular expressions," on page 351 for guidance on creating regular expression filters.

c) **Define an override filter** indicates that you want to provide a more specific filter comparison that overrides the base characteristics previously defined. This additional comparison string is used to reverse the filter result. When the filter is **Inclusive**, the override acts as an exclusion qualifier for the filter expression. When the filter is **Exclusive**, the override acts as an inclusion qualifier for the filter expression. (For more about **Inclusive** and **Exclusive**, see step "9" on page 108, and the examples that follow). The override filter has the following properties:

- **Starting offset** defines the position in the attribute string where the comparison is to begin.
- **Comparison string** defines the pattern string against which the attribute is matched.

  Type a regular expression that is used by the agent to filter the data read from the file. The records that match the filter pattern are eliminated from the records that are returned to

the monitoring environment, or are the only records returned. The result depends on whether you choose for the filter to be inclusive or exclusive.

d) **Replacement value** can be used to alter the raw attribute string with a new value. See Appendix F, “ICU regular expressions,” on page 351 for more details about special characters that can be used.

e) **Replace first occurrence** replaces the first occurrence that is matched by the comparison string with new text.

f) **Replace all occurrences** replaces all occurrences that are matched by the comparison string with new text.

ii) Click **OK**.

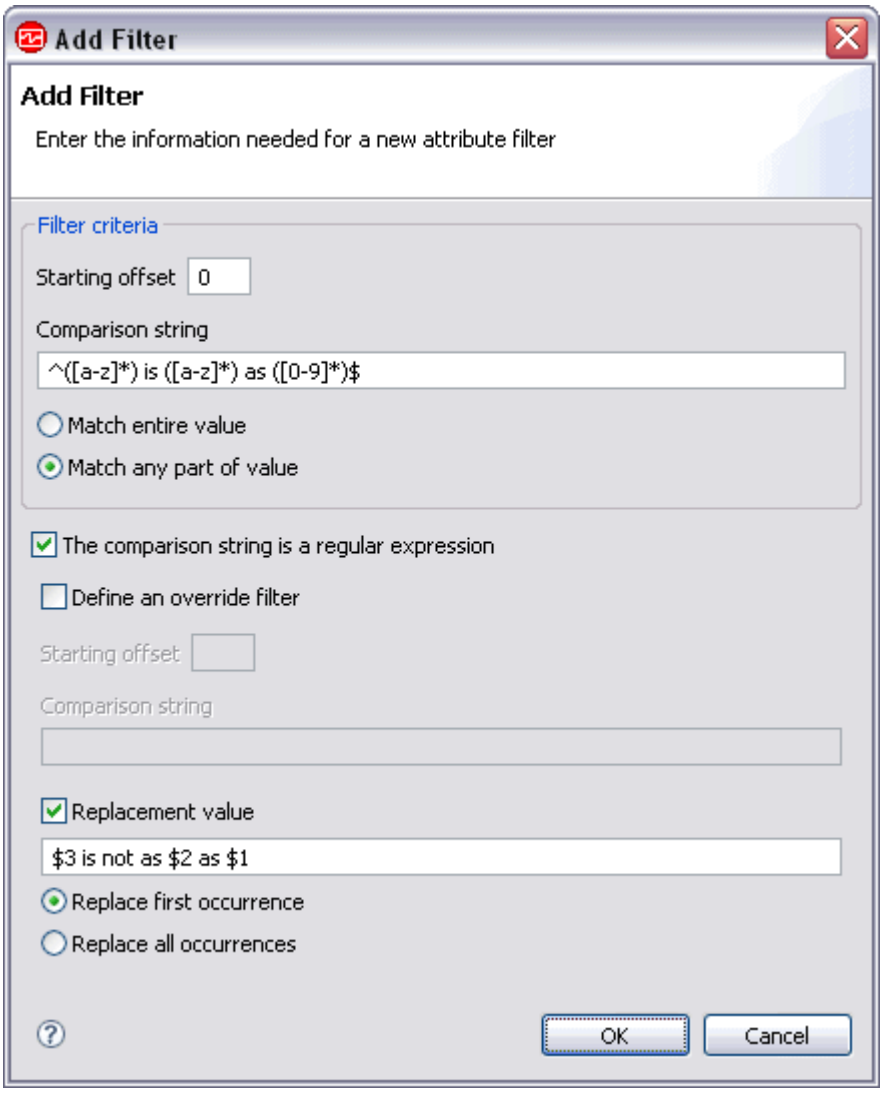


*Figure 8. **Add Filter** example 1*

If the attribute string is `abc is easy as 123`, then the replaced string that is displayed in the Tivoli Enterprise Portal or IBM Cloud Application Performance Management console as `123 is not as easy as abc`.

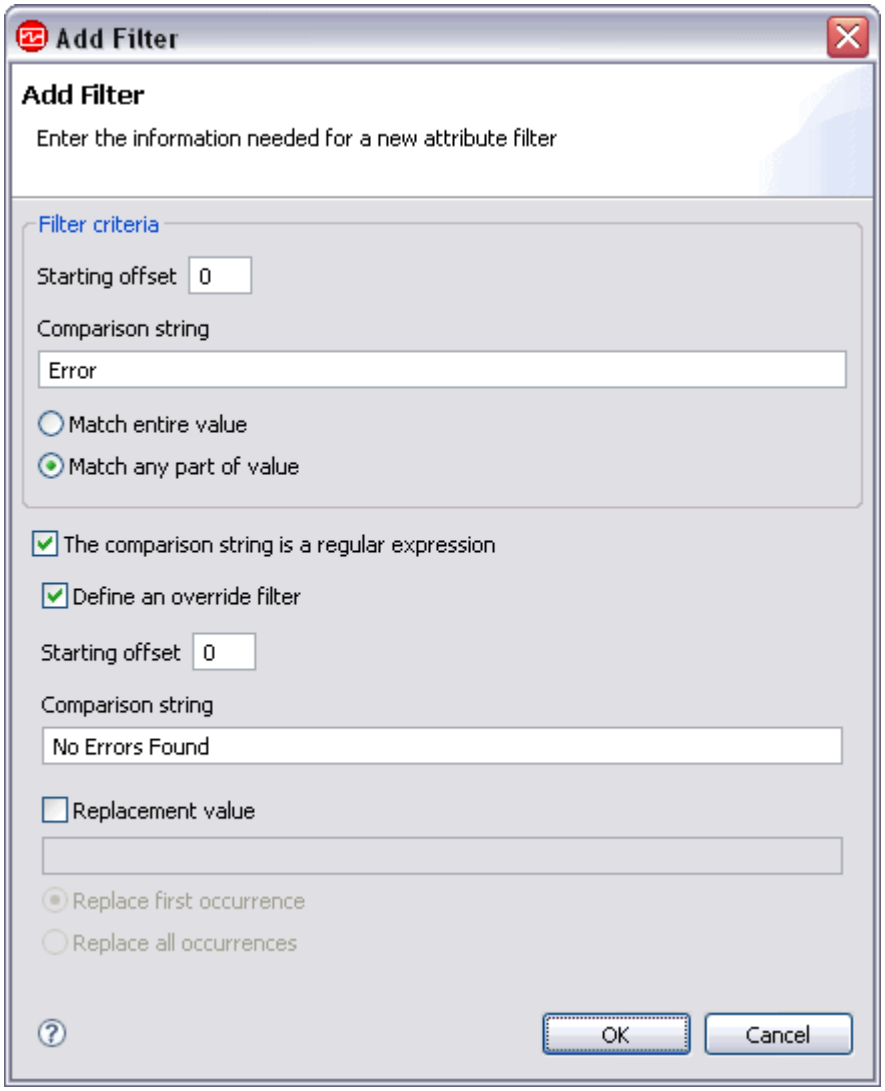


*Figure 9. **Add Filter** example 2*

If the attribute string is `Unrecoverable Error reading from disk`, and the filter is **Inclusive**, then the attribute is displayed in the Tivoli Enterprise Portal or IBM Cloud Application Performance Management console. If the attribute string is `No Errors Found during weekly backup` and the filter is **Inclusive**, then the attribute is not displayed.

d) In the **Field Identification** section of the **Advanced Log File Attribute Information** page, specify how to override the attribute group field delimiters for this one attribute only. Click one of the attribute filters, and complete the required fields for the option:

- **Number of characters**: Enter the limit for the number of characters.
- **Tab separator** specifies the use of tab separators.
- **Separator Text**: Enter the separator text that you want to use.
- **Begin and End Text** Enter both **Begin** text and **End** text.

e) In the **Summary** section of the **Advanced Log File Attribute Information** page, click the **Include attribute in summary attribute group** check box to add the attribute to the summary attribute group.

This attribute group is produced when a user turns on log attribute summarization.

f) Click **OK**.

10. If you used the test function in step (“7” on page 108), the **Select key attributes** page is displayed. On the **Select key attributes** page, select key attributes or indicate that this data source produces only one data row.

    For more information, see (“Selecting key attributes” on page 15).
11. Do one of the following steps:
    - If you are using the New Agent wizard, click **Next**.
    - Click **Finish** to save the data source and open the Agent Editor.

    Note: When a log file attribute group is added to an agent with the default minimum Tivoli Monitoring version (6.2.1) or later, a `Log File Status` attribute group is included. For more information about the `Log File Status` attribute group, see (“Log File Status attribute group” on page 322).

Log file parsing and separators

You can change the default separator that is used to separate one or more attributes in a log file record.

When you create a log file attribute group, a separator is by default assigned. The default separator is a tab. The separator is used by the agent to parse and delimit the data for each attribute in the data row. You can change the default attribute separator to be:

- A fixed number of characters
- A space
- A different character or characters
- A specific beginning and end text
- An XML element.

You change the default separator that is used for all attributes in the group in the following ways:

1. When you are creating the attribute group, on the **Log File Information** page.
2. After you create the attribute group, by opening the **Agent Editor** > **Data Sources** tab, selecting the attribute group and choosing a separator in the **Field Identification** area.

You can also optionally assign specific separators to one or more individual attributes. You can assign specific separators for individual attributes to use:

- A fixed number of characters.
- A tab separator
- A space separator
- A different character or characters
- A specific beginning and end text.

You change the separator that is used for individual attributes in the following ways:

1. By selecting **Advanced** on the **Attribute Information** page when you are creating an attribute.
2. By opening the **Agent Editor** > **Data Sources** tab, selecting the attribute and selecting **Advanced** on the **Log File Attribute Information** tab.

Example 1 - Simple log file output

Some log file records have clear and regular separators, for example:

```
one,two,three
```

Here the `","` character is a clear and regular separator between the three pieces of data on the row. In this case, select **Separator Text** and specify `","` as the default separator for the attribute group. There is no need to change or define other separators.

Defining this separator for a log file that contains the data row that is shown earlier in this example is shown in the following output:

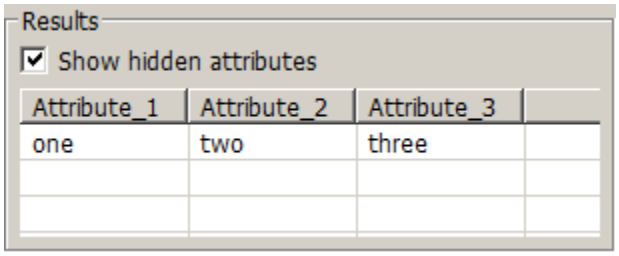


Figure 10. Example attribute value output when Agent parses a simple log file data row.

Example 2 - Complex log file output

Some log files can contain data rows that have irregular or changing separators, for example:

```
one,two,three,[four]12:42,five
```

In this example an assignment of separators to attribute definitions that you can use is:

1. In the previous example you set the default separator to `","`. This separator is used for all attributes unless you over-ride it with a specific separator. In this example the default separator of `","` is correct to use again for the first three attributes in the row.
2. For the fourth attribute, assume the string between the`"["` and `"]"` is a value that you want to extract. In this case when you define the fourth attribute, you assign a separator type **Begin and End Text** with begin and end text values of `"["` and `"]"`.
3. For the fifth attribute, assume that you want to extract the values between the `"]"` and `":"` characters. In this case when you define the fifth attribute, you assign separator type **Separator Text** set to `":"`.
4. For the sixth attribute, your default attribute group separator `","` is fine again.
5. For the seventh attribute, you do not need to specify a separator as it is the last attribute.

Defining these separators on a log file that contains the data row that is shown earlier in this example is shown in the following output:

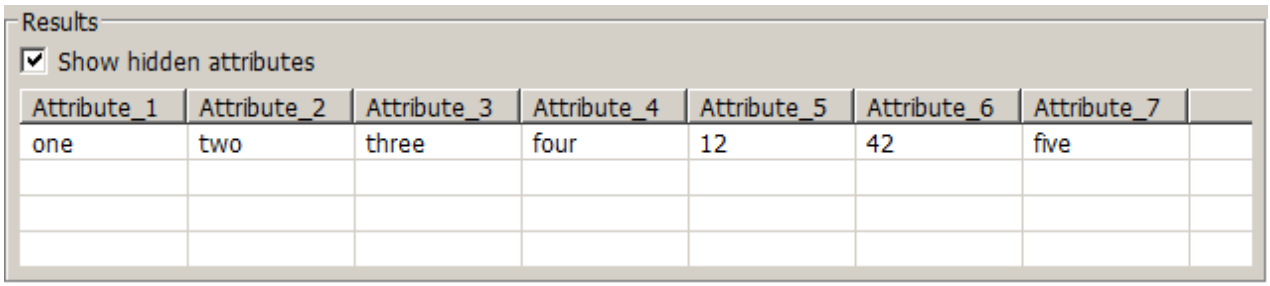


Figure 11. Example attribute value output when Agent parses a complex log file data row.

The procedure to define the attribute separators is described under step “5” on page 105 of “Monitoring a log file” on page 104.

Testing log file attribute groups

You can use Agent Builder to test the log file data set (attribute group) that you created. If no attributes are defined for the group, the testing process defines them automatically.

Before you begin

If any attributes are already defined for this data set and you want to define attribtes automatically during testing, use the agent editor to remove all the existing attributes from the data set. For instructions, see “Removing attributes” on page 39.

Procedure

1. You can start the Testing procedure in the following ways:
   - During agent creation click **Test Log File Settings** on the **Log File Information** page.
   - After agent creation, select an attribute group on the Agent Editor **Data Source Definition** page and click **Test Log File Settings**. For more information about the Agent Editor, see Chapter 4, "Using the Agent Editor to modify the agent," on page 17.

   After you click **Test Log File Settings** in one of the previous two steps, the **Parse Log** window opens.
2. Select the source of the log data for testing:
   - **Use attribute group settings**: use the file name and location specified in the data source. By default, the data source processes only the information that is added to the log file after the testing process is started. You can use this option if the log file is being updated in real time.
   - **Specify a sample file**: provide a sample log file. With this setting, the testing procedure parses the entire contents of the log file. With this option, you can test the data source and create the attributes for it immediately, based on an existing sample. Specify the path and name of the file in the **Log file name** field or use the **Browse** button to select the file.
3. Optional: Before you start testing, you can set environment variables and configuration properties.

   For more information, see ("Attribute group testing" on page 229).
4. Click **Start Agent**.

   A window opens indicating that the Agent is starting. When the agent starts, it monitors the configured log file for new records
5. To test your agent's data collection, generate new records in the monitored log file.

   When new records are added to the log file, the agent parses them according to its configuration and updates the corresponding attribute values in its cache.
6. To simulate a request from Tivoli Enterprise Portal or SOAP for agent data, click **Collect Data**.

   The **Parse Log** window collects and shows any new attribute values in the agent's cache since it was last started. An example data collection is shown in Figure 12 on page 114

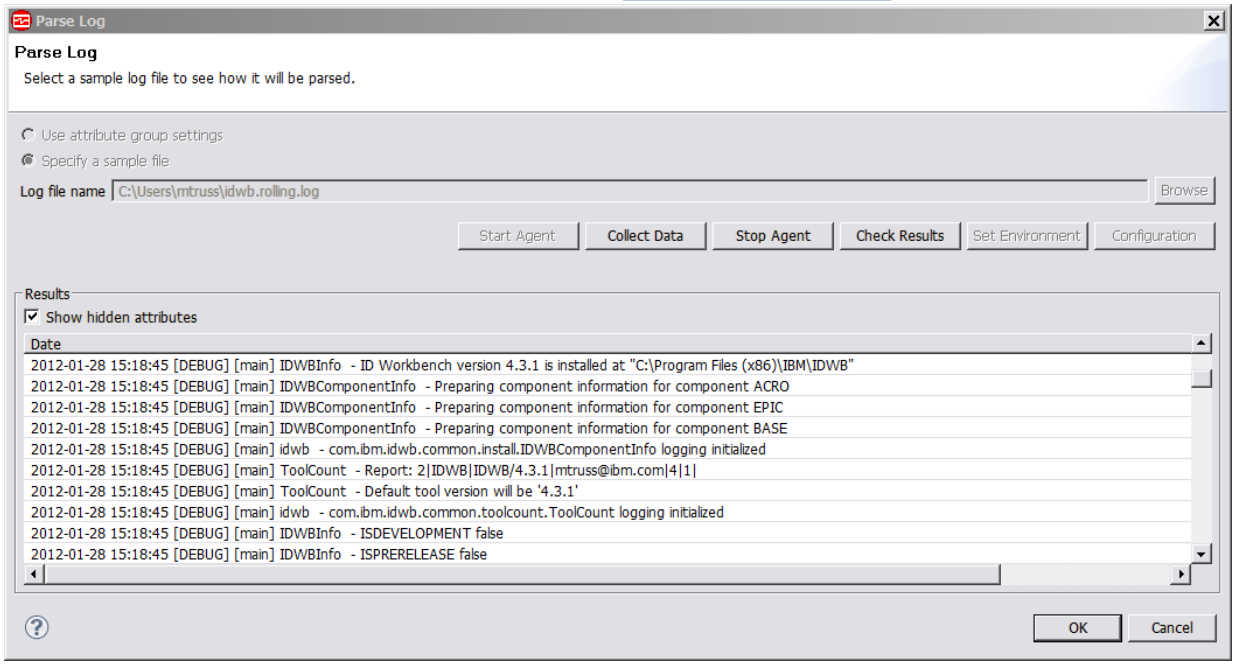


Figure 12. Parse Log window that shows parsed log file attribute values

7. Optional: Click **Check Results** if the returned data is not as you expected.

   The **Data Collection Status** window opens and shows you more information about the data. The data that is collected and shown by the Data collection Status window is described in "Performance Object Status node" on page 280

8. The agent can be stopped by clicking **Stop Agent**.
9. Click **OK** or **Cancel** to exit the **Parse Log** window. Clicking **OK** saves any changes that you made.

Related concepts

"Testing your agent in Agent Builder" on page 229
After you use Agent Builder to create an agent, you can test the agent in Agent Builder.

Monitoring an AIX Binary Log

You can define a data source to monitor AIX binary error logs through the `errpt` command. You can also configure it to filter and summarize the data. The resulting events are placed in a data set.

About this task

Log Monitoring supports the monitoring of AIX binary error logs through the `errpt` command. The `errpt` command generates an error report from entries in an error log. It includes flags for selecting errors that match specific criteria. This support for the monitoring of AIX binary error logs through the `errpt` command is modeled on the support for the same function in the Tivoli Monitoring UNIX Logs Agent (product code `kul` or `ul`).

When you supply the Agent Builder with an **`errpt`** command string, it processes the events that result from running this command. Agent Builder enforces the same constraints on this command that the Monitoring Agent for UNIX Logs does. In particular, you must use the **`-c`** (concurrent mode) option so that the command runs continuously, and you cannot use the **`-t`** option or the following options that result in detailed output: **`-a`**, **`-A`**, or **`-g`**.

An Agent Builder agent that monitors the AIX **`errpt`** command automatically includes the same information as a Monitoring Agent for UNIX Logs does. For more information about the attribute groups for AIX binary error logs, see "AIX Binary Log attribute group" on page 293.

Procedure

1. On the **Agent Initial Data Source** page or the **Data Source Location** page, click **Logged data** in the **Monitoring Data Categories** area.
2. In the **Data Sources** area, click **AIX Binary Log**.
3. Click **Next**.
4. On the **Binary Log Information** page, enter an `errpt` command.

   The default value is:

   ```
   errpt -c -smmddhhmmyy
   ```

   The agent searches for the 'mmddhhmmyy' string and replaces it with the actual date and time on startup. Only the first occurrence of the string is replaced.

   You can supply your own `errpt` command but Agent Builder enforces the same constraints on this command that the Monitoring Agent for UNIX Logs does. In particular, you must use the **`-c`** (concurrent mode) option so that the command runs continuously, and you cannot use the **`-t`** option or the following options that result in detailed output: **`-a`**, **`-A`**, or **`-g`**.
5. (Optional) Click **Advanced** to select filtering and summarization options for events. For more information, see "Controlling duplicate events" on page 259.
6. Do one of the following steps:
   - If you are using the **Agent** wizard, click **Next**.
   - Click **Finish** to save the data source and open the Agent Editor.

Related reference

"AIX Binary Log attribute group" on page 293

The AIX Binary Log attribute group displays events from the AIX Binary Log as selected by the provided `errpt` command string.

Monitoring a Windows Event Log

You can define a data source to collect data from a Windows event log. You can configure it to filter the data. The resulting events are placed in the Event Log data set.

About this task

You can collect data from the Windows event log by using the type, source, or ID of events. You use these parameters to filter the log events that the Windows system gathered. The agent compares each new event in the monitored event log against the specified filter. If the event matches one of the event types, event sources, and event IDs specified in the filter, it passes.

For example, if the Event log filter is for the Application log, specify **Error** as the event type. This choice matches all events that are logged to the Application log with an event type value of `error`. If you add the **Diskeeper** and **Symantec AntiVirus** event sources, the agent matches all error events from either of these sources. You can add specific event IDs to refine the filter further. No direct association exists between the event type, event source, and event ID. If one of the values for each matches an event, the event matches.

By default, only events that are generated after the agent starts are processed. However, you can enable the agent when it restarts to process log events that are generated while the agent is shut down. For more information about enabling the agent to process events generated while the agent is shut down, see step "6" on page 116.

Procedure

1. On the **Agent Initial Data Source** page or the **Data Source Location** page, click **Logged Data** in the **Monitoring Data Categories** area.
2. In the **Data Sources** area, click **Windows Event Log**.
3. Click **Next**.
4. On the **Windows Event Log** page, select the name from one of the logs in the **Windows Event Log name** list, or type a name for the event log.

   The list is constructed from the set of logs on the current system, for example:

   ```
   Application
   Security
   System
   ```
5. In the **Windows Event Log** page, specify whether you want to filter the results by using one or more of the following mechanisms:

   - "Filtering by event type" on page 117
   - "Filtering by event source" on page 118
   - "Filtering by event identifier" on page 118

   Note: You must select at least one of these filter criteria.
6. To process log events that are generated while the agent is shut down, on an agent restart, click **Offline Event Settings** on the **Windows Event Log** page.

   The **Windows Event Log Bookmark Settings** window opens.
7. Select one of the following bookmarking options:

   Note: These options apply to all Windows event logs being monitored.

   - **Do not collect any offline events**: Events that are generated while the agent is shut down are not processed. This option is the default option.

- **Collect all offline objects**: All events that are generated while the agent is shut down are processed.
- **Specify custom collection settings**: You can enter a value to throttle the processing of old events that are based on a time value, or number of events, or both. By using this option, you ensure that the monitoring environment is not overloaded with events when the agent starts.

  For example, if 100 is entered in **The maximum number of events to collect** field and 30 is entered in the **Restrict collection based on a time interval (in seconds)** field. The number of events that are processed is either the last 100 events that are generated before the agent starts, or any event that is generated within 30 seconds of agent start. Which result depends on the variable that is matched first.

  When you enter a value for the maximum number of events to collect, the *CDP_DP_EVENT_LOG_MAX_ BACKLOG_EVENTS* environment variable is added. When you enter a value to restrict collection that is based on a time interval, the *CDP_DP_EVENT_LOG_MAX_BACKLOG_TIME* environment variable is added. When either or both of these variables are added, the *eventlogname_productcode_instancename_subnodename*`.rst` file is created containing the last event record that is processed for the event log. This file is in the *%CANDLE_HOME%*`\tmaitm6\logs` directory and is used when the agent is restarted to process old events that are generated while the agent was shut down.

8. If you want to set global options for the data source, click **Global Options** on the **Windows Event Log** page

   The **Global Windows Data Source Options** window opens.
9. Select the **Include remote Windows configuration properties** check box if you want to include this option, and click **OK**.

   For information about Windows remote connection configuration for Windows data sources, see "Configuring a Windows remote connection" on page 207.
10. After you specify the filter and click **OK**, on the **Windows Event Log** page, do one of the following steps:
    - If you are using the **Agent** wizard, click **Next**.
    - Click **Finish** to save the data source and open the Agent Editor. The name of the new Windows Event Log is shown on the **Agent Editor Data Source Definition** page.

What to do next

For information about Windows remote connection configuration for Windows Event Log data sources, see "Configuring a Windows remote connection" on page 207.

Filtering by event type

Filter Windows Event Log results by event type

Procedure

1. In the **Windows Event Log** page, select **Filter by event type**.
2. Select one or more of the following Event types:
   - **Information**
   - **Warning**
   - **Error**
   - **Success Audit**
   - **Failure Audit**
3. Click **Finish** to complete.

Filtering by event source

Filter Windows Event Log results by event source

Procedure

1. Select **Filter by event source** and click **Add** in the **Event sources** area of the **Windows Event Log** page.

   The **Event Source** window opens.
2. Make one of the following choices.
   - Type the event source name and click **OK**.
   - Click Browse **Browse** to find and select an event source from a list and click **OK**.

     The name that you selected is shown in the **Event Source** window.

     Note:

     a. To sort the list of event sources, click the column heading.

     b. To refresh the information in the window, click the **Refresh** icon.

     c. To search for specific event sources, click the **Search** (binoculars) icon.
3. Click **OK** to see the new event source filter in the Event sources list in the **Windows Event Log** window.

Filtering by event identifier

For the Windows Event Log data source, you can filter events by event identifier.

About this task

To filter by event identifier, use the following procedure:

Procedure

1. Select **Filter by event identifier** and click **Add** in the **Event identifiers** area of the **Windows Event Log** window.

   The **Event Identifier** window is displayed.
2. If you know that you want to monitor specific events from an application, specify the numbers of the event as the application defines it. Type an integer as the event identifier and click **OK**.

   The new numeric event identifier filter is displayed in the Event identifiers list in the **Windows Event Log**.

   Note: Each event identifier must be defined individually.
3. If you want to modify a Windows event log, select it and click **Edit**.
4. If you want to delete a Windows event log, select it and click **Remove**.
5. You can add more event logs to the list, or click **Finish**.

Monitoring a command return code

You can define a data source to monitor an application or system by using a *command return code*. The agent runs the command, collects the return code, and adds the result to the Availability data set.

About this task

A user-created script, executable file, query, or system command can return a code. A command return code is an application-specific mechanism for determining whether the application or monitored system is available. The agent runs the specified command and determines the state of the application or monitored system by examining the return code.

The command must present a unique return code for each descriptive state. The command must also define a message to be used by the agent for each of these return codes. The command can use environment and configuration variables within the user created script, executable file, query, or system command. The command must not use environment or configuration variables on the command-line invocation of the command, with only the following exceptions available: *AGENT_BIN_DIR*, *AGENT_ETC_DIR*, *AGENT_LIB_DIR*, *CANDLE_HOME*, and *CANDLEHOME*.

Procedure

1. On the **Agent Initial Data Source** page or the **Data Source Location** page, select **Command or script** in the **Monitoring Data Categories** area.
2. In the **Data Sources** area, click **A command return code**.
3. Click **Next**.
4. On the **Command Return Code** page, **Command return code information** area, type the display name.
5. Use the following substeps to define and describe command lines that you want your command return code to use.

   Note: Define a command for each operating system that is supported by the agent. Commands can be shared, but the total set of operating systems for all of the commands must equal the set of agent supported operating systems.

   a) Click **Add** in the **Commands** area of the **Command Return Code** window to open the **Command Information** window.

   b) Type a command line and select an operating system from the list in the **Operating Systems** area of the **Command Information** window.

   Note:

   i) For a Windows command, you must type the full name of the command. For example, `command_to_run.bat` and not just `command_to_run`.

   ii) Place quotation marks around the name so that it is not parsed by the command interpreter. For example, type `"this is a test.bat"argument` and not `this is a test.bat argument`.

   iii) You can click a command and click **Edit** to modify it, or click **Remove** to delete it.

   c) Click **Add** in the **Return Codes** area of the **Command Information** window.

   d) Select a return code type from the list that is shown in the **Return Code Definition** window

   You can assign the following states to the test return codes:

   - `ALREADY_RUNNING`
   - `DEPENDENT_NOT_RUNNING`
   - `GENERAL_ERROR`
   - `NOT_RUNNING`
   - `OK`
   - `PREREQ_NOT_RUNNING`
   - `WARNING`

   e) Type a numeric value for the return code type that you selected.

   The return code value is an integer that specifies a defined return code for the command return code. For portability between operating systems, use a return code value of `0 - 255`. For a command that runs only on Windows, the return code value can be `-2147483648 - 2147483647`.

   f) Define a message for each return code so that the message and code can be shown together. Click **Browse** to set up the message text.

The message window lists messages that are defined in the agent. The **Messages** (list) window opens.

Note:

i) You can select text that was entered previously by selecting it in the list of message texts instead of clicking **Browse**. Then, continue to Step 5k.

ii) Until you define messages, the list remains blank. You can use **Edit** to alter a defined message and **Remove** to delete one or more messages that you defined.

g) In the **Messages** (list) window, click **Add**

The **Message Definition** window opens.

Note: The message identifier is automatically generated for you.

h) Enter some text that describes the meaning of the new message in the **Message text** field.

i) Click **OK**.

The **Messages** (list) window opens showing the new message .

j) To verify the message and make it permanent, select it in the list and click **OK**.

The new return code type, value, and text are shown in the **Return Code Definition** window.

k) If you want this return code to be available to other commands on other operating systems for this command return code, select **Global return code applies to all commands**. If you want this return code to be available only to this command, leave **Local return code applies only to this command** selected.

l) Click **OK** in the **Return Code Definition** window.

m) Define at least two return codes before you leave the **Command Information** window. One return code to indicate no problems with the availability, another to indicate whether a problem occurred. If you want to add another return code, return to step c.

n) Optional: In the **Command Information** window, **Command files** area, click **Add** if you want to select one or more scripts or executable files for the agent to run.

The file or files are copied into the project folder of the agent under `scripts/`*`operating system`*, where *operating system* is a variable that depends on what you selected in the **Operating Systems** area of the **Command Information** window. These files are also packaged and distributed with the agent. To edit the definition of an existing command file, or the original command file since copied into the project, select the file and click **Edit**. See ("Editing a command file definition" on page 122).

o) Click **OK** in the **Command Information** window.

Note: The command files table is where you define any external files that you want to include in the agent package. These files are copied into the project directory and packaged with the agent for distribution.

6. If you have other return codes that are not already defined, define and describe global return codes that your command return code can use.

a) Click **Add** in the **Global return codes** area of the **Command Return Code** page.

Note: The return codes that are defined here are global. This means that the return codes are appropriate for all of the commands that are defined for the command return code. (They are not shared between command return codes). In addition, you can define return codes when you enter the command information. The return codes that are defined here can be global or local. Local return codes are only appropriate for this specific command. This hierarchy is useful if you have a return code that is the same across all operating systems. (For instance, a return code of 0 means that everything is functioning correctly. You can define it at the global level, and then all defined commands interpret 0 in this way.) If none of the other operating systems return a 5, you can define the return code of 5 only for the Windows command. If you define a return code at the local command level that is already defined at the global level, the command level is used. You can use this method to override return codes on specific operating systems. For instance, if on all UNIX operating systems, a return code of 2 means one thing, but, on Windows, it means something

different. You can define a return code of 2 at the global level as expected by the UNIX operating systems. Then, in the command for Windows, you can redefine return code 2 for the meaning on Windows.

b) Select a return code type from the list that is shown in the **Return Code Definition** window.

You can assign the following states to the test return codes:

- `ALREADY_RUNNING`
- `DEPENDENT_NOT_RUNNING`
- `GENERAL_ERROR`
- `NOT_RUNNING`
- `OK`
- `PREREQ_NOT_RUNNING`
- `WARNING`

c) Type a numeric value for the return code type that you selected. The return code value is an integer that specifies a defined return code for the command return code.

d) Click **Browse** to set up the message text and its associated meaning. You must define a message for each return code so that the message and code are shown together.

The **Messages** window lists messages that are defined in the agent.

Note:

i) Until you define messages, the list remains blank. You can use **Edit** to alter a defined message and **Remove** to delete one or more messages you defined.

ii) You can select text that was entered previously by selecting it in the **Message text** list instead of clicking **Browse**. Then, continue to Step 6h.

e) In the **Messages** (list) window, click **Add** to see a **Message Definition** window, where you can type text that describes the meaning of the new message.

f) Click **OK**.

g) The **Messages** (list) window opens with the new message. To verify the message and make it permanent, select it in the list and click **OK**.

h) When the new text, type, and value are shown in the **Return Code Definition** window, click **OK**.

i) On the Command Return code page, when you finish defining the return codes and commands for all supported operating systems, do one of the following steps:

- If you are using the New Agent wizard, click **Next** or click **Finish** to save the data source and open the Agent Editor.
- If you are using the New Agent Component wizard, click **Finish** to return to the Agent Editor.

What to do next

If you want to use the data from this data source in the summary dashboard for IBM Cloud Application Performance Management, you must create a filtered data set (attribute group) based on the Availability data set and configure it as providing a single row. Use the NAME field to select the row for your process.

In the new filtered attribute group, select the Status field and specify the severity values for it.

For instructions, see:

- “Creating a filtered attribute group” on page 182
- “Specifying severity for an attribute used as a status indicator” on page 44
- Chapter 12, “Preparing the agent for Cloud APM,” on page 219

Editing a command file definition

You can change the command file that is imported into the project, or import changes to the existing command file into the project.

Procedure

1. Select the file in the **Command files** area of the **Command Information window**.
2. Click **Edit** to open the **Import Command File** window.

   From the **Import Command File** window, you can get the status of the command file. You can also change the location of the original source file, and recopy the source file into the agent.
3. Choose one of the following steps:

   - Click **OK** to schedule a copy of the file to occur the next time that the agent is saved.
   - Click **Copy Immediately** to copy the file without first saving the agent.

   Note: The **Copy Immediately** option is not available when you access the **Import Command File** window from the New Agent wizard.

File Separation & Consolidation

You can use the Separate and Consolidate functions to move files in and out of operating system-specific folders in the agent.

When a file is first added to the agent, a single copy is added in the `scripts/all_windows` folder, the `scripts/all_unix` folder, or the `scripts/common` folder. The `scripts/common` folder is used if the file is used on both Windows and UNIX.

To place different copies of the file on different operating systems (for example, a binary executable file), click **Edit** and click **Separate**. The file is removed from the common folder and copied into operating system-specific folders. Then, you can replace individual copies of the file with ones appropriate for specific operating systems.

Note: Java resource files must remain in the `scripts/common` folder. You cannot click **Separate** to make separate copies of Java resource files for individual operating systems.

If you separated the files into operating-system-folders, you can use **Consolidate** to move them back into a common folder. If you created the agent in an Agent Builder version that did not support common folders, use **Consolidate** to move them back into a common folder. If any of the copies of the file differ from one another, you are prompted to select the file to use as the common file. All other copies are discarded.

Monitor output from a script

You can define a data source to collect data from a script or external program. Use it when application data is not available through a standard management interface or when you need to provide a summary of multi-row data in a single row. The agent runs the script and collects its output. Each line in the script output is parsed into a row of the resulting data set.

Data can be collected from either a local or remote system. The output of the script or program must contain only values for each attribute within the attribute group. To return multiple rows of data, the data for each row must be separated by a line break. The attributes in each row of data are separated by the separators you define. For more information about separators, see “Script parsing and separators” on page 123

The command can use environment and configuration variables within the user-created script, executable file, query, or system command. The command cannot use environment or configuration variables on the command-line invocation of the command, with only the following exceptions available: `AGENT_BIN_DIR`, `AGENT_ETC_DIR`, `AGENT_LIB_DIR`, `CANDLE_HOME`, and `CANDLEHOME`.

The agent monitors script output that is written by using the same locale and code page that the agent runs in.

Collecting script data from a remote system

To collect script or program data from a remote system, Agent Builder uses a Secure Shell (SSH)

To collect data from a remote system, Agent Builder creates a Secure Shell (SSH) session and starts the script or external program on the remote system. The agent establishes and logs on to an SSH session. The agent then uploads the scripts to the remote system, starts the script or external program, and retrieves the output. The agent can be configured to keep the session open or reestablish the session for each invocation. If the session is kept open, the script can be reused or uploaded for each invocation. By default, a single SSH session is used and the scripts are reused for each invocation.

Agent Builder supports use of only SSH Protocol Version 2 with Rivest, Shamir, and Adleman (RSA) or Digital Signature Algorithm (DSA) keys. The agent is either authenticated by user name and password, or by public key authentication. The generation and distribution of the public keys is an administrative task that must be done outside of the agent and Agent Builder.

To run a Take Action command that is written against a Secure Shell (SSH) enabled script data provider on the remote system, see “SSHEXEC action” on page 369.

Restriction: If your agent was built with an Agent Builder version before 6.3 and it has a script data provider that uses SSH, the provider fails when run with IBM Tivoli Monitoring version 6.3 or later. To resolve this issue, rebuild the agent with the current version of Agent Builder.

The restriction is because IBM Tivoli Monitoring version 6.3 uses a newer version of the Global Secure ToolKit (GSKit) API. You must rebuild the agent with Agent Builder 6.3 or later to run it with IBM Tivoli Monitoring version 6.3 or later. If you build the agent with Agent Builder 6.3, it can also run with earlier versions of IBM Tivoli Monitoring.

Script parsing and separators

You can change and assign specific script separators to one or more attributes.

When you create a script attribute group, a single character text separator is by default assigned. The default separator is " ; ". The separator is used by the agent to parse and delimit the data for each attribute in the data row. You can change the default separator to use a different character. You can also assign specific separators to one or more individual attributes.

You can assign specific separators for individual attributes that:

- Take a fixed number of bytes from the output.
- Separate one attribute from the next with a custom separator, which can be more than one character.
- Delimit an attribute value with a string at the beginning and end of the value.
- Return the rest of the text as the attribute value (whether it contains embedded separators or not).

You can use one or more of these separators to extract attribute values from the data rows.

Example 1 - Simple script output

Some scripts can output data rows with clear and regular separators, for example:

```
Row One;1;2
Row Two;3;4
Row Three;5;6
```

Here the " ; " character is a clear and regular separator between the three pieces of data on each row. In this case, the default separator is fine, so there is no need to change or define other separators. It is not

difficult to imagine a similar script output where the separator is a different character, as in the following example.

```
Row One-1-2
Row Two-3-4
Row Three-5-6
```

In this example the separator is changed from a ";" character to a "-" character. In this case when you define the attributes, change the default separator to use the "-" character.

Example 2 - Complex script output

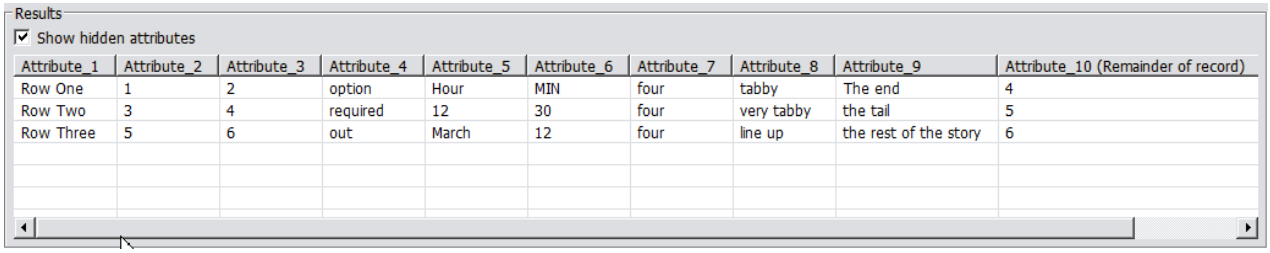
Some scripts can output data rows that have irregular or changing separators, for example:

```
Row One;1;2;[option]Hour:MIN;fourtabby     The end;4
Row Two;3;4;[required]12:30;fourvery tabby the tail;5
Row Three;5;6;[out]March:12;fourline up    the rest of the story;6
```

In this example an assignment of separators to attribute definitions that you can use is:

1. Initially the default separator ";" is fine for the first three attributes in each data row. In this case, you assign the separator type **Separator Text** set to ";" when you define each attribute, this setting is the default one.
2. For the fourth attribute, assume the string between the"[" and "]" is a value that you want to extract. In this case when you define the fourth attribute, you assign a separator type **Begin and End Text** with begin and end text values of "[" and "]".
3. For the fifth attribute, assume that you want to extract the values between the "]" and ":" characters. In this case when you define the fifth attribute, you assign separator type **Separator Text** set to ":".
4. For the sixth attribute, the default separator ";" is fine again, accept the default.
5. For the seventh attribute, you would like to extract the string in the next four characters "four". There is not a clear separator at the end of this string. You can assign a number of characters to define the separation from the next attribute. You assign a separator type **Number of characters**, and specify four characters as the length.
6. For the eighth attribute you would like to extract the strings tabby, very tabby and line up. In this case, you can assume that all of these strings are followed by a tab character. In this case, you assign a separator of type **Tab separator**.
7. For the ninth attribute, you revert again to the default separator type to extract the remaining text to this attribute.
8. For the 10th attribute, you specify **Remainder of record** to assign the remainder of the data row to this attribute

Defining these separators on a script that outputs the data rows that are shown earlier in this example is shown in the following output:


Results

Show hidden attributes

Attribute_1	Attribute_2	Attribute_3	Attribute_4	Attribute_5	Attribute_6	Attribute_7	Attribute_8	Attribute_9	Attribute_10 (Remainder of record)
Row One	1	2	option	Hour	MIN	four	tabby	The end	4
Row Two	3	4	required	12	30	four	very tabby	the tail	5
Row Three	5	6	out	March	12	four	line up	the rest of the story	6

Figure 13. Example attribute value output when Agent parses complex script output.

The procedure to define the attribute separators is described under step “10” on page 127 of “Steps for monitoring output from a script” on page 125.

Steps for monitoring output from a script

Configure your agent to receive data from a script data source.

Before you begin

See "Monitor output from a script" on page 122

About this task

Use the following procedure to monitor output from a script:

Procedure

1. On the **Agent Initial Data Source** page or the **Data Source Location** page, select the option **Command or script** in the **Monitoring Data Categories** area.
2. In the **Data Sources** area, click **Output from a script**.
3. Click **Next**.
4. On the **Command List** page , click **Add** to display a **Command Information** window.

   Note: Selecting the **Enable data collection using SSH** check box enables SSH for this attribute group. If this check box is not selected, the attribute group runs locally.

   Note: If a command exists that can be run on the operating system on which the Agent Builder is running, the **Test** option is enabled. You can use **Test** to test a command that you defined.
5. In the **Command Information** area in the **Command Information** window, type a command name with the necessary arguments in the **Command** field, and a separator in the **Separator** field.

   Note:

   a. Scripts in Windows are frequently started without specifying the `.bat` or `.cmd` extension on the command line. For remote execution, a shell environment must be installed and you must specify the `.bat` or `.cmd` in the script data source command for the script to run. `Cygwin` is an example of a shell environment that is available for Windows. Linux, Red Hat, and AIX. To verify that a shell environment exists, SSH or log on to the remote host and enter the command:

   ```
   PATH=$PATH:. <command>
   ```

   If the command runs, then a shell environment exists.

   b. Use quotation marks around the name so that it is not parsed by the command interpreter. For example, `this is a test.bat argument` becomes:

   ```
   "this is a test.bat" argument
   ```

   c. Environment variables and configuration variables can be used in the user-provided script, but cannot be part of the command line that starts the script. The following variables are exceptions to this rule:

   AGENT_BIN_DIR
   The directory where the agent places binary files or scripts

   AGENT_ETC_DIR
   The directory where the agent places configuration files

   AGENT_LIB_DIR
   The directory where the agent places shared libraries or dynamic-link libraries

   CANDLEHOME
   The Linux or UNIX Tivoli Monitoring installation directory

   CANDLE_HOME
   The Windows Tivoli Monitoring installation directory

d. If the SSH data collection option is being used, the command line is run relative to the user's home directory on the remote system. If you are uploading scripts or executables to the remote system, they are copied to the location specified in the agent's environment variable *CDP_SSH_TEMP_DIRECTORY*. The location defaults to the user's home directory on the remote system. On some systems, you might need to define the command line with a relative path, such as `./Script.sh`.

6. In the **Operating Systems** area, select one or more operating systems. When you collect data from a remote system by using SSH, Operating Systems is a property of the system on which the agent is installed. It is not the Operating System of the remote system. It is advised that you select the **All operating systems** check box when you use the SSH data collection features.
7. Optional: If one or more user-defined files are necessary to run the command, click **Add** in the Command files area to specify the files from your system.

   The files are copied into the project folder of the agent under `scripts/`*`operating system`*, where *operating system* is a variable that depends on what you selected in the **Command Information** window. These files are also packaged and distributed with the agent. If you want to edit the definition of a command file you already added, or changed the contents of, select the file and click **Edit**. See "Editing a command file definition" on page 122.
8. Click **OK**. The **Command List** page is displayed.
9. To test the command, use the following steps:
   a) Click **Test** to open the command information and display the **Test Command** window. To test the script on a remote system, select a system from the **Connection name** list or click **Add** to add the host name of a system.

   b) Use the **Test Command** window to change the command, default separator, and attribute separators, and to view how these changes affect the data that is returned.

      i) Type the command and separator in the fields if they are not already entered.

      Note: You can specify other separators by using the **Attribute Information** window at attribute creation time or by using the Agent Editor to modify an existing attribute. For more information about the Agent Editor, see Chapter 4, "Using the Agent Editor to modify the agent," on page 17 and for more information about manipulating data source and attributes, see Chapter 5, "Editing data source and attribute properties," on page 35

      ii) Before you start testing, you can set environment variables and configuration properties. For more information, see ("Attribute group testing" on page 229).

      iii) Click **OK** to return to the **Test Settings** window.

      iv) Click **Start Agent**. A window indicates that the Agent is starting.

      v) To simulate a request from Tivoli Enterprise Portal or SOAP for agent data, click **Collect Data**. The Agent Builder runs your command. If you specified a remote system, provide a user ID and password. Even if the return code is not 0, the Agent Builder parses the results of the command in the same way the agent does.

      vi) The **Test Settings** window collects and displays any data in the agent's cache since it was last started. The initial names of the attributes are **Attribute_1, Attribute_2**, and so on; however, you can modify the properties of the attributes by clicking the appropriate column heading.

      vii) Click **Check Results** to view the return code from the command, the unparsed data, and any error messages that were returned.

      viii) The agent can be stopped by clicking **Stop Agent**.

      ix) Click **OK** to return to the **Command Information** window.

      If you change the command or the separator, the appropriate command is updated to reflect those changes.

      If this window was opened when you created the script data source, the attributes were added to the new script data source.

If this window was opened from an existing script data source, then any changes to the attributes are made to the script data source. Any additional attributes are added, but any extra attributes are not removed. These options affect only the attributes that are parsed from the script output. Any derived attributes are not affected. If any of these attributes become invalid based on the attributes they reference, you can update or remove derived attributes manually. The derived attribute formula is displayed and not the actual result value.

Note: If the attribute group exists, to start a test, complete the following procedure

a. Select the attribute group on the **Agent Editor Data Sources Definition** page.

b. Select the script to be tested from the Command List

c. Click **Test** and follow the procedure at step "9" on page 126

10. If you skipped testing the command in step ("9" on page 126), use the following steps:

a) On the **Command List** page with the completed command information, click **Next**.

b) On the **Attribute Information** page, complete the attribute name and type information by using (Table 5 on page 40). Select **Add additional attributes** to add further attributes

c) On the **Attribute Information** page, use the **Script Attribute Information** tab to choose a specific data separator for this attribute.

The standard separator `;` is selected by default. You can choose a number of other separators such as a string, a number of characters, a tab, or a space. You can also choose to use a different string separator for the beginning and end of the data. Finally, you can also choose **Remainder of record** to assign the remainder of the record to the attribute. For more information about script parsing and separators, see "Script parsing and separators" on page 123.

11. Do one of the following steps:

- If you are using the **Agent** wizard, click **Next**.
- Click **Finish** to save the data source and open the Agent Editor.

12. You can add attributes and supply the information for them. For more information, see "Creating attributes" on page 37.

In addition to the fields applicable to all data sources (described in "Fields and options for defining attributes" on page 40), the **Data Sources Definition** page for the Script data source has the following options:

Command List
Provides access to the commands and scripts to start during data collection.

Add
Allows the user to add a command to be started by this attribute group.

Edit
Allows the user to edit an existing command entry.

Remove
Allows the user to delete an existing command entry.

Test
Allows the user to access the test environment for this attribute group.

Enable data collection using SSH
Selecting this check box enables SSH for this attribute group. If this check box is not selected, the attribute group runs locally.

For information about SSH remote connection configuration for script data sources, see "Configuring a Secure Shell (SSH) remote connection" on page 210.

Monitoring data from Java Database Connectivity (JDBC)

You can define a data source to receive data from a JDBC database. The agent runs an SQL query to collect data from the database. Each column that is returned by the query is an attribute in the resulting data set.

About this task

The JDBC data provider supports the following database servers:

- IBM DB2® 9.x and 8.x
- Microsoft SQL Server 2008, 2005, and 2000
- Oracle database 11g and 10g

Agent Builder does not include the JDBC drivers for these databases. The JDBC drivers are a set of JAR files that are provided by the vendor that are necessary to establish a JDBC connection to the database. For convenience, here are links to where those drivers can be downloaded:

- IBM DB2: JDBC drivers are included with the database server installation in a subdirectory named `java` located under the main DB2 installation directory.
- Microsoft SQL Server website at www.microsoft.com
- Oracle database: Oracle Database JDBC (http://www.oracle.com/technetwork/database/features/jdbc/index.html)

Note: An important thing to remember is that the JDBC data provider can remotely monitor your Database servers. A Java runtime environment and JDBC driver JAR files for the database server you are connecting to must be on the system where the agent runs.

The following versions of Java are supported:

- Oracle Corporation Java Version 5 or later
- IBM Corporation Java Version 5 or later

Procedure

1. On the **Agent Initial Data Source** page or the **Data Source Location** page, click **Data from a server** in the **Monitoring Data Categories** area.
2. In the **Data Sources** area, click **JDBC**.
3. Click **Next**.
4. On the **JDBC Information** area in the **JDBC Information** page, click **Browse** to connect to a database and build your SQL Query.

   Use the JDBC Browser to connect to a database and view its tables so you can build an SQL query that collects the data you need. When you select a table and columns, a query is generated for you and attributes are added for each of the columns returned by the query. You can modify and test the query that is generated to make sure the data that is returned is what you need.

   Note: You can also manually create the JDBC data source without clicking **Browse**. If you want to manually create the data source, specify the query and click **Next**. You must define an attribute for each column returned by the query, in the order that the columns are returned.

   With the JDBC data provider, you can run SQL queries and stored procedures against a database to collect monitoring data. When you specify an SQL query to collect data, you can include a where clause in your SQL statement to filter the data that is returned. The SQL statement can also join data from multiple tables. In addition to SQL select statements, the JDBC data provider can run stored procedures. For information about running stored procedures, see “Stored procedures” on page 133.
5. The first time the Browser opens, the Java Database Connectivity (JDBC) Browser window indicates that no connections are selected. You must add a connection. Click **Add** and follow the Steps to add a connection.

If you already defined a connection, that connection is used and you can proceed to Step "6" on page 129.

Note: The **Status** field shows the status of the current connection.

Use the following steps to add a connection:

a) On the **JDBC Connections** page , click **JDBC Connection**, and click **Next**.

b) On the **Connection Properties** page, complete the fields as follows:

Connection Name
Name of the JDBC connection. Type a unique name for this connection. You use this name to reference the connection in the browser.

Database Type
Type of database. Select the database product to which you are connecting. For example, to connect to the IBM DB2 database, select **DB2**.

User Name
Must be defined with at least read access to the database, but does not have to be the database administrator

Password
Must be defined with at least read access to the database, but does not have to be the database administrator

Host name
Host name on which the database server is running. With JDBC, you can monitor remote databases so you are not restricted to monitoring databases on the local system.

Port
Port on the host name on which the database server is listening.

Database
Name of the database to which to connect.

Jar Directory
Directory containing the JDBC JAR files used to connect to the database. Type the path name, or click **Browse** to locate the directory.

c) Optional: Select the **Save the password in the Agent Builder workspace** check box if you want to save the password for this connection.

d) Optional: Select the **Set as agent configuration defaults** check box if you want the defaults for this application server type to be copied from these properties.

If you are building the agent on a system that is similar to your monitored systems, it is advisable to check this box. If you do not check this box, the user who configures the agent sees an empty field. The user must then determine the values for all of the information without default values.

e) Click **Test Connection** to create a connection to the database that uses the configuration parameters you specified.

A message on the **Connection Properties** page indicates whether the connection succeeds.

f) When you have a working connection, click **Finish**.

6. In the **Java Database Connectivity (JDBC) Browser** window, a connection is made to the configured database. The tables that are contained in the database are shown in the **Database Tables** area. Select a database table to see the columns that are contained in that table in the **Columns in the selected table** area.

Note:

a. Click the binoculars icon to search for a table in the **Database Tables** list.

b. All tables are shown by default. You can filter the tables that are shown by selecting a different filter option. The available filter options are shown in Table 11 on page 130.

Table 11. Filter options	
Filter option	**Description**
All	Show all tables
User	Show only user tables
System	Show only system tables
View	Show only database views

Note: If you want to retrieve specific columns, select only these columns. If you select the table, Agent Builder automatically builds a query that gathers all of the columns from the table and creates attributes for all the columns that are currently in the table.

You can select columns in the following ways:

- Select the table and get the default query for all columns.
- Select columns to get only those columns.

7. Optional: Modify the enumeration values that are set for Error, Missing data, and No value in the **Attribute Information** page.

   Modify the values to avoid any overlap with legitimate values that might be returned from database table columns.

8. Optional: Click **Test** on the **Java Database Connectivity (JDBC) Browser** window to test and modify the SQL statement.

   The **Run the SQL statement** window opens.

   a) Enter or modify the SQL statement in the **SQL statement** field.

   b) Click **Run** to run the SQL statement.

      The results are displayed in the **Results** area. Continue to modify and test the statement until you are satisfied with the data that is returned.

   c) Click **OK** to save the statement, create the correct attributes, and return to the **JDBC Information** window.

9. Optional: Click **Test** on the **JDBC Information** window to test the attribute group in a more realistic agent environment. For more information about testing JDBC attribute groups, see “Testing JDBC attribute groups” on page 134. If you change the JDBC statement during this test, you must also adjust the attributes so that there is one attribute per column returned by the JDBC statement, in the correct order.
10. Optional: You can create a filter to limit the data that is returned by this attribute group by clicking **Advanced**. For more information about filtering data from an attribute group, see “Filtering attribute groups” on page 45
11. On the **JDBC Information** page, **Operating Systems** section, select the operating systems, and click **Next**. See “Specifying operating systems” on page 56 for information about which operating systems to select.

    Note: Click **Insert Configuration Property** to select a property to insert. For more information, see (Chapter 10, “Customizing agent configuration,” on page 205).
12. On the **Select key attributes** page, select key attributes or indicate that this data source produces only one data row. For more information, see “Selecting key attributes” on page 15.
13. If you want to test a data source that you previously defined, in the Agent Editor window, select the **Data Sources** tab and select a JDBC data source. In the **JDBC Attribute Group Information** area, click **Test**. For more information about testing, see “Testing JDBC attribute groups” on page 134.
14. If you want to view the configuration sections that were automatically generated, click the **Insert Configuration Property** tab of the Agent Editor.

    You can change the labels or default values for these properties to match the defaults that the user sees when they initially configure the agent.

15. Optional: Complete the **Attribute Information** page; for details, see "Fields and options for defining attributes" on page 40. Do this step if you chose to manually create the JDBC data source without clicking Browse in step "4" on page 128.

    The Agent Builder JDBC data source supports collecting data from most SQL types. The information in Table 12 on page 131 describes the type of attribute that is created by the JDBC Browser when it detects a column of one of these types. These data types are the supported types for use with a monitoring agent.

Table 12. Supported SQL data types for use with a monitoring agent

SQL data type	IBM Tivoli Monitoring attribute that is created
BIGINT	This data type is a 64-bit gauge value in IBM Tivoli Monitoring. If you select IBM Tivoli Monitoring V6.2 compatibility, it is a 32-bit gauge.
DECIMALDOUBLEFLOATNUMERICREAL	These SQL Types are created as 64-bit gauge attributes in IBM Tivoli Monitoring. If the database metadata contains a scale value, that value is used; otherwise, the scale is set to 1. If you select IBM Tivoli Monitoring V6.2 compatibility, the attribute is a 32-bit gauge.
BITINTEGERSMALLINTTINYINT	The following SQL types are created as 32-bit gauge attributes in IBM Tivoli Monitoring.
BOOLEAN	This value is a 32-bit gauge in IBM Tivoli Monitoring with enumerations for TRUE and FALSE.
TIMESTAMP	Data in columns of this type are converted to a 16-byte IBM Tivoli Monitoring time stamp attribute.
TIMEDATECHARLONGVARCHARVARCHAR	These SQL types are all treated as string attributes by the browser. The column size is used as the attribute size up to 256, which is the default string attribute size for the JDBC browser.

    Note: If you collect data from a data type that is not listed, a string attribute is used by default. The agent also tries to collect the data from the database as a string.

    Modify the enumeration values that are set for Error, Missing data, and No value in the **Attribute Information** page, if required. Modify the values to avoid any overlap with legitimate values that might be returned from database table columns.

JDBC configuration

When you define a JDBC data source in your agent, some configuration properties are created for you.

If you define a JDBC data source in your agent, the agent must use Java to connect to the JDBC database server. Java configuration properties are added to the agent automatically. The following Java configuration properties are specific to the agent runtime configuration:

- *Java Home*: A fully qualified path that points to the Java installation directory
- *JVM Arguments*: Use this parameter to specify an optional list of arguments to the Java virtual machine.
- *Trace Level*: This parameter defines the amount of information to write to the Java trace log file. The default is to write only Error data to the log file.

Note: Agent Builder does not require the Java properties because it uses its own JVM and logging, which are configured through the JLog plug-in.

If you define a JDBC data source in your agent, the following required, common configuration fields are added to the agent automatically:

- *JDBC database type*: Type of database to which you are connecting, IBM DB2, Microsoft SQL Server, or Oracle Database Server.
- *JDBC user name*: User name that is used to authenticate with the database server.
- *JDBC password*: Password that is used to authenticate with the database server.
- *Base paths*: List of directories that are searched for JAR files that are named in the *Class Path* field, or directories that are named in the *JAR directories* field, that are not fully qualified. Directory names are separated by a semi-colon (`;`) on Windows, and by a semi-colon (`;`) or colon (`:`) on UNIX systems.
- *Class path*: Explicitly named JAR files to be searched by the agent. Any files that are not fully qualified are appended to each of the Base Paths until the JAR file is found.
- *JAR directories*: List of directories that are searched for JAR files. Directory names are separated by a semi-colon (`;`) on Windows, and by a semi-colon (`;`) or colon (`:`) on UNIX systems. The JAR files in these directories do not have to be explicitly identified; they are found because they are in one of these directories. Subdirectories of these directories are not searched. Any directories that are not fully qualified are appended to each of the Base Paths until the directory is found.

The runtime configuration also requires that you specify some additional details to connect to the database. You can choose how to specify the remaining configuration items, either as a JDBC URL or as basic configuration properties (the default):

- URL configuration option
  - JDBC connection URL: Vendor-specific connection URL that provides details on which host the database is located and the port number to which to connect. The URL format typically looks as follows:

    ```
    jdbc:identifier://server:port/database
    ```

    see the JDBC driver vendor documentation for the different URL formats.
- JDBC Basic Properties option (default)

  JDBC server name: Host name that the database server is running on.
  JDBC database name: Name of the database on the host where the connection is made.
  JDBC port number: Port number on which the database server is listening.

Note: With the JDBC data provider, you can monitor multiple database types in the same agent by using subnodes. To monitor in this way, you must carefully define the Subnode Configuration Overrides. If you monitor multiple database types, the following configuration settings are likely to be different:

- JDBC database type
- JDBC user name
- JDBC password

If you are using the basic configuration option, you must also define overrides for the following properties on the **Subnode Configuration Overrides** page:

- JDBC server name
- JDBC port number
- JDBC database name

To define the configuration overrides for your subnode, see Chapter 9, “Using subnodes,” on page 187 for more details about accessing the **Subnode Configuration Overrides** page. When you configure the agent at run time, all of these properties must be configured for each new subnode instance that is created.

In addition to configuration overrides, your agent must also point to JDBC drivers for each database type that you plan to connect to from your subnodes. The *JAR directories* parameter is the most convenient way to point to your JDBC drivers. List the directories that contain the JDBC drivers by using a semicolon to separate each directory. For example, if you are connecting to DB2 and Oracle databases with the agent, you must specify a *JAR directories* value similar to this example: `C:\Program Files\IBM \SQLLIB\java;C:\oracle\jdbc`.

Stored procedures

Example SQL and DB2 stored procedures that you can use with the JDBC data provider.

The JDBC data provider can process the result sets returned by a stored procedure. String or integer input parameters can be passed to the stored procedure. The following syntax runs a stored procedure:

```
call[:index] procedureName [argument] ...
```

Where:

index
An optional integer that specifies which result set is to be used by the data provider. This parameter is useful when the stored procedure returns multiple result sets and you want to collect only the values from one of the result sets. If an index is not specified, data from each result set is collected and returned.

procedureName
The name of the stored procedure that is to be run by the JDBC data provider.

argument
An input argument to the stored procedure. Multiple arguments must be separated by a space. If the argument contains a space character, enclose the entire argument in double quotation marks. If the argument can be parsed as an integer, it is passed to the stored procedure as an integer argument. Any argument that is enclosed in double quotation marks is passed as a string argument.

SQL Server Samples

`call sp_helpdb`

Runs the procedure `call sp_helpdb` which requires no arguments. Data from all returned result sets are included in the data that is returned by the data provider.

`call:2 sp_helpdb master`

Runs the procedure `sp_helpdb` with the master argument. This argument is a string input argument. Only data from the second result set that is returned by the stored procedure is included in the data that is returned by the data provider.

When the index is not specified, data from all returned results sets is collected. You must ensure that the data returned in these cases is compatible with the attributes you define. Agent Builder creates attributes from the first returned result set, and any further result sets are expected to be compatible with the first one.

DB2 stored procedure

Here is a sample DB2 function that is written in SQL. This function demonstrates how to return results that can be processed by the Agent Builder JDBC data provider:

```
-- Run this script as follows:
-- db2 -td# -vf db2sample.sql

-- Procedure to demonstrate how to return a query from
-- a DB2 stored procedure, which can then be used by
-- an Agent Builder JDBC provider. The stored procedure
-- returns the following columns:
-- Name              Description              Data Type
-- current_timestamp  The current system time  timestamp
-- lock_timeout       The lock timeout         numeric scale 0
-- user               The user for the session String 128 characters long
```

```
DROP procedure db2sample#

CREATE PROCEDURE db2sample()
  RESULT SETS 1
  LANGUAGE SQL
BEGIN ATOMIC

  -- Define the SQL for the query
  DECLARE c1 CURSOR WITH HOLD WITH RETURN FOR
  SELECT CURRENT TIMESTAMP as current_timestamp,
CURRENT LOCK TIMEOUT as lock_timeout, CURRENT USER as user
  FROM sysibm.sysdummy1;

  -- Issue the query and return the data
  OPEN c1;
END#
```

This function can be called from Agent Builder by using the same syntax that is defined for other stored procedures. In this case, you define `call db2sample` as your JDBC statement to run this stored procedure.

Oracle stored procedures

Oracle stored procedures do not return result sets. Instead, you must write a function that returns an Oracle reference cursor. Here is a sample Oracle function that is written in PL/SQL that demonstrates how to return results that can be processed by the Agent Builder JDBC data provider:

```
CREATE OR REPLACE FUNCTION ITMTEST
RETURN SYS_REFCURSOR
IS
        v_rc SYS_REFCURSOR;
BEGIN
  OPEN v_rc FOR SELECT * FROM ALL_CLUSTERS;
  RETURN v_rc;
END;
```

This function can be called from Agent Builder by using the same syntax that is defined for other stored procedures. In this case, you define `call ITMTEST` as your JDBC statement to run this stored procedure. Because the Oracle function must return a cursor reference, only one result set can be processed by Oracle functions. This means that the index option is not supported for Oracle because there is no way to return multiple result sets.

Testing JDBC attribute groups

You can test the JDBC attribute group that you created, within Agent Builder.

Procedure

1. You can start the Testing procedure in the following ways:
   - During agent creation click **Test** on the **JDBC Information** page.
   - After agent creation, select an attribute group on the Agent Editor **Data Source Definition** page and click **Test**. For more information about the Agent Editor, see Chapter 4, “Using the Agent Editor to modify the agent,” on page 17.

   After you click **Test** in one of the previous two steps, the **Test JDBC Statement** window is displayed.
2. Optional: Before you start testing, you can set environment variables, configuration properties, and Java information.

   For more information, see “Attribute group testing” on page 229. For more about JDBC configuration properties, see (“JDBC configuration” on page 131).
3. Click **Start Agent**.

   A window indicates that the Agent is starting.
4. To simulate a request from Tivoli Enterprise Portal or SOAP for agent data, click **Collect Data**.

The agent queries the database with the specified SQL query. The **Test JDBC Statement** window collects and shows any data in the agent's cache since it was last started.

Note: The order of the returned data is significant; for example, the data value in the first returned column is always assigned to the first attribute. If you change the JDBC statement, you must add, remove, or reorder the attributes to match the columns returned by the statement.

5. Optional: Click **Check Results** if the returned data is not as you expected.

   The **Data Collection Status** window opens and shows you more information about the data. The data that is collected and displayed by the Data collection Status window is described in "Performance Object Status node" on page 280
6. Stop the agent by clicking **Stop Agent**.
7. Click **OK** or **Cancel** to exit the **Test JDBC Statement** window. Clicking **OK** saves any changes that you made.

Related concepts

"Testing your agent in Agent Builder" on page 229
After you use Agent Builder to create an agent, you can test the agent in Agent Builder.

Monitoring system availability by using Ping

You can define a data source to test a list of network devices by using the Internet Control Message Protocol (ICMP) echo ping. The host name or IP address of the devices you want to test are listed in one or more device list files. A separate Ping configuration file specifies the path to each device list file. Then, the name of the Ping configuration file is set in the agent runtime configuration. The results include the status of each network device.

Before you begin

Create device list files and a ping configuration file (see "Configuration files" on page 136).

About this task

Part of network management involves the ability to determine whether systems respond to an Internet Control Message Protocol (ICMP) ping. Use this data source to monitor basic online or offline status for a set of servers or other critical devices in your environment. Monitoring with ping is simple and low-overhead. To monitor a list of devices, add the Ping data collector to your agent.

Procedure

1. On the **Agent Initial Data Source** page or the **Data Source Location** page, click **Network management data** in the **Monitoring Data Categories** area.
2. In the **Data Sources** area, click **Ping**.
3. Click **Next**.
4. In the **Operating Systems** area in the **Ping Information** window, select the operating systems.
5. Optional: You can test this attribute group by clicking **Test**. For more information about testing, see "Testing Ping attribute groups" on page 137
6. Optional: You can create a filter to limit the data that is returned by this attribute group by clicking **Advanced**. For more information about filtering data from an attribute group, see "Filtering attribute groups" on page 45
7. Do one of the following steps:

   a) If you are using the **Agent** wizard, click **Next**.

   b) Click **Finish** to save the data source and open the Agent Editor.
8. For more information about adding attributes, see ("Creating attributes" on page 37).

Results

For more information about the attribute group for Ping, see “Ping attribute group” on page 309.

Configuration files

You provide the agent with the list of devices to ping by using configuration files.

The agent requires two types of configuration files.

Device list file

Includes a list of devices to ping. If you have many devices, you can divide them across multiple device list files. The agent starts a separate thread for each device list file and cycles through the files in parallel. It cycles through each file every 60 seconds or every 30 seconds plus the time it takes to ping the list, whichever is longer.

The syntax of the device list file is as follows:

```
LISTNAME=list_name
device_name or host_name
device_name or host_name
device_name or host_name device_name or host_name
```

Where *list_name* is a description for the devices in that file. If no list name is defined, the name of the device list file is used. The list name does not need to be the first entry in the file. However, if the file has multiple list name definitions, the last definition is used.

There is no limit to the number of devices you can include in a device list file. However, including too many entries defeats the purpose of having a targeted list of critical devices and increases the overall workload. It might be more difficult to retrieve the status of each device within the 60-second monitoring interval.

At the start of each cycle, the agent checks the last modification time of the device list file. If the last modification time of the file is more recent than the last time the agent read the file, the agent rereads the file without requiring a restart.

Ping configuration file

Specifies the location of each device list file. Use the fully qualified path or a path relative to the location of the ping configuration file. The ping configuration file is passed as a runtime configuration parameter to the agent.

Example

In the following example, devices are divided into two files.
The `/data/retailList.txt` file contains the following entries:

```
LISTNAME=Retail
frontend.mycompany.com
productdb.mycompany.com
```

The `/data/manufacturingList.txt` file contains the following entries:

```
LISTNAME=Manufacturing systems
manufloor.mycompany.com
stats.supplier.com
```

The ping file, `/data/pinglists.txt`, contains the following entries:

```
/data/retailList.txt
/data/manufacturingList.txt
```

Network Management configuration property

After a ping data source is added, the configuration is displayed on the **Runtime Configuration Information** page of the Agent Editor.

The **Network Management** configuration section of the **Runtime Configuration Information** page contains the following property:

Table 13. Network Management configuration properties

Name	Valid values	Required	Description
`Ping configuration file`	Path to a file	No. If this file is not provided, the `KUMSLIST` file is used from the agent bin directory.	The path to the file that contains a list of files, each containing a list of hosts to monitor by using ICMP pings.

Testing Ping attribute groups

You can test the Ping attribute group that you created within Agent Builder.

Procedure

1. You can start the Testing procedure in the following ways:
   - During agent creation click **Test** on the **Ping Information** page.
   - After agent creation, select an attribute group on the Agent Editor **Data Source Definition** page and click **Test**. For more information about the Agent Editor, see Chapter 4, "Using the Agent Editor to modify the agent," on page 17.

   After you click **Test** in one of the previous two steps, the **Test Settings** window opens.
2. Optional: Before you start testing, you can set environment variables and configuration properties. For more information, see "Attribute group testing" on page 229.
3. Click **Browse** to select a Ping configuration file. For more about Ping configuration files, see "Configuration files" on page 136
4. Click **Start Agent**. A window indicates that the Agent is starting.
5. To simulate a request from the monitoring environment for agent data, click **Collect Data**. The agent pings the devices that are specified in the device list file, which is referenced from the Ping configuration file.
6. The **Test Settings** window collects and shows any data in the agent's cache since it was last started.
7. Optional: Click **Check Results** if the returned data is not as you expected.

   The **Data Collection Status** window opens and shows you more information about the data. The data that is collected and shown by the Data collection Status window is described in "Performance Object Status node" on page 280.
8. Stop the agent by clicking **Stop Agent**.
9. Click **OK** or **Cancel** to exit the **Test Settings** window. Clicking **OK** saves any changes that you made.

Related concepts

"Testing your agent in Agent Builder" on page 229
After you use Agent Builder to create an agent, you can test the agent in Agent Builder.

Monitoring HTTP availability and response time

You can configure a data source to monitor the availability and response time of selected URLs. Use a configuration file to define a list of URLs. Set the name of the file in the agent runtime configuration. In IBM Tivoli Monitoring, you can also use Take Action commands to add and remove monitored URLs. The status for each URL is added as a line in the resulting data set.

About this task

For each URL you monitor, the results provide general information about the HTTP response to the HTTP request. The results include whether it can be retrieved, how long it takes to retrieve, and the size of the response. If the response content is HTML, information is also provided about the page objects within the URL.

You can monitor URLs that use the HTTP, HTTPS, FTP, and file protocols. You specify the URLs to monitor in the HTTP URLs file, or through Take Action options.

Important: At the time of release, Take Action commands are not available in an IBM Cloud Application Performance Management environment. They are available only in a Tivoli Monitoring environment.

This data source requires a Java runtime environment. The following versions of Java are supported:

- Oracle Corporation Java Version 5 or later
- IBM Corporation Java Version 5 or later

Use the following procedure to create an attribute group to monitor a list of URLs:

Procedure

1. On the **Agent Initial Data Source** page or the **Data Source Location** page, click **Data from a server** in the **Monitoring Data Categories** area.
2. In the **Data Sources** area, click **HTTP**.
3. Click **Next**.
4. On the **HTTP Information** page , select one or more operating systems in the **Operating Systems** area.
5. Optional: Click **Test** to test this attribute group. For more information about testing, see "Testing HTTP attribute groups" on page 145
6. Optional: Click **Advanced** to create a filter to limit the data that is returned by this attribute group. For more information about filtering data from an attribute group, see "Filtering attribute groups" on page 45
7. Do one of the following steps:
   a) If you are using the **Agent** wizard, click **Next**.
   b) Click **Finish** to save the data source and open the Agent Editor.

Results

The HTTP data source creates two attribute groups: Managed URLs and URL Objects. You can add, modify, or delete attributes.

Related tasks

"Creating attributes" on page 37
You can add new attributes to a data set.

Related reference

"HTTP attribute groups" on page 312
The two HTTP attribute groups, `Managed URLs` and `URL Objects`, are used to receive information from URLs and the objects within theses URLs.

HTTP tables

Reference information about the default HTTP attribute groups.

The two attribute groups that are created by the HTTP data source are:

`Managed URLs`
The Managed URLs table provides availability and response time data about each URL being monitored.

URL Objects
The URL Objects table contains a separate URL entry for each embedded object. For example, the `.gif` and `.jpg` files that might be used in the website that is listed in the Managed URL report.

For information about the syntax that is used in the Managed URLs and URL Objects tables, see ("Specific fields for HTTP attributes" on page 139).

When you want to monitor the response time and availability of specific objects within a website, review the contents of the URL Objects table. The URL Objects table monitors a specific list of objects that are detected in downloaded HTML files. The following table lists the HTML elements that are searched for objects to monitor and the attributes within these elements that reference the objects:

Table 14. HTML elements searched for objects to monitor

HTML element	Attribute containing object to be monitored
`img`	`src`
`script`	`src`
`embed`	`src`
`object`	`codebase` or `data`
`body`	`background`
`input`	`src`

In the following example HTML extract, the object that is monitored is the image that is referenced by the `src` attribute of the `img`element.

```
<img src="/v8/images/id-w3-sitemark-simple.gif" alt="" width="54" height="33" />
```

A full URL to the image is calculated based on the URL to the source document.

Note: If you do not want to monitor objects that are found in a web page, in the URL Monitoring configuration section, set the **Page object collection** property to **No**.

Specific fields for HTTP attributes

In the **Attribute information** page , there are two fields for HTTP attributes that define how data is collected from the URL. The **Attribute Type** field can be any value from a list that controls the information about the URL that is returned. Some attribute types require a value in the **Type Value** field.

The following table describes all of the attribute types for the Managed URLs attribute group, and the type value when one is required:

Table 15. HTTP Attribute Information - Managed URLs

Attribute type	Description	Type value	Data type that is returned	Differences with FTP and file protocols
`XPath Query`	Runs an XPath query on the content that is returned from a URL connection. The query must be written to return data useful for an attribute, not a list of nodes.	The XPath query to run against the content that is obtained from a URL connection.	The data that is returned can be a string, a numeric, or a timestamp value. If the data is in the XML DateTime format, you can specify timestamp as the attribute type. The agent converts the value to a Candle Timestamp.	None
`Response Time`	The amount of time in milliseconds that it took to download the content from the requested URL.	None	Integer (number of milliseconds)	None
`Response Message`	The HTTP response message that is returned by the server.	None	String	The response message applies only if the URL uses the HTTP or HTTPS protocols.
`Response Code`	The HTTP response code that is returned by the server.	None	Integer	The response code applies only if the URL uses the HTTP or HTTPS protocols. It is always 0 for file or FTP URLs.
`Response Length`	The size of the content in bytes that is downloaded from the requested URL	None	Integer (size in bytes)	None
`Response Header`	The response header can be used to retrieve a value from one of the URL response header fields. The argument specifies which field is requested.	The response header to collect.	String	Generally FTP and file protocols do not have any headers that can be collected.

Table 15. HTTP Attribute Information - Managed URLs (continued)				
Attribute type	**Description**	**Type value**	**Data type that is returned**	**Differences with FTP and file protocols**
`Request URL`	The connection is made to this URL. All of the response keywords provide information about the connection to this URL. The XPath Query can be used to obtain information that is obtained from the content that is returned by accessing this URL.	None	String	None
`Page Objects`	The number of objects that are discovered on the monitored HTML page that are monitored by the URL Objects attribute group.	None	Integer	None
`Total Object Size`	The total size of the objects that is monitored in the URL Objects attribute group for this web page.	None	Integer (in bytes)	None
`Alias`	The user specified alias for this URL.	None	String	None
`User`	The user specified data for this URL.	None	String	None

The following table describes the attribute types for the URL Objects attribute group:

Table 16. HTTP Attribute Information - URL Objects				
Attribute type	**Description**	**Type value**	**Data type that is returned**	**Differences with FTP and file protocols**
`URL`	The URL that is monitored in the Managed URLs table.	None	String	None
`Object Name`	The URL for the object that is monitored within the **HTML** page.	None	String	None

Table 16. HTTP Attribute Information - URL Objects (continued)				
Attribute type	**Description**	**Type value**	**Data type that is returned**	**Differences with FTP and file protocols**
`Object Size`	The size in bytes of the content that is downloaded from the Object Name URL.	None	Numeric	None
`Object Response Time`	The time in milliseconds it took to download the page object.	None	Numeric	None

Monitoring a URL

You can start monitoring any URL by including it in the URLs file or by using the HTTP URL Add Take Action option.

URLs file

The URLs file specified in configuration can be in any directory. If this file does not exist or is empty, then you can start URL monitoring by using Take Actions. For more information, see "Take Action option" on page 142. If you already have a Tivoli Universal Agent that uses the Tivoli Universal Agent HTTP Data Provider, you can reuse the KUMPURLS file. When you are configuring the agent, point to your KUMPURLS file.

The following table provides examples of how URLs are entered in the URLs file, depending on the method by which they were added.

Table 17. URLs file entries	
URLs	**Added by**

`www.bbc.co.uk`
`http://weather.com`
`www.ibm.com`
	Manually adding entries to the file. If no protocol is specified, as in the `www.ibm.com` example, http is assumed.

`ftp://userid:password@ftpserver/index.html`	Manually added by using File Transfer Protocol (FTP)
`http://www.ibm.com USER=ibm ALIAS=ibm`	Using the HTTP URL Add Take Action
`file:/tmp/samples.html USER=samples \` `ALIAS=samples`	Using a HTTP URL Add Take Action that uses FTP
`http://google.com INTERVAL=60 CACHE=50 \` `USER=google ALIAS=search`	Example from the Tivoli Universal Agent KUMPURLS file

When you directly edit the URLs file, your changes are implemented when the agent does its next data collection.

Take Action option

You can also specify URLs to monitor through a Take Action option that is called `HTTP URL Add`.

Restriction: This option is not available in the current release of IBM Cloud Application Performance Management, because you can not start Take Action commands manually.

When this option is selected, a window is displayed where you can specify the following parameters:

URL
A required parameter that represents the URL itself. You can type this parameter with or without the `http://`or `https://`prefix.

Alias
An optional parameter that you can specify to associate a more meaningful name to a URL. No spaces are allowed in this parameter. If this parameter is not completed, the Alias Name defaults to blank.

User_Data
An optional parameter that you can specify to enter data about the URL. If this parameter is not completed, the `User_Data`defaults to `INITCNFG`.

After you complete the information and close the window, assign the `HTTP URL Add` action to the destination managed system that is associated with the agent. Monitoring begins immediately for the new URL. The URL is also added to the URLs file so that it continues to be monitored across agent restarts.

A corresponding Take Action option is named `HTTP URL Remove`. Use the `HTTP URL Remove` action to immediately stop monitoring for a particular URL. The removed URL is also deleted from the URLs file. The **HTTP URL Remove** window requests only the URL and `User_Data` values. The URL and `User_Data` values must match the values that are seen in the Tivoli Enterprise Portal or the Remove action fails. For example, if you omitted the `http://` from the URL field of the Add action, you must include it in the URL field of the Remove action. If you did not specify `User_Data`, you must specify `INITCNFG` as seen in the Tivoli Enterprise Portal.

If a URL is added manually to the URLs file, you can delete it with the Take Action. If you delete with the Take Action, you must specify the values as seen in the Tivoli Enterprise Portal. For example, if you added `www.ibm.com` to your URLs file, the Tivoli Enterprise Portal displays `http://www.ibm.com` as the URL and `INITCNFG` as the `User_Data`. To remove the URL with the Take Action, you must use the values that are seen in the Tivoli Enterprise Portal.

After you complete the information and close the window, assign the `HTTP URL Remove` action to the destination managed system that is associated with the agent.

Monitor `https://` URLs

The HTTP data source can monitor only secure `https://` URLs that do not require scripted access or interactive prompting.

If the `https://` URL can be retrieved with a standard HTTP Get call, then it can be monitored.

Proxy server

If the system where the agent is running requires a proxy to access the SOAP data provider, you must specify proxy server configuration properties.

For more information, see “Proxy Server configuration” on page 144.

HTTP configuration

Reference information about HTTP configuration.

After an HTTP data source is added, the configuration is displayed on the **Runtime Configuration** page of the Agent Editor. Configuration sections are added for URL Monitoring, for Proxy Server authentication, and for Java.

URL Monitoring configuration

The URL Monitoring configuration section contains the following properties:

Table 18. URL Monitoring configuration properties			
Name	**Valid values**	**Required**	**Description**
HTTP URLs file	Path to a file	Yes	The path to the file that contains a list of URLs.
Page Object Collection	Yes, No The default value is Yes.	No	Whether to download objects that are found in a web page and collect data from them.

Proxy Server configuration

The Proxy Server configuration section contains the following properties:

Table 19. Proxy Server configuration properties			
Name	**Valid values**	**Required**	**Description**
Proxy Hostname	String	No	The proxy host name to be used for HTTP connections.
Proxy User Name	String	No	The user name for the proxy server.
Proxy Port	Positive integer The default value is 80.	No	The HTTP port number of the proxy server.
Proxy Password	Password	No	The password for the proxy server.

Note: If the **Proxy Hostname** property is blank, no proxy is used.

Java configuration

If you define an HTTP data source in your agent, the agent must use Java to connect to the HTTP server. Java configuration properties are added to the agent automatically. The following Java configuration properties are specific to the agent runtime configuration. The Agent Builder does not require the Java properties because it uses its own JVM and logging, which are configured through the JLog plug-in):

Table 20. Java configuration properties			
Name	**Valid values**	**Required**	**Description**
Java Home	Fully qualified path to a directory	No	A fully qualified path that points to the Java installation directory.
Trace Level	Choice (The default value is Error)	Yes	Use this property to specify the trace level that is used by the Java providers.
JVM Arguments	String	No	Use this property to specify an optional list of arguments to the Java virtual machine.

Testing HTTP attribute groups

You can test the HTTP attribute group that you created, within Agent Builder.

Procedure

1. Start the Testing procedure in the following ways:
   - During agent creation click **Test** on the **HTTP Information** page.
   - After agent creation, select an attribute group on the Agent Editor **Data Source Definition** page and click **Test**. For more information about the Agent Editor, see Chapter 4, "Using the Agent Editor to modify the agent," on page 17

   After you click **Test** in one of the previous two steps, the **HTTP Test** window is displayed.
2. Click **Browse** to select the HTTP URLs file. For more information about URLs files, see "URLs file" on page 142.
3. Optional: Set environment variables, configuration properties, and Java information before you start testing.

   For more information, see "Attribute group testing" on page 229. For more information about HTTP configuration, see "HTTP configuration" on page 143.
4. Click **Start Agent**.

   A window indicates that the Agent is starting.
5. To simulate a request from Tivoli Enterprise Portal or SOAP for agent data, click **Collect Data**.

   The agent monitors the URLs defined in the HTTP URLs file. The **HTTP Test** window displays any data that is returned.
6. Optional: Click **Check Results** if the returned data is not as you expected.

   The **Data Collection Status** window opens and shows you more information about the data. The data that is collected and displayed by the Data collection Status window is described in "Performance Object Status node" on page 280
7. Stop the agent by clicking **Stop Agent**.
8. Click **OK** or **Cancel** to exit the **HTTP Test** window. Clicking **OK** saves any changes that you made.

Related concepts

"Testing your agent in Agent Builder" on page 229
After you use Agent Builder to create an agent, you can test the agent in Agent Builder.

Monitoring data from a SOAP or other HTTP data source

You can define a data source to receive data from an HTTP server (for example, using the SOAP protocol). The data source sends an HTTP request to an URL and parses the response (in XML, HTML, or JSON formats) into the attributes of the resulting data set. You can select the data that is retrieved from the request.

About this task

By using the SOAP data source, you can specify an HTTP URL and send a GET, POST, or PUT request. For POST or PUT requests, you can specify the associated POST data. An XML, HTML, or JSON response is retrieved and parsed, and the data is exposed to the monitoring environment in attributes. You can define the attributes as all of the values within a particular element. Or you can define custom `XPath` values to specify how to populate individual attributes. You can also combine the two mechanisms.

Use the following procedure to collect and parse XML, HTML, or JSON responses from a URL:

Procedure

1. On the **Agent Initial Data Source** page or the **Data Source Location** page, click **Data from a server** in the **Monitoring Data Categories** area.
2. In the **Data Sources** area, click **SOAP**.
3. Click **Next**.
4. On the **SOAP Information** page, enter a URL.

   The default value is:

   ```
   http://${KQZ_HTTP_SERVER_NAME}:${KQZ_HTTP_PORT_NUMBER}
   ```

   Note: You can use a configuration variable or multiple configuration variables that resolve to a URL. Click **Insert Configuration Property** to select a property to insert. For more information, see Chapter 10, “Customizing agent configuration,” on page 205.
5. Select a request type. The default request type is Get. For Post and Put requests, enter the data to be processed.

   Note: For Post and Put requests, the **Insert Configuration Property** is enabled. Click **Insert Configuration Property** to include a configuration variable in the data to be processed. For more information, see (Chapter 10, “Customizing agent configuration,” on page 205).
6. Click **Browse**

   Note: If after you enter a URL and select a request type, you do not want to use the SOAP browser to build the definition, enter a **Row Selection XPath**. You enter the **Row Selection XPath** in the **SOAP Information** window. Next, define all of the attributes for the attribute group.
7. In the **SOAP Browser** window, do the following steps:
   a) Enter a URL and select a request type if you did not already do so.
   b) Click **Configuration** to set any configuration properties that are referenced in the URL or other fields.
   c) Click **Connect** to obtain data from the SOAP provider.

      When you connect to the URL, a list of XML elements for this URL is shown in a Document Object Model (DOM) tree. An HTML or JSON response is converted to XML and displayed as a DOM tree. For details about conversion of a JSON response to XML, see “XML representation of JSON data” on page 149. In the WebSphere Application Server example in (Figure 14 on page 147), the following URL was entered:

      ```
      http://nc053011.tivlab.raleigh.ibm.com:9080/wasPerfTool/servlet
      /perfservlet?module= \threadPoolModule
      ```

      The `PerformanceMonitor` XML element is shown. This element is the top-level XML element in the XML document that is returned by the SOAP provider.

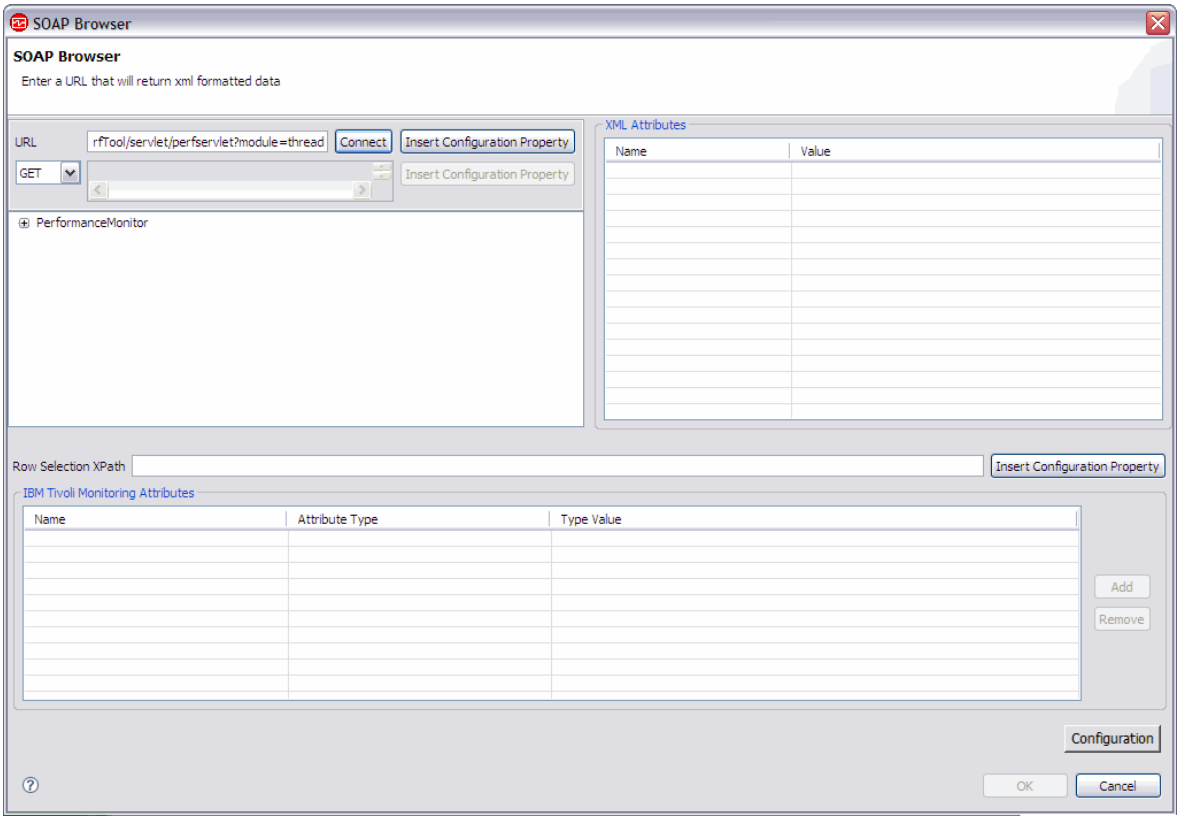


*Figure 14. **SOAP Browser** window*

d) In the DOM tree, find and select the XML node that you want to set as the **Row Selection XPath**.

In the WebSphere Application Server example in (Figure 15 on page 148), the `PerformanceMonitor/Node/Server/Stat/Stat/Stat` node is selected. This node represents a row of data in the attribute group. When you select a node in the DOM tree and click **Add**, you get all of the attributes and elements defined on that node of the tree. (You click **Add** in the **Agent Attributes** area).

When a node is selected, the **XML Attributes** area shows any XML attributes defined for the selected node. Select an XML attribute and click **Add** to include this attribute in the list of Agent Attributes.

Note: If more than one row of data is expected, the XPath must map to a node set. Where the Row Selection XPath returns a node that is set with only one item, the attribute group contains only one row.

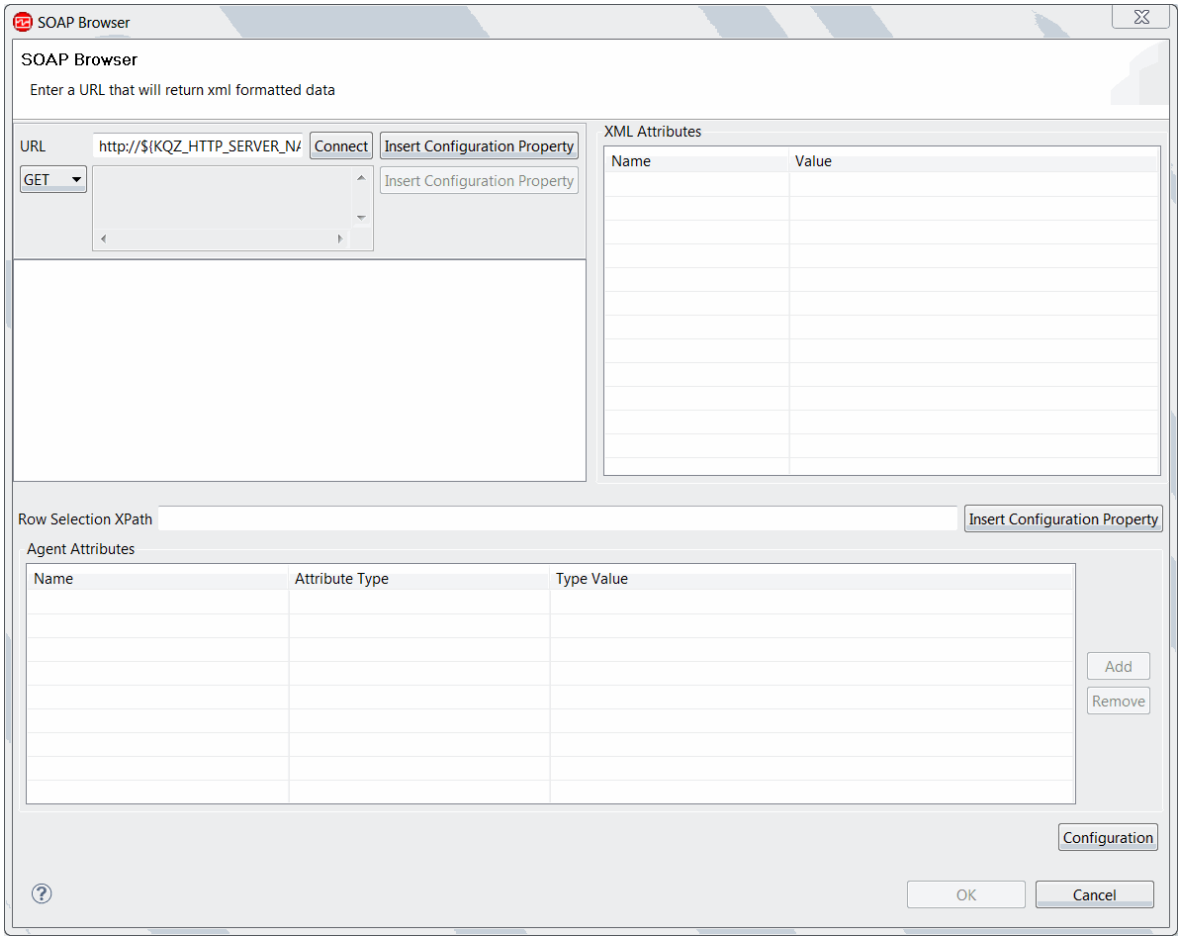


*Figure 15. **SOAP Browser** window*

e) Click **Add** in the Agent Attributes area.

The list of agent attributes is shown and the **Row Selection XPath** field is filled.

The XPath for each agent attribute is used to map XML nodes or elements to agent attributes. In the WebSphere Application Server example in theFigure 16 on page 149, the first attribute in the list of agent attributes, `Stat`, is not of use and would be removed.

You can edit the name and XPath for an agent attribute in the **Type Value** field. For more information about using XPaths, see “XPath options” on page 151

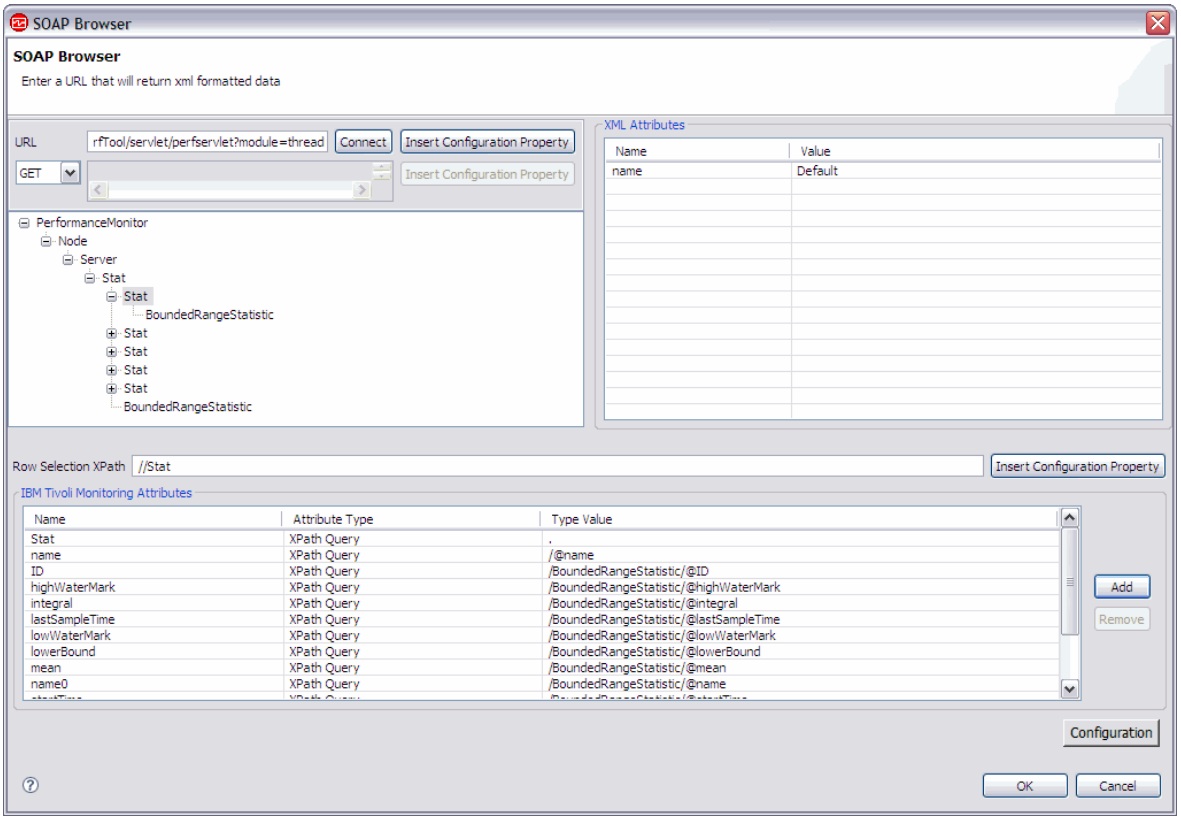


*Figure 16. **SOAP Browser** window*

f) In the **SOAP Browser** window, click **OK** to save your changes and return to the **SOAP Information** window.

8. In the **SOAP Information** window, click **Next**.
9. If you did not use **Browse** earlier and you entered the **URL** and **Row Selection XPath** in the **SOAP Information** window, the **Attribute Information** page is shown. Specify the information for the first attribute on the **Attribute Information** page, and click **Finish**. You can then specify more attributes by using the Agent Editor. For more information about creating attributes, see ("Creating attributes" on page 37).
10. If you used the **Browse** function in step "6" on page 146, the **Select key attributes** page is shown. On the **Select key attributes** page, select key attributes or indicate that this data source produces only one data row. For more information, see "Selecting key attributes" on page 15.
11. Optional: You can test this attribute group by clicking **Test**. For more information about testing, see "Testing SOAP attribute groups" on page 153
12. Optional: You can create a filter to limit the data that is returned by this attribute group by clicking **Advanced**. For more information about filtering data from an attribute group, see "Filtering attribute groups" on page 45
13. Do one of the following steps:
    a) If you are using the **Agent** wizard, click **Next**.
    b) Click **Finish** to save the data source and open the Agent Editor.

XML representation of JSON data

If the HTTP request returns JSON data, the data provider converts the data to XML.

The data provider converts the name of a JSON attribute to the element name. For a JSON attribute of a simple type, it converts the value to text data within the element. Embedded JSON objects are converted to embedded XML elements. Any subordinate attributes are converted to subordinate elements.

The root XML element is `JSON_document`.

If a JSON attribute name contains characters that are invalid in an element name, the data provider modifies it to produce a valid element name. The data provider also adds a `JSON_name` attribute to the element. The value of the attribute is the original JSON attribute name.

For every element of a JSON array, the data provider creates a `JSON_`*xxx*`_array_element` XML element, where *xxx* is the name of the array. The value of the array element is converted into text within the XML element. A `JSON_index` attribute is added to each XML element; the value of the attribute is the index of the array element within the array.

The data provider adds the following attributes to every element:

- `JSON_level`: the level of the node within the JSON file. The root of the tree, represented by the `JSON_document` tag, is level 1.
- `JSON_type`: the type of the JSON node (object, array, string, or number).

Specific fields for SOAP attributes

In the **Attribute Information** window, there are two fields for SOAP attributes that define how data is collected from the SOAP response.

The **Attribute Type** field can be any value from a list that controls the information about the response that is returned. Some attribute types require a value in the **Type Value** field. The default attribute type is XPath Query, which runs an XPath query against the SOAP server response content. The type value is the XPath query that is run. The following table describes all of the attribute types and the type value when one is required:

Table 21. SOAP Attribute Information

Attribute type	Description	Type value	Returned data type	Differences with FTP and file protocols
`XPath Query`	Runs an XPath query on the content that is returned from a URL connection. The query must be written to return data useful for an attribute, not a list of nodes.	The XPath query to run against the content that is obtained from a URL connection. If a row selection query was defined, this XPath query must be relative to the row selection query.	The data that is returned can be a string, a numeric, or a timestamp value. The Agent Builder browser for SOAP generally detects the correct data type for the attribute from the data that is being browsed. If the data is in XML DateTime format, you can specify timestamp as the attribute type and the agent converts the value to a Candle Timestamp.	None
`Response Time`	The amount of time in milliseconds that it took to download the content from the requested URL.	None	Integer (number of milliseconds)	None

Table 21. SOAP Attribute Information (continued)

Attribute type	Description	Type value	Returned data type	Differences with FTP and file protocols
`Response Message`	The HTTP response message that is returned by the server.	None	String	The response message applies only if the URL uses the HTTP or HTTPS protocols.
`Response Code`	The HTTP response code that is returned by the server.	None	Integer	The response code applies only if the URL uses the HTTP or HTTPS protocols. It is always 0 for file or FTP URLs.
`Response Length`	The size of the content in bytes that was downloaded from the requested URL	None	Integer (size in bytes)	None
`Response Header`	The response header can be used to retrieve a value from one of the URL response header fields. The argument specifies which field is requested.	The response header field to collect.	String	Generally FTP and file protocols do not have any headers that can be collected.
`Request URL`	The connection was made to this URL. All of the response keywords provide information about the connection to this URL. The XPath Query can be used to obtain information that is obtained from the content that is returned by accessing this URL.	None	String	None

XPath options

Using XML Path Language, you can select nodes from an XML document. A few of the possible uses of XPaths for the SOAP data sources include:

- Using predicates in the XPath to identify the XML elements that correspond to rows of data in the IBM Tivoli Monitoring attribute group. You can use predicates in the XPath that maps XML elements or attributes to Tivoli Monitoring attributes, as in the following example:

  ```
  Stat[@name="URLs"]/CountStatistic[@name="URIRequestCount"]/@count
  ```

  Where there are multiple location steps in the XPath, each location step can contain one or more predicates. The predicates can be complex and contain boolean values or formula operators. For example:

  ```
  //PerformanceMonitor/Node/Server[@name="server1"]/Stat/Stat/Stat[@name=
  "Servlets"]/Stat
  ```

- Including node set functions in the XPath, if a row contains multiple XML elements of the same type. And if the position of an XML element in the node list determines the Tivoli Monitoring attribute the element maps to. Examples of node set functions are, `position()`, `first()`, `last()`, and `count()`.
- Doing simple data transformation, such as substring. If you specify the following substring:

  ```
  substring(myXMLElement,1,3)
  ```

  the XPath returns the first three characters of the XML element, myXMLElement.

You can specify elements outside the context of the Row Selection XPath by using two periods, `(..`, as in the following example:

```
/../OrganizationDescription/OrganizationIdentifier
```

SOAP configuration

After a SOAP data source is added, the configuration is displayed on the **Runtime Configuration** page of the Agent Editor.

Configuration sections are added for HTTP Server, for Proxy Server, and for Java. For information about Proxy server configuration, see (“Proxy Server configuration” on page 144). For information about Java configuration, see “Java configuration” on page 144.

HTTP Server

The HTTP Server configuration section contains the following properties:

Table 22. HTTP Server configuration properties

Name	Valid values	Required	Description
HTTP user name	String	No	The HTTP user
HTTP password	Password	No	The HTTP server password
HTTP server name	String (The default value is localhost)	No	The host or IP address of the HTTP server
HTTP port number	Numeric (The default value is 80)	No	The host or IP address of the HTTP server
Certificate validation enabled	True, False (The default value is True)	Yes	Disabling certificate validation is potentially insecure
HTTP trust store file	Path to a file	No	The HTTP trust store file

Table 22. HTTP Server configuration properties (continued)			
Name	**Valid values**	**Required**	**Description**
HTTP trust store password	The HTTP trust store password	No	The HTTP trust store password

Proxy server

If the system where the agent is running requires a proxy to access the SOAP data provider, you must specify proxy server configuration properties. For more information, see "Proxy Server configuration" on page 144.

Testing SOAP attribute groups

You can test the SOAP attribute group that you created, within Agent Builder

Procedure

1. You can start the Testing procedure in the following ways:
   - During agent creation click **Test** on the **SOAP Information** page.
   - After agent creation, select an attribute group on the Agent Editor **Data Source Definition** page and click **Test**. For more information about the Agent Editor, see Chapter 4, "Using the Agent Editor to modify the agent," on page 17

   After you click **Test** in one of the previous two steps, the **Test SOAP Collection** window is displayed.
2. Optional: Before you start testing, you can set environment variables, configuration properties, and Java information.

   For more information, see "Attribute group testing" on page 229. For more information about SOAP configuration, see "SOAP configuration" on page 152.
3. Change the URL, Row Selection XPath, and request type.
4. Click **Start Agent**.

   A window indicates that the Agent is starting.
5. To simulate a request from Tivoli Enterprise Portal or SOAP for agent data, click **Collect Data**. This action populates the Results table and you can preview how the data is parsed and shown in columns in the Tivoli Enterprise Portal.

   In the Results area, you can change the attribute definitions and reload the data to see how your changes affect the attribute group. You can right-click in a column results area to display options to edit the attribute. The attribute edit options are:

   - **Edit Attribute**
   - **Hide Attribute**
   - **Insert Attribute Before**
   - **Insert Attribute After**
   - **Remove**
   - **Remove Subsequent Attributes**
   - **Remove All**
6. Optional: Click **Check Results** if the returned data is not as you expected.

   The **Data Collection Status** window opens and shows you more information about the data. The data that is collected and shown by the **Data Collection Status** window is described in "Performance Object Status node" on page 280.
7. Stop the agent by clicking **Stop Agent**.

8. Click **OK** or **Cancel** to exit the **Test SOAP Collection** window. Clicking **OK** saves any changes that you made.

Related concepts

"Testing your agent in Agent Builder" on page 229
After you use Agent Builder to create an agent, you can test the agent in Agent Builder.

Monitoring data by using a socket

You can define a data source to collect data from an external application by using a TCP socket. The application must initiate the TCP connection to the agent and send data in a structured XML format. Depending on the application, the data source can produce a data set with a single row, multiple rows, or event data.

About this task

Use the socket data source to provide data to the agent from an external application, running on the same system as the agent. The external application can send data to the agent anytime it wants to. For example, you can develop a command-line interface that allows a user to post data to an attribute group when it is run. Another option is to modify a monitored application to send updates to the agent. The agent does not start or stop the application that is sending data to the socket; this action is controlled by the user.

There are some limitations with the socket data source:

- By default only connections to the local host (127.0.0.1) are possible. For more information about configuring your agent to accept connections from a remote host, see "Remote socket port connection" on page 162.
- There is no mechanism in the socket API for the client to determine what subnodes are available. The client can send data for a specific subnode, but it must already know the subnode name.

Use the following procedure to create an attribute group to collect data by using a Transmission Control Protocol socket (TCP) socket.

Procedure

1. On the **Agent Initial Data Source** page or the **Data Source Location** page, click **Custom programs** in the **Monitoring Data Categories** area.
2. In the **Data Sources** area, click **Socket**.
3. Click **Next**.
4. On the **Socket Information** page, enter an Attribute group name.
5. Enter a help text for the attribute group.
6. Select whether the attribute group **Produces a single data row**, **Can produce more than one data row**, or **Produces events**. For more information, see "Sending data" on page 156.
7. In the Socket Information section, select a **Code page**. For more information, see "Character sets" on page 159.
8. Optional: Click **Advanced** to modify the advanced properties for the attribute group. The **Advanced** option is active when you select that the attribute group **Can produce more than one data row**, or **Produces events**.
9. Click **Next**.
10. On the **Attribute Information** page, specify the first attribute for the attribute group. For more information about creating attributes, see "Creating attributes" on page 37.
11. Click **Next**.
12. Optional: On the **Global Socket Data Source Information** page, **Error Codes** section, you can define the error codes that the socket client can send when it cannot collect data. For more information, see ("Sending errors instead of data" on page 157). To define an error code, use the following steps:

a) In the **Error Codes** section, click **Add**. An error code has a limit of 256 characters. Only ASCII letters, digits, and underscores are allowed. No spaces are allowed.

b) In the **Socket Error Code Definition** window, enter a display value that is shown in the **Performance Object Status** attribute group.

c) Enter an internal value. The internal value must be an integer from 1,000 to 2,147,483,647.

d) You must define a message text for each error. You can use message text that was entered previously by selecting it from the list. Click **OK** to return to the **Global Socket Data Source Information** page. The message text is used in the agent log file.

If no suitable message text is available, click **Browse** to set up the message text. The Messages (list) window opens. The message window lists messages that are defined in the agent. Until you define messages, the list remains blank. You can use **Edit** to alter a defined message and **Remove** to delete one or more messages that you defined.

e) In the Messages (list) window, click **Add** to see a **Message Definition** window. In the **Message Definition** window type, the text that describes the meaning of the new message and select the message type.

Note: The message identifier is automatically generated for you.

f) Click **OK**.

g) The Messages (list) window opens, with the new message. To verify the message and return to the **Global Socket Data Source Information** page, click **OK**.

13. Optional: In the **Supplemental Files** section of the **Global Socket Data Source Information** page, you can add files that are packaged with the agent. These files are copied to the agent system when the agent is installed.

The **File Type** column describes how each file is expected to be used. Three possible uses are described in the following table:

Table 23. File types for supplemental files

File Type	Description
Executable	Select this option if you want to include an executable file with the agent. The agent does not use these files.
Library	Select this option if you to include a library with the agent. The agent does not use these files.
Java resource	Select this option to include Java resources with the agent. The agent does not use these files.

For information about where the Supplemental Files are installed with your agent, see (“New files on your system” on page 246).

Click **Edit** to edit the imported file. For more information, see (“Editing a command file definition” on page 122).

14. Optional: You can test this attribute group by clicking **Test**. For more information about testing, see “Testing socket attribute groups” on page 163

15. Optional: If the data source is sampled, you can create a filter to limit the data that is returned by this attribute group by clicking **Advanced**. The data source is sampled when you did not select "Produces events" on the **Socket Information** page. For more information about filtering data from an attribute group, see “Filtering attribute groups” on page 45

16. Do one of the following steps:

a) If you are using the **Agent** wizard, click **Next**.

b) Click **Finish** to save the data source and open the Agent Editor.

Select the operating systems on which the agent listens to data from socket clients in the **Operating Systems** section of the **Socket Provider Settings** page. To open the page, click **Socket Provider Settings** in the outline view or click **Global Settings** in the Agent Editor on any socket attribute group page.

Note: Error codes and supplemental files can be updated in the **Error Codes** and **Supplemental Files** sections of the **Socket Provider Settings** page.

Sending socket information to the agent

When your agent contains one or more socket attribute groups, the agent opens a socket and listens for data from clients.

The application that sends socket data to the agent connects to a port that is defined in the agent. The port is either the value that is set by an agent configuration property or an ephemeral port that is allocated automatically by TCP/IP. For more information about socket ports and configuration, see "Socket configuration" on page 161.

The data that is received must follow a structured XML format. The following XML information flows are possible by using the socket data source:

- Send one or more rows of data to the agent for a sampled attribute group
- Send a row of data to the agent for an attribute group that Produces events
- Send an error code to the agent instead of data.
- Send a task prefix registration to the agent
- Receive a task request from the agent
- Send a task response to the agent

Sending data

An attribute group is defined to receive sampled data or event data. When you create the attribute group, you specify an option that indicates whether the data to be received:

- Produces a single data row
- Produce more than one data row
- Produces events

If you select **Produces a single data row** or **Can produce more than one data row**, that is a sampled attribute group. If you select **Produces events**, then your attribute group sends an event to the monitoring environment each time that a row is received.

When you view sampled data in the Tivoli Enterprise Portal or IBM Cloud Application Performance Management console, you see the latest set of collected rows. The data that is displayed for an event attribute group is the contents of a local cache that is maintained by the agent. For event data, the agent adds the new entry to the cache until the size is reached when the oldest one is deleted. For sampled data, the agent replaces the contents of the cache every time you send data.

If you select **Produces events** or **Produces a single data row**, you must send only one row of data to the agent for that attribute group in each message. You can send as many events as you want, send each event in a separate message.

Normally sampled data is collected by the agent on request, but the socket client provides updated samples on its own schedule. You can update a sampled attribute group (single row or multiple row) as often as you require. When the data is requested by Tivoli Monitoring or IBM Cloud Application Performance Management, the agent provides the latest data.

If there are missing rows of data for the socket attribute group in the Tivoli Enterprise Portal or IBM Cloud Application Performance Management console, check the errors in the log file. Also, if the data in the attribute group is not as expected, check the errors in the log file. The socket data source attempts to

process whatever it can from the input. For example, if the client sends three well-formed rows and one that is not valid (for example, malformed XML), you see:

- Three rows of data in the attribute group
- An error is logged for the malformed row in the agent's log file
- Since valid rows were returned, the Performance Object Status shows a status of NO_ERROR

For both event and sampled data, the data is sent to the agent as a single XML data flow from the socket client. Data that is sent from a socket client must always be terminated with a newline character: '\n'. The agent reads data until it sees the newline character and then an attempt is made to process what was received. Any data received that cannot be processed is discarded. The following is a sample of how you would send two rows of data to the agent for an attribute group named abc:

```
<socketData><attrGroup name="abc"><in><a v="1"/><a v="no"/><a v="5"/></in><in> \
<a v="3"/><a v="yes"/><a v="5"/></in></attrGroup></socketData>\n
```

This sample sends two rows of data to the agent where each row contains three attributes. The order of the attributes is important and must follow the order that is defined in your attribute group. The only exception to this is that the derived attributes must be skipped, regardless of where they are in your attribute group.

If the attribute group is defined in a subnode, then the subnode instance ID must be identified when data is sent to the agent. The subnode instance ID is identified by using the subnode attribute in the socketData element. A convention must be adopted for configuring subnode instance IDs for use by the socket client since the client cannot query instance IDs or configuration properties. Data sent to a subnode which is not configured is ignored.

Here is a sample:

```
<socketData subnode="app1"><attrGroup name="abc"><in><a v="1"/><a v="no"/><a v="5"/>
</in><in> \<a v="3"/><a v="yes"/><a v="5"/></in></attrGroup></socketData>\n
```

In this sample, the data is sent to the subnode with an instance ID equal to "app1". "app1" is not the managed system name, but the instance identifier that is specified when the subnode instance is configured.

The following XML elements make up the socket data:

socketData
The root element. It has one optional attribute that is called subnode that specifies the subnode instance ID.

attrGroup
This element identifies the attribute group that the socket data is for. The name attribute is required and is used to specify the attribute group name.

in
This element is required to identify a new row of data. All of the attribute values for a row of data must be children of the same in element.

a
The a element identifies an attribute value. The v attribute is required and is used to specify the attribute value.

Sending errors instead of data

Sometimes the application that posts socket data might not be able to collect the data necessary for an attribute group. In this case, instead of sending data to the agent, an error code can be returned. The error code gives you a way to tell the monitoring environment about your problem. An example error is:

```
<socketData><attrGroup name="abc"/><error rc="1000"/></attrGroup></socketData>\n
```

The error code must be defined in the agent in a list that is common to all of the socket attribute groups. When the agent receives an error code, the defined error message is logged in the agent log file. In

addition, the attribute group named Performance Object Status has an Error Code attribute is updated with the Error Code Type. The Error Code Type is defined for the error code you send.

For the previous example, you must define the Error Code Value of 1000 in the agent. See the following sample error code definition:

Table 24. Sample error code

Error Code Value	Error Code Type	Message
1000	APP_NOT_RUNNING	The application is not running

When the error code is sent, a message similar to the following is logged in the agent log file:

```
(4D7FA153.0000-5:customproviderserver.cpp,1799,"processRC") Received error code 1000
 from client. \Message: K1C0001E The application is not running
```

If you select Performance Object Status query from the Tivoli Enterprise Portal, the **Error Code** column for the row **abc** attribute group shows the value `APP_NOT_RUNNING` in that table.

Sending an error to a sampled attribute group clears any data that was previously received for that attribute group. Sending data to the attribute group causes the error code to no longer be displayed in the Performance Object Status attribute group. You can also send an error code of 0 to clear the error code from that table.

Sending an error to an attribute group that produces events does not clear the cache of events that were previously sent.

Handling `take action` requests

The socket client can register to receive `take action` requests from the agent when the action command matches a certain prefix. Any action that does not match is handled by the agent. The prefix must not conflict with actions that the agent is expected to handle, so use the agent product code as the prefix. `Take actions` provided with the Agent Builder are named after the data source that the `take action` uses. For example, the `JMX_INVOKE take action` operates on the JMX data source. Another example is the `SSHEXEC take action` which uses the SSH script data provider. Since these actions do not use the product code, the product code is a safe prefix to use as the `take action` prefix.

The socket client must be long running and leave the socket open. It must send a registration request for the prefix and listen for requests from the socket. The agent ensures that a timeout does not occur on the socket of a long-running client, even if no data is flowing. The following is a sample registration request:

```
<taskPrefix value="K42"/>\n
```

In this sample, any `take action` command that is received by the agent that begins with `"K42"` is forwarded to the socket client that initiated the registration. The following shows a sample `take action` request that the socket client might receive:

```
<taskRequest id="1"><task command="K42 refresh" user="sysadmin"/></taskRequest>\n
```

The `id` is a unique identifier that the agent uses to track requests that are sent to clients. When the socket client responds to the task, it must provide this identifier in the `id` attribute of the `taskResponse` element.

The socket client must process the action and send a response. A sample response is:

```
 <taskResponse id="1" rc="1"/>\n
```

If the action completes successfully, an `rc` attribute value of `0` is returned. The value of `rc` must be an integer, where any value other than 0 is considered a failure. The task return code value is logged to the agent log file and shown in the Take Action Status query that is included with the agent. The dialog that is displayed on the Tivoli Enterprise Portal after an action is run does not show the return code. That dialog

indicates whether the `take action` command returned success or failure. The agent log or Take Action Status query must be viewed to determine the actual return code if a failure occurred.

It is the agent developer's responsibility to document, create, and import any actions that are supported by the socket clients that are used with an agent. If users send unsupported actions to the socket client, the client must be developed to handle those scenarios in an appropriate manner. If users define more actions that start with the registered prefix, they are passed to the client. The client must be developed to handle those scenarios in an appropriate manner.

There is a timeout that controls how long the agent waits for a response from the socket client. The setting is an environment variable that is defined in the agent that called `CDP_DP_ACTION_TIMEOUT` and the default value is 20 seconds.

Note: The error code messages that are defined for socket data source attribute groups are not used for `take actions`. You can return the same return code values. However, the agent does not log the message that is defined or affect the `Error Code` field in the Performance Object Status attribute group.

Encoding of socket data

The socket client encodes data that is sent to the agent.

It is important to be aware of how your socket client is encoding data that is being sent to the agent.

Special characters

Data sent to the agent must not contain any newline characters except at the end of each event or data sample. Newline characters that occur inside of attribute values must be replaced with a different character or encoded as shown in (Table 25 on page 159). You must also be careful not to break the XML syntax with your attribute values. The following table shows the characters that occur in the attribute values that you encode:

Table 25. Characters to encode in attribute values

Character	Header
&	`&`
<	`<`
>	`>`
“	`"`
'	`'`
\n	`
`

Note: The agent uses the newline character to separate responses received from a client. Unexpected newline characters prevent data from being parsed correctly.

The agent does not contain a full-featured XML parser so you must not use special encoding for characters not in (Table 25 on page 159). For example, do not encode `¢` or `&cent` in place of a cent sign ¢.

Character sets

In addition to encoding special characters, the agent must know what code page was used to encode your data. Define each socket attribute group to indicate whether you are sending the data to the agent as **UTF-8** data or as **Local code page**. Be aware of how your client is sending data. If you use a client that is written in Java, specify **UTF-8** as the encoding on the writer you use to send data to the agent. Specify **UTF-8** as the **Code Page** for your attribute group. **Local code page** means the local code page of the agent. If the data is sent over a remote socket, it must conform to the local code page of the agent or use UTF-8.

Numeric Data

Be aware of how you are formatting your numeric attribute values. The numeric values that you send to the agent must not contain any special characters. One example is the thousands separator character. Other examples are currency symbols or characters that describe the units of the value. If the agent encounters a problem when it is parsing numeric data, it logs an error that indicates the issue. The Performance Object Status Error Code is not set when an attribute fails to parse. The following is an example error message from the agent log:

```
(4D3F1FD6.0021-9:utilities.cpp,205,"parseNumericString") Invalid characters :00:04 \
found getting numeric value from 00:00:04, returning 0.000000
```

Note: For information about how a time stamp attribute must be formatted, see (“Time stamp” on page 42).

Socket errors

Errors are written to the agent log file for problems that occur with data received from a socket client.

Other errors that are logged are `take actions` that return a value other than `0`. Error values that are sent by the socket client are logged along with the message associated with the error code.

The Performance Object Status for the attribute group is set when the socket client sends an error return code to the agent. Some other values can be seen in addition to the ones defined by the agent. The following table describes other “Error Code” values you are likely to encounter with socket attribute groups:

Table 26. Performance Object Status values

Error Code	Description
NO_ERROR	No error occurred. Indicates that there are no problems with the attribute group. Problems with a row of sampled data do not cause the state to change from NO_ERROR. You must validate the number of rows that are shown and the attribute values even when you see NO_ERROR as the error code.
NO_INSTANCES_RETURNED	A socket client sent no rows of data for a sampled attribute group. Not an error. It indicates that there are no instances of the resources that are being monitored by this attribute group.
XML_PARSE_ERROR	The agent failed to parse data that is received from the client. See the agent log for more details.
OBJECT_CURRENTLY_UNAVAILABLE	The client sent the agent an error code that was not defined in the global list of error codes.
GENERAL_ERROR	A problem occurred collecting data from the client, usually because the client did not reply to the request within the timeout interval. See the agent trace log for more details. The client can also specify GENERAL_ERROR as an error code, but it is better if a more detailed error code is defined.

Socket configuration

After you add a socket data source to your agent, you can configure the agent to accept data from a specified socket port.

About this task

After you add a Socket data source, the configuration is displayed on the **Runtime Configuration** page of the Agent Editor. The Socket configuration section contains the following property:

Table 27. Socket configuration property

Name	Valid values	Required	Description
Port number	0 or any positive integer The default value is 0	Yes	The port that the agent uses to listen on for data from socket clients. A value of 0 indicates that an ephemeral port is to be used.

The agent writes the value of the port that is being used to a file. Socket clients that run on the agent computer can later read this file to determine which port to connect to. The file that the port is written to is named *kxx_instanceName*_cps.properties, where: *kxx* is the three character product code of the agent and *instanceName* is the agent instance name for a multiple instance agent. If the agent is not a multiple instance agent, this part of the name is not included so the file name is *kxx*_cp.properties.

In Windows, the file is written to the %CANDLE_HOME%\TMAITM6 directory for 32-bit installations or %CANDLE_HOME%\TMAITM6_x64 for 64-bit installations. In UNIX, the file is written to /tmp.

Procedure

1. Optional: Set the environment variable CDP_DP_HOSTNAME to the host name or IP address of your network interface, if your system has multiple interfaces:
   a) Go to the Agent Editor **Agent Information** view and select **Environment Variables**.
   b) Click **Add** and select CDP_DP_HOSTNAME from the list of environment variables by using the Name field.
   c) Set the host name or IP address in the Value field.
2. Start your agent.

   When the agent is started, it binds to the interface that is defined by the CDP_DP_HOSTNAME environment variable. If CDP_DP_HOSTNAME is not set, the agent binds to the default host name.

   If you want the agent to bind to a defined port instead of an ephemeral port, you can set the configuration property **Port Number** (CP_PORT).

   To set the port number configuration property, use the following steps:

   a) Go to the Agent Editor **Runtime Configuration** view.
   b) In the **Runtime Configuration Information** pane select **Configuration for Socket** > **Socket** > **Port Number**
   c) Enter a port number value in **Default value**.

      If you do not enter a value, a value of 0 is used. A value of 0 indicates that an ephemeral port is used.

Remote socket port connection

You can configure your agent to accept data from a remote socket port. The agent must run on a system that has a network interface connection to a remote system.

Procedure

1. Set the value of the environment variable CDP_DP_ALLOW_REMOTE to YES by completing the following steps.
   a) Go to the Agent Editor **Agent Information** page and select **Environment Variables**.
   b) Click **Add** and select CDP_DP_ALLOW_REMOTE from the list of environment variables by using the **Name** field.
   c) Set the **Value** field to YES.
2. Follow the procedure that is detailed in "Socket configuration" on page 161.

   Restriction:

   - The data that is sent between the socket application and the agent:
     - Must conform to the XML syntax defined for a socket data provider. For more information, see " Encoding of socket data" on page 159.
     - Must be encoded in UTF-8.
     - Is sent in clear text (unencrypted). If the data contains sensitive information, the communication must be secured through an SSH tunnel or other mechanism outside the agent.
   - The agent processes data that is received from any remote hosts, so the environment must be secured with appropriate firewall or network traffic filters.

Results

You can run code that implements a socket data provider on any system which can connect to the system where the agent is running.

Sample script for socket

This samples script demonstrates how a socket client might be written.

Perl sample

The following sample Perl script connects to a socket and sends data. This sample was written for an agent that runs on UNIX, with the product code k00 and an attribute group called SocketData.

```
#!/usr/bin/perl -w
# SocketTest.pl
# A simple Agent Builder Socket client using IO:Socket
#------------------------

use strict;
use IO::Socket;

# Initialize socket connection to the agent
#-----------------------
my $host = '127.0.0.1';
my $port = 0;
# This sample is for an agent with the k00 product code.  The product code is
# used in the following line to find the file containing the port number to use.
open PORTFILE, "/tmp/k00_cps.properties" || die "Port file not found $!\n";
while (<PORTFILE>) {
    if (/^CP_PORT=([0-9]+)/) {
        $port = $1;
    }
}

if ($port == 0) {
    die "Could not find port to use to connect to agent.\n";
```

```
}

my $sock = new IO::Socket::INET( PeerAddr => $host, PeerPort => $port,
Proto => 'tcp'); $sock or die "no socket :$!";

# The following call sends 2 rows of data to the agent.  Each row contains 1
# String attribute and 3 numeric attributes.
syswrite $sock, "<socketData><attrGroup name=\"SocketData\"><in><a v=\"A message
from perl\"/>  \<a v=\"1\"/><a v=\"2\"/><a v=\"123\"/></in><in><a v=\"More from
perl\"/><a v=\"456\"/>  \<a v=\"123\"/><a v=\"789\"/></in></attrGroup>
</socketData>\n";

close $sock;
```

Testing socket attribute groups

You can test the socket attribute group that you created, within Agent Builder.

Before you begin

To test the attribute group, you need a socket client to send data. An example socket client that is written with perl script can be seen at "Sample script for socket" on page 162

Restriction: Unlike most other attribute groups, you cannot test the socket attribute group while it is being created. You can test the attribute group when you complete its creation.

Procedure

1. Select an attribute group on the Agent Editor **Data Source Definition** page after agent creation and click **Test**. For more information about the Agent Editor, see Chapter 4, "Using the Agent Editor to modify the agent," on page 17.

   After you click **Test** in one of the previous two steps, the **Test Socket Client** window is displayed.
2. Optional: Set environment variables and configuration properties before you start testing.

   For more information, see "Attribute group testing" on page 229.
3. Click **Start Agent**. A window indicates that the Agent is starting.
4. When the agent starts, it listens for socket data according to its configuration.
5. To test your agent 's data collection, you now generate socket data that matches the agents configuration.

   You can generate socket data by using a socket client.

   When the agent receives socket data that matches its configuration, it adds the data to its internal cache.
6. To simulate a request from Tivoli Enterprise Portal for agent data, click **Collect Data**.

   The **Test Socket Client** window collects and displays any data in the agent's cache since it was last started.
7. Click **Check Results** if something does not seem to be working as expected.

   The **Data Collection Status** window opens and shows you more information about the data. The data that is collected and displayed by the Data collection Status window is described in "Performance Object Status node" on page 280
8. Stop the agent by clicking **Stop Agent**.
9. Click **OK** or **Cancel** to exit the **Test Socket Client** window. Clicking **OK** saves any changes that you made.

Related concepts

"Testing your agent in Agent Builder" on page 229

After you use Agent Builder to create an agent, you can test the agent in Agent Builder.

Use the Java API to monitor data

You can define a data source to use the Java API to interact with a long-running application on the Java platform. The agent starts the application at startup and interacts with it periodically. When you build the agent, Agent Builder creates the source code for the application. You must customize the code to gather the correct data. Depending on the code, the data source can produce multiple data set that can contain a single row, multiple rows, or event data.

About this task

Use the Java API data source and the Java programming language to collect data that cannot be collected by using other Agent Builder data sources. The agent starts the Java application and sends a shutdown request when it is time to shutdown. The Java application must exit only when it is requested to do so.

An agent that contains Java API attribute groups interfaces with the Java application process. The Java application uses the Java Provider Client API to interface with the agent. For information about the API, see the Javadoc on the Tivoli Monitoring Knowledge Center. Using the Java API you can:

- Connect to the agent process and register for attribute groups that are supported by the Java application
- Receive and reply to a request for sampled data
- Send data asynchronously for an attribute group that produces events
- Send an error for an attribute group where data collection is failing
- Support attribute groups in subnodes with configured subnode instances
- Receive and reply to a "Take Action" request

Use the following procedure to create an attribute group which collects data in a Java application and sends it using the Java API. The procedure shows how to create a sample Java application to use as a starting point for your Java application.

Procedure

1. On the **Agent Initial Data Source** page or the **Data Source Location** page, click **Custom programs** in the **Monitoring Data Categories** area.
2. In the **Data Sources** area, click **Java API**.
3. Click **Next**.
4. On the **Java API Information** page, enter an Attribute group name.
5. Enter a help text for the attribute group.
6. Select whether the attribute group **Produces a single data row**, **Can produce more than one data row**, or **Produces events**. This choice affects the sample Java application that is created at the end of the wizard. For more information, see “Sending data” on page 156.
7. Optional: Click **Advanced** to modify the advanced properties for the attribute group. **Advanced** is available when you select that the attribute group **Can produce more than one data row**, or **Produces events**.
8. Click **Next**.
9. On the **Attribute Information** page, specify the first attribute for the attribute group. For more information about creating attributes, see (“Creating attributes” on page 37).
10. Select **Add additional attributes** and click **Next** to add other attributes to the agent. References to the attributes are incorporated into the sample Java application that is created at the end of the wizard.
11. Click **Next**.
12. On the **Global Java API Data Source Information** page, enter a Class name and a JAR file name.

The class name is a fully qualified class name whose main method is called when Java is started. The sample Java application is created with the main Java method in this class.

The JAR file is the archive that contains the Java classes that comprise the Java application. The JAR file is packaged and installed with the agent.

13. Optional: Define the error codes that the Java application can send, on the **Global Java API Data Source Information** page, **Error Codes** section. These error codes are sent by the Java application when it cannot collect data.

    Restriction: An error code has a limit of 256 characters. Only ASCII letters, digits, and underscores are allowed. No spaces are allowed.

    a) Click **Add** in the Error Codes section.

    b) In the **Java API Error Code Definition** window, enter a display value.

    c) Enter an internal value. The internal value must be an integer from 1,000 to 2,147,483,647.

    d) Define a message text for each error. You can use message text that was entered previously by selecting it from the list. Click **OK** to return to the **Global Java API Data Source Information** page.

    The message is logged in the agent log file.

    e) If no suitable message text is available, click **Browse** to set up the message text.

    The Messages (list) window is displayed. The message window lists messages that are defined in the agent. Until you define messages, the list remains blank. You can use **Edit** to alter a defined message and **Remove** to delete one or more messages that you defined.

    f) In the Messages (list) window, click **Add** to see a **Message Definition** window. In the **Message Definition** window, you can type the text that describes the meaning of the new message and select the message type.

    Note: The message identifier is automatically generated for you.

    g) Click **OK**.

    h) The Messages (list) window is displayed with the new message. To verify the message and return to the **Global Java API Data Source Information** page, click **OK**.

14. Optional: In the **Supplemental Files** section of the **Global Java API Data Source Information** page, you can add files that are packaged with the agent and copied to the agent system on agent installation. The Java provider client API JAR file is not listed here; it is automatically copied to the agent system. The **File Type** column describes how each file is expected to be used. Three possible uses are described in the following table (Table 28 on page 165). Click **Edit** to edit the imported file. For more information, see (“Editing a command file definition” on page 122).

Table 28. File types for supplemental files

File type	Description
Executable	Select this option if you want to include an executable file with the agent. The agent does not use this file, but it is in the path for the Java application to use.
Library	Select this option if you to include a library with the agent. The agent does not use this file, but it is in the library path for the Java application to use.
Java resource	Select this option to include Java resources with the agent. The agent does not use this file, but it is in the class path for the Java application to use.

Note: When a Java resource supplemental file is added to the Agent Builder, the file is automatically added to the project class path. The Java compiler uses the supplemental file to resolve any references that your code has, to classes in the resource.

For information about where the Supplemental Files are installed with your agent, see ("New files on your system" on page 246).

15. Optional: Create a filter to limit the data that is returned by this attribute group, if the data is sampled. Create a filter by clicking **Advanced**.

    Note: The data is sampled if you did not select **Produces events** on the **Java API Information** page.

    For more information about filtering data from an attribute group, see "Filtering attribute groups" on page 45.

16. Optional: Add configuration properties to the subnode.

    If you are adding this data source to a subnode, the **Subnode Configuration Overrides** page is shown so you can add configuration properties to the subnode. At least one configuration property is needed under the subnode for the sample Java application to be created. At least one configuration property is needed because the sample uses a configuration property to distinguish one subnode instance from another.

17. Do one of the following steps:

    a) If you are using the **Agent** wizard, click **Next**. Complete the wizard as required.

    b) Otherwise, click **Finish** to save the data source and open the Agent Editor. Then, in the main menu, select **File** > **Save**.

    At this point, Agent Builder creates the source code for the monitoring application. The code is located in the `src` subdirectory of the project directory. Edit this code to create your monitoring application.

What to do next

Select the correct operating systems on the **Java API Settings** page. Make this selection if this attribute group and the Java application, run on operating systems different from the operating systems that are defined for the agent. To open the page, click **Java API Settings** in the outline view or click **Global Settings** in the Agent Editor on any Java API attribute group page.

Note: Error codes and supplemental files can be updated later in the **Error Codes** and **Supplemental Files** sections of the **Java API Settings** page.

Running the Java application

Information about the initialization of the Java application and its dependencies

Initializing the Java application

The agent starts the Java application while the agent is starting and initializing. Configuration settings are used to control which Java run time is used to start the process. Java virtual machine arguments and the Java logging level can also be specified in the configuration. For more information about Java API configuration, see "Java API configuration" on page 175. The Java process inherits the environment variables that are defined for the agent. Runtime configuration settings are also placed in the environment and can be queried by using API calls.

The Java application must be a long-running process. It must not terminate unless it receives a shutdown request from the API. If the Java application does terminate after it is registered with the agent, the agent will attempt to restart the Java application up to three times. If data collection is successfully resumed, this restart count is reset. The agent logs an error when a Java application terminates and when a restart is initiated.

Note: If the Java application terminates before attribute group registration is completed, no restart is attempted.

Dependencies

A Java application must use a Java runtime environment. The following versions of Java are supported:

- Oracle Corporation Java Version 5 or later
- IBM Corporation Java Version 5 or later

Java must already be installed on the agent system when the agent is configured and started. The JAR file that contains the API used to communicate with the agent is included with the agent `runtime` and included in the `classpath` of the JVM. Any additional JAR files that are needed by your Java application must be defined as Supplemental Files to the Java API attribute groups. Any supplemental files that have a *File Type* of *Java resource* are automatically added to the base `classpath` of the Java application, along with the Java API JAR file.

Any JAR files that are necessary for the runtime operation of the Java application that are not included with the agent, must be included in the *Classpath for external jars* configuration setting.

Generated sample Java application

A reference that describes the code the Agent Builder generates and the code you must add or replace for the resources you want to monitor.

When you create an agent with one or more Java API data sources, the Agent Builder generates Java application source code. The code is generated in the agent project and follows the structure of your agent. You must add your own Java code to the generated application. Your code collects data for sampled attribute groups, handles events to be posted to event-based attribute groups, reports errors if problems are encountered, and runs tasks. The generated application supplies the agent with data, but it is sample data, to be replaced with data obtained from the resources you want to monitor.

A sample agent is assumed that has the following characteristics:

- Product code: K91
- Java API Main class: `agent.client.MainClass`
- Agent data source structure as shown in (Figure 17 on page 167):

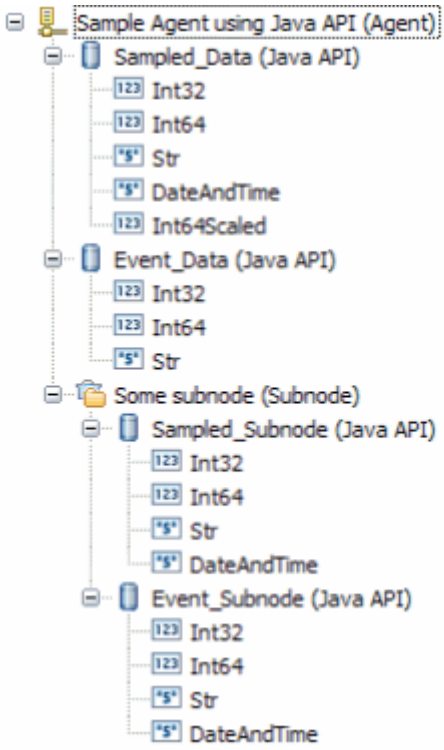


Figure 17. Sample agent structure

- *Some subnode* configuration property: `K91_INSTANCE_KEY`

Class structure

The generated Java application separates, to a great degree, code that interfaces with the agent from code that interfaces with the resources you are monitoring. It contains files that you modify, and files that you do not modify.

The following Java classes are created by the Agent Builder:

`MainClass (agent.client package)`
The class that you specified on the **Global Java API Data Source Information** page. This class contains a main method and a method that handles *take action* requests. This class inherits from the helper class described next. You must modify this class to interface with resources you want to monitor and the actions you want to take.

`MainClassBase (agent.client package)`
A helper class which initializes the connection to the server, registers attribute groups, and waits for requests from the server. Do not modify this class.

`Sampled_Data`, `Sampled_Subnode`, `Event_Data`, and `Event_Subnode` classes `(agent.client.attributeGroups package)`
There is one class for each Java API attribute group which handles data collection requests for the attribute group or generates events for the attribute group. These classes each inherit from one of the helper classes described next. You must modify these classes to gather data from the resources you want to monitor.

`Sampled_DataBase`, `Sampled_SubnodeBase`, `Event_DataBase`, and `Event_SubnodeBase` classes `(agent.client.attributeGroups package)`
Helper classes, one for each Java API attribute group, which define the structure of the attributes of the group in an internal class. Do not modify these classes.

`ICustomAttributeGroup` interface `(agent.client.attributeGroups package)`
An interface that defines public methods in each attribute group class. Do not modify this interface.

The classes which you can modify are never overwritten by the Agent Builder. The Agent Builder creates them only if they do not exist.

The helper classes and the interface are overwritten each time the Agent Builder is saved. As you modify and save the agent, the helper classes are updated to reflect any structural changes to the Java API attribute groups. The interface and helper classes contain a warning in the header that reminds you not to modify the file.

Initialization and cleanup

The main method in `MainClass` is called when the agent is started. It creates a `MainClass` instance and then enters the long-running method to receive and handle agent requests.

Most of the initialization and cleanup code must be added to `MainClass`. In the constructor, add initialization that is needed to create or access your resources. You might want to open connections to remote resources, create handles, or initialize data structures.

Before the agent terminates, the `stopDataCollection` method is called. If you want to close connections or cleanup before the Java application ends, add that code to the `stopDataCollection` method.

If initialization is needed only for a particular attribute group, that initialization can be added to the constructor of the attribute group class. Similarly, if any cleanup is needed only for a particular attribute group, that cleanup code can be added to the `stopDataCollection` method of the attribute group.

Any code in the Java application can use the logger object to write log entries. (The main helper class creates a protected logger object in its constructor. The attribute group helper objects create a protected reference to that logger in their constructors). The logger object uses the Java trace log utility. Errors and detailed trace information can be obtained from the trace log that is created by the logger. The trace information is important for troubleshooting problems with the provider.

When `stopDataCollection` is called, if you pass the cleanup work to another thread, wait for that thread to finish before you return from the `stopDataCollection` method. Otherwise, the cleanup work can be abruptly terminated when the process ends because the main thread completed.

One of the agent configuration settings is for the Java trace level. The following table shows the values that you can set in the `JAVA_TRACE_LEVEL` configuration property. If the API created the logger for you, the table shows the Level that is used by the logger.

Table 29. Java trace level options

Configured trace level	Java logging trace level	Description
Off	OFF	No logging is done.
Error	SEVERE	Trace problems that occurred in the Java application.
Warning	WARNING	Trace errors and potential errors.
Information	INFORMATION	Trace important information about the Java application.
Minimum Debug	FINE	Trace high-level details necessary to analyze the behavior of the Java application.
Medium Debug	FINER	Trace details about the program flow of the Java application.
Maximum Debug	FINEST	Trace all details about the Java application.
All	ALL	Trace all messages.

The name of the log file that is created by the Java application in this example is `k91_trace0.log`. If the agent is a multiple instance agent, the instance name is included in the log file name.

Note: Do not write messages to standard error or to standard out. On Windows systems, these messages are lost. On UNIX and Linux systems, this data is written to a file that does not wrap.

Collecting sampled attribute group data

The class for a sampled attribute group (one that collects one or more data rows) contains a `collectData` method, for example, `Sampled_Data.collectData`. This method is called whenever data is requested by the agent.

The helper class of the attribute group defines an inner class that is called Attributes. This class has one field for each attribute that is defined in your attribute group. Derived attributes are not included since they are calculated by the agent. The data types of attribute fields are Java equivalents to the Tivoli Monitoring attribute types, as shown in (Table 30 on page 169).

Table 30. The data types of attribute fields and their IBM Tivoli Monitoring attribute type equivalents

Tivoli Monitoring type	Data type of attribute field
String	String
Numeric, 32 bit, no decimal adjustment	int
Numeric, 64 bit, no decimal adjustment	long
Numeric, non-zero decimal adjustment	double
Time stamp	Calendar

The `collectData` method must:

1. Collect the appropriate data from the resource that is being monitored.
2. Create an Attributes object.
3. Add the data to the fields of the Attributes object.
4. Call the `Attributes.setAttributeValues` method to copy the data to an internal buffer.
5. Repeat steps 1 - 4 as necessary for each data row. (You can skip steps 1 - 4 altogether and return no rows. In this case, the Error Code column of the Performance Object Status table has a value of NO_INSTANCES_RETURNED. For more information about error codes, see (“Error codes” on page 172).
6. Call `AgentConnection.sendData`to send the data to the agent, or call `sendError` to discard data that is copied from calls to `setAttributeValues`and send an error code instead.

You must collect the data from your resource (Step 1), replacing the sample data that is used in the generated application.

To populate the Attributes object, you can pass the data in using the Attributes constructor (as is done in the generated application). Alternatively use the zero-argument constructor to create an Attributes object and then assign the fields of the attributes object to the attribute values you collected. Fields have the same name as the attributes, though they start with a lowercase letter.

Collecting sampled data for a subnode

If a sampled attribute group is in a subnode, there are presumably multiple resources that you are monitoring (a different one for each subnode). You must determine which resource to collect data from. There must be one or more configuration properties that identify which resource is being monitored.

For this example, it is assumed that one configuration property, `K91_INSTANCE_KEY`, contains a value that identifies the resource from which data must be collected.

Use the following steps to find the correct resource:

1. Get the instance ID of all configured subnodes by calling `AgentConnection.getConfiguredSubnodeInstanceIDs`. Each subnode that is configured has a unique instance ID.
2. For each instance ID, get the `K91_INSTANCE_KEY` configuration property by calling `AgentConnection.getSubnodeConfigurationProperty`.
3. Find the resource that is represented by the value in `K91_INSTANCE_KEY`.

These steps might be done in the `collectData` method before the series of steps that are detailed in (“Collecting sampled attribute group data” on page 169).

Alternatively, you might want to do these steps in the attribute group class constructor and establish a direct mapping from instance ID to resource. An example attribute group class constructor is the `Sampled_Subnode` constructor. This procedure also gives you the opportunity to create handles or open connections that might be used through the life of the agent. Creating handles or open connections can make access to your resources more efficient.

The generated code creates sample resource objects of type `MonitoredEntity` in the constructor, and adds them to a `configurationLookup` map. You must remove the `MonitoredEntity` inner class, and replace the `MonitoredEntity` objects with objects that access your own resources. If you choose to do the entire lookup procedure in the `collectData` method, you can remove the `configurationLookup` map from the class.

If you choose to use the constructor, to map the subnode instance ID to your resource, the steps in the `collectData` method are:

1. Retrieve the instance ID of the subnode from the request parameter, by calling `Request.getSubnodeInstanceID`.
2. Retrieve the resource object from the map that is created in the constructor.
3. Perform the series of steps that are detailed in “Collecting sampled attribute group data” on page 169 to send data to the agent.

An arbitrary subnode property is chosen in the Agent Builder example, in this case K91_INSTANCE_KEY. If not the correct property, or more than one property is needed to identify the correct resource, you must choose the properties to identify the resource.

Sending events

For attribute groups that generate events, there is no periodic call to a collectDatamethod. Events are sent by your application as your resource posts them.

As an example of producing events, the generated code for an event-based attribute group creates and starts a thread which runs from an internal class named SampleEventClass. The event-based attribute group that is used in the example is the Event_Dataclass. The thread periodically wakes up and sends an event. If you want to periodically poll your resource for events, you can use the structure of the Event_Data class as it was generated:

1. From the Event_Data constructor, create and start a thread.
2. In the run method of the thread, loop until the agent terminates.
3. Sleep for a time before you check for events. You might want to change the polling interval of 5000 milliseconds to a number that makes sense for your agent.
4. Determine whether one or more events occurred. The generated application does not check, but always posts a single event.
5. For each event that must be posted, get the event data to be posted.
6. Create and populate the Attributes object (like the collectData method did for a sampled attribute group).
7. Call the Attributes.sendEventData method. Events consist of a single row, so only a single event can be sent at a time.

Alternatively, if you are working with a Java API that reports events from its own thread, you can initialize that thread in the Event_Data constructor. You can also register your own event-handling object with the event-handling mechanism of your resource. In your event handler, use the following steps:

1. Get the event data to be posted.
2. Create and populate the Attributes object.
3. Call the Attributes.sendEventData method.

In this case, you do not have to create your own thread in the Event_Data class nor would you need the SampleEventClass class.

Sending events in a subnode

When an event is detected for a subnode attribute group, the Java application must post the event to the correct subnode.

For this example, it is assumed that one configuration property, K91_INSTANCE_KEY, contains a value that identifies an instance of a resource which can produce events. It is also assumed that the value of the K91_INSTANCE_KEY property is retrieved along with data to be posted in the event. To do retrieve the property and data, the Java application does the following steps:

1. Gets the event data to be posted, along with the “instance key”.
2. Creates and populates the Attributes object.
3. Gets a list of all configured subnode instance IDs by calling AgentConnection.getConfiguredSubnodeInstanceIDs.
4. For each subnode instance, fetches the value of K91_INSTANCE_KEY by calling AgentConnection.getSubnodeConfigurationProperty.
5. When the value of K91_INSTANCE_KEY is found which matches the value that is obtained with the event data, remembers the corresponding subnode instance ID.
6. Calls Attributes.sendSubnodeEventData, passing the remembered subnode instance ID.

The generated application does not do the lookup described in steps 4 and 5, but instead posts an event to the attribute group of every subnode. This behavior is probably not the correct one for a production agent.

Take action commands

Take action commands are defined either in the Tivoli Enterprise Portal or by using the tacmd createaction command. The actions can be imported into the agent's Agent Builder project so that they are created when the agent is installed. For more information about importing take action commands, see (Chapter 17, “Importing application support files,” on page 257).

The generated Java application registers for any actions that begin with the product code of the agent, for example, K91Refresh. This registration is done in the main helper class (MainClassBase) from the registerActionPrefix method. If you want to register other prefixes, or not register for actions at all, override the registerActionPrefix in (MainClassBase).

When the agent wants to run an action which starts with a prefix that your agent registered, the MainClass.takeAction method is called. You add code to call Request.getAction(), do the appropriate action, and then call AgentConnection.sendActionReturnCode to send the return code from your action. A return code of 0 means the action is successful, any other return code means the action failed.

Handling exceptions

The collectData and takeAction methods can throw any Java exception, so you can allow your collection code to throw exceptions without catching them. The handleException method (for collectData) or handleActionExceptionmethod (for takeAction) is called when the helper class gets the exception.

For collectData exceptions, you must call AgentConnection.sendError when an exception occurs or when there is a problem in data collection. The generated application passes an error code of GENERAL_ERROR. However, you must replace this error code with one defined by your agent that best describes the problem that was encountered. For more information about adding error codes, see Step (“13” on page 165).

For takeAction exceptions, you must call AgentConnection.sendActionReturnCode with a non-zero return code.

Some of the AgentConnection methods throw exceptions that are derived from com.ibm.tivoli.monitoring.agentFactory.customProvider.CpciException. The handleException method is not called if a CpciException is thrown during the collecting of data as the helper class handles the exception.

Note: If you choose to catch your exceptions inside the collectData method rather than using the handleException method, ensure any CpciException is rethrown. You ensure CpciException is rethrown so it can be handled by the base class.

Error codes

A typical response to an exception or other resource error is to send an error code to the agent by calling the AgentConnection.sendError method. An error for an event-based attribute group can be sent at any time. An error for a sampled attribute group can be sent only in response to a collect data request, and in place of a sendData call.

If you send an error to the agent, the following happens:

1. An error message is logged in the agent trace log. This error message includes the error code and the message that is defined for that error code.
2. There is a Performance Object Status query that can be viewed to obtain status information about your attribute groups. The Error Code column is set to the Error Code type defined for the error you sent. The error status clears after data is successfully received by the agent for the attribute group. If you

reply to a collect data request with a `sendData` call but you included no data rows, you get `NO_INSTANCES_RETURNED` in the Error Code column.

The following table describes some error codes that are internal to the agent that you can expect to see in certain situations:

Table 31. Internal error codes for the agent

Error Code	Description
`NO_ERROR`	There are no problems with the attribute group currently.
`NO_INSTANCES_RETURNED`	The Java application responded to a data collection request but provided no data. Not providing data is not an error. It generally indicates that there are no instances of the resource that are being monitored by the attribute group.
`OBJECT_NOT_FOUND`	The agent tried to collect data for an attribute group that is not registered through the client API. This error can mean that the application failed to start or did not initiate the attribute group registration when the agent tried to collect data.
`OBJECT_CURRENTLY_UNAVAILABLE`	The application sent the agent an error code that is not defined in the global list of error codes.
`GENERAL_ERROR`	A problem occurred collecting data from the application, usually because the application did not reply to the request within the timeout interval. See the agent trace log for more details. The application can also specify `GENERAL_ERROR` as an error code, but it is better if a more detailed error code is defined.

Changes to the agent

Certain changes to the agent require you to make corresponding changes to the Java application. If the structural changes are complex, you can delete any or all of the Java source files before you save the agent. You can also delete the files if you want to start over without the customizations you made,

The following table describes required modifications to the Java application source files after certain changes are made in the Agent Builder when the agent is saved.

Table 32. Changes to an agent that require modifications to the Java source

Agent change	What Agent Builder does	Manual changes that are needed in the Java source
Change of the main class package name	• Generates all classes in the new package structure. • Removes all helper classes from the old package.	• Port main and attribute group class content from the classes in the old package to the classes in the new package. • Remove the classes from the old package after migration is complete.

Table 32. Changes to an agent that require modifications to the Java source (continued)

Agent change	What Agent Builder does	Manual changes that are needed in the Java source
Change of the main class name	• Creates new main classes. • Removes old main helper class.	• Port main class content to the new class. • Update references to the class name from the attribute group classes.
Addition of a Java API attribute group	• Creates classes for the new attribute group. • Adds registration for the new attribute group in the main helper class.	Overwrite sample code with custom logic in the attribute group class.
Removal of a Java API attribute group	Removes registration from the main helper class.	• Remove the attribute group class or port customized logic to some other class. • Remove the attribute group helper class.
Renaming of a Java API attribute group	• Creates classes for the new name of the attribute group. • Updates registration for the renamed attribute group in the main helper class.	• Port customized logic in the attribute group class with the old name to the attribute group class with the new name. • Remove the attribute group class with the old name. • Remove the attribute group helper class with the old name.
Addition of an attribute to a Java API attribute group	Updates the Attributes inner class in the attribute group helper class.	Collect data for the new attribute in the attribute group class.
Removal of an attribute from a Java API attribute group	Updates the Attributes class in the attribute group helper class.	Remove data collection for the former attribute in the attribute group class.
Renaming of an attribute in a Java API attribute group	Updates the attribute name in the Attributes class in the attribute group helper class.	Update any references to the attribute name in the Attributes class (often there are no references because the Attributes constructor, with positional arguments, is used).
Reordering of attributes in a Java API attribute group	Updates the attribute order in the Attributes class in the attribute group helper class.	Update the argument order in any calls to the Attributes constructor.

Some of the changes that are mentioned in the previous table can be streamlined if you use the Eclipse Refactor - Rename action. Use this action on all the affected names (including helper class names) before you save the changed agent.

Use of the Java API

The Java API is used throughout the generated Java application to communicate with the agent. Often your only direct interaction with the Java API is to modify a parameter of an existing method call. For example, changing a posted error code from GENERAL_ERROR to an error code defined in your agent.

If you want to do more extensive coding with the Java API, you can view Javadoc from the Eclipse text editor. You can view Javadoc while you edit the Java code by doing the following steps:

1. Highlight a package, class, or method name from the API.
2. Press **F1** to open the Eclipse Help view.
3. Select the Javadoc link.

You can also see a brief description from the Javadoc by hovering over a class or method name. Javadoc for the API can also be found on the Tivoli Monitoring Knowledge Center, see Javadoc.

The classes for the Java API are in `cpci.jar`. The `cpci.jar` file is automatically added to the Java Build Path of the project when an agent which contains a Java API attribute group is created. The file is also added when an agent that contains a Java API attribute group is imported. The file is also added when a Java API attribute group is added to an existing agent. The`cpci.jar` is also automatically packaged with each agent that contains a Java API attribute group and added to the CLASSPATH of the Java application.

Java API configuration

When you define a Java API data source in your agent, some configuration properties are created for you.

If you define a Java API data source in your agent, the agent must use Java to connect to the Java API server. Java configuration properties are added to the agent automatically. The following Java configuration properties are specific to the agent runtime configuration:

Table 33. Java configuration properties

Name	Valid values	Required	Description
Java home	Fully qualified path to a directory	No	A fully qualified path that points to the Java installation directory.
Java trace level	Choice	Yes	Use this property to specify the trace level that is used by the Java providers.
JVM arguments	String	No	Use this property to specify an optional list of arguments to the Java virtual machine.
Class path for external jars	String	No	Path containing any JAR files that are not included with the agent, but are necessary for the runtime client operation.

These configuration variables are available on the **Runtime Configuration Information** page of the Agent Editor under **Configuration for Java Virtual Machine (JVM)**, and **Configuration for Java API**.

Testing Java application attribute groups

You can test the Java application attribute group that you created, within Agent Builder.

Before you begin

Restriction: Unlike most other attribute groups, you cannot test the Java application attribute group while it is being created. You can test the attribute group when it is added to the agent and the agent is saved. Saving the agent causes the Java code to be generated for the attribute group.

Procedure

1. Select an attribute group on the **Agent Editor Data Source Definition** page after agent creation and click **Test** .

   For more information about the Agent Editor, see Chapter 4, “Using the Agent Editor to modify the agent,” on page 17

   After you click **Test** in one of the previous two steps, the **Test Java Client** window is displayed.
2. Optional: Set environment variables, configuration properties, and Java information before you start testing. For more information, see “Attribute group testing” on page 229. For more about default Java runtime configuration properties, see “Java API configuration” on page 175.
3. Click **Start Agent**. A window indicates that the Agent is starting.
4. To simulate a request from Tivoli Enterprise Portal or SOAP for agent data, click **Collect Data**.

   The agent monitors the Java Client for data. The **Test Java Client** window displays any data that is returned.
5. Optional: Click **Check Results** if the returned data is not as you expected.

   The **Data Collection Status** window opens and shows you more information about the data. The data that is collected and displayed by the Data collection Status window is described in “Performance Object Status node” on page 280
6. Stop the agent by clicking **Stop Agent**.
7. Click **OK** or **Cancel** to exit the **Test Java Client** window. Clicking **OK** saves any changes that you made.

Related concepts

“Testing your agent in Agent Builder” on page 229
After you use Agent Builder to create an agent, you can test the agent in Agent Builder.

Chapter 7. Creating data sets from existing sources

When at least one data set exists, you can create a new data set using data from an existing data set.

The option to create a new data set is available on the **Agent Initial Data Source** page and on the **Data Source Location** page. You can create a data set by using existing data sources in the following ways:

1. Joining data from two existing data sets (attribute groups). For more information, see "Joining two attribute groups" on page 177.
2. Filtering data from an existing data set (attribute group). For more information, see "Creating a filtered attribute group" on page 182.

Tip: The option to join two data sets is available only after two or more data sets are created.

Joining two attribute groups

Create an attribute group from two other attribute groups.

About this task

Joining attribute groups is most useful when the agent collects data from two different types of data sources. For example, the agent might collect data WMI and PerfMon, or SNMP and script data sources. Each set of attributes might be more useful when used together in one Tivoli Enterprise Portal view.

For example, assume that your attribute groups are defined as follows:

```
First_Attribute_Group
    index integer
    trafficRate    integer
    errorCount    integer
```

```
Second_Attribute_Group
    index2    integer
    name string
    traffic string
```

One definition provides you with counters (like Perfmon) and the other provides you with identification information. Neither attribute group is useful to you by itself. However, if you can combine both attribute groups by using the index to match the appropriate rows from each, you have a more useful attribute group. You can use the combined attribute group to display the name, type, and metrics together.

This same mechanism can be used to add tags to information collected through normal attribute groups. The information can then be more easily correlated in an event system when a problem is detected. For example, a company wants to manage all its servers by collecting common data and by using common situations to monitor the health of the servers. It also wants to be able to identify the servers with more information that tells it what application is running on a particular server. It wants to have control of the values that are used on each server, but it does not want to create different agents for each application. It can accomplish this control by creating an additional attribute group in its single agent as follows:

```
Application_Information
    application_type     integer
    application_name    string
    application_group    string
```

This attribute group would be defined as a script attribute group that gathers its values from agent configuration. You can specify different values for each agent instance and use one agent to manage all of their systems. This attribute group would then be joined to all the source attribute groups where this application information might be needed. The information is then available in the Tivoli Enterprise Portal, situations, events, and warehoused data.

© Copyright IBM Corp. 2010, 2021

When you join two attribute groups, a third attribute group is created. This attribute group contains all the attributes that are contained within the source attribute groups.

The results of a join operation vary depending on the number of rows that each source attribute group supports. If both attribute groups are defined to return only a single row of data, then the resulting joined attribute group has one row of data. The single row contains all attributes from both source attribute groups.

Table 34. source attribute group one (single row)

Attribute1	**Attribute2**	**Attribute3**
16	some text	35

Table 35. source attribute group 2 (single row)

Attribute4	**Attribute5**	**Attribute6**	**Attribute7**
5001	more data	56	35

Table 36. Resulting join

Attribute1	**Attribute2**	**Attribute3**	**Attribute4**	**Attribute5**	**Attribute6**	**Attribute7**
16	some text	35	5001	more data	56	35

Suppose that one source attribute group is defined to return only one row (single-row) while the other can return more than one row (multi-row). The resulting joined attribute group contains the same number of rows as the multi-row source attribute group. The data from the single-row attribute group is added to each row of the multi-row attribute group.

Table 37. source attribute group one (single row)

Attribute1	**Attribute2**	**Attribute3**
16	some text	35

Table 38. source attribute group two (more than one row)

Attribute4	**Attribute5**	**Attribute6**	**Attribute7**
user1	path1	56	35
user2	path2	27	54
user3	path3	44	32

Table 39. Resulting join

Attribute1	**Attribute2**	**Attribute3**	**Attribute4**	**Attribute5**	**Attribute6**	**Attribute7**
16	some text	35	user1	path1	56	35
16	some text	35	user2	path2	27	54
16	some text	35	user3	path3	44	32

Finally, assume that both source attribute groups are defined to return more than one row. You must identify an attribute from each of the source attribute groups on which to join. The resulting attribute group contains data rows where the attribute value in the first attribute group matches the attribute value from the second attribute group.

Table 40. source attribute group one (more than 1 row)		
Attribute1	**Attribute2**	**Attribute3**
16	some text	35
27	more text	54
39	another string	66

Table 41. source attribute group 2 (more than 1 row)			
Attribute4	**Attribute5**	**Attribute6**	**Attribute7**
user1	path1	56	35
user2	path2	27	54
user3	path3	44	32

Table 42. Resulting join (joining on Attribute3 and Attribute7)						
Attribute1	**Attribute2**	**Attribute3**	**Attribute4**	**Attribute5**	**Attribute6**	**Attribute7**
16	some text	35	user1	path1	56	35
27	more text	54	user2	path2	27	54

With Agent Builder, you can also join user-defined attribute groups to the Availability attribute group if there are any availability filters defined in your agent. For more information about the data that is contained in the Availability attribute group, see (“Availability node” on page 275).

You can create this type of attribute group by accessing the menu on the data sources tree by right-clicking and then selecting **Join Attribute Groups**.

Procedure

1. On the **Data Source Definition** page, right-click one of the attribute groups you would like to join and select **Join Attribute Groups**.

   This option is only visible if there are at least two attribute groups defined. Having an availability filter defined counts as having an attribute group defined.

   The **Attribute Group Information** page is displayed.

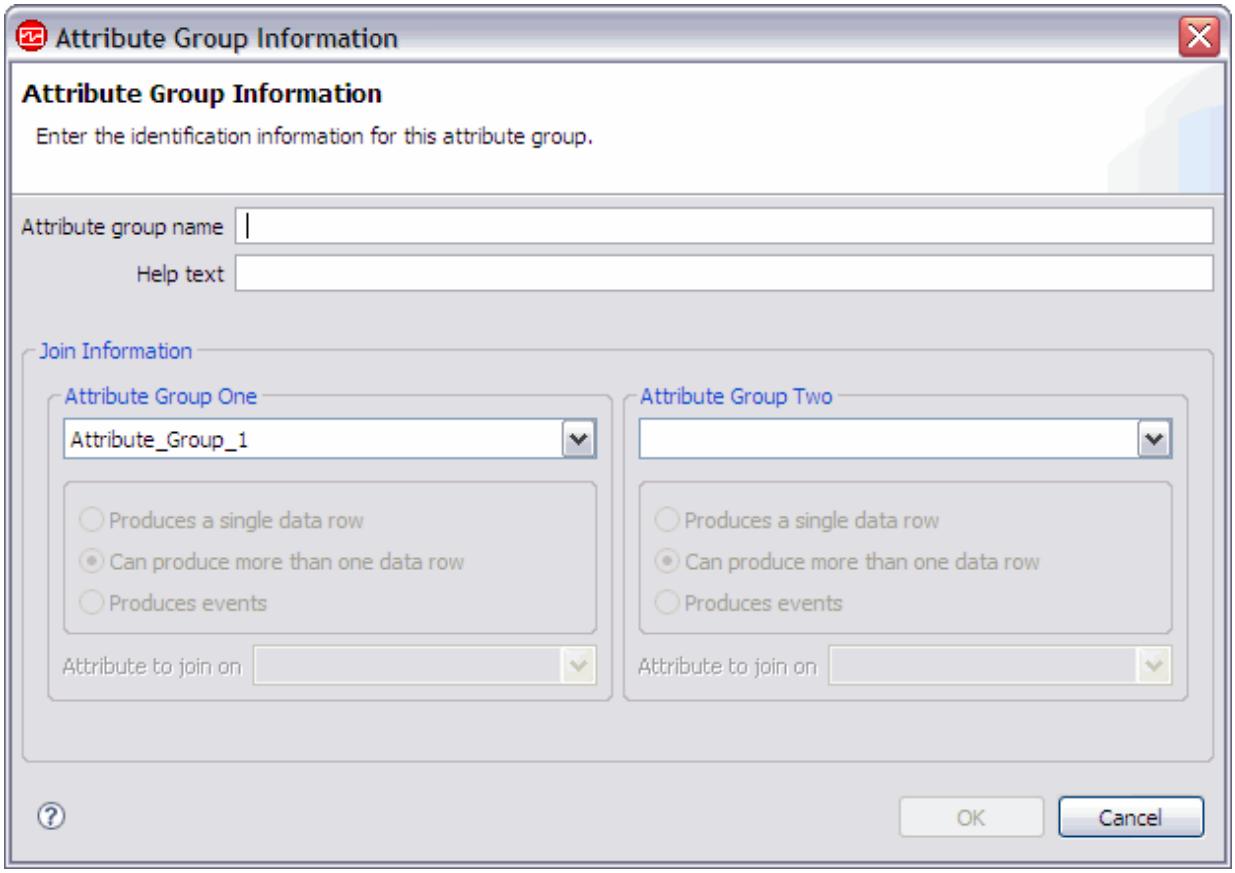


*Figure 18. **Attribute Group Information** pageAttribute Group Information window*

2. In the **Join Information** area, select the two attribute groups you would like to join. Select the attribute groups by choosing from the groups available in the **Attribute Group One** and **Attribute Group Two** lists.

   For each attribute group, either **Produces a single data row** or **Can produce more than one data row** is selected for you. This selection is locked and depends on how the source attribute groups were originally defined.

   Note: There are restrictions on which attribute groups can be joined:

   - You cannot join an attribute group in one subnode type to an attribute group in another subnode type.
   - You can join only an event attribute group to a single row non-event attribute group.

   a) Select the attribute that you want to join on for each attribute group when both attribute groups show **Can produce more than one data row**, under **Attribute to join on**.

   The **Attribute group name** and **Help** fields are filled for you using information from the chosen attribute groups. If you want to, you can change these entries.
3. Click **OK**.

Results

The joined attribute group that you created is added to the **Attribute Group Information** area of the **Data Source Definition** page

Manipulating attributes in joined attribute groups

Using attributes in joined attribute groups can impose rules on how those attributes are manipulated.

Deleting an attribute group

An attribute group cannot be deleted if it is referenced in a joined attribute group unless the joined attribute group is also being deleted.

Deleting an attribute

An attribute cannot be deleted if its parent attribute group is referenced in a joined attribute group and one of these statements is true:

- The attribute is defined as a join attribute in the joined attribute group.
- The attribute is used in any derived attribute in the joined attribute group.

Joined attributes cannot be deleted. Only derived attributes, if any are added, can be deleted from the joined attribute group.

Reordering attributes

The order of the joined attributes is fixed by the order of the source attributes. The joined attribute list cannot be reordered. Only derived attributes, if any, can be reordered.

When the version of an agent, is committed, source, and derived attributes cannot be reordered or removed. Attributes added in a new version of the agent, whether source or derived attributes, will come after all committed attributes. For more information, see “Committing a version of the agent” on page 33.

Adding an attribute

New joined attributes cannot be explicitly added. Only derived attributes can be explicitly created.

Removing availability filters

The last availability filter cannot be removed if the Availability attribute group is referenced in a joined attribute group.

Joined attributes

Manipulate information that relates to joined attributes

Procedure

- Change the attribute name and help text of the joined attribute can be changed so that they are different from the source attribute:
  a) Select the attribute in the joined attribute group in the **Attribute Group Information** pane of the **Data Source Definition** page.
  b) Enter the new name and help text.
- The joined attribute can be shown or not shown on the Tivoli Enterprise Portal by selecting or clearing the **Display attribute in the Tivoli Enterprise Portal** check box. The check box is in the **Joined Attribute Information** section of the **Data Source Definition** page. This choice is irrespective of whether the source attribute is shown on the Tivoli(r) Enterprise Portal.
- Any attribute or combination of attributes (that are shown on the Tivoli Enterprise Portal) can be marked as key attributes by selecting the **Key attribute** check box. This choice is independent of whether the attributes are key attributes in the source attribute groups. The choice is also independent of whether the source attributes are shown on the Tivoli(r) Enterprise Portal.

- Attribute type information for joined attributes is taken from the source attributes and cannot be changed in the joined attribute. In the **Joined Attribute Group Information** section of the agent editor (Figure 19 on page 182), click **Locate source attribute** to go to the source attribute.

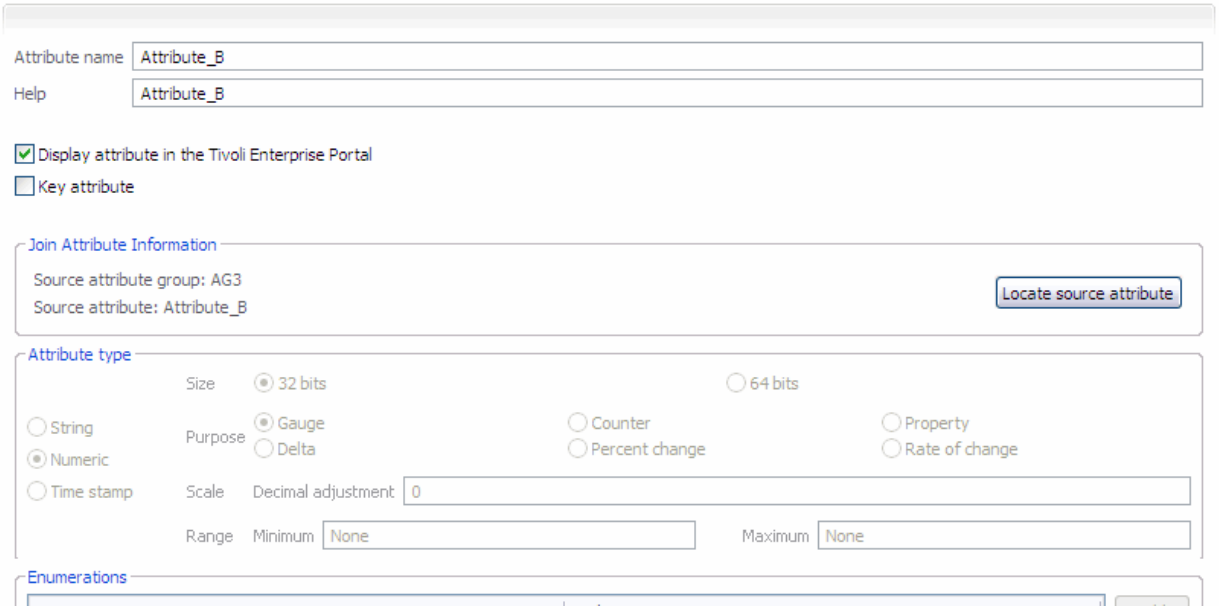


Figure 19. Locating source attribute information

Any changes to the source attribute groups are reflected in the joined attributes. If the source attribute groups change, those attributes are automatically updated under the joined attribute group. This automatic update also occurs if a different attribute group is set as the source attribute group. Changes to a source attribute type are copied to the joined attribute. Changes to a source attribute name or help text are copied to the joined attribute. However, such source attribute changes are not copied after you change the name or help text of a joined attribute.

Creating a filtered attribute group

Create a filtered attribute group (data set) by filtering rows of data from an existing attribute group. If an existing data set returns multiple rows, you can create a filtered group returning one row for use with IBM Cloud Application Performance Management.

About this task

A filtered attribute group has the same columns as the source attribute group, but can exclude some of the rows. It uses a selection formula to determine which rows to include.

To provide status and summary information for Cloud APM, you need to use a data set that returns a single row. For details, see Chapter 12, “Preparing the agent for Cloud APM,” on page 219. If the source information is in a data set that returns multiple rows, you can create a filtered attribute group that returns a single row.

For example, the process, Windows service, and command return code data sources provide information as rows in the single Availability data set. You can create a filtered attribute group, using the NAME field in the selection formula. The group includes status for the necessary application. Define it as returning one row. Then you can use this attribute group as the summary data set for Cloud APM.

A filtered attribute group is also useful when a base data source query returns data that you prefer to divide into separate groups. Examples of such data sources are Windows Performance Monitor, SNMP and WMI.

For example, assume that a data source can return the following data:

```
Name    Type  Size  Used  Free
Memory  MEM      8     4     4
Disk1   DISK   300   200   100
Disk2   DISK   500   100   400
```

This is a table that reports on the storage that exists on the system and it includes both memory and disk space. You might prefer to break down the table into memory and disk as separate tables. You can break down the table by creating two base attribute groups. Each of these base attribute groups collects the same data and filters out the rows you do not want. However, that is not the most efficient way to do things. Instead, you define one base attribute group that returns both the memory and disk usage data together. Next, define two filtered attribute groups. Each uses the same base table as its source. One includes a filter where `Type=="MEM"` and the other includes a filter where `Type=="DISK"`.

In the example, for the filtered attribute group where `Type=="MEM"`, the returned data is:

```
Name    Type  Size   Used  Free
Memory  MEM      8      4     4
```

and where `Type=="DISK"`, the returned data is:

```
Name    Type  Size  Used  Free
Disk1   DISK  300   200   100
Disk2   DISK  500   100   400
```

Note: Attributes groups whose data is event-based cannot be used to create filtered attribute groups. Only attribute groups whose data is sampled can be used.

Procedure

1. Click **Existing data sources** in the **Monitoring Data Categories** area on the **Agent Initial Data Source** page or the **Data Source Location** page

   Note:

   - You reach the **Agent Initial Data Source** page by using the new agent wizard. For more information, see Chapter 3, “Creating an agent,” on page 13.
   - You can reach the **Data Source Location** page by right-clicking an agent in the **Data Source Definition** page of the **Agent Editor** and selecting **Add Data Source**.
2. Select **Filter an attribute group's data rows** in the **Data Sources** area.
3. Click **Next**

   The **Filter Information** page is displayed.
4. Select a **Source attribute group** from the list.
5. Enter a **Selection formula** to filter the data from the attribute group you selected.
   For example, in the **Filter Information** page that is shown earlier, the selection formula filters data rows where the Type attribute is equal to `"DISK"`. Data rows whose Type attribute does not match `"DISK"` are discarded. The selection formula that you enter must evaluate to a Boolean result, true, or false.

   Note: In the **Filter Information** page, you can click **Edit** to enter or modify the formula by using the Formula Editor. For more information about the Formula Editor, see “Formula Editor” on page 45.
6. Click **Next**.
7. Select **Produces a single data row** or **Can produce more than one data row**.
   a) If you selected **Can produce more than one data row**, select a key attribute or attributes from the list.
8. Click **Finish**.

Chapter 8. Creating a navigator group

In an IBM Tivoli Monitoring environment, use Navigator groups to group several related data sources (attribute groups) together so that workspaces can be created that show views that combine the data sources. You can create a navigator group while you create an agent by using the New Agent wizard at the base agent level. You can also create a navigator group while you define a subnode by using the New Agent Component wizard.

About this task

For example, you might be able to collect file system data from more than one data source. It can be useful to create one workspace that shows views of all file system data from those different data sources.

Navigator groups are also a good way to hide data sources on the Tivoli Enterprise Portal. You might decide that metrics collected from two data sources are most useful if the data sources are joined to create a new combined data source. You want to see only the combined data in the Joined data source. You can create a navigator group that contains all three data sources and create a workspace that contains views to display only the combined data source. The two original data sources are effectively hidden from view in the Tivoli Enterprise Portal. See Chapter 7, “Creating data sets from existing sources,” on page 177 for information about joining data sources.

Note: When you group data sources in a navigator group, Tivoli Monitoring does not associate a query with the navigator group. It is assumed that you define a default workspace for the navigator group to display the data sources in a useful format.

A navigator group can be defined in the base agent or in a subnode. A navigator group cannot contain another Navigator group.

Navigator groups have no effect in an IBM Cloud Application Performance Management environment.

Procedure

1. Take one of the following steps:
   - When creating a new agent using the **Agent** wizard, on the **Agent Initial Data Source** page, click **Data source groupings** in the **Monitoring Data Categories** area.
   - With an existing agent, take the following steps in the Agent Editor:
     a. Click the **Data Sources** tab to open the **Data Source Definition** page.
     b. Select the agent and click **Add to selected**.
     c. On the **Data Source Location** page, in the **Monitoring Data Categories** area, click **Data source groupings**.
2. In the **Data Sources** area, click **A navigator group**.
3. Click **Next**.
4. On the **Navigator Group Information** page, type the navigator group name and the text for the Help you want associated with the name, and click **Next**.

   Note: Agent Builder automatically creates navigator groups in certain situations. The following navigator group name is reserved:

   - Availability

   .
5. On the **First Navigator Group Data Source** page, select the first source of monitoring data for the new navigator group. Click a category in the **Monitoring Data Categories** list and a data source in the **Data Sources** list. Then, click **Next**.

© Copyright IBM Corp. 2010, 2021

Tip: You can create the data source as usual. Alternatively, click **Existing data sources** and choose to move one or more data sources that you already created into the navigator group.

6. If you want to create a data source within a navigator group, on the **Data Source Definition** page, select the navigator group, and click **Add to Selected**.
7. If you want to move existing data sources into the navigator group, on the **Data Source Definition** page, select the navigator group, and click **Add to Selected** and on the **Navigator Group Data Source** page, select **Existing data sources**. In the **Currently Defined Data Sources** page, select the data sources.
8. If you want to remove a data source from a navigator group, do one of the following steps on the **Data Source Definition** page:
   - Select the data source, and drag it to the root of the data sources tree.
   - Select the data source, and clicking **Remove**.
9. If you want to create a navigator group, do one of the following steps on the **Data Source Definition** page:
   - Click **Add to Agent**.
   - Select a subnode and click **Add to Selected**.

Chapter 9. Using subnodes

You can use subnodes to monitor multiple application components from a single agent instance.

You can build a single agent that accomplishes the following tasks by using subnodes:

- Monitors each instance of a software server that is running on a system instead of having to use separate instances of the agent, one per software server instance.
- Monitors several different remote systems instead of having to use separate instances of the agent, one for each remote system.
- Monitors several different types of resources from one agent instead of having to build and deploy several different agents.
- In IBM Tivoli Monitoring, displays an additional level in the Tivoli Enterprise Portal physical Navigation tree that allows further grouping and customization. Moreover, you can define Managed System Groups for another level of granularity with situations.
- In IBM Cloud Application Performance Management, provides several different resources, displaying different summary and detail dashboards. Subnode resources can be displayed as peers or subcomponents of the agent resource. You can include these resources in applications independently.

You can create subnode types in Agent Builder. Each type must correspond to a different type of resource that an agent can monitor. Add data sources and data sets to the subnode type for a particular monitored resource.

When you deploy the agent on a monitored host and configure it, you can create one or more instances of each subnode type. Each instance of a subnode must correspond to an instance of a server, a remote system, or whatever resource the subnode type was designed to monitor. All subnode instances of a single subnode type have attribute groups and workspaces that have an identical form. However, each subnode instance has data that comes from the particular resource that is being monitored.

When you configure the agent on the monitored host, you can determine the number of subnode instances. Some configuration data can apply to the agent as a whole, but other configuration data applies to a single subnode instance. Configure each subnode instance differently from the other subnode instances so that they do not monitor the exact same resource and display the exact same data.

In an IBM Tivoli Monitoring environment, a subnode instance is displayed within the agent in the Navigation Physical view in the Tivoli Enterprise Portal. Workspaces display the data that is produced by a subnode instance and situations can be distributed to one or more instances of a subnode. A managed system list is automatically created that contains all instances of the subnode, just like the Managed System List that is created for an agent.

In an IBM Cloud Application Performance Management environment, you can display both agent and subnode instances as monitored resources. Each subnode instance becomes a separate resource. For details, see “Subnodes in IBM Cloud Application Performance Management” on page 192.

Because agents built with Agent Builder create the subnode instances that are based on configuration values, these subnodes have the same life span as the agent. There is still just one heartbeat that is done for the agent, not a separate heartbeat for each subnode. Therefore, by using subnodes you can significantly increase the potential scale of the monitoring environment. The alternative is to use multiple agent instances, which can limit the potential scale of the IBM Tivoli Monitoring or IBM Cloud Application Performance Management environment.

Adding or removing a subnode requires reconfiguring the agent. To reconfigure the agent, you need to stop and restart it, involving all subnodes. You can define the agent as a multi-instance agent; in this case, you can start and stop a single instance, and leave the other instances running.

Along with data sets in subnodes, an agent can define agent-level data sets that are located outside of a subnode.

© Copyright IBM Corp. 2010, 2021

In the Tivoli Enterprise Portal Navigator tree, a subnode type is displayed under the agent name, and subnode instances are displayed under a subnode type. Subnodes are identified by a Managed System Name (MSN) just like agents, for example `94:Hill.cmn`.

For example, in the Navigator tree in Figure 20 on page 188, **Watching Over Our Friends** is an agent with three resources (**Boarders**, **Common Areas**, and **Kennel Runs**) and two subnode types (**Common Area** and **Kennel Run**). Two of these resources have subnode types that are defined for them (**Common Area** and **Kennel Run**). A subnode is not required for the third resource (**Boarder**), which is represented by a single row in a table at the base agent level. The Common Area subnode type has three subnode instances: `94:Hill:cmn`, `94:Meadow:cmn`, and `94:Tree:cmn` representing three common areas in the kennel. The Kennel Run subnode type has four subnode instances: `94:system1:run`, `94:system2:run`, `94:system4:run`, and `94:system5:run` representing four kennel runs.

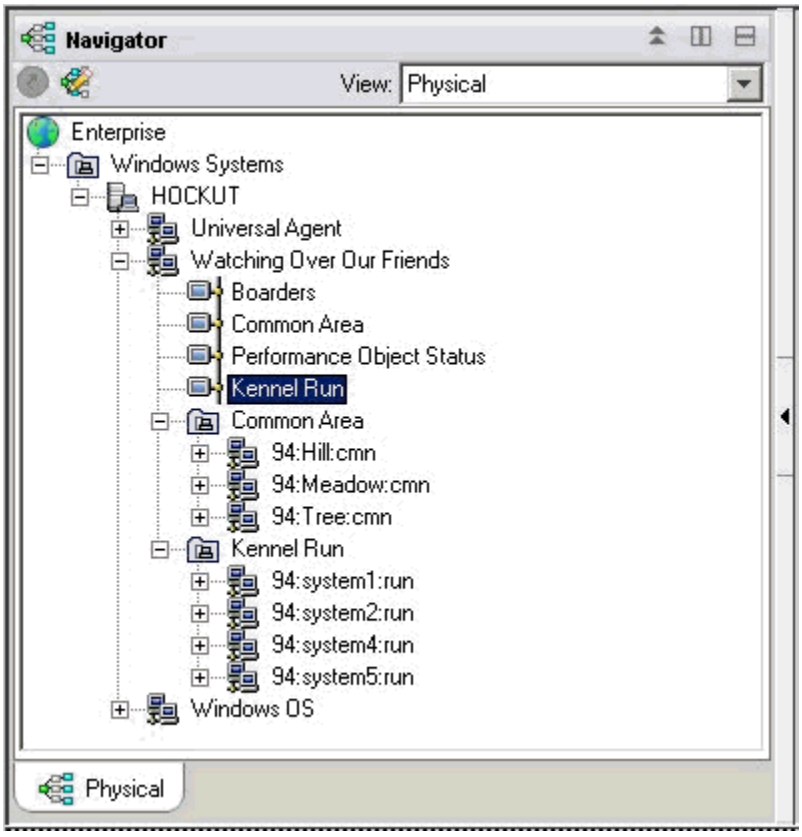


Figure 20. Subnodes in the Navigator tree

There are two ways that a single agent can use subnodes:

- The agent can have different subnodes of the same type.
- The agent can have subnodes of different types.

Subnodes for the same data from different sources

You can use subnodes of the same type to represent multiple instances of a monitored resource type. Each subnode of the same type includes the same attribute groups and the correct values for the specific monitored resource instance. The number of subnodes varies based on agent configuration. The example in Figure 21 on page 189 shows the monitoring of different systems.

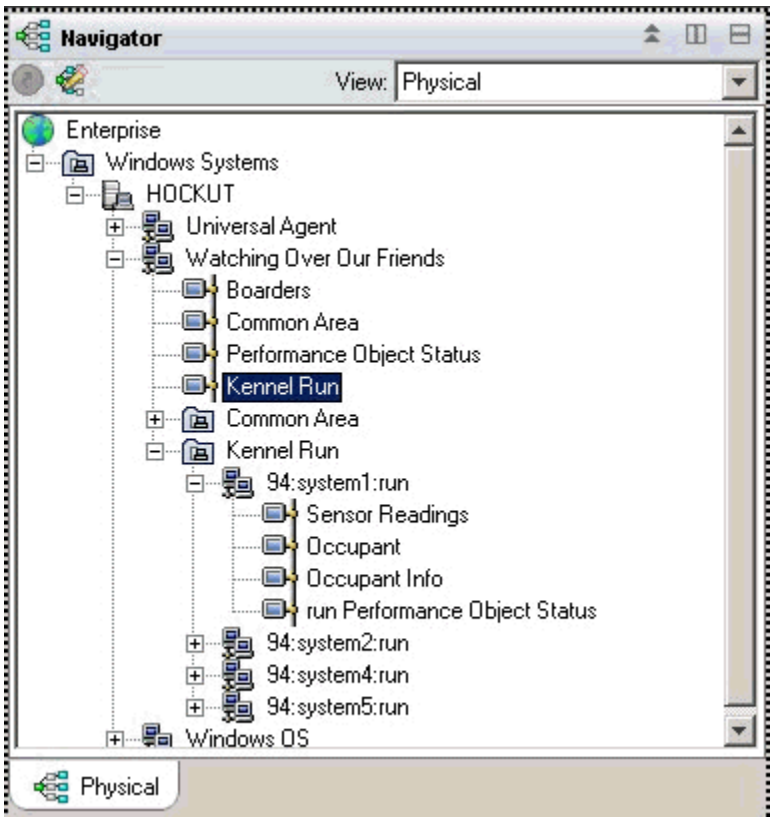


Figure 21. Subnodes monitoring different systems

Subnodes for multiple types of data

When one agent monitors multiple types of monitored resources, you can create a subnode type for each of the resource types. Each subnode includes the information that is defined in that subnode type. The following example shows two subnode types. Each type is monitoring a different type of resource, with different types of data available for each resource:

- Common Area
- Kennel Run

The agent in Figure 22 on page 190 runs one copy of each subnode type. A particular agent might create any subset of the defined agents. Subnodes can be used to mimic Tivoli Monitoring V5 profiles.

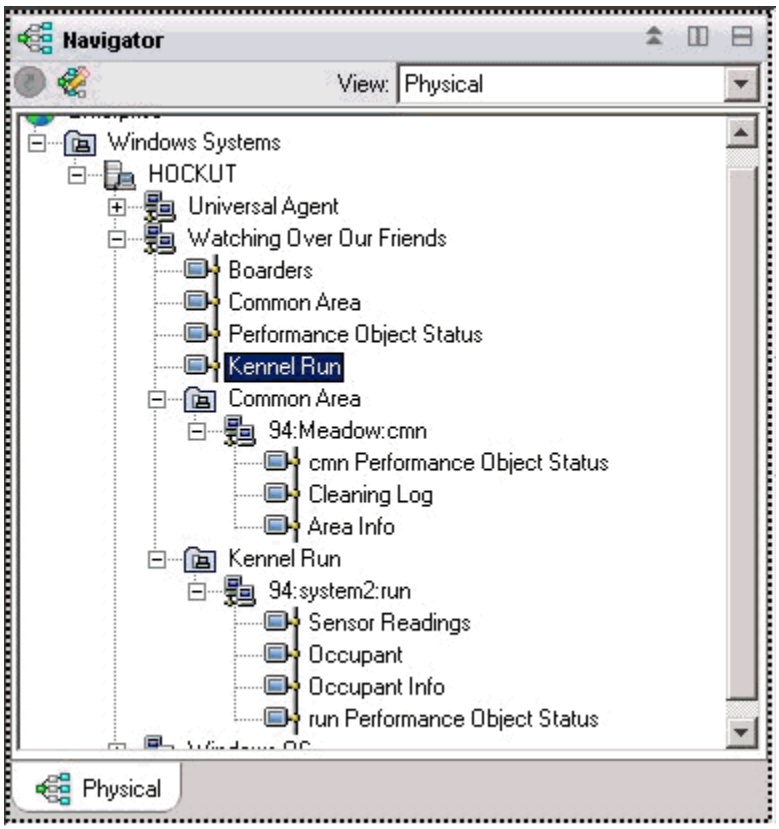


Figure 22. Subnode types in Navigator tree

Both ways of using subnodes can be used in the same agent, where each type can have more than one subnode instance.

Figure 22 on page 190shows two types of subnodes that monitor two types of resources: Common Areas and Kennel Runs. In addition, there are several subnodes that are defined for each type. There are three subnodes of type Common Area; these subnodes have the following IDs: Meadow, Hill, and Tree. There are also four subnodes of type Kennel (each collecting data from a different system that is dedicated to a Kennel Run); these subnodes have the following IDs: system1, system2, system4, and system5.

Note: The first 24 characters of subnode IDs must be unique for all instances of the subnode type in the IBM Tivoli Monitoring installation.

Data Providers in subnodes

A subnode can contain any mixture of data from the different data provider types. Most current Agent Builder data providers can be used in a subnode, including the following data providers:

- WMI
- Perfmon
- Windows Event Log
- SNMP
- SNMP Events
- JMX
- ICMP ping
- Script
- Log
- CIM

- JDBC
- HTTP
- SOAP
- Socket
- Java API

A subnode can also contain a joined attribute group that combines data from two other attribute groups from the same subnode or from agent-level attribute groups.

Status of subnodes

There are two ways to determine status for a subnode agent. The first way is to look at the data that is displayed in the Performance Object Status attribute group. This attribute group displays the status for each of the other attribute groups at the same level in the agent. The Performance Object Status attribute group at the agent level displays the collection status for the other attribute groups at the agent level. The Performance Object Status attribute group in each subnode displays the collection status for the attribute groups in that subnode.

Agent Builder also creates one attribute group for each subnode type, which displays one row for each configured subnode of that type. In the example in (Figure 23 on page 191), four subnodes are running to collect data.

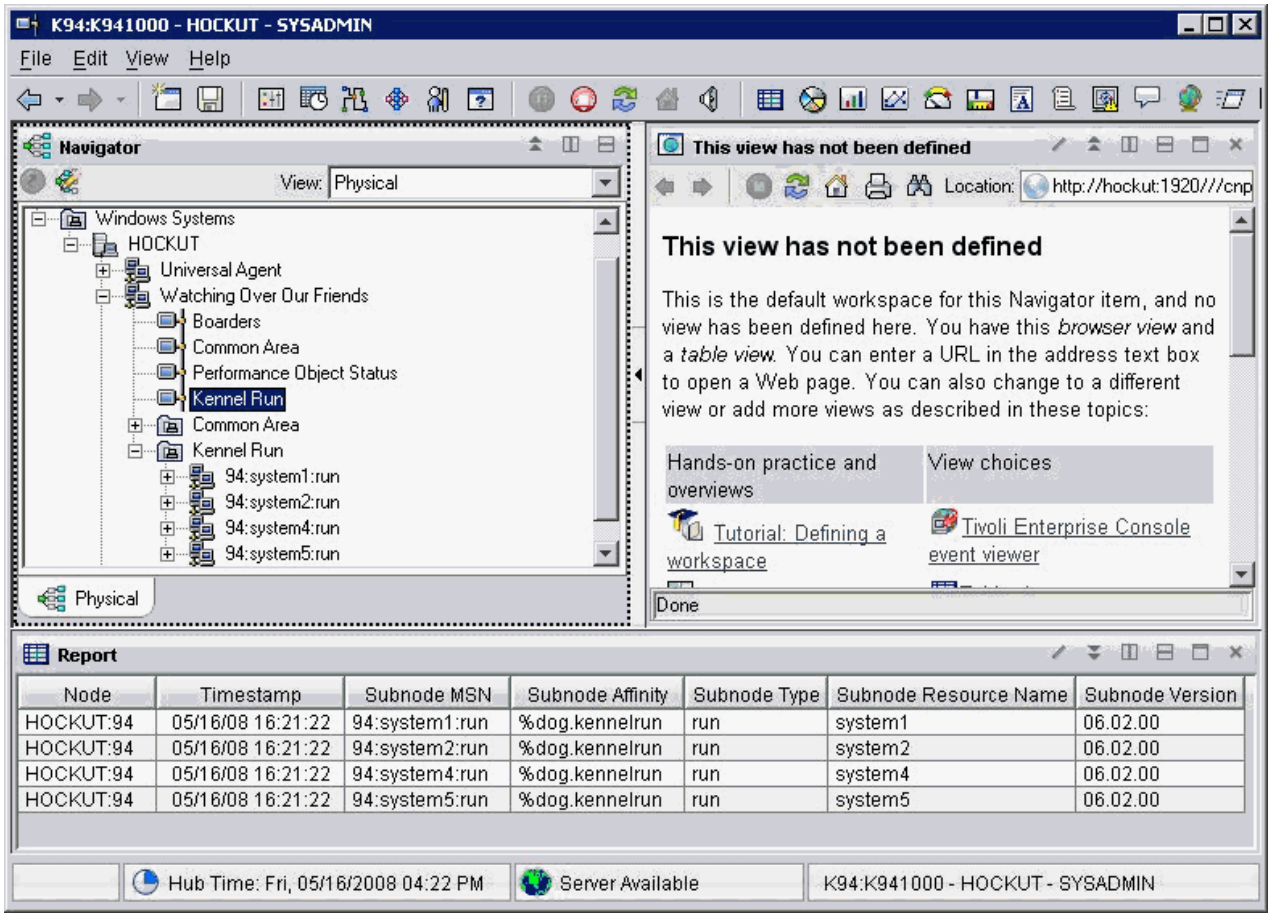


Figure 23. Monitoring multiple subnode instances of the same subnode type

In the IBM Tivoli Monitoring environment, the **Performance Object Status** subnode contains data visible in the Navigator tree and can have situations that monitor the status of the other data collections.

In the IBM Cloud Application Performance Management environment, you can create thresholds to monitor **Performance Object Status** data.

The example in Figure 24 on page 192 shows a case where the data collection failed (the script shell command was not found). Typically, any value other than NO_ERROR indicates that there is a problem. For each of the data collectors that are defined in the subnode, there is one row in the table.

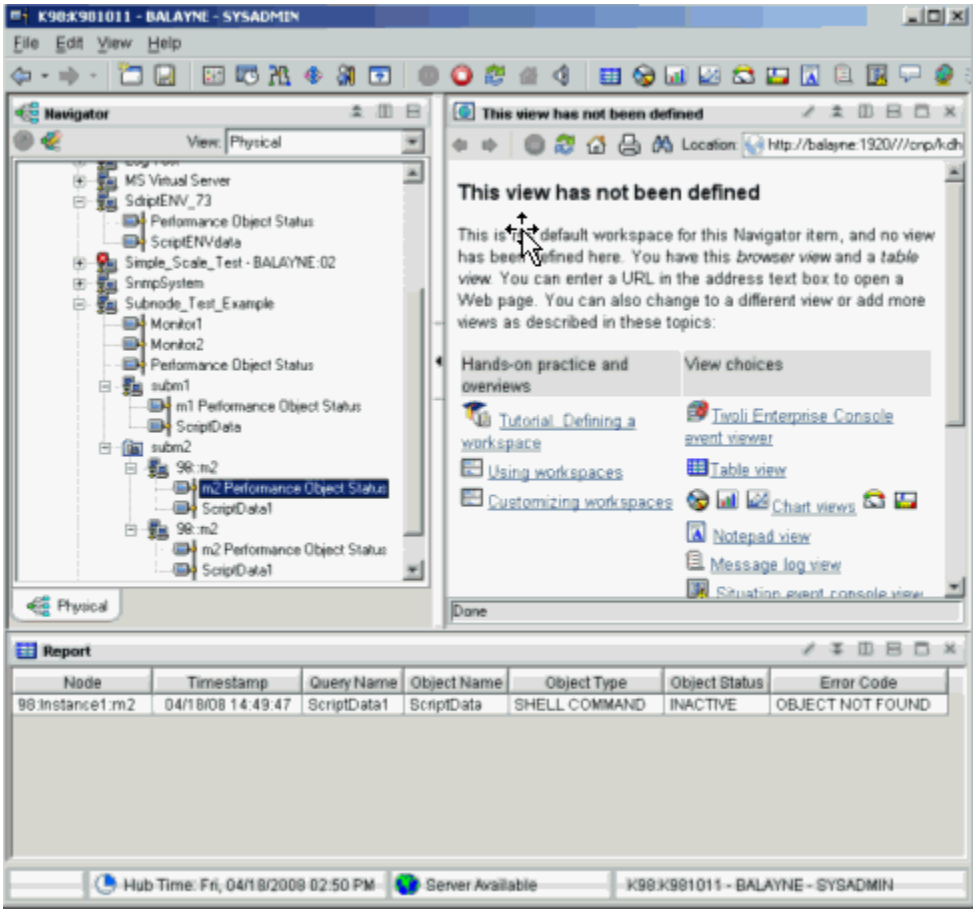


Figure 24. Example: data collection in a subnode

Subnodes in IBM Cloud Application Performance Management

In IBM Cloud Application Performance Management, you can define either the agent instance or a subnode instance or both as monitored resources, and each resource corresponds to a summary dashboard.

Subnode dashboards can not display agent-level data. To display agent-level data in this environment, define a summary dashboard for the agent.

Depending on the settings you select, agent and subnode resources can appear at the same level, with no hierarchical distinction, or subnode resources can be listed as children to agent resources.

For instuctions about configuring agent and subnode resources, see Chapter 12, “Preparing the agent for Cloud APM,” on page 219.

Creating subnodes

You can create a subnode when creating or editing an agent.

Procedure

1. Take one of the following steps:

- When creating a new agent using the **Agent** wizard, on the **Agent Initial Data Source** page, click **Data source groupings** in the **Monitoring Data Categories** area.
- With an existing agent, take the following steps in the Agent Editor:
  a. Click the **Data Sources** tab to open the **Data Source Definition** page.
  b. Select the agent and click **Add to selected**.
  c. On the **Data Source Location** page, in the **Monitoring Data Categories** area, click **Data source groupings**.

2. In the **Data Sources** area, click **A Subnode Definition**
3. Click **Next**.
4. Complete the **Subnode Information** page as follows to define the new subnode:
   a) In the **Name** field, type the name of the subnode you are creating.
   b) In the **Type** field, enter 1-3 characters (by using numbers, letters, or both) to identify the type of the subnode you are creating.
   c) In the **Description** field, type a description for the subnode you are creating.
   d) Click the **Show nodes attribute group for this type of subnode** check box to hide or display the availability attribute group. For more details about this attribute group, see "Availability node" on page 275.
   e) Click **Next**.
5. Complete the **Initial Subnode Data Source** page to select a data source as the first item in the new subnode. Click a category in the **Monitoring Data Categories** list and a data source in the **Data Sources** list. Then, click **Next**.

   Tip: You can create the data source as usual. Alternatively, you can move one or more data sources that you already created into the navigator group. To move data sources, click **Existing data sources** and, in the **Currently Defined Data Sources** page, select the data sources.

   Important: You can not include process, Windows service, or command return code data sources in a subnode. As a workaround, you can write a script that determines the necessary process or service information and use a script output data source.
6. If your agent contains custom configuration properties or if the selected data source requires configuration, use the **Subnode Configuration Overrides** page to choose the configuration properties.

   In the **Subnode Configuration Overrides** page, choose the configuration properties that you want for the subnode at the agent level. Then, choose the configuration properties that you want to vary for each subnode.

   Use **Move**, **Copy**, and **Remove** to specify the configuration properties as described in "Configuring a subnode" on page 194.
7. Click **Next**.

   The **Data Source Definition** page is displayed.

Subnode configuration

When a subnode type is defined, a single configuration section is defined specifically for that subnode.

There are several ways that a subnode configuration section differs from other configuration sections:

- The set of properties in a subnode section can be duplicated, so there are multiple sets of properties. Each set of properties forms its own section. The layout of all sections is identical, but different values can be entered in each section.

  In contrast, the properties in other sections (which are referred to as agent-level sections) are shown only one time during runtime configuration. They do not form subsections and cannot be duplicated or removed.

  See "Subnode configuration example" on page 197 for GUI and command-line examples of configuring subnodes.

- For each copy of a subnode section that is created at runtime configuration, the agent creates a separate subnode instance. All of those subnode instances are of the same type.
- The property names in subnode sections can be duplicates of property names in agent-level sections. When duplicate names occur, the subnode property value overrides the agent-level property value.
- In IBM Tivoli Monitoring V6.2.1 and later, a subnode section can have default property values that apply to all instances of subnodes of that type. This feature makes it possible to have a three-level lookup of a single property value as follows:
  1. The agent obtains the property value from the subnode instance subsection.
  2. If no value is configured at the subnode instance level, the property value is obtained from the subnode default level.
  3. If no value is configured at either of those two levels, then the property value is obtained from an agent-level section.

See "Subnode configuration example" on page 197 for GUI and command-line examples of configuring subnodes.

Configuring a subnode

Use the **Subnode Configuration Overrides** page to configure a subnode data source.

Before you begin

Use the steps in "Creating subnodes" on page 192 to create a subnode.

About this task

When you add a data source to a subnode, the **Subnode Configuration Overrides** page is presented if the data source requires configuration. It shows custom configuration properties and any other configuration properties that are applicable to the subnode type.

Procedure

- In the **Subnode Configuration Overrides** window, choose the configuration properties that you want for the subnode at the agent level. Also, choose the configuration properties that you want to vary for each subnode.
- Use **Copy >>** to copy configuration properties so that they are both at the agent level and the subnode level.

  The agent looks for a value first at the subnode level, and if it does not find a value, it looks at the agent level. If a property at both levels is a required property, it is required only at the agent level, it is optional at the subnode level.
- Use **Move >>** to move properties from the agent level to the subnode level. **Move>>** is not available for properties that are required by an agent-level data source or by a subnode of a different type.
- Use **Remove** to remove one of the two lists. Properties can be removed only if they are listed at both the agent-level and the subnode level. This function cannot be used to remove a property completely.
- Use **<< Copy** to copy a property from the subnode level to the agent-level.
- Use **<< Move** to move a property from the subnode to the agent-level.

What to do next

You can change the configuration for an existing subnode by using the Agent Editor.

Subnode configuration overrides

Use Subnode Configuration Overrides to override agent configuration properties with subnode-specific properties.

The procedure in "Configuring a subnode" on page 194 describes how to manage subnode configuration for automatically generated properties. Managing custom configuration properties is similar. Any custom configuration properties that are defined are displayed in the **Subnode Configuration Overrides** window.

When you copy or move a custom property from the subnode level to the agent-level, you are prompted for the section to place the property in. You can select an existing custom section, or enter the name of a new custom section.

Selecting subnode configuration properties

Without subnodes, all instances of a data source type share the configuration parameters. For example, all SNMP attribute groups connect to the same host by using the same community name. With subnodes, each instance of a subnode can connect to a different host if the SNMP_HOST property is placed at the subnode level.

Selecting properties to be overridden at the subnode level is an important consideration when you are developing an agent. If too many properties are selected, the subnode configuration section becomes cluttered and difficult to manage. If too few properties are selected, then the agent functions might be limited when someone wants to vary a property from one subnode to the next.

The following properties cannot be copied to the subnode level. (All attribute groups in all subnodes and in the base agent must use the same SNMP version and JMX connection type):

- SNMP version
- JMX MBean server connection type
- Java home
- Java trace level
- JVM arguments
- Class path for external JAR files
- Socket data source port number
- JMX or JDBC class path settings

Advanced subnode configuration

Use advanced subnode configuration to override an agent configuration property in a subnode.

About this task

There is an option in IBM Tivoli Monitoring V6.2.1 and later agents that you can enable to override properties from any agent-level configuration section in a subnode instance. On the **Subnode Configuration Overrides** page, there is a check box labeled **Allow any configuration property to be overridden in any subnode**. For more information, see ("Subnode configuration overrides" on page 195). For this option to be enabled, you must select **6.2.1** as the **Minimum ITM version** when you name your agent ("Naming and configuring the agent" on page 13). If you choose this option, each subnode instance can override any property from any agent-level configuration section. But this property can be overridden only from the GUI and not from the **itmcmd** command line.

Procedure

The **Allow any configuration property to be overridden in any subnode** option causes an **Advanced** field that contains a list to be displayed on each subnode configuration panel. The initial selection in the **Advanced** field provides the brief directions: **Select a section to override values**.

- When you click the list, you see a list of all the non-subnode sections that contain configuration properties.
- Select a section.

  The properties from that section are temporarily added to the subnode panel. The value of any property that you change is added to the set of properties that are defined for the subnode. A data source in the subnode looks for property values in the subnode before it looks in the agent-level sections. .

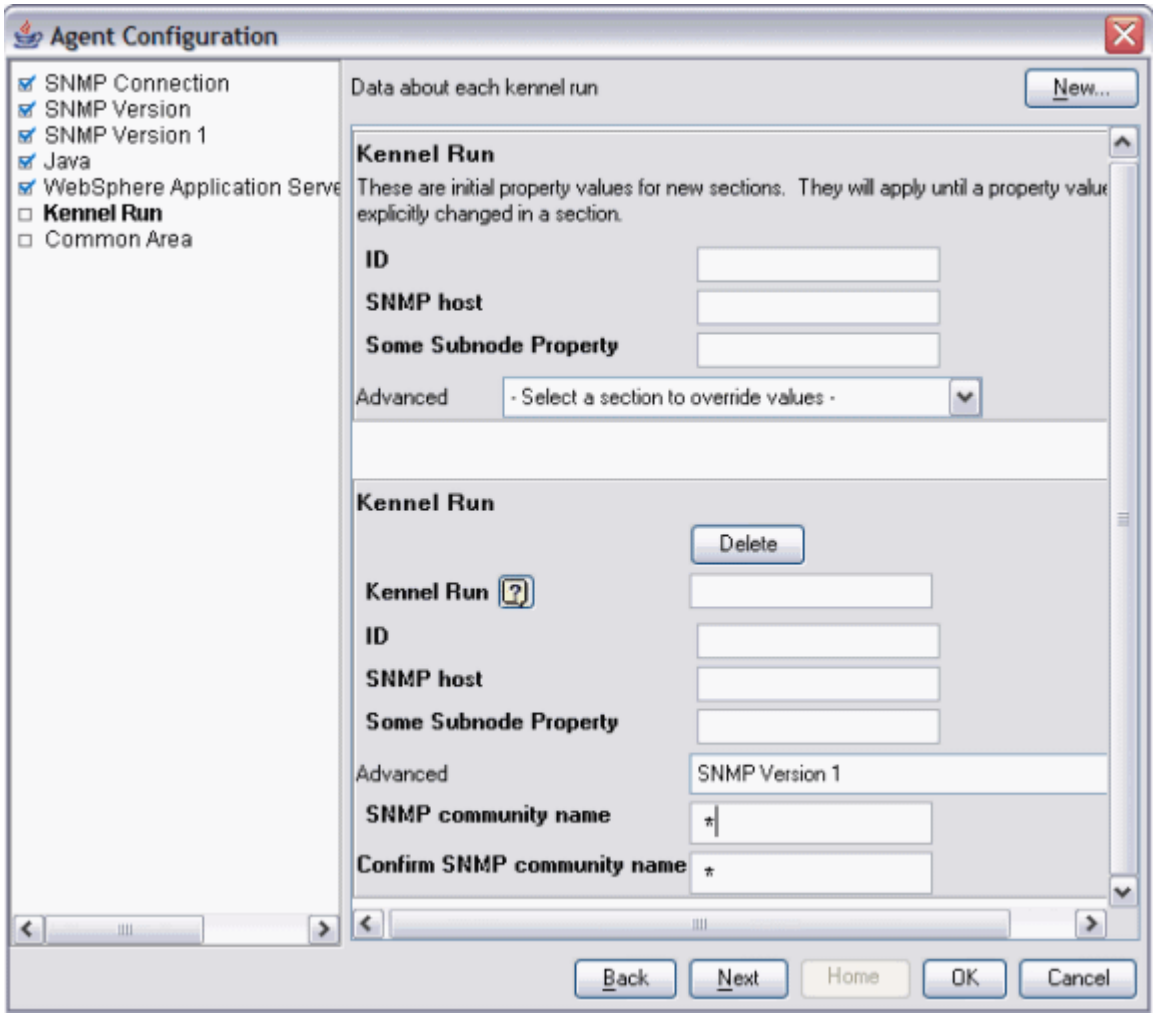


Figure 25. SNMP Version 1 Properties expanded

The following further information applies to overriding properties from agent-level sections:

- Properties that were copied to the subnode section are not shown when the agent-level section is selected in the **Advanced** list. For example, in Figure 25 on page 196, **SNMP host** is not displayed after the **Advanced** list because it was copied to the subnode properties and is already displayed.
- Sections that contain no properties to override do not have a selection in the **Advanced** list.
- Overridden values that you enter for one section are retained even if you select a different section to display different properties.
- Select **Allow any configuration property to be overridden in any subnode** to enable this feature in your agent.

Configuring a subnode from the command line

In the IBM Tivoli Monitoring environment, you can also configure a subnode by using the command line.

Before you begin

For more information about subnode configuration, see “Subnode configuration” on page 193

About this task

Procedure

- To configure a subnode instance from the command line, use the following command:

```
tacmd configureSystem -m HOSTNAME:00 -p
section_name:subnode_instance_id.property_name=value
```

Where:

section_name
Same as the subnode type

subnode_instance_id
ID for the subnode that is defined during configuration.

property_name
Name of the configuration property

value
Value for the property

Subnode configuration example

How to configure a sample agent with one defined subnode.

Example:

This example shows how to configure a sample agent that has one subnode named Example Subnode of type `exs` and the following three configuration properties:

- `Agent Cfg` (actual property name is `K00_AGENT_CFG`) is defined only at the agent level.
- `Subnode Cfg` (actual property name is `K00_SUBNODE_CFG`) is defined only in the example subnode.
- `Overridable Cfg` (actual property name is `K00_OVERRIDABLE_CFG`) is defined at the agent level and was copied to the example subnode.

(Figure 26 on page 198) shows these configuration properties on the **Runtime Configuration Information** page of the Agent Editor.

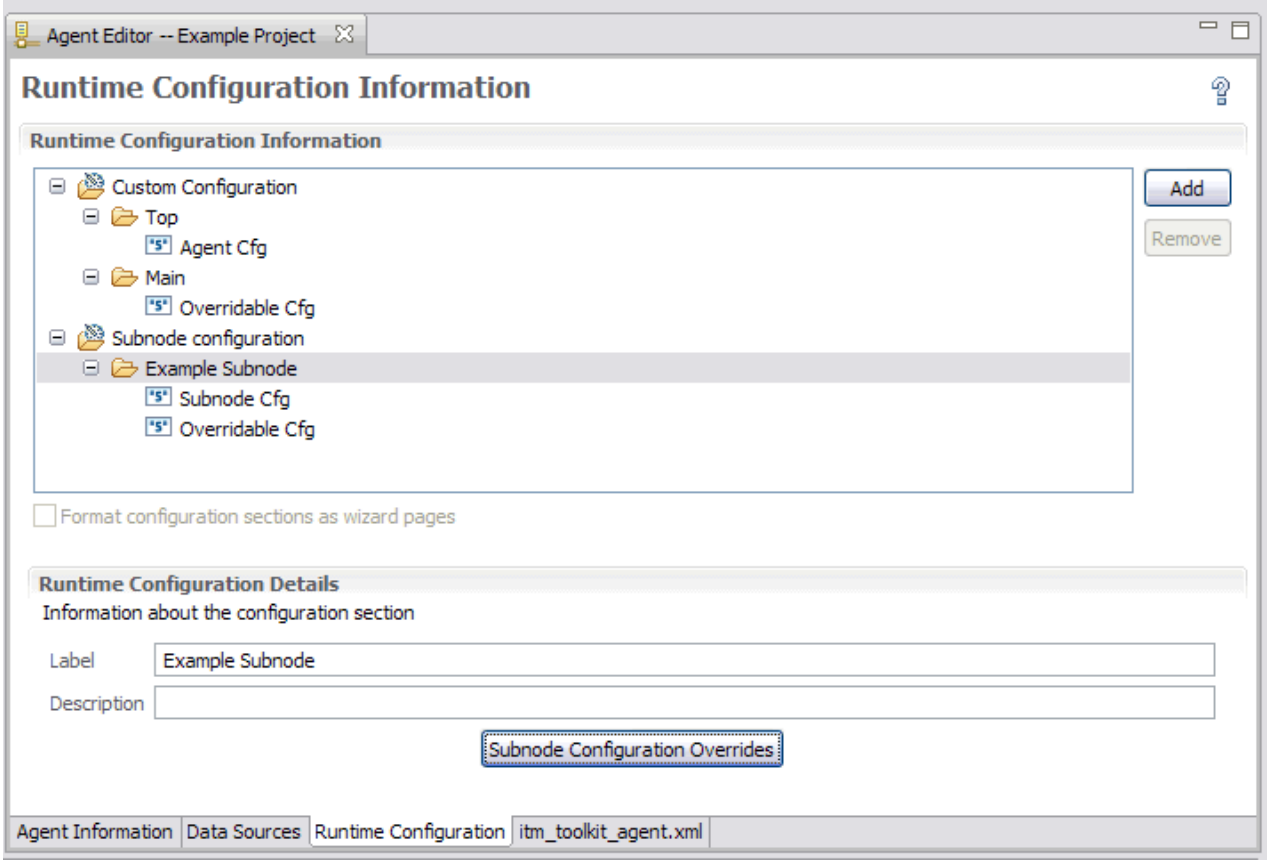


Figure 26. Configuration property definitions in the Agent Builder

When this example agent is configured, the first page that is displayed is the **Top** section, which contains the **Agent Cfg** property as shown in (Figure 27 on page 199). Because this property is an agent-level property, it is shown one time during agent configuration. Any instance of the Example Subnode can see this property value, but all instances see the same value.

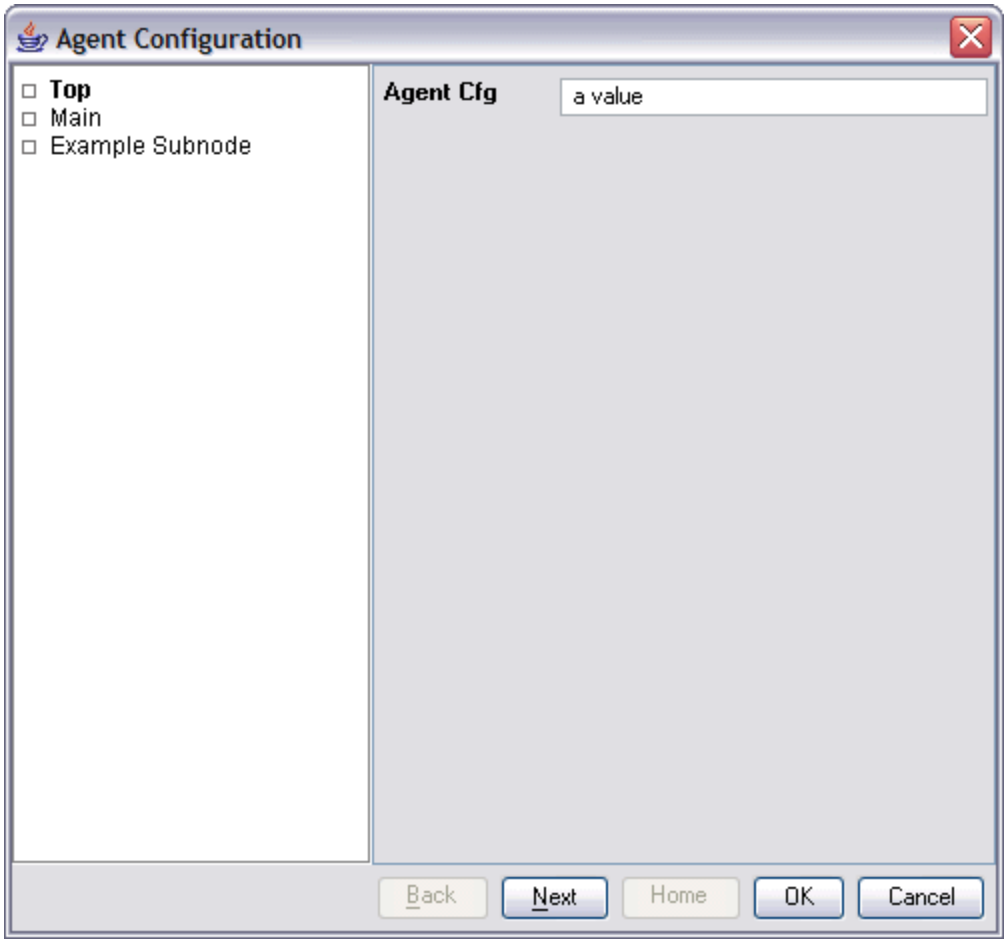


*Figure 27. **Top** section with agent-level configuration for the **Agent Cfg** property*

If you are configuring from the Tivoli Enterprise Monitoring Server command line, the **Agent Cfg** property can be set by using the following command:

```
tacmd configureSystem -m HOSTNAME:00 -p "TOP.K00_AGENT_CFG=a value"
```

The next section that is displayed is the **Main** section as shown in Figure 28 on page 200. It is also an agent-level section and contains the agent-level **Overridable Cfg** property. This property differs from the **Agent Cfg** property because this property was copied to the Example Subnode in the Agent Builder. This means that a default value for the property can be entered on the **Main** page. However, any Example Subnode instance can override the value that is entered here with a different value.

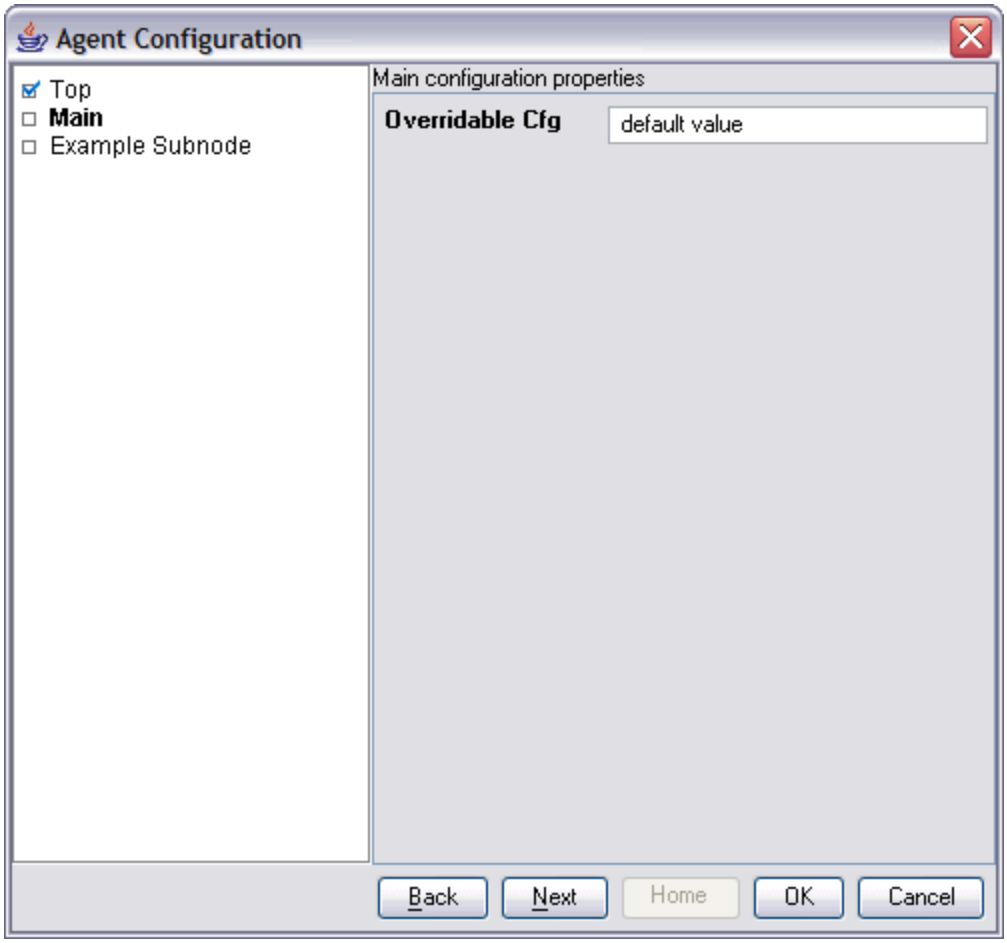


*Figure 28. **Main** section with the agent-wide default value for the **Overridable Cfg** property*

If you are configuring from the Tivoli Enterprise Monitoring Server command line, this property can be set by using the following command:

```
tacmd configureSystem -m HOSTNAME:00 -p "MAIN.K00_OVERRIDABLE_CFG=default value"
```

You can place both of these properties in the same agent-level section. You can decide how many custom agent-level sections to create and how to distribute custom properties among them.

The next section that is displayed is the **Example Subnode** section as shown in Figure 29 on page 201. Because this agent is being configured for the first time, there are no defined subnode instances and no subnode instance subsections are shown. The initial property values subsection is shown, although it is optional and some subnode types might not show it. Because the initial property values subsection is shown, default values can be entered for any of the configuration properties. The **Overridable Cfg** property already has a default value that was obtained from the agent-level property of the same name.

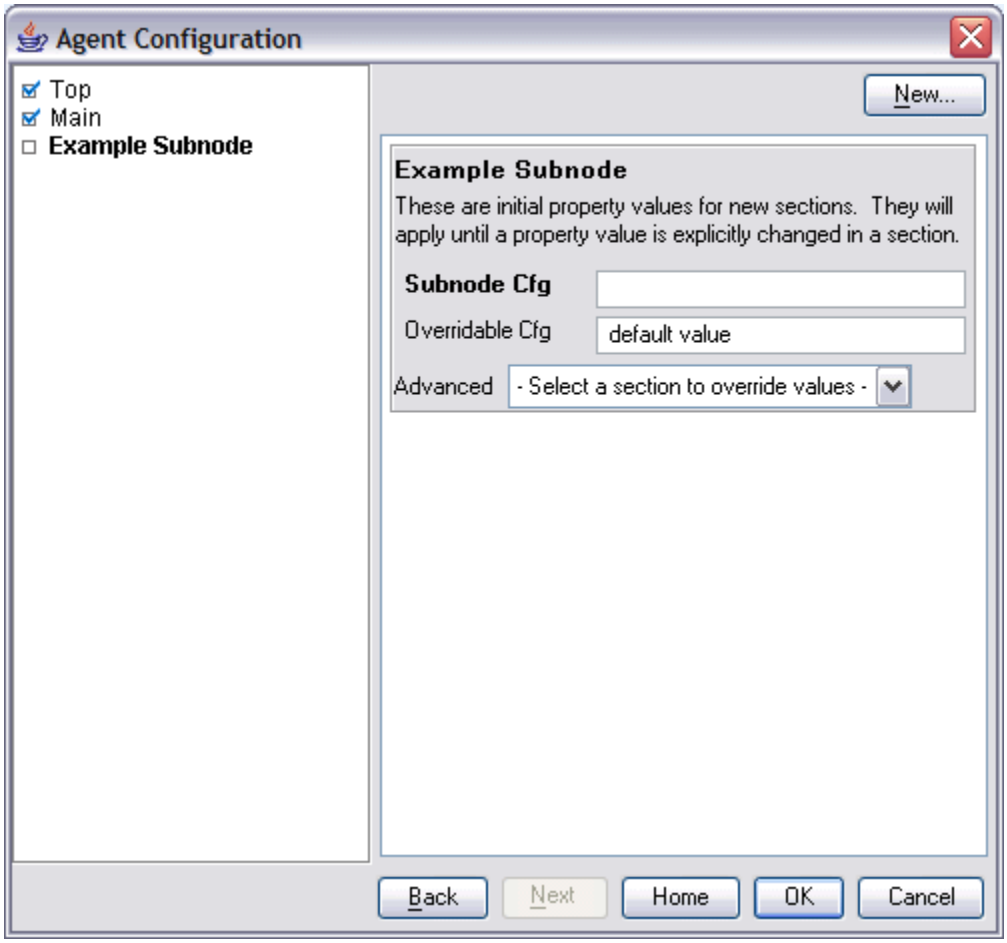


*Figure 29. **Example Subnode** section page with no subnode*

Subnode instances are defined by doing the following actions on the empty **Example Subnode** section page (Figure 30 on page 202):

1. In the initial **Example Subnode** section, in the **Subnode Cfg** field, type the following default string for the property: `sub-default value`.
2. Click **New**. An **Example Subnode** subsection is displayed after the initial properties subsection.
3. In the **Example Subnode** field, type the following subnode instance ID: do.
4. Click **New**. A second **Example Subnode** subsection is shown after the first.
5. In the second **Example Subnode** field, type the following subnode instance ID: `re`.
6. In the **Subnode Cfg** field, type the following value for the **Subnode Cfg** property: `sc override`.
7. In the **Overridable Cfg** field, type the following value for the **Overridable Cfg** property: `oc override`.

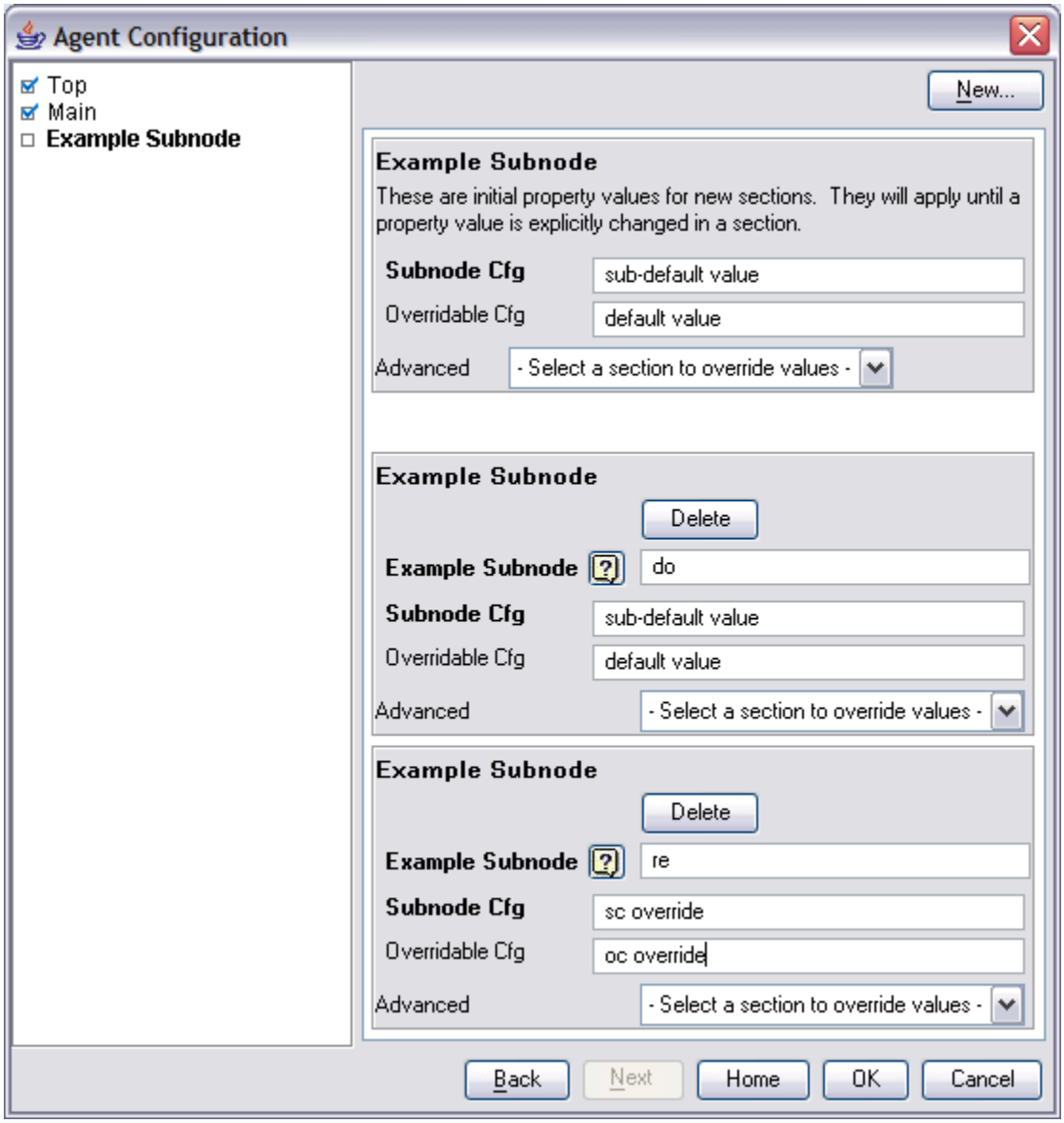


*Figure 30. **Example Subnode** section page with two subnode instances defined*

The two new subsections cause the agent to create two subnode instances when it is started. Because the properties of the **do** subnode subsection were not changed, the default property values are used by that subnode instance. Since different values were entered for the properties in the **re** subsection, the **re** subnode instance uses those values that were typed.

You can set a default value from the Tivoli Enterprise Monitoring Server command line with the following command:

```
tacmd configureSystem -m HOSTNAME:00 -p "exs.K00_SUBNODE_CFG=sub-default value"
```

The format for setting subnode default values is exactly like the format for setting agent-level properties, except that the section name identifies a subnode section.

You can create the subnode instances from the Tivoli Enterprise Monitoring Server command line with the following command:

```
tacmd configureSystem -m HOSTNAME:00 -p "exs:do.K00_OVERRIDABLE_CFG=default value" \
 "exs:re.K00_SUBNODE_CFG=sc override" "exs:re.K00_OVERRIDABLE_CFG=oc override"
```

The subnode instance ID is inserted between the section name and property name. When you use the command line to create a subnode instance, at least one property must be specified, even if all the

properties use default values. Otherwise, default values are not required to be specified on the command line when you define subnode instances.

All of the agent configuration properties can be set in a single command. The following command is equivalent to all of the preceding individual commands:

```
tacmd configureSystem -m HOSTNAME:00 -p "TOP.K00_AGENT_CFG=a value" \
 "MAIN.K00_OVERRIDABLE_CFG=default value" \
 "exs.K00_SUBNODE_CFG=sub-default value" \
 "exs:do.K00_OVERRIDABLE_CFG=default value" \
 "exs:re.K00_SUBNODE_CFG=sc override" "exs:re.K00_OVERRIDABLE_CFG=oc override"
```

Subnodes and Windows data sources

Choose to include Windows Remote Connection properties in the agent or not.

About this task

If an agent has Windows data sources at the agent level and not in subnodes, including Windows Remote Connection configuration properties in the agent are optional. Windows data sources are Windows Event Log, Windows Management Instrumentation, Windows Performance Monitor. If configuration properties are not included, these data sources monitor the local Windows system by default and need no configuration. By default, no Windows data sources are included in any subnode.

To choose whether to include Windows Remote Connection properties in the agent, do the following steps:

Procedure

1. On the **Windows Management Instrumentation (WMI) Information** page, click **Global Options** when data source properties are displayed. Select **Global Options** either while you are creating the data source or from the Agent Editor **Data Sources** page.
2. In the **Global Windows Data Source Options** window, select **Include Windows Remote Connection configuration** if you want to include these properties in the agent.

Subnodes and Script data sources

Subnode instance configuration properties are accessed in subnode scripts just as they are in agent-level scripts.

Scripts have access to all agent-level configuration properties and all subnode instance configuration properties. If an agent-level property is overridden at the subnode level, the script has access only to the subnode level property value.

Chapter 10. Customizing agent configuration

Customize the configuration of process, log file, and script data sources.

Before you begin

If you are adding SNMP, JMX, CIM, JDBC, HTTP, and SOAP data sources to your agent, configure these data sources as described in the following sections:

- “Monitoring data from a Simple Network Management Protocol (SNMP) server” on page 73
- “Monitoring Java Management Extensions (JMX) MBeans” on page 83
- “Monitoring data from a Common Information Model (CIM)” on page 102
- “Monitoring data from Java Database Connectivity (JDBC)” on page 128
- “Monitoring HTTP availability and response time” on page 137
- “Monitoring data from a SOAP or other HTTP data source” on page 145

About this task

Use this task to customize the configuration of process, log file, and script data sources so an agent can access the application that it is monitoring.

All agents must be configured before they can be started. All agents must have basic configuration information such as the method of connecting to the Tivoli Enterprise Monitoring Server. Many times, an agent must have more configuration information so it has access to information specific to the system on which it is running. For example, if you must know the installation location of a software product, add configuration properties to prompt for this information. Another example of information you might prompt for is the user ID and password to access an interface,

Custom configuration is defined by the agent developer. It is not required for all agents, but can be used in the following areas of data collection:

- Matching an argument in a Process Monitor
- Matching the command line in a Process Monitor
- Forming a log file path or name
- Defining an environment variable in a script

Note: Certain data sources such as JMX and SNMP add this configuration automatically.

Note: When data source specific configuration is added automatically by the Agent Builder, this configuration is added in English only.

If during data source definition your agent requires system-specific information for an area of data collection, **Insert Property** or **Insert Configuration Property** is shown.

For example, when you create an attribute group that monitors a log file, **Insert Configuration Property** is shown.

Procedure

1. Click **Insert Configuration Property** to display the **Configuration Properties** window,
2. In the **Configuration Properties** window, click a property, and click **Add**.

   Note: Initially there are no configuration properties that are defined for the agent.
3. In the **Runtime Configuration Property** window, complete the following fields:

   a) In the **Section** area, complete the following fields:

© Copyright IBM Corp. 2010, 2021

Label
Text that describes the properties

Description
(optional) Description of the properties

b) In the **Property** area, complete the following fields:

Label
Text that is displayed in the agent configuration panel that identifies the information you must enter.

Environment variable
The environment variable is displayed in the **Environment variable** field and is updated as you type in the label field. The Agent Builder automatically constructs the name of the environment variable from the product code and the label. If you want to change the environment variable independently from the label, you can clear **Match Label**.

Description
(optional) Description of the property that is being defined.

Type
Type of information that is collected, one of the following options:

String
For any alphabetic information that must be collected (for example, installation locations, user names, and host names).

Password
For any information that must be encrypted when stored. In addition to providing encryption of the data, the data that is typed into the text box is obscured by asterisks. In addition, you are required to type this information twice to validate the data.

Numeric
For any numeric information (for example, port numbers).

Choice
For a list of specific values. This option enables the Choices table. You can define specific values by clicking **Add**. The values that are entered are displayed in the agent configuration panel as a group of selections, you can make only one selection from the group.

Read Only Text
Displays text when you configure the agent, but no information is collected.

Separator
Displays a horizontal separator, but no information is collected.

File Browser
Collects a string, which is a file name. Click **Browse** to browse the file system for the wanted file.

Default value
(Optional) Specify the value that is shown in the configuration panel at run time when the agent is configured for the first time. If you want a default value for UNIX/Linux that is different from a default value for Windows, click **Multiple Values**.

In the **Configuration Property Default Values** window , specify the default values that you want for Windows systems and for UNIX and Linux systems.

Note: Support for multiple default values is a feature that is only supported in IBM Tivoli Monitoring V6.2.1 and higher. If your agent is compatible with IBM Tivoli Monitoring V6.2, a prompt warns you about this requirement and you can cancel or continue with V6.2.1 compatibility enabled.

Required
Check this field if the user must enter a value when the agent is configured. Clear this field if it is optional for the user to enter a value.

c) To add a choice, click **Add**

4. In the **Configuration Property Value** window, complete the **Label** and **Value** fields.

   The Label is displayed as one of the choices. If this choice is taken, the value becomes the property value.

5. Click **OK**.

   The new configuration section and property are displayed in the **Configuration Properties** window under **Custom Configuration**.

6. Optional: To add another property to an existing section, select the section or an existing property in the section and click **Add**. You make the selection in the runtime configuration tree of the **Configuration Properties** window.
7. Complete the fields for the new property. (Complete the same fields as in step "3" on page 205).
8. Click **OK**. The property that you most recently added is selected.
9. Keep the selection or select the property that you want to insert into the log file name.
10. Click **OK**. The property is inserted into the log file name.

    You can then continue through the wizard to complete defining your log file attribute group.

    Note: Even though a configuration property is defined in the context of a log file name, it can be used in other locations. For instance, another location that accepts a configuration property is a script data source. This flexibility means that you can access the value for the configuration element **File Information** with the script variable *$K00_APPLICATION_LOG_FILE* if the product code is K00. You can also use the Windows batch file variable *%K00_APPLICATION_LOG_FILE%*.

Changing configuration properties by using the Agent Editor

Use the Agent Editor to change configuration properties of your agent.

About this task

This task provides information about viewing, adding, and changing configuration properties by using the Agent Editor.

Procedure

1. Click the **Runtime Configuration** tab.
2. Select a configuration section, and click **Add**.

   Add works just like it does in Chapter 10, "Customizing agent configuration," on page 205. There is no **Edit** selection because a configuration section or property is edited when it is selected.

3. Select a configuration property to display the **Runtime Configuration Details** area.
4. In the **Runtime Configuration Details** area, edit the fields to configure the property.

Configuring a Windows remote connection

Information about Configuring a Windows remote connection

About this task

Windows Management Instrumentation (WMI), Windows Performance Monitor (Perfmon), and Windows Event Log data sources can monitor data on the system where the agent is installed. These data sources can also monitor data on remote Windows systems. These three data source types are known as Windows data sources. If these Windows data sources are monitoring data remotely, they all share Windows Remote Connection configuration properties for the agent level where they are defined.

If you define a Windows data source in the base level of your agent, Windows Remote Connection configuration properties are not added to the agent automatically. They are not added to maintain compatibility with earlier versions of agents that might use the Windows data provider before remote

monitoring was enabled. The Windows data source in your agent monitors data on the local Windows system where the agent is installed.

If you define a Windows data source in a subnode in your agent, Windows Remote Connection configuration properties are added to the agent automatically. The Windows data source must support Windows Remote Connection if it is in a subnode. You cannot clear the option until all windows data sources are removed from all subnodes in the agent. Each instance of a subnode might be configured to monitor a different remote Windows system. All Windows data sources in the subnode share Windows Remote Connection configuration properties.

To configure a base agent to remotely monitor a single remote Windows system, use the following procedure.

Procedure

1. In the Agent Editor **Data Source Definition** window, click **Global Options**.
   The **Global Windows Data Source Options** window opens.
2. Select **Include Windows Remote Connection configuration**.
3. Click **OK**.

Results

The following connection-specific configuration properties can be accessed from the Agent Editor **Runtime Configuration Information** page by selecting **Configuration for Windows remote access** > **Windows Remote Connection**

Remote Windows host
Host name of remote Windows computer

Remote Windows password
Password for remote Windows

Remote Windows `DOMAIN\user name`
User name for the remote Windows host

What to do next

You can view, add, and change the configuration properties by using the Agent Editor. For instructions, see “Changing configuration properties by using the Agent Editor” on page 207. If a Windows data source is defined in a subnode, you can also specify Subnode Configuration Overrides. For instructions, see “Subnode configuration” on page 193.

Creating a user with Windows Management Instrumentation (WMI) permissions

You can add and configure a user on a Windows system with permissions to allow WMI browsing.

About this task

If your agent collects data from a remote system by using Windows Management Instrumentation (WMI), it requires permissions to access WMI data on the remote system. The agent can access WMI data on a remote system when you provide credentials of an account with permissions to access WMI data on the system. The procedure applies to Windows 7, Windows 2008 Server and Windows Vista.

Note: Your agent can also access data on a remote Windows system by using Windows Performance Monitor (Perfmon), and Windows Event Log data sources. However, in the case of Windows Performance Monitor (Perfmon), and Windows Event Log data sources, you must provide Administrator credentials for the remote system.

Procedure

1. Create a user account:
   a. Go to Windows **Start** > **Administrative Tools** > **Computer Management**. The **Computer Management** window opens.
   b. Expand **Local Users and Groups**.
   c. Right-click the **Users** folder and select **New User**.
   d. Complete the user details and click **Create** and **Close**.
2. Configure the group membership for the new user account:
   a. In the **Computer Management** window, select the **Users** folder.
   b. Right-click the new user account and select **Properties**.
   c. Click the **Member Of** tab.
   d. Click **Add**.
   e. Click **Advanced**.
   f. Click **Find Now**.
   g. Select the following groups:
      - Distributed COM Users
      - Performance Log Users
      - Remote Desktop Users

      Tip: Press Ctrl and click to select multiple groups.
   h. Click **OK** until you return to the **Computer Management** window.
   i. Select **File** > **Exit** to exit the **Computer Management** window.
3. Assign Distributed Component Object Model (DCOM) rights:
   a. Go to Windows **Start** > **Administrative Tools** > **Component Services**. The **Component Services** window opens.
   b. Expand **Component Services** > **Computers** > **My Computer**.
   c. Right-click **My Computer** and select **Properties**. The **My Computer Properties** window opens.
   d. Click the **COM security** tab.
   e. In the **Access Permissions** area, click **Edit Limits**
   f. In **Distributed COM Users**, verify that **Local Access** and **Remote Access** are selected.
   g. Click **OK** to save settings.
   h. In the **My Computer Properties** window, **Launch and Activation Permissions** area, click **Edit Limits**
   i. In **Distributed COM Users**, verify that **Local Launch**, **Remote Launch**, **Local Activation**, and **Remote Activation** are selected.
   j. Click **OK** to save settings and click **OK** again to close the **My Computer Properties** window.
   k. Select **File** > **Exit** to exit the **Component Services** window.
4. Configure the WMI namespace security assignments
   a. Go to Windows **Start** > **Run...**.
   b. Enter `wmimgmt.msc` and click **OK**.
   c. Right-click **WMI Control (Local)** and select **Properties**.
   d. Click the **Security** tab.
   e. Click **Security**.
   f. Click **Add**.

g. Click **Advanced**.
h. Click **Find Now**.
i. Select the new user account, and click **OK** until you return to the **Security for Root** window.
j. Click **Advanced** and select the newly added user account.
k. Click **Edit**.
l. From the **Apply to:** menu selection, select **This namespace and subnamespaces**.
m. In **Execute Methods**, verify that **Enable Account**, **Remote Enable**, and **Read Security** are selected.
n. Click **OK** until you return to the **wmimgmt** window.
o. Select **File** > **Exit** to exit the **wmimgmt** window.

What to do next

For more information about collecting WMI data from a remote system, see "Monitoring data from Windows Management Instrumentation (WMI)" on page 69.

Configuring a Secure Shell (SSH) remote connection

Information about configuring an SSH remote connection

About this task

Script data sources can monitor data on the system where the agent is installed and also on remote systems. If script data sources are monitoring data remotely, they all share SSH remote connection configuration properties for the agent level where they are defined. Earlier versions of an agent might use the script data provider before remote monitoring was enabled. To maintain compatibility with earlier versions of agents, SSH remote connection configuration properties are not automatically added to the agent. The script data source in your agent monitors data on the local system where the agent is installed.

If you define a Script data source in a subnode and you select **Enable data collection using SSH**, you can configure each subnode instance to monitor a different remote system. All script data sources in the subnode share SSH remote connection configuration properties.

If you want the agent to remotely monitor a remote system, use the following procedure.

Procedure

In the Agent Editor **Data Source Definition** window for the script data source, select **Enable data collection using SSH**.

Results

The following connection-specific configuration properties can be accessed from the **Agent Editor**, **Runtime Configuration Information** page by selecting **Configuration for Secure Shell (SSH)** > **SSH Remote Connection**

Network address
The IP address or host name of the remote computer.

SSH Port Number
The IP port number on which the SSH server is running. The default value is 22.

Authentication Type
Type of authentication to use when you are logging on to the remote SSH server. You can choose Password or Public Key.

Disconnect from the remote system after each collection interval
An option to determine whether the script data provider drops the login session to the remote system after it collects data. By default, the value is No.

Remove script from the remote system after each collection interval
An option to delete the script from the remote system after each data collection interval. By default, the value is No.

If the Authentication Type is set to Password, the following configuration properties can be accessed from the **Agent Editor**, **Runtime Configuration Information** page by selecting **Configuration for Secure Shell (SSH)** > **Password**:

Username
User name for the remote system

Password
Password for the remote system

If Authentication Type is set to Public Key, the following configuration properties can be accessed from the **Agent Editor**, **Runtime Configuration Information** page by selecting **Configuration for Secure Shell (SSH)** > **Public Key**:

Username
User name that is associated with the public key file

Public Keyfile
Public key file that is associated with the user

Private Keyfile
Private key file that is associated with the user

Password
Password that is used to unlock the private key file

What to do next

You can view, add, and change the configuration properties by using the Agent Editor. For instructions, see "Changing configuration properties by using the Agent Editor" on page 207. If the SSH Remote Connection configuration properties are included in a subnode, you can also specify Subnode Configuration Overrides. For instructions, see "Subnode configuration" on page 193.

Chapter 11. Creating workspaces, Take Action commands, and situations

After installing an agent in an IBM Tivoli Monitoring environment, you can create workspaces, queries, Take Action commands, and situations for your monitoring solution.

The situations, workspaces, Take Action commands, and queries that you create can be included in the installation package. To have one installation image for situations, workspaces, and the agent itself, the situation, and workspace files must be in the same project as the agent. The Agent Builder provides a wizard to create the appropriate files in the agent project. For information about importing application support files, see Chapter 17, “Importing application support files,” on page 257.

Creating situations, Take Action commands, and queries

Find information to help create situations, Take Action commands, and queries.

To create situations, Take Action commands, and queries, use the Tivoli Enterprise Portal and the embedded Situation editor. For detailed information about how to create situations, see the Tivoli Enterprise Portal User's Guide. You can also use the help documentation that is installed with your Tivoli Enterprise Portal Server. An Agent Builder monitoring agent can recognize and perform special processing for a set of specific Take Action commands. For more information about these special Take Action commands, see Appendix J, “Take Action commands reference,” on page 369.

Situations for system monitor agents are created differently from the Enterprise situations that are created with the Tivoli Enterprise Portal Situation editor or the **tacmd createSit** command. For system monitor agents, private situations are created in a local private situation configuration XML file for the agent. For information about creating situations for system monitor agents, see "Private situations" in the "Agent Autonomy" chapter of the *IBM Tivoli Monitoring Administrator's Guide*.

Creating workspaces

Place the Tivoli Enterprise Portal in the Administrator mode to create workspaces that you can export and include in your solution.

About this task

Build the workspaces in the environment from which they are used. When you build workspaces, change the display settings on your computer to build workspaces at the minimum resolution that is normally used in your environment. Building workspaces at a greater resolution can create views that are too cluttered to be used reasonably at lesser resolutions.

To create workspaces that you can export and include in your solution, the Tivoli Enterprise Portal must be placed in the "Administrator" mode. To place the Tivoli Enterprise Portal in "Administrator" mode, use the following steps:

Procedure

1. Go to the *ITM_INSTALL*/CNP directory and open the `cnp.bat` file.

   If you used the default installation, the directory is `C:\IBM\ITM\CNP`. In the `cnp.bat` file, you must update the `set _CMD= %_JAVA_CMD%` line to include option `-Dcnp.candle.mode="$_KCJ_$"`.

   If you want to create extensions on Linux or AIX systems, use the following path:

   ```
   /opt/IBM/ITM/li263/cj/bin/cnp.sh
   ```

   Where *li263* is the operating system on which the Tivoli Enterprise Portal is running.

© Copyright IBM Corp. 2010, 2021

The updated `set _CMD= %_JAVA_CMD%` looks similar to the following example:

```
set _CMD= %_JAVA_CMD% -Dcnp.candle.mode="$_KCJ_$" -Xms64m -Xmx256m -showversion -noverify
 -classpath %CPATH% -Dkjr.trace.mode=LOCAL -Dkjr.trace.file=C:\IBM\ITM\CNP\LOGS\kcjras1.log
-Dkjr.trace.params=ERROR -DORBtcpNoDelay=true -Dibm.stream.nio=true
-Dice.net.maxPersistentConnections=16 -Dice.net.persistentConnectionTimeout=1
-Dcnp.http.url.host=SKINANE -Dvbroker.agent.enableLocator=false  -Dnv_inst_flag=%NV_INST_FLAG
%
-Dnvwc.cwd=%NVWC_WORKING_DIR% -Dnvwc.java=%NVWC_JAVA% candle.fw.pres.CMWApplet
```

Note: The command is shown here on multiple lines for formatting reasons only.

2. Open a new Tivoli Enterprise Portal Client, and log in with the `sysadmin` user ID.
3. Set the "sysadmin" user ID in "Administrator" mode. In the Tivoli Enterprise Portal, select **Edit** > **Administer Users**. Select **sysadmin** and then under the **Permissions** tab, select **Workspace Administration**. Select the **Workspace Administration Mode** check box.

   If you make the selection correctly, ***ADMIN MODE*** is displayed in the desktop title bar.

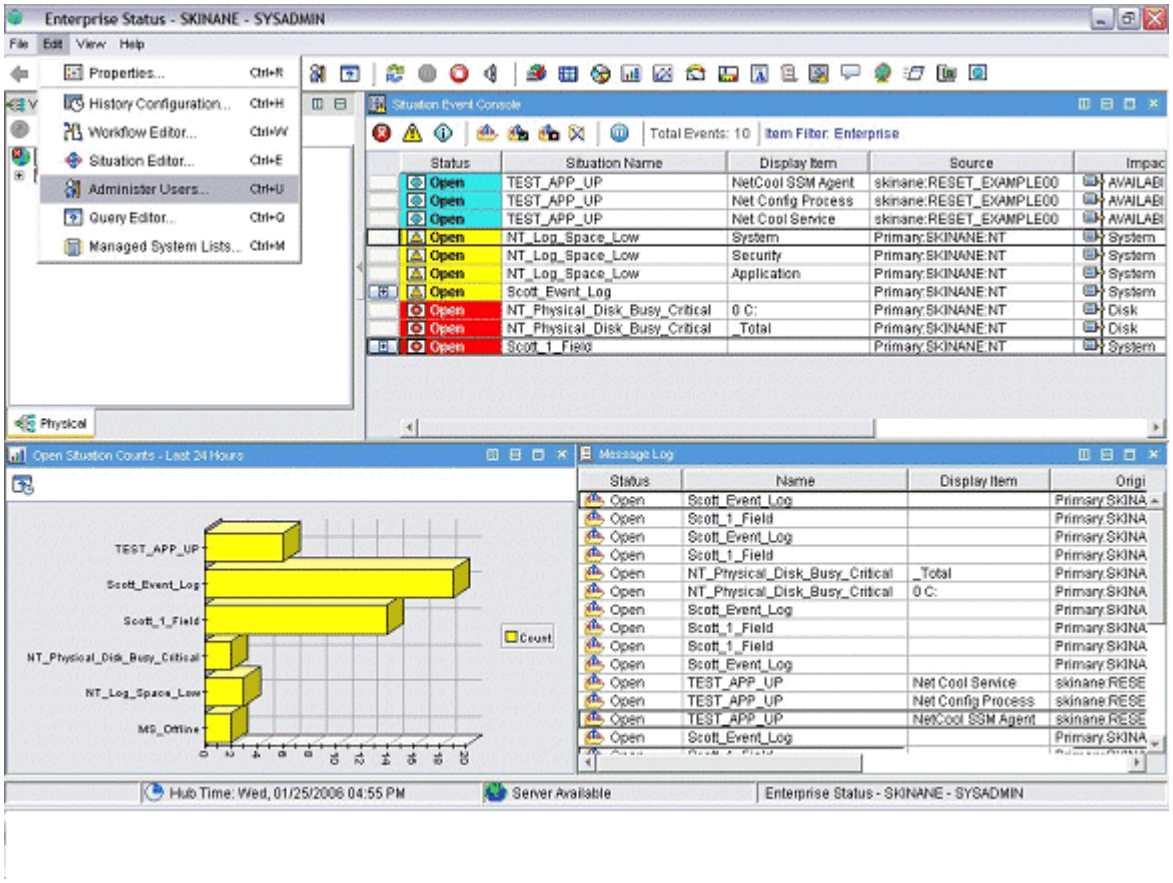


Figure 31. Setting the `sysadmin` user ID

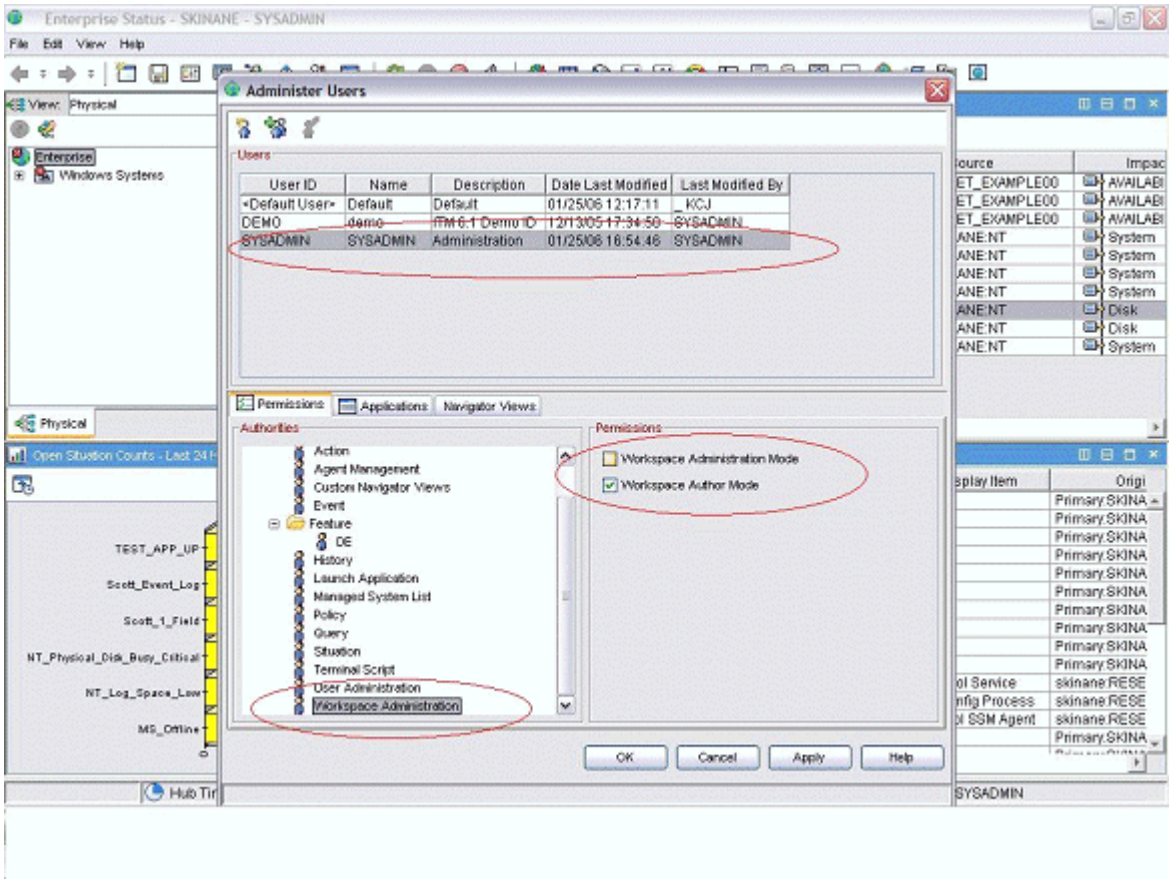


Figure 32. Setting the `sysadmin` user ID (continued)

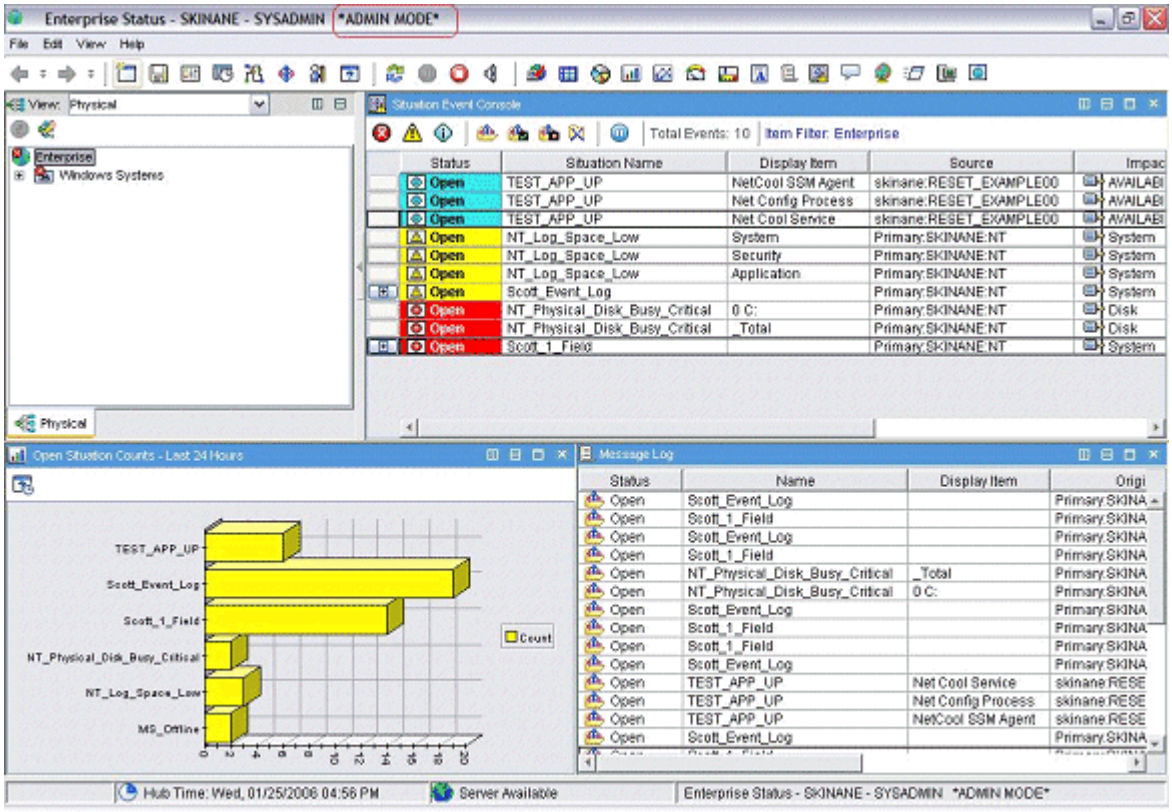


Figure 33. Setting the sysadmin user ID (continued)

What to do next

After you are in "Administrator" mode as depicted in (Figure 33 on page 216), you are now ready to create workspaces for your application. For information about how to customize and create workspaces, see the Tivoli Enterprise Portal User's Guide. Alternatively, use the help documentation that is installed with your Tivoli Enterprise Portal component.

If you want your workspaces to be "read-only" and to not be deleted by a customer, set the "not-editable" and "non-deletable" properties for each workspace. In the workspace properties, you must select the following properties:

- **Do not allow modifications**
- **Product-provided by IBM (mark as non-deletable)**

You can go to the properties by either viewing a workspace or clicking the icon with the controls on it. You can also go to one of the view property pages and then going to the workspace level in the properties tree. If you have more than one workspace for each navigator item, remember to set the properties for each workspace. As indicated in the following example screen capture:

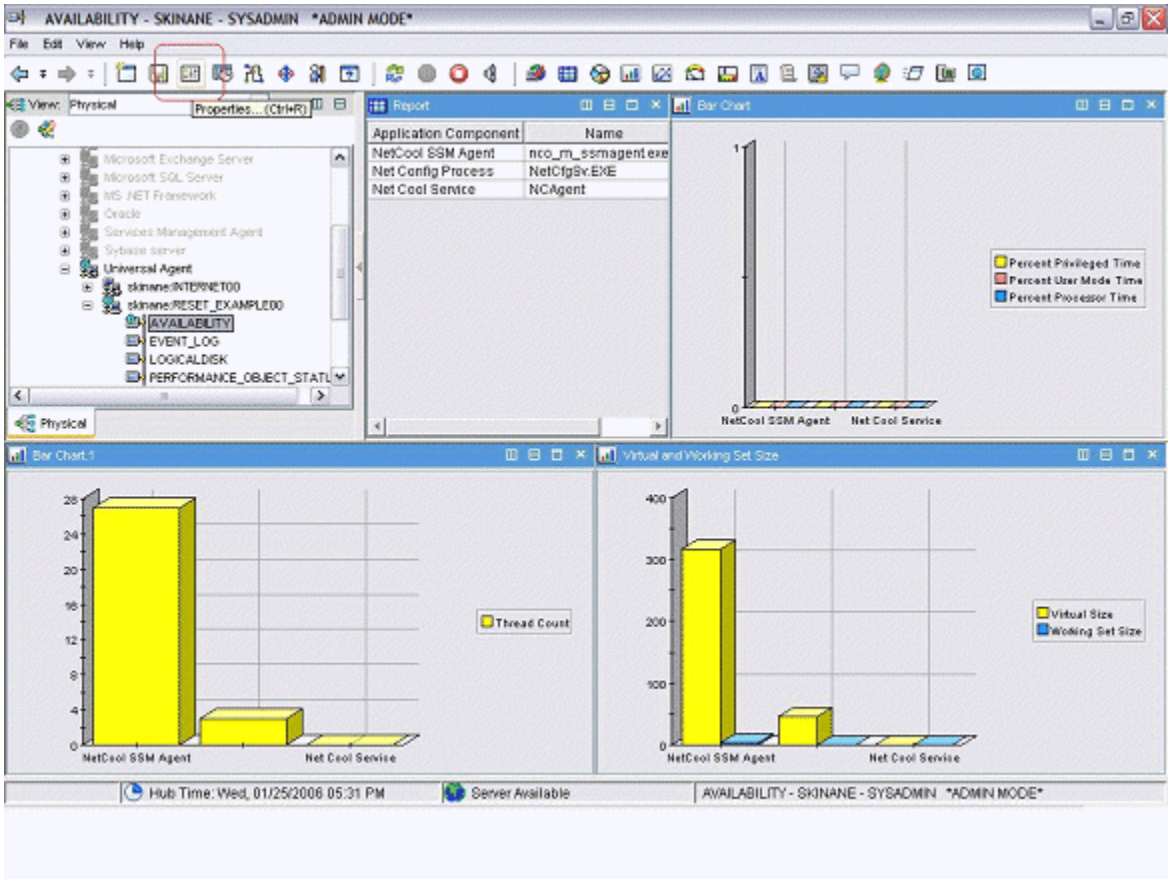


Figure 34. Setting workspace properties

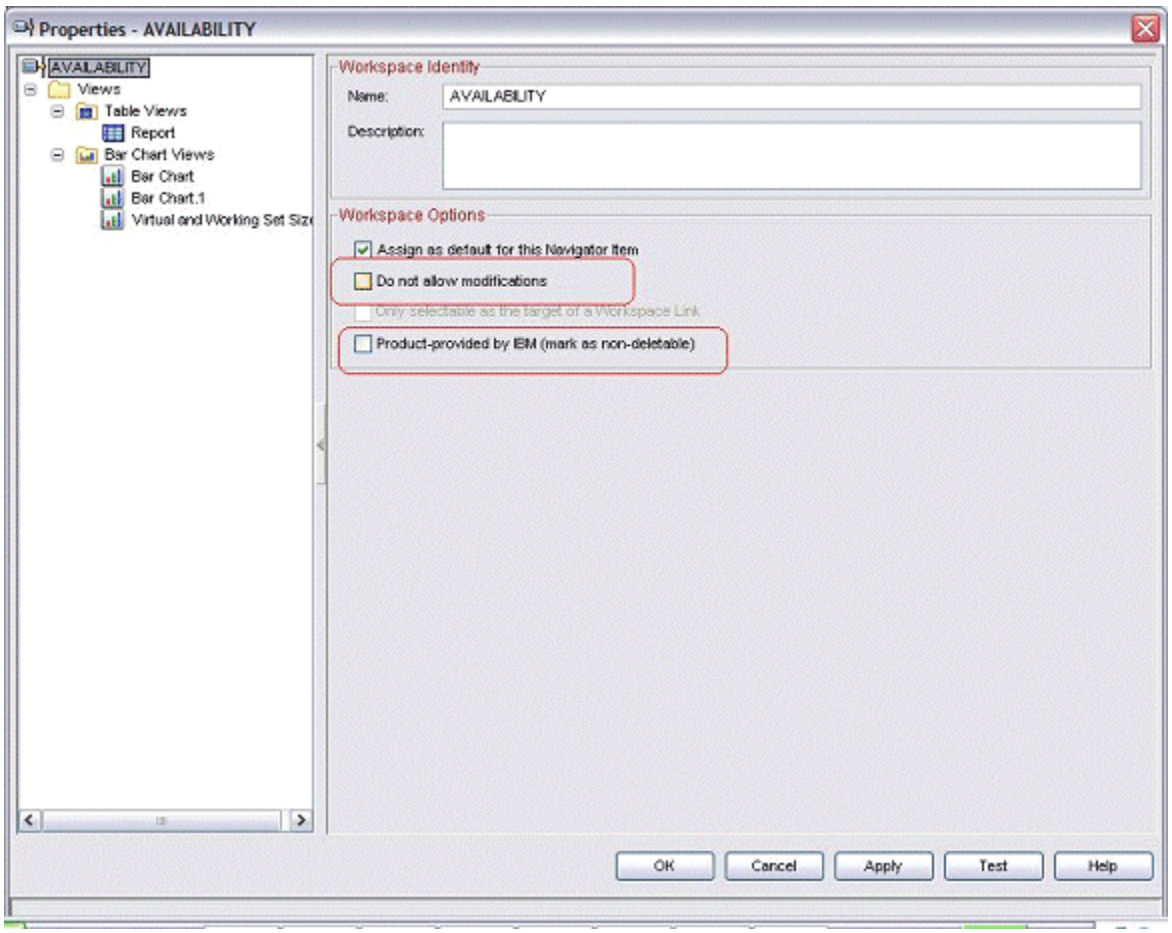


Figure 35. Setting workspace properties (continued)

Chapter 12. Preparing the agent for Cloud APM

If you want to use your agent with IBM Cloud Application Performance Management, you need to prepare it using the **Dashboard Setup** wizard. This wizard configures the information that you can see in the summary and detail dashboards in Cloud APM. It also sets the resource information that Cloud APM requires for the agent.

Before you begin

In order to prepare the agent for Cloud APM successfully, you need to ensure that the agent provides the following data:

- One or more data sets (attribute groups) that produce one row of data. You can use the attributes from these data sets to populate the summary dashboard.

  Important: To include any information in the summary dashboard, you need to provide it in a data set that produces a single row of data. Some data sources create data sets that produce multiple rows of data; for example, the process, Windows service, and command return code data sources place data into the single Availability data set, which produces multiple rows. In such cases, you need to create a filtered data set producing one row in order to include the data in a summary dashboard. For instructions, see “Creating a filtered attribute group” on page 182.
- A numeric attribute within one of these data sets that indicates the status of the monitored service (normal, warning, critical, or other similar status values). You must define status severity values for this attribute. For instructions about defining status severity values, see “Specifying severity for an attribute used as a status indicator” on page 44.
- If the port number on which the monitored application provides service is fixed, you must know the port. If the port might change between different deployments, one of the data sets that produce one row of data must contain a numeric field that indicates the port.
- If the agent can be installed on a host to monitor a server that is running on a different host, a string attribute within one of these data sets that indicates the server IP address. If the agent always monitors the host where it is running, such an attribute is not required.

  Tip: If an attribute that provides the host name is available, you can create a derived attribute for the IP address by using the `nameToIpAddress` function. For information about creating a derived attribute, see “Creating derived attributes” on page 37. For information about the function, see “ipAddressToName” on page 51.

If the agent has subnodes, these requirements apply to each subnode for which you want to create a dashboard.

About this task

Cloud APM monitors *resources*. A resource corresponds to instance of the agent, or sometimes a subnode. To define a resource, you need to supply a resource type name, server name, IP address, and port number that apply to the monitored service.

Cloud APM displays a summary dashboard for every monitored resource. The summary dashboard includes a status indicator; with this indicator (usually green, yellow, or red for normal, warning, or critical status) the user can see the status of the resource at a glance. The same dashboard can contain a few other high-level health metrics.

On the summary dashboard, data is displayed as single items. Therefore, the data set with this data must produce only one row.

Optionally, a detail dashboard can be available for the agent. The user can click the summary dashboard to view the detail dashboard. The detail dashboard can display tables, so data from any data set can be used on this dashboard.

© Copyright IBM Corp. 2010, 2021

You must select the attributes that are displayed on the summary dashboard (including the status indicator) and on the detail dashboard.

Important: The data in the attributes that you select is automatically passed from the agent to the Cloud APM server every minute. Specifying too much data can lead to overloading of the network, the server, or the monitored host. Select the required attributes only. For example, if a joined data set or a derived attribute must be displayed, do not specify the source attributes as well.

Important: No data other than these attributes is passed to Cloud APM. You cannot view or use other data in Cloud APM, except for thresholds, which are monitored at the agent level. If you use other data in thresholds, you might not be able to view the threshold status in the Cloud APM console.

Procedure

1. From the **Agent Information** view, click the **Dashboards** link.
2. Under **Dashboard Components**, select **Show agent components in the dashboard**.

   Tip: Alternatively, if you are creating an agent for use exclusively with IBM Tivoli Monitoring, you can select **No dashboard presence for this agent**. In this case, do not complete the subsequent steps of this procedure. You can not install such an agent in a Cloud APM environment.
3. Click the **Dashboard Setup Wizard** link.
4. If the agent has subnodes, define the arrangements of agent and subnode resources in Cloud APM:
   - Select **Base agent instances** to display the base agent (data outside of subnodes) as a resource.
   - For every subnode, select **Subnode "*name*" instances** to display this subnode as a resource.
   - Optionally, for any of the selected subnodes, select **Show as child of agent**. In this case, the subnode resource is displayed as a child under the agent resource in lists in the Cloud APM console.

   Cloud APM displays a summary and detail dashboard for each of the components you selected.

   Important: If you run the wizard again and unselect an agent or subnode, the resources for the agent or subnode are not removed automatically. To remove the resources, expand **Resources** in the Outline view, select the resources to be deleted, and press the Delete key on the keyboard.
5. In the **Attribute Selection - Status** page, select the attribute that indicates the status of the monitored service. Numeric attributes from groups that return a single data row are available.

   Tip: Alternatively, if you do not want to display status in the dashboard, unselect **Provide status for this agent**.
6. In the same page, you can select whether you want to display additional data in the summary and detail dashboards:
   - To display additional high-level health metrics in the summary dashboard, ensure the **Select additional attributes to display in this agent's summary information** box is selected. Otherwise, clear the box.
   - To display additional data in the detail dashboard, ensure the **Select additional attributes to display in this agent's detail information** box is selected. Otherwise, clear the box. (Typically, select this box, as a detail dashboard is required to display enough data to make a monitoring agent meaningful).

   Click **Next**.
7. If you selected **Select additional attributes to display in this agent's summary information**, in the **Attribute Selection - Summary** page, select up to four additional attributes to include in the summary dashboard. Attributes from groups that return a single data row are available. Click **Next**.
8. If you selected **Select additional attributes to display in this agent's detail information**, in the **Attribute Selection - Details** page, select the attributes to include in the detail dashboard. All attributes in the agent are available; to avoid performance issues, include as few attributes as possible. Click **Next**.

9. In the **Resource Type** page, enter the server type that you are monitoring, for example, `Email server` or `SampleCo Database Server`. Click **Next**.
10. In the **Attribute Selection - Software Server Name** page, enter a fixed software server name in the **Fixed Name** field or select an attribute from your agent that gives the software server name. This name is displayed to the user for this particular monitored instance, for example, the name of the JBoss application server instance. Click **Next**.

    Important: Do not run two or more monitoring agents, agent instances, or subnodes with the same software server name on the same monitored host. If your agent has instances or subnodes, ensure that a unique software server name is generated for every instance or subnode. If two different agents produce the same software server name, do not install them on the same monitored host.
11. In the **Attribute Selection - IP address** page, select an attribute from your agent that specifies the IP address (not host name) of the primary interface connection that the monitored server or application uses. For example, the HTTP connection for an HTTP server or the database client connection for a database server. Alternatively, select **Use the agent's IP address** to use the address of the host where the agent runs. Click **Next**.
12. In the **Attribute Selection - Port** page, enter the port on which the monitored application provides service or select a numeric attribute from your agent that specifies this port. Click **Finish**.
13. If you selected both an agent and a subnode or more than one subnode as resources, click **Next** to enter dashboard and resource information for the next component (agent or subnode). If the **Next** button is disabled, you entered the information for all necessary components; click **Finish** to complete the wizard.

Results

When you install the agent on a monitored host, you can view the summary and detail dashboards in the **Status Overview** tab.

Important: There can be a delay of up to 30 minutes between installation of the agent and availability of the dashboards, especially if this is the first time that this agent type and version is installed in your environment.

Click the summary dashboard for the agent to view the detail dashboard. By default, all information in the detail dashboard is displayed as tables.

You can use the **Attribute Details** tab to configure custom display of this information as tables and charts.

Chapter 13. Preparing the agent for Cloud Pak for Multicloud Management

If you want to use your agent with IBM Cloud Pak for Multicloud Management, you need to prepare it using the **Resource wizard**. The Resource wizard creates a resource definition. You can also establish relationships between defined resources using the **Relationship wizard**.

Defining resources

Use the Resource wizard to create or modify resource definitions for your agent so that you can see their monitoring data in the Resources dashboard on the Cloud Pak console.

Before you begin

The resource definition process reads the agent definition and allows you to build resources. But if you then edit the agent definition to add an attribute, rename an attribute, add an attribute group, or delete either one, these changes are not automatically reflected in the resource definition. For this reason, ensure that you have the correct data before you run the Resource wizard.

About this task

Resources are defined by resource types, and when the agent runs it will create resource instances that are displayed in the console. Resource types define a set of properties and metrics that provide the identity and metrics related to a managed resource.

Use the Resource wizard to create or edit a resource definition for the agent. The resource definition then automatically generates the widgets for the Resources dashboard. In the Resource wizard, you define an attribute group; this attribute group provides the data for the basic widgets in the Resources dashboard. Optionally, you can also add components; components are extra tables of data that appears in the Resources dashboard below the Events timeline widget. You can add as many component tables as you need.

When you run the Resource wizard, the following widgets are created and displayed in the IBM Cloud Pak for Multicloud Management Resources dashboard:

- Events timeline
- Line graphs that display data for the "important" metrics (see step “5” on page 224 for a description)
- Tabular display of the metrics in the main resource attribute group
- Optional (s) for each component created
- Related resources widget

In addition, if you have a status attribute in the attribute group that defines a resource, a threshold is automatically created in the Cloud Pak console.

Procedure

Complete these steps to define or edit a resource for the agent:

1. Select **Agent Definition**>**Resources**, the **Introduction** window is displayed, click **Next**.
2. In the **Resource Information** window, select **Add new Resource**, click **Next**.
3. In the **Attribute Group Selection** window, select an attribute grouping. At a minimum, the attribute grouping needs to identify the resource.
4. In the **Resource Information** window, enter a name and description for the resource. The name that you enter corresponds to the name you see in the Resources types list in the Resources dashboard.

© Copyright IBM Corp. 2010, 2021

The description also appears in the resource list in the Resource tab. You can use any characters for name or description.

5. In the **Attributes** window, you can assign different characteristics to your attributes.
   - **Not Available:** Use this to eliminate the attribute completely from both the Resources dashboard and the Threshold manager.
   - **Not uploaded to the server, available for threshold calculation:** Use this to indicate that the attribute data isn't uploaded by default. Use this if you don't want this attribute to be displayed in the Resources dashboard, but you do want to be able to create thresholds against it in the Threshold manager.
   - **Metric describes the state of the resource instance at a given time**. Use this to assign a string as a metric. Usually all strings are properties and all numbers are metrics (unless you earlier explicitly specified otherwise in the Attribute editor, for more information, search for 'purpose' in table 1 in the “Numeric aspects of attributes” on page 42 topic.
   - **Property describes the resource instance:** Use this to assign a numeric as a property. By default all the strings are properties and all the numbers are metrics.
   - **Display Name:** By default the display name and key property is the node name. This is indicated with "D" in the **Opt** column. When node name is the display name, you see the origin node name from IBM Tivoli Monitoring displayed in the selection lists in the Resources dashboard. If there is an attribute that is more suited to being the display name, select Display name for that attribute.
   - **Important**: This indicates that a metric is represented in a line chart in the Resources dashboard. The label and grouping for the line chart is based on the unit that you specified when you defined the attribute. For more information, see “Numeric aspects of attributes” on page 42. It is recommended indicating 1 - 4 metrics as important.
6. If you want more tables to present extra data in the resource, add a component, you add one component for each table you want to add to a resource. In the **Component selection** window, select **Add a new component**. Click **Next**.
7. In the **Component Information** window, enter a name and description for the component. The name that you enter corresponds to the name that is given to the table in the Resources dashboard. You can use any characters for name or description. Click **Next**.
8. Repeat step “3” on page 223 and “5” on page 224. Click **Finish**.

Results

You can now view the Resource dashboard in the Cloud Pak console. Representations of your resource definition are presented in these Resource dashboard widgets:

- Events timeline
- Line graphs that display data for the "important" metrics (see step “5” on page 224 for a description)
- Tabular display of the metrics in the main resource attribute group
- Optional (s) for each component created
- Related resources widget

.

Building resource relationships

After you have created your resource definitions, you can create a relationship between two resources using the **Relationship wizard**.

About this task

The Relationship wizard guides you through setting up a resource relationship for enriching the interaction and display of data in the Resource dashboard between two resources. For example, you can set up a relationship for an application resource that runs on the Operating system (the target resource). You can

then see the linkage in the Resource dashboard when the operating system is slow or fails, and how it affects the application.

Procedure

Complete these steps to create or edit a resource relationship:

1. Select **Agent Definition** > **Relationship** to open the **Introduction** window, and then click **Next**.
2. In the **Relationship Selection** window that displays, choose to add or modify a relationship:
   - To define a new relationship, click **Add a new relationship**, and click **Next**.
   - To edit an existing relationship, select it from the list, click **Modify selected relationship**, and click **Next**.
3. Create or edit the resource relationship definition:
   a) For **Source resource**, select the main resource.
   b) For **Relationship type**, select the option that describes the relationship: is a federation of, runs on (and depends on), is a member of, defines, has been assigned to, manages, or uses.
   c) For **Target resource**, select the target resource.
   d) If you selected a source resource or target resource that is a *single-instance* resource (does not specify keys), click **Finish**.
   e) If you selected a source and target that are both *multi-instance* resources, click **Next**, and select which resource properties to use to match the related resources.

      You can add multiple properties if you want multiple key attributes that match.

   Note: To define a relationship to an *operating system*, select by selecting the check box for **Relationship is with the operating system the agent is running on**.

Results

You can observe the relationship in the Resource dashboard for your agent in the Cloud Pak console.

Chapter 14. Data Definition Designer

For information about the Data Definition Designer, see The Data Definition Designer guide.

© Copyright IBM Corp. 2010, 2021

Chapter 15. Testing your agent in Agent Builder

After you use Agent Builder to create an agent, you can test the agent in Agent Builder.

Test the agent to ensure that the monitoring data you are expecting is the data that is being displayed. By testing your agent, you can learn to modify or tweak settings in the agent to ensure that the data displayed is beneficial and accurate.

You can test your agent in Agent Builder by using the following methods:

1. Begin by using the attribute group test function of Agent Builder to test individual attribute groups one at a time. For more information, see “Attribute group testing” on page 229.
2. After you complete attribute group testing, you can use the agent test function of Agent Builder to test all attribute groups in your agent together. For more information, see “Full agent testing” on page 232.

Important: When testing your agent in Agent Builder, you can see the following special values for numeric attributes:

- -1: a general error
- -2: missing data
- -3: no value (for example, NULL was returned by a database)

Attribute group testing

You can use attribute group testing to test the attributes groups of the agent you created with Agent Builder, one attribute group at a time. You can test many attribute groups before you complete the attribute group definition. For example, you can initiate testing from the **IBM Tivoli Monitoring Agent Wizard** when you are defining the attribute groups of a new agent. You can also initiate testing from the **IBM Tivoli Monitoring Agent Component Wizard** when you are adding attribute groups to an existing agent.

Before you begin

Before you start testing an attribute group, you can optionally:

- Set attribute group testing preferences. For more information, see “Attribute group testing - preferences” on page 231.
- Set environment variables, configuration properties, and where applicable Java information. For more information, see “Attribute group testing - configuration” on page 231.

About this task

Agent Builder supports an attribute group test function for most data sources

Procedure

- Start the Testing procedure in the following ways:
  1. During agent or attribute group creation click **Test** on the relevant data source Information page.
  2. After agent creation, select an attribute group on the Agent Editor **Data Source Definition** page and click **Test** . For more information about the Agent Editor, see Chapter 4, “Using the Agent Editor to modify the agent,” on page 17.

  After you click **Test** in one of the previous two steps, the Attribute group Test window is displayed. This window is different for different data sources,

  Agent Builder supports an attribute group test function for most data sources.

© Copyright IBM Corp. 2010, 2021

For more information about the test procedures for specific attribute groups, see the following Testing sections:

- Windows Management Instrumentation (WMI), for more about the WMI test procedure, see "Testing WMI attribute groups" on page 71
- Windows Performance Monitor (Perfmon), for more about the Perfmon test procedure, see "Testing Perfmon attribute groups" on page 73
- Simple Network Management Protocol (SNMP), for more about the SNMP testing, see "Testing SNMP attribute groups" on page 77
- Simple Network Management Protocol (SNMP) event sender, for more about the SNMP event test procedure, see "Testing SNMP event attribute groups" on page 82
- Java Management Extensions (JMX), for more about the JMX test procedure, see "Testing JMX attribute groups" on page 101
- Common Information Model (CIM), for more about the CIM test procedure, see "Testing CIM attribute groups" on page 104
- Log file, for more about the log file test procedure, see "Testing log file attribute groups" on page 113
- Script, for more about the script test procedure, see "Steps for monitoring output from a script" on page 125
- Java Database Connectivity (JDBC), for more about the JDBC test procedure, see "Testing JDBC attribute groups" on page 134
- Internet Control Message Protocol (ICMP) ping, for more about the ICMP test procedure, see "Testing Ping attribute groups" on page 137
- Hypertext Transfer Protocol (HTTP) Availability, for more about the HTTP test procedure, see "Testing HTTP attribute groups" on page 145
- SOAP, for more about the SOAP test procedure, see "Testing SOAP attribute groups" on page 153
- Transmission Control Protocol socket (TCP) socket, for more about the socket test procedure, see "Testing socket attribute groups" on page 163
- Java application programming interface (API), for more about the Java API test procedure, see "Testing Java application attribute groups" on page 176

Some data sources do not have an attribute group test function, for example:

- When you can use the Agent Builder browser to view live data on a system. For example, you can view the processes that are currently running on the system (processes). Other examples are when you can view the services that are installed on the system (windows services) and the Windows Event Logs that are present.
- There is little or no customization that you can do in the agent (AIX Binary Log, command return code).
- Joined and Filtered attribute groups cannot be tested by using the attribute group test function because these groups are based on multiple attribute groups.

Note:

1. Use the full agent test to test data sources that cannot be tested by using the attribute group test function. For more information about the full agent test, see "Full agent testing" on page 232.
2. When you test data sources, after you click **Collect Data**, data might not be displayed at all or might not be current after the first click. In such cases, click **Collect Data** a second time to display current data.

- Debugging:

  Each data source that is tested has a test directory that is created for it by Agent Builder. This directory is used for the test runtime environment of the data source. Log files that relate to tests run on the data source are stored under this directory. The log files can be useful to help debug issues that are found during testing.

Note:

1. The location of the test log file is shown as a status message in the **Test** window after you click **Start Agent** and also after you click **Stop Agent**.
2. All test data source directories are deleted when the Agent Builder is shut down.

Attribute group testing - preferences

Set preferences before you test an attribute group.

About this task

Before you start testing an attribute group, you can optionally set some preferences that determine how attributes are treated during testing.

Procedure

1. Select **Window** > **Preferences** from the Agent Builder menu bar.
   The **Preferences** window opens.
2. Select **Agent Builder**.
   The preferences that are associated with testing attribute groups are shown:

   Show data types changed dialog when testing
   When selected, Agent Builder suggests changes to the data type of an attribute. Agent Builder suggests changes when the data type of an attribute does not match the data that is returned by a test for that attribute. For example, if the string length defined for an attribute is too short to hold a value that is returned by a test. In this example, Agent builder suggests redefining the attribute to have a longer string length. When this option is cleared, Agent Builder does not check or suggest data types during testing. This option is selected by default.

   Maximum script or log attributes created
   The value that is entered in this field determines the maximum number of attributes that Agent Builder parses during the initial test of a log file or script attribute group. The default value is 25.
3. When you are finished setting your preferences, click **OK** to save your settings and close the **Preferences** window.
   If you want to restore the default settings, click **Restore Defaults** before you click **OK**

Attribute group testing - configuration

Set environment variables, configuration properties, and Java information before you test an attribute group.

About this task

Before you start testing an attribute group, you can optionally set environment variables, configuration properties, and where applicable Java information from the data source Test window. The Java information is a subset of the configuration data. Some environment variables have special values that are set by default for attribute group testing. For more information about environment variables with special values for attribute group testing, see “Test Environment variables” on page 236.

Procedure

1. Optional: Click **Set Environment** from the data source **Test** window.
   The **Environment Variables** window opens. When populated, the **Environment Variables** window lists all of the environment variables that are used during the running of the test. The initial view of the Environment variable window contains any existing environment variables that are defined in your agent. It also contains any environment variables that you added from previous tests of this agent.
   a) Click **Add** or **Edit** to add or edit individual variables.

b) Click **Remove** to remove individual variables, or **Restore Default** to restore default variables and remove all others.

c) Click **OK** to save your changes and return to the **Test** window.

2. Optional: Click **Configuration** from the data source **Test** window. The **Runtime Configuration** window opens.

   a) Click **Edit Agent Configuration** to add a configuration property or to edit existing agent configuration properties by using the **Configuration Properties** window.

   b) Select a configuration property and click **Edit** to edit an existing configuration property that relates to the attribute group you are testing.

   c) Select a configuration property and click **Restore Default** to restore a configuration property to its default value.

   Important: If a JMX data source connects to a remote WebSphere Application Server, ensure that a local WebSphere Application Server is installed and that the Java location is set to the JRE that this server uses. For details about setting up the connection, see "Monitoring Java Management Extensions (JMX) MBeans" on page 83.

3. Click **OK** to save your changes and return to the **Test** window.

4. **Note:** You can set Java information for following types of attribute groups:

   - Java Management Extensions (JMX)
   - Java Database Connectivity (JDBC)
   - Hypertext Transfer Protocol (HTTP) Availability
   - SOAP
   - Java application programming interface (API)

   The Java information is a subset of the configuration data described in step "2" on page 232

   Optional: Click **Java Information** from the data source **Test** window.

   The **Java Information** window opens.

   a) Enter Java Information.
   For example, Browse to or type the location of the Java Runtime Environment (JRE), select a **Java trace level**, or enter **JVM arguments**

   b) Click **OK** to save your changes and return to the **Test** window.

Full agent testing

Use full agent testing to test all attribute groups of your agent together. You can also use full agent testing to test data sources that cannot be tested by using the attribute group test function.

About this task

You can use full agent testing to run the agent in the same way it runs in IBM Tivoli Monitoring without needing an IBM Tivoli Monitoring installation.

Important: On Windows systems, If you want to run a full test of the agent inside Agent Builder (see "Full agent testing" on page 232), ensure that the 32-bit version of the operating system on which you are running the Agent Builder, that is, 32-bit Windows, is selected in the Agent Information window. On Linux systems, the 64-bit version must be selected.

Procedure

1. Open the **Agent Test** perspective:

   a) In the agent editor, open the **Agent Information** tab.

   b) Click **Test the agent**.

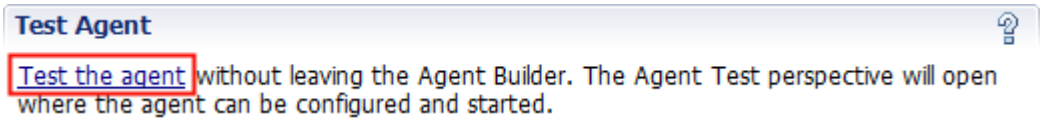


*Figure 36. **Test Agent** section of the **Agent Editor**, **Agent Information** page.*

Alternatively, from the Agent Builder menu select **Window** > **Open Perspective** > **Other**, select **Agent Test** and click **OK**

The **Agent Test** perspective opens (Figure 38 on page 235). The **Agent Test** view shows agents that you have opened in the agent editor; you can test any of these agents. An **Attribute Group Test** view is also displayed; this view is initially empty. The **Attribute Group Test** view shows data that is collected from a selected attribute group when the agent is running.

Tip: If no agents are being edited, the **Agent Test** perspective is empty. To populate the view, go to the **IBM Tivoli Monitoring** perspective and open an agent in the **Agent Editor**. When an agent is opened in the **Agent Editor** return to the **Agent Test** perspective to test the agent.

2. Optional: Configure environment variables and configuration properties before you start the test.

   You can access the **Environment Variables** and **Runtime Configuration** windows in two ways from the **Agent Test** view:

   - Right-click the agent in the **Agent Test** view to open a selection menu. You can select **Set Environment** from the menu to open the **Environment Variables** window. You can select **Configuration** from the menu to open the **Runtime Configuration** window.
   - Click the view menu icon on the **Agent Test** view toolbar to access the **Set Environment** and **Configuration** menu items as in the previous choice.

   For more information about using the **Environment Variables** and **Runtime Configuration** windows, see “Attribute group testing” on page 229.

   Important:

   a. The agent is populated automatically with the last set of configuration that relates to each tested attribute group.

   b. Some environment variables can have different default values for attribute group testing and for full agent testing. For more information about environment variables with special values for attribute group testing, see (“Test Environment variables” on page 236).

   c. If a JMX data source connects to a remote WebSphere Application Server, ensure that a local WebSphere Application Server is installed and that the Java location is set to the JRE that this server uses. For details about setting up the connection, see “Monitoring Java Management Extensions (JMX) MBeans” on page 83.

   d. In a Java API, JDBC, JMX, HTTP, or SOAP data source, you can use the **Java** > **JVM arguments** setting to control agent trace logging. Set the following value:

   ```
   -DJAVA_TRACE_MAX_FILES=files -DJAVA_TRACE_MAX_FILE_SIZE=size
   ```

   where *files* is the maximum amount of trace log files that are kept (the default value is 4) and *size* is the maximum log file size in kilobytes (the default value is 5000). For example, you can set the following value:

   ```
   -DJAVA_TRACE_MAX_FILES=7 -DJAVA_TRACE_MAX_FILE_SIZE=100
   ```

   In this case, the agent writes 100 kilobytes into the first log file, then switches to the second log file, and so on. After writing seven log files of 100 kilobytes each, it overwrites the first log file.

   e. If your agent has subnodes, in an installed version you can set different configuration values for different subnodes and separately for the base agent attribute groups. However, in full agent testing configuration you can only set every configuration value once; the setting applies to the base agent and any subnodes. You can only test one instance of every subnode.

3. In the **Agent Test** view, select the agent that you want to test and click the **Start Agent** icon.

   A window indicates that the agent is starting. When the agent starts, its attributes groups are shown as children of the agent in the **Agent Test** view. The attribute groups are indicated by the attribute group icon .

   The status attribute groups that give information about the agent (**Performance Object Status**, **Thread Pool Status** and **Take Action Status**) are also shown as children of the agent in the **Agent Test** view. The status attribute groups are indicated by the information icon.

   You can start and run more than one agent at the same time.

   The **Stop Agent** icon becomes available when the agent is started.

   If your agent has subnodes or navigator groups, they are shown as nodes in the **Agent Test** view. Subnode definitions are shown under the agent. A subnode instance node is shown under the subnode definition node. Attribute groups and navigator groups are shown under the subnode instance node. For example:

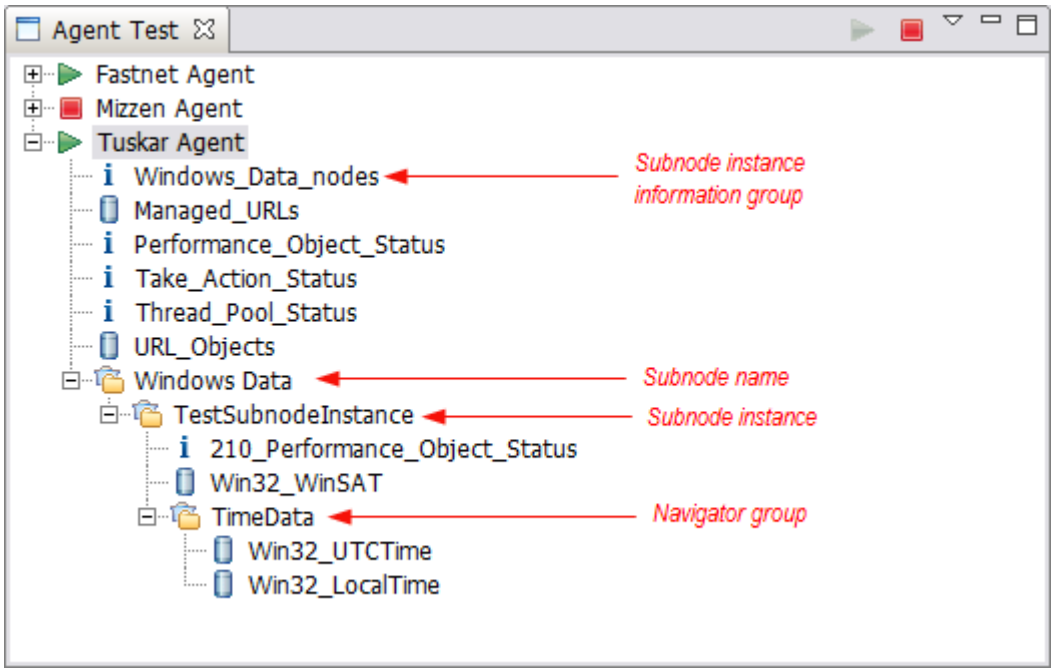


*Figure 37. **Agent Test** view with example subnode and navigator group highlighted.*

   You can right-click on any of the nodes in the **Agent Test** view to access menu selections like **Edit** and **Stop Agent**. **Edit** opens the **Data Source Definition** for the selected node in the **Agent Editor**.

   Note: Changes that you make with the **Agent Editor** are not visible in the running agent until you stop and restart the agent.

4. In the **Agent Test** view, select the first attribute group that you want to test.

   When you select an attribute group, a data collection begins for the selected attribute group. If the collection takes some time, a window indicates that the data collection is in progress. When the data collection is complete the collected data is displayed in the **Attribute Group Test** view, for example:

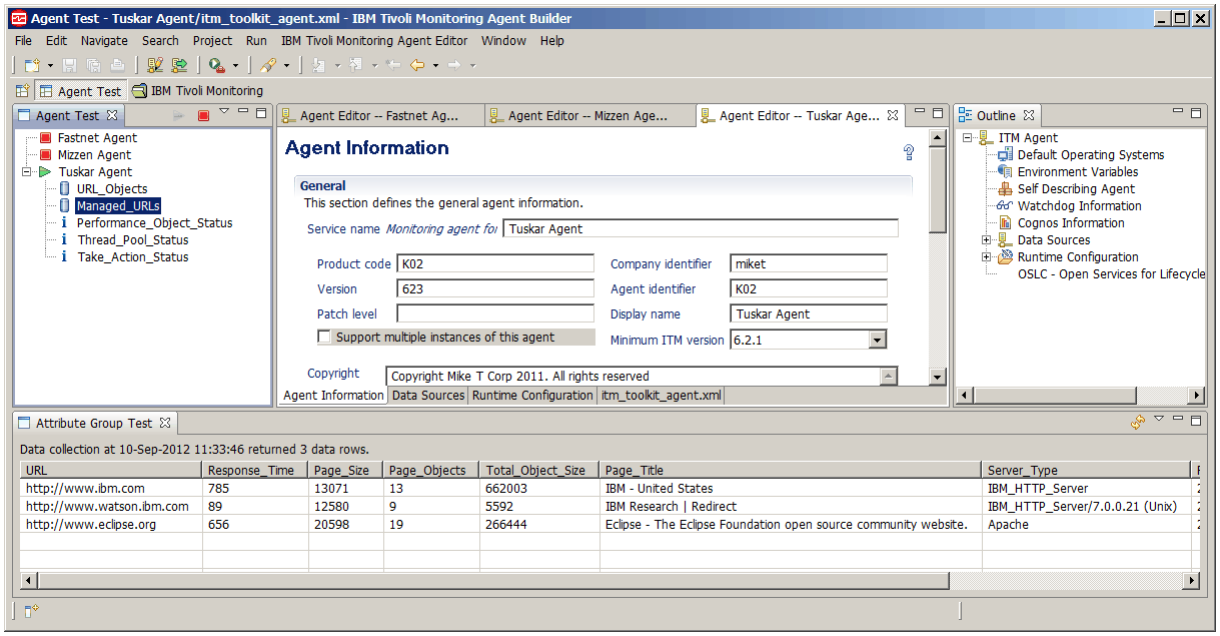


*Figure 38. **Agent Test** perspective*

If no data is displayed, a message `0 data rows returned` is shown in the **Attribute Group Test** view. There are several reasons why the agent might not return data. These reasons include:

- There is no data
- Incorrect definition
- Incorrect configuration

You can check the reason why no data is returned by looking at the value of the **Error_Code** in the **Performance Object Status** attribute group. For more information about viewing the **Performance Object Status** attribute group, see step "9" on page 236

To collect data for another attribute group in the running agent, select the required attribute group.

When you select an attribute group in the **Agent Test** view, the corresponding attribute group is displayed in the **Agent Editor** view.

5. Optional: Run a second data collection, after the initial data collection, for certain attribute group types, to get useful data values.

   To run a data collection, click the collect data icon in the **Attribute Group Test** view.

   If the collection takes some time, a window indicates that a data collection is in progress. When the data collection is complete, the newly collected data is displayed in the **Attribute Group Test** view.

6. Optional: Click an attribute column heading in the **Attribute Group Test** view to open the **Attribute Information** in the **Agent Editor Data Source Definition** tab. You can also access the same **Attribute Information** by right-clicking on any data cell in the table and choosing **Edit** from the menu.

   You can edit properties of the attribute in the normal way. Changes that you make are not visible in the running agent until you stop and restart the agent.

7. Optional: Open multiple **Attribute Group Test** views at the same time.

   To open an additional **Attribute Group Test** view, click the view menu icon on the **Attribute Group Test** view toolbar and then select **Open view for attribute group**.

   Note: When an additional **Attribute Group Test** view is opened, it displays the same attribute information as the original **Attribute Group Test** view. You can then select another attribute group in the **Agent Test** view to display different attribute group information in the original **Attribute Group Test** view. The first time another **Attribute Group Test** view is opened, it opens in the same location

as the original view but with its own tab. If you want to see the two views simultaneously, you can drag the tab to another location in the workspace.

8. Optional: Select the subnode instance information attribute group, if your agent has subnodes, to see how the subnodes are listed in your agent (Figure 37 on page 234). Selecting the subnode instance information attribute group shows subnode instance information in the **Attribute Group Test** view (for all online subnodes of the selected type).
9. Optional: To see more information about the operation of the agent, you can select the **Performance Object Status** and **Thread Pool Status** attribute groups in the **Agent Test** view. These status attribute groups are indicated by the information icon . Select these groups to see status information about earlier data collections for your attribute groups.

   For example:

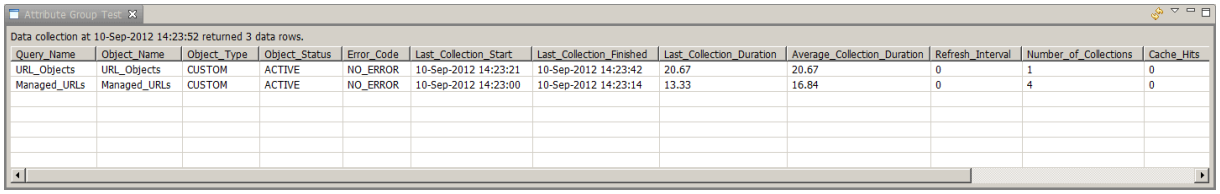


*Figure 39. The **Attribute Group Test** view that shows more information (Performance Object Status) about data collections for the **Managed_URLs** and **Managed_Nodes** attribute groups*

10. When you are finished testing your agent, click the stop agent icon

Test Environment variables

Use these environment variables to control the behavior of the agent during testing.

Environment variables are dynamic named values that determine how the agent runs. For attribute group test, some agent environment variables are set to special values. The special values are used so that the agent responds in a way that suits the testing of a single attribute group. For full agent test special values are not used and instead the default values are used. The default values mean that the agent behaves as it normally would, which is more appropriate to full agent testing.

The environment variables that have special values for attribute group testing are summarized in the following table. For more information about all agent environment variables, see ("List of environment variables" on page 19). For more information about setting environment variables, see ("Environment variables" on page 19).

Table 43. Environment variables

Environment variable	Default value (full agent test)	Attribute group test value	Reason for changed value for attribute group test
CDP_DP_INITIAL_COLLECTION_DELAY	varies	1	This value applies to an agent with a thread pool. This value is the time in seconds that the thread pool waits before the initial data collection request is sent to a data provider. **Note:** If CDP_DP_INITIAL_COLLECTION_DELAY is not set, the thread pool waits for a time that is specified by CDP_DP_REFRESH_INTERVAL or CDP_ATTRIBUTE_GROUP_REFRESH_INTERVAL. This wait time is the same time the thread pool waits between data collections, and might be too long to wait for the first data collection.

Table 43. Environment variables (continued)

Environment variable	Default value (full agent test)	Attribute group test value	Reason for changed value for attribute group test
CDP_DP_CACHE_TTL	55	1	When set to 1 a Collect Data request is much more likely to cause the data provider to collect data immediately. Otherwise it might return cached data that is up to 60 seconds old.

Chapter 16. Installing your agent into a monitoring infrastructure for testing and use

After you test your agent in Agent Builder, you can install the agent into an existing IBM Tivoli Monitoring, IBM Cloud Application Performance Management or IBM Cloud Pak for Multicloud Management environment for further testing and for use.

Installing and testing your agent in a monitoring infrastructure has the following benefits:

- You can configure and test multiple instances of an agent that run simultaneously.
- You can configure and test multiple instances of subnodes that run simultaneously.
- In a Tivoli Monitoring environment, you can build workspaces, situations, actions, and queries in the Tivoli Enterprise Portal.

Important: Deploy initial versions of your agent into a test version of the monitoring infrastructure. On Tivoli Monitoring, use a separate monitoring server and portal server. On Cloud APM, use a test cloud account or a separate test deployment of the on-premises monitoring server. Deploy the final version of your agent on a production infrastructure.

If you deploy a version of the agent on the production monitoring infrastructure and then change any data sets in the agent, the new version might conflict with the older version on the server. In this case it might be impossible to use any version of the agent.

Installing an agent

There are two methods for installing the agents that you create with Agent Builder.

1. To test your agent with a monitoring infrastructure that is running on the same system as the Agent Builder, you can install the agent into the local Tivoli Monitoring or Cloud APM installation.
2. To test or use the agent with a Tivoli Monitoring or Cloud APM system that is not running on the same system as the Agent Builder, you can generate a compressed file (*agent package*) that you can transfer to the other systems and deploy.

Note:

1. With Tivoli Monitoring, after you install an agent, you can see performance metrics in the Tivoli Enterprise Portal tables. For support of situations or workspaces, see Chapter 17, “Importing application support files,” on page 257.
2. With Tivoli Monitoring, after you install the agent, you can use the Tivoli Enterprise Portal to verify the data from the agent. For more information, see “Changes in the Tivoli Enterprise Portal” on page 250. If after you view the data in the Tivoli Enterprise Portal, you want to modify the agent, see Chapter 4, “Using the Agent Editor to modify the agent,” on page 17.
3. For an agent that supports Linux or UNIX, generate the installer image on a Linux or UNIX system because a Linux or UNIX system creates the files with the appropriate permissions.

Installing an agent locally

Install the agent into a monitoring environment on the local system where Agent Builder is running.

About this task

Complete the following steps to install your agent into a monitoring environment on the local system:

1. Click the `itm_toolkit_agent.xml` file from the Project Explorer navigation tree of Agent Builder by using one of the following methods:
   a. Right-click the `itm_toolkit_agent.xml` file and select **IBM** > **Generate Agent**.

© Copyright IBM Corp. 2010, 2021

b. Select the `itm_toolkit_agent.xml` file and select the **Generate Agent** icon on the toolbar.

c. Double-click the `itm_toolkit_agent.xml` file and select **Agent Editor** > **Generate Agent**.

2. In the **Generate Agent Wizard** window, in the **Install the Agent Locally** section, enter the installation directory for the monitoring infrastructure. The Agent Builder completes the value that is found in the `CANDLE_HOME` environment variable. If this variable is not set, the default value for Windows, `C:\IBM\ITM`, is displayed.

   The check boxes are enabled as follows:

   Install the agent
   Enabled if the Agent Builder detects an appropriate Tivoli Enterprise Monitoring Agent or a IBM Cloud APM agent in the specified location. An appropriate agent is one that supports the local operating system and is the correct minimum version.

   Install the TEMS support
   Enabled in a Tivoli Monitoring environment if the Agent Builder detects a Tivoli Enterprise Monitoring Server in the specified location.

   Install the TEPS support
   Enabled in a Tivoli Monitoring environment if the Agent Builder detects a Tivoli Enterprise Portal Server in the specified location.

3. Select the components to install (agent, Tivoli Enterprise Monitoring Server support, Tivoli Enterprise Portal Server support).

4. In a Tivoli Monitoring environment, if the Tivoli Enterprise Monitoring Server or Tivoli Enterprise Portal Server is installed on the local computer and you are installing the support files for these servers, you can choose whether to restart the servers.

   In this case, the **Restart TEMS without credentials** and **Restart TEPS** check boxes are active in the **Install the Agent Locally** section of the Generate Agent wizard. You can clear the check boxes to install the support without recycling the servers.

   When you clear the **Restart TEMS without credentials** check box, you are prompted for the Tivoli Enterprise Monitoring Server user ID and password. Enter these details and click **Logon**. If you are running Tivoli Monitoring with security off, enter "sysadmin" for the user ID, leave the password blank, and click **Logon**.

   Alternatively, to continue without entering credentials, click **Logon** without specifying a user ID and password or click **Cancel**. If you complete these steps, the Tivoli Enterprise Monitoring Server is recycled.

   Important: To install support files without recycling the Tivoli Enterprise Monitoring Server, ensure that the Tivoli Enterprise Monitoring Server is running.

5. Select the agent components to generate. You can select **Base Agent**, **Cognos Reporting**, or both.

6. In a IBM Cloud APM environment, you can provide security signing for self-describing agents. Click **Edit all jar signing preferences**. You can add a time stamp to signed JAR files and specify the time stamping authority. Specify details about your Java Keystore File.

   Note: You must create the Java Keystore File by using Java tools. For example, to generate a private key and certificate with a corresponding public key in a Java Keystore File, you can run this command:

   - *ab_install_path*`/jre/bin/keytool -genkeypair -keystore` *keystore_file_path* `-storepass` *key_store_password* `-alias` *key_store_alias* `-dname "CN=`*common_name*`, OU=`*organizational_unit*`, L=`*city_or_locality*`, ST=`*state_or_province*`, C=`*country*`" -keypass` *key_password*

   Where:

   - *ab_install_path* is the location where Agent Builder is installed
   - *keystore_file_path* is the path where an existing JKS keystore is located, or where one is created
   - *key_store_password* is the password that is needed to access any items in this keystore
   - *key_store_alias* is a name that identifies this key within the keystore (defaults to `"mykey"`)

- *key_password* the password that is needed to access this particular key (defaults to `key_store_password`)

The certificate must be included in the keystore for the server.

7. When you complete the **JAR Signing** details, click **OK**.
8. Click **Finish**.
9. Configure and start the agent. For more information, see “Configuring and starting the agent in an IBM Tivoli Monitoring environment” on page 243 or “Configuring the agent” on page 245 and “Starting and stopping the agent” on page 246 in a IBM Cloud APM environment.

   For Tivoli Monitoring v6.2 FP1 or later, you can install the Tivoli Enterprise Monitoring Server and Tivoli Enterprise Portal Server support without restarting the servers. In this case, the **Restart TEMS without credentials** and **Restart TEPS** check boxes are active in the **Install the Agent Locally** section of the Generate Agent wizard. You can clear the check boxes to install the support without recycling the servers. When you clear the **Restart TEMS without credentials** check box, you are prompted for the Tivoli Enterprise Monitoring Server user ID and password. Enter the Tivoli Enterprise Monitoring Server user ID and password and click **Logon**. If you are running Tivoli Monitoring with security off, enter "sysadmin" for the user ID, leave the password blank, and click **Logon**. You can also continue without entering credentials (click **Logon** without specifying a user ID and password or click **Cancel**. Doing so causes the Tivoli Enterprise Monitoring Server to be recycled).

   Note: The Tivoli Enterprise Monitoring Server must be running to install support files without recycling the Tivoli Enterprise Monitoring Server.

Creating the agent package

You can use Agent Builder to create a compressed agent installation package.

About this task

An agent package contains all the fines necessary to run the agent, as well as the installation and configuration scripts. The package also includes support files for the monitoring environment.

You can use an agent package to install the agent into the IBM Tivoli Monitoring and IBM Cloud Application Performance Management environments.

Procedure

1. Click the `itm_toolkit_agent.xml` file from the **Project Explorer** navigation tree of Agent Builder by using one of the following methods:
   - Right-click the `itm_toolkit_agent.xml` file and select **IBM** > **Generate Agent**.
   - Select the `itm_toolkit_agent.xml` file and select the **Generate Agent** icon on the toolbar.
   - Double-click the `itm_toolkit_agent.xml` file and select **Agent Editor** > **Generate Agent**.
2. Enter the name of the directory where you want to place the output (a compressed package or expanded files) in the **Generate Agent Image** section.
3. Select the **Keep intermediate files** check box to keep the generated expanded files separate from the zip or tar file.
4. Select the **Create a ZIP file** check box to create a compressed file in the specified directory. The compressed zip file is named `smai-`*`agent_name-version`*`.zip` for Windows systems by default.
5. Select the **Create a TAR file** check box to create a tar file in the specified directory. The compressed tar file is named `smai-`*`agent_name-version`*`.tgz` for UNIX and Linux systems by default.
6. Select the agent components to generate. You can select **Base Agent**, **Cognos Reporting**, or both.

   Important: For the IBM Cloud Application Performance Management environment, do not select **Cognos Reporting**, because the reports are currently not supported and including the reports increases the size of the package.

7. You can optionally provide security signing for agent application files. If you want to provide security signing, select **Sign self-describing support JAR**. Click **Edit all jar signing preferences**. You can add a timestamp to signed jar files and specify the time stamping authority. Specify details about your Java Keystore File.

   Important: You can create the Java Keystore File by using Java tools. For example, to generate a private key and certificate with a corresponding public key in a Java Keystore File, you can run this command:

   - `ab_install_path/jre/bin/keytool -genkeypair -keystore keystore_file_path -storepass key_store_password -alias key_store_alias -dname "CN=common_name, OU=organizational_unit, L=city_or_locality, ST=state_or_province, C=country" -keypass key_password`

   Where:

   - *ab_install_path* is the location where Agent Builder is installed
   - *keystore_file_path* is the path where an existing JKS key store resides, or where one will be created
   - *key_store_password* is the password needed to access any items in this key store
   - *key_store_alias* is a name identifying this key within the key store (defaults to `"mykey"`)
   - *key_password* is the password needed to access this particular key (defaults to `key_store_password`)

   Include this certificate in the server key store.
8. Click **Finish**.

Installing the package in an IBM Tivoli Monitoring environment

To test or use the agent in the IBM Tivoli Monitoring environment, use the generated package to install the agent on the monitored systems, hub Monitoring Server systems, and Portal Server system.

Before you begin

Before installing the agent on a monitored system, ensure that the Tivoli Monitoring operating system agent is present and working. For information about installing Tivoli Monitoring agents, see Installing monitoring agents in the Tivoli Monitoring Knowledge Center.

Important: To display agent information in the Tivoli Enterprise Portal, you must install the following components:

- The agent on all monitored systems
- Tivoli Enterprise Monitoring Server support files on the hub Tivoli Enterprise Monitoring Servers
- Tivoli Enterprise Portal Server support files on the Tivoli Enterprise Portal Server
- Tivoli Enterprise Portal support files on the Tivoli Enterprise Portal Server and, if applicable, any Tivoli Enterprise Portal desktop clients

Procedure

1. Copy the compressed file, which is named *product_code*`.zip` for Windows systems or *product_code*`.tgz` for UNIX and Linux systems by default, onto the system where you want to install the agent.
2. Extract the file to a temporary location.

   Note: Linux UNIX For UNIX and Linux systems, this temporary location must not be `/tmp/`*product_code*, where the product code is lowercase.

   You can install the agent remotely by using the compressed file.

- Linux On a Linux system, use the following command to extract your `.tgz` file:

  ```
  tar -xvzf filename
  ```

- UNIX On an AIX system, use the following command to extract your `.tgz` file:

  ```
  gunzip filename
  tar -xvf filename
  ```

3. Run the appropriate installation script.
   - To install the agent, Tivoli Enterprise Monitoring Server, Tivoli Enterprise Portal Server, and Tivoli Enterprise Portal support all at the same time:

     ```
     InstallIra.bat/.sh itm_install_location [[-h Hub_TEMS_hostname] -u
     HUB_TEMS_username -p Hub_TEMS_password]
     ```

   - To install the agent without installing support files:

     ```
     installIraAgent.bat/.sh itm_install_location
     ```

   - To install the Tivoli Enterprise Monitoring Server support:

     ```
     installIraAgentTEMS.bat/.sh itm_install_location [[-h Hub_TEMS_hostname] -u
     HUB_TEMS_username -p Hub_TEMS_password]
     ```

   - To install the Tivoli Enterprise Portal Server and Tivoli Enterprise Portal support:

     ```
     installIraAgentTEPS.bat/.sh itm_install_location
     ```

   The installation location, *itm_install_location*m must be the first argument and is mandatory on all scripts: `installIra.bat/.sh`, `installIraAgent.bat/.sh`, `installIraAgentTEMS.bat/.sh`, and `installIraAgentTEPS.bat/.sh`. This is the location where Tivoli Monitoring components are installed on this system.

   Other arguments are optional.

   If you install Monitoring Server support files and do not provide a user ID is not provided, the Tivoli Enterprise Monitoring Server is recycled.
4. Configure and start the agent, see “Configuring and starting the agent in an IBM Tivoli Monitoring environment” on page 243.

What to do next

If you changed the layout of your agent in a way that causes navigator items to be moved or removed, restart the Tivoli Enterprise Portal Server and Tivoli Enterprise Portal. The restart ensures that your changes are correctly recognized.

Configuring and starting the agent in an IBM Tivoli Monitoring environment

After installing an agent on a monitored system in the IBM Tivoli Monitoring, configure and start the agent.

Procedure

1. Open the **Manage Tivoli Monitoring Service**.

   The new entry **Monitoring Agent for *agent_name*** is displayed.
2. Right-click the entry and select **Configure Using Defaults**. Click **OK** to accept the defaults if you are prompted.

   Important:

   a. On UNIX systems, the option to select is **Configure**.

   b. For multi-instance agents, when you are configuring, you are prompted for an instance name.

Tip: If your agent uses a JMX data source to connect to a remote WebSphere Application Server, ensure that WebSphere Application Server is also installed on the host that is running the agent and set the `Java home` setting to the Java runtime environment that the local WebSphere Application Server uses.

Tip: For a Java API, JDBC, JMX, HTTP, or SOAP data source, you can use the **Java** > **JVM arguments** setting to control agent trace logging. Set the following value in this setting:

```
-DJAVA_TRACE_MAX_FILES=files -DJAVA_TRACE_MAX_FILE_SIZE=size
```

where *files* is the maximum number of trace log files that are kept (the default value is 4) and *size* is the maximum log file size in kilobytes (the default value is 5000). For example, you can set the following value:

```
-DJAVA_TRACE_MAX_FILES=7 -DJAVA_TRACE_MAX_FILE_SIZE=100
```

In this case, the agent writes 100 kilobytes into the first log file, then switches to the second log file, and so on. After writing seven log files of 100 kilobytes each, it overwrites the first log file.

If you added runtime configuration elements to your agent, or if you selected a data source, then you are presented with configuration panels. You use these panels to collect the required information for your agent.

3. Right-click the agent entry and select **Start**
4. Open the Tivoli Enterprise Portal and go to the new agent.

Installing and using an agent in an IBM Cloud Application Performance Management environment

To test or use the agent in the IBM Cloud Application Performance Management environment, use the generated package to install the agent on all monitored systems. In some cases, you need to configure the agent before it can be started. You can start and stop the agent as necessary.

Installing the agent

Use the installation package prepared by Agent Builder to install the agent on all monitored systems.

Before you begin

Ensure that an agent for IBM Cloud Application Performance Management, usually the operating system agent, is already present on the monitored system and working.

Windows On Windows systems, use an Administrator command line shell to install and configure agents. To start an Administrator shell, select **Command Prompt** from the Windows Programs menu, right-click, and click **Run as Administrator**.

Procedure

1. Extract the package to a temporary directory and change to this directory.
2. Install the agent by using the following command, depending on your operating system:
   - Windows On Windows systems, `installIraAgent.bat` *agent_install_location*
   - Linux UNIX On Linux and UNIX systems, `./installIraAgent.sh` *agent_install_location*

   Where *agent_install_location* is the installation location of the existing agent. The default location is:
   - Windows On Windows systems, `C:\IBM\APM`
   - Linux On Linux systems, `/opt/ibm/apm/agent`
   - AIX On AIX systems, `/opt/ibm/apm/agent`

Important: If you have added any custom configuration properties in the **Runtime Configuration** window of the Agent Editor, if the agent supports multiple instances, or if the agent uses any predefined data source that needs configuration (for example, a user ID and password), you must configure the agent before it can start. If an agent does not require configuration, it starts automatically after installation.

Configuring the agent

If you have added any custom configuration properties in the Runtime Configuration window of the Agent Editor, if the agent supports multiple instances, or if the agent uses any predefined data source that needs configuration (for example, a user ID and password), you must configure the agent before it can start.

Before you begin

Windows On Windows systems, use an Administrator command line shell to install and configure agents. To start an Administrator shell, select **Command Prompt** from the Windows Programs menu, right-click, and click **Run as Administrator**.

About this task

In the configuration process, you can:

- Set the instance name to create or change an instance, if the agent supports multiple instances.
- Set any configuration properties that are available for the agent.
- Create and configure subnodes, if the agent supports subnodes.

Windows On Windows systems, to set any configuration properties or create any subnodes, you must use the silent configuration procedure. A sample silent configuration response file is located in the *install_dir*\samples directory and is named *agentname*_silent_config.txt. Create a copy of this file and set the configuration variables as necessary.

Linux UNIX On Linux and UNIX systems, you can optionally use the silent configuration procedure. Alternatively, you can use the interactive procedure. If you start the configuration command without a response file name, the configuration utility prompts you for the configuration values.

Procedure

1. Change to the *install_dir*/bin directory.
2. Run the following command to configure the agent:
   - If the agent does not support multiple instances:
     - Windows On Windows systems, *name*-agent.bat config [*response_file*]
     - Linux UNIX On Linux and UNIX systems, ./*name*-agent.sh config [*response_file*]
   - If the agent supports multiple instances:
     - Windows On Windows systems, *name*-agent.bat config *instance_name* [*response_file*]
     - Linux UNIX On Linux and UNIX systems, ./*name*-agent.sh config *instance_name* [*response_file*]

   Where:

   - *instance_name* is the name of the instance. If an instance with this name does not exist, the instance is created. If the instance already exists, it is reconfigured. You must create at least one instance to use the agent.
   - *response_file* is the name of the silent configuration response file.

Tip: If your agent uses a JMX data source to connect to a remote WebSphere Application Server, ensure that WebSphere Application Server is also installed on the host that is running the agent and set the `Java home` setting to the Java runtime environment that the local WebSphere Application Server uses.

Tip: For a Java API, JDBC, JMX, HTTP, or SOAP data source, you can use the **Java** > **JVM arguments** setting to control agent trace logging. Set the following value in this setting:

```
-DJAVA_TRACE_MAX_FILES=files -DJAVA_TRACE_MAX_FILE_SIZE=size
```

where *files* is the maximum number of trace log files that are kept (the default value is 4) and *size* is the maximum log file size in kilobytes (the default value is 5000). For example, you can set the following value:

```
-DJAVA_TRACE_MAX_FILES=7 -DJAVA_TRACE_MAX_FILE_SIZE=100
```

In this case, the agent writes 100 kilobytes into the first log file, then switches to the second log file, and so on. After writing seven log files of 100 kilobytes each, it overwrites the first log file.

Starting and stopping the agent

To monitor a system, ensure that the agent is started on the system. You can start and stop the agent at any time. If the agent supports multiple instances, you can start and stop every instance independently.

Procedure

1. Change to the *install_dir*`/bin` directory.
2. Run the following command to start the agent:
   - If the agent does not support multiple instances:
     - Windows On Windows systems, *name*`-agent.bat start`
     - Linux UNIX On Linux and UNIX systems, `./`*name*`-agent.sh start`
   - If the agent supports multiple instances:
     - Windows On Windows systems, *name*`-agent.bat start` *instance_name*
     - Linux UNIX On Linux and UNIX systems, `./`*name*`-agent.sh start` *instance_name*
3. Run the following command to stop the agent:
   - If the agent does not support multiple instances:
     - Windows On Windows systems, *name*`-agent.bat stop`
     - Linux UNIX On Linux and UNIX systems, `./`*name*`-agent.sh stop`
   - If the agent supports multiple instances:
     - Windows On Windows systems, *name*`-agent.bat stop` *instance_name*
     - Linux UNIX On Linux and UNIX systems, `./`*name*`-agent.sh stop` *instance_name*

Agent post-generation and installation results

Installation of an Agent Builder agent creates and changes certain files on your system. In an IBM Tivoli Monitoring environment, you can also see changes in the Tivoli Enterprise Portal.

New files on your system

After you generate and install the agent that you created with Agent Builder, you can see the following new files on your agent system:

Note: *xx* denotes the two character product code.

Windows

Windows systems:

TMAITM6\k*xx*agent.exe
Agent binary

TMAITM6\K*xx*ENV
Environment variable settings

TMAITM6\K*xx*.ref
Agent provider configuration

TMAITM6\SQLLIB\k*xx*.his
SQL description of agent attribute information

TMAITM6\SQLLIB\k*xx*.atr
Agent attribute information

TMAITM6*xx*_dd_*version*.xmll
Product description

TMAITM6*xx*_dd.properties
Product name

TMAITM6\kxxcma.ini
Agent service definition file

TMAITM6*your files*
Supplemental files included from the Java API or Socket data sources with a file type of *executable* or *library*. Scripts included from the Script or Command return code data sources.

Linux UNIX

UNIX/Linux systems:

registry/*xx*architecture*.ver
Internal versions and prerequisites file

***architecture*/*xx*/bin/*xx*_dd_*version*.xml**
Product description

***architecture*/*xx*/bin/k*xx*agent**
Agent binary

***architecture*/*xx*/bin/*xx*_dd.properties**
Product name

***architecture*/*xx*/work/k*xx*.ref**
Agent provider configuration

***architecture*/*xx*/tables/ATTRLIB/k*xx*.atr**
Agent attribute information

***architecture*/*xx*/hist/k*xx*.his**
SQL description of agent attribute information

architecture*/*xx*/bin/*your files
Supplemental files included from the Java API or Socket data sources with a file type of *executable*. Scripts included from the Script or Command return code data sources.

architecture*/*xx*/lib/*your files
Supplemental files included from the Java API or Socket data sources with a file type of Library.

config/.*xx*.rc
Internal setup file

config/*xx*.environment
Environment settings

config/*xx*_dd_*version*.xml
Product description

config/*xx*_dd.properties
Product name

config/.ConfigData/k*xx*env
Environment variable settings

Note: Run the following command to find out the architecture of the system:

```
cinfo -p xx
```

where *xx* is the two-character product code.

For example, for a Solaris 8 64-bit system that is running an agent with product code 19, here is the output:

```
# /opt/ibm/apm/agent/bin/cinfo -p 19

*********** Fri Aug 17 11:23:58 EDT 2007 ******************
User : root Group: other
Host name : guadalajara  Installer Lvl:06.20.00.00
CandleHome: /opt/IBM/ITM
***********************************************************
Platform codes:
sol286 : Current machine
sol286 : Product (19)
tmaitm6/sol286 : CT Framework (ax)
```

The line in bold is the relevant one. The string before the colon, `sol286`, indicates the architecture in use for this agent. This string is different for different combinations of operating system and computer hardware type. The agent must be previously installed for this feature to work.

The following files are for Java-based data sources. These files are created only if the agent contains JMX, JDBC, HTTP, or SOAP data sources:

- `cpci.jar`
- `jlog.jar`
- `common/jatlib-1.0.jar`

The following files are for JMX runtime support. These files are created only if the agent contains JMX data sources:

- `common/jmx-1.0.jar`
- `common/connectors/jboss/connJboss-1.0.jar`
- `common/connectors/jsr160/connJSR160-1.0.jar`
- `common/connectors/was/connWas-1.0.jar`
- `common/connectors/weblogic/connWeblogic-1.0.jar`

The following file is for JDBC runtime support. These files are created only if the agent contains JDBC data sources:

- `common/jdbc-1.0.jar`

The following file is for HTTP or SOAP runtime support. These files are created only if the agent contains HTTP or SOAP data sources:

- `http-1.0.jar`

The following files are for the Java API runtime support. These files are created only if the agent contains a Java API data source:

- `cpci.jar`
- `custom/`*`your JAR file`* The name of this JAR file is specified in the **Global settings** of a Java API data source.
- `custom/`*`your JAR file`* Supplemental files with a file type of Java resource.

The same files exist on Windows, UNIX, and Linux systems for Java-based data sources, but they are in different directories:

- Windows Windows path: `TMAITM6\kxx\jars`
- Linux UNIX UNIX/Linux path: *`architecture`*`/`*`xx`*`/jars`

The following files are for log file monitoring runtime support. These files are created only if the agent contains log file data sources:

- Windows On Windows systems: TMAITM6\kxxudp.dll
- Linux On Solaris/Linux systems: *architecture*/*xx*/lib/libkxxudp.so
- On HP-UX systems: *architecture*/*xx*/lib/libkxxudp.sl
- UNIX On AIX systems: *architecture*/*xx*/lib/libkxxudp.a

The following files are for SSH script monitoring runtime support. These files are created only if the agent contains a script data source that is enabled for SSH collection:

- Windows On Windows systems: `TMAITM6\kxxssh.dll`
- Linux On Solaris/Linux systems: *`architecture`*`/`*`xx`*`/lib/libkxxssh.so`
- On HP-UX systems: *`architecture`*`/`*`xx`*`/lib/libkxxssh.sl`
- UNIX On AIX systems: *`architecture`*`/`*`xx`*`/lib/libkxxssh.a`

Changes in the Manage Tivoli Enterprise Monitoring Services window

After installing an agent in a IBM Tivoli Monitoring environment, you can see an entry for the agent in the **Manage Tivoli Enterprise Monitoring Services** window. The entry name is **Monitoring Agent for *agent_name***.

Important: Manage Tivoli Enterprise Monitoring Services is not supported in the IBM Cloud Application Performance Management environment.

Windows On Windows systems, this entry contains a **Task/Subsystem** column that identifies whether your agent supports multiple instances:

- A single instance agent displays a new application in the **Manage Tivoli Enterprise Monitoring Services** window. The name of the application is **Monitoring Agent for *agent_name***. A service is created for the agent (Figure 40 on page 250). The **Task/Subsystem** column contains the value **Primary**.
- A multiple instance agent displays a new application template in the **Manage Tivoli Enterprise Monitoring Services** window. The name of the template is **Monitoring Agent for *agent_name***. A service is not created for the agent until you create an instance of the agent from this template. The **Task/Subsystem** column contains the value **Template** to indicate that this entry is a template that is used to create instances of the agent.

Linux UNIX On Linux and UNIX systems, the entry for the agent is the same whether your agent supports multiple instances or not.

Note: The following screens are for a Windows system. UNIX and Linux systems have similar screens.

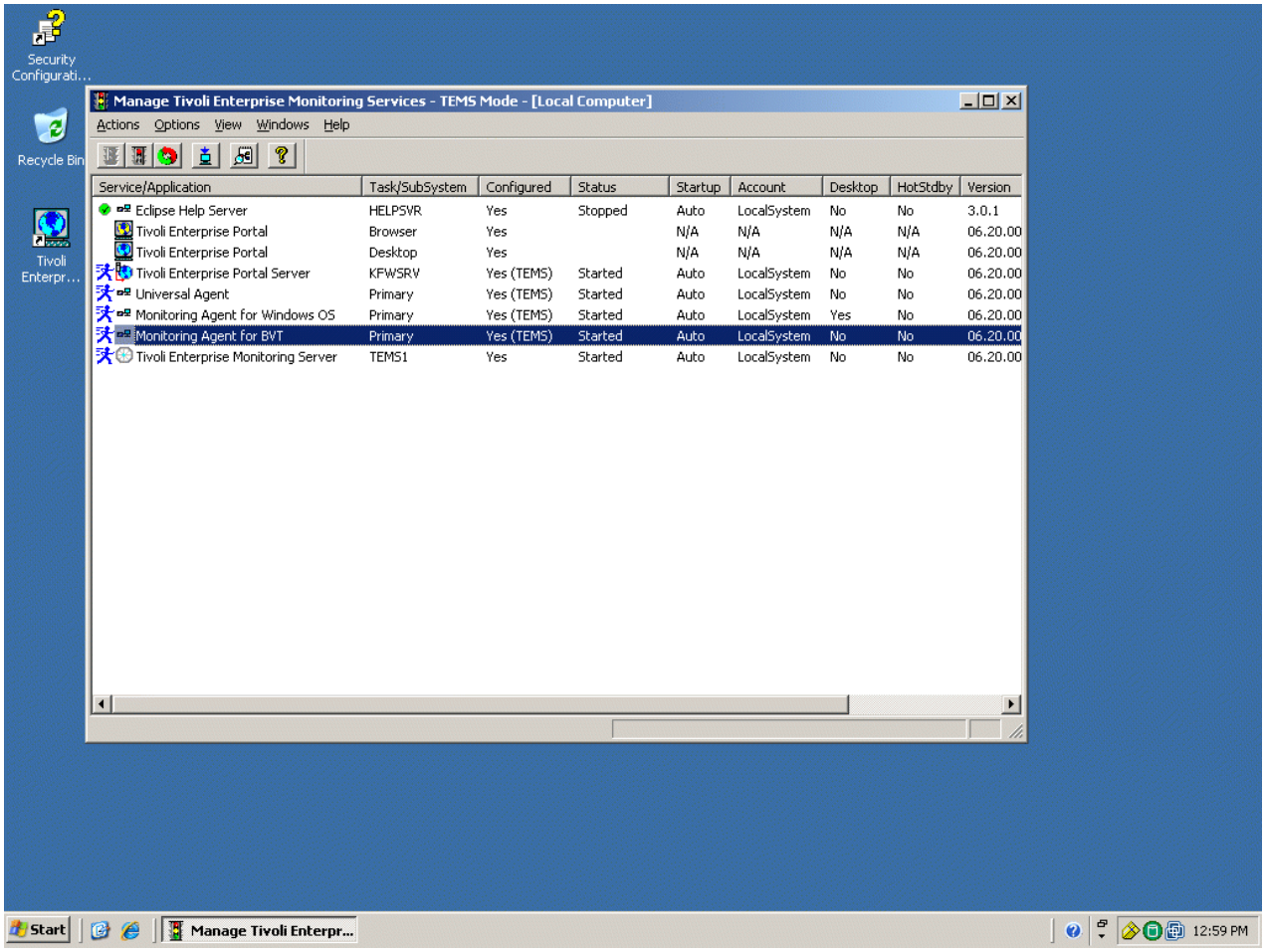


Figure 40. Manage Tivoli Enterprise Monitoring Services window

Changes in the Tivoli Enterprise Portal

In an IBM Tivoli Monitoring environment, after you install and start the agent, click the green **Refresh** icon in Tivoli Enterprise Portal. Then you can view the new agent. You can see the following changes in the portal:

- A new subnode for the agent in the Tivoli Enterprise Portal physical view.
- Nodes for every navigator group and data source that you defined by using the Agent Builder (Figure 41 on page 251).

  Note: For each navigator item, you must define a default query.

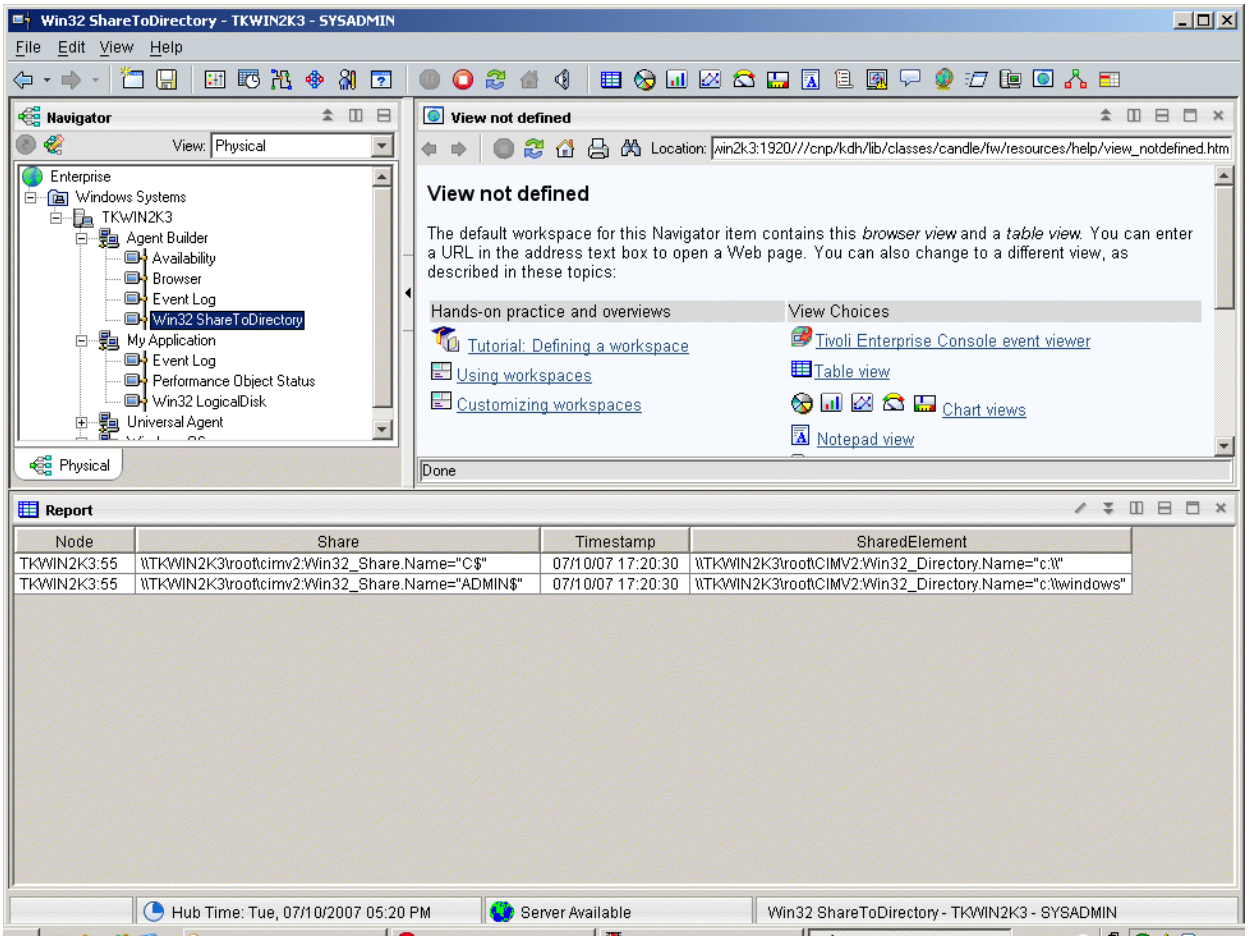


Figure 41. Nodes for attribute groups in the new agent.

- If your agent contains subnodes, an expandable node is present for each subnode that is defined in your agent. The following nodes are shown under the expandable node:
  - xxx performance object status, where xxx is the three-letter subnode type
  - Nodes for every Navigator group and data source that you defined in the subnode
  - xxx event log node if you have event logs
  - xxx JMX monitors node if you have JMX and you included JMX monitors
- The following automatic node:
  - An availability node if your agent contains an availability data source (Figure 42 on page 252)

    Note: This node behaves differently depending on the contents of the agent. If the agent monitors only availability, the availability node represents the availability data source. If the agent monitors availability and performance, the availability node becomes the navigator item that represents the availability and performance object status data sources.

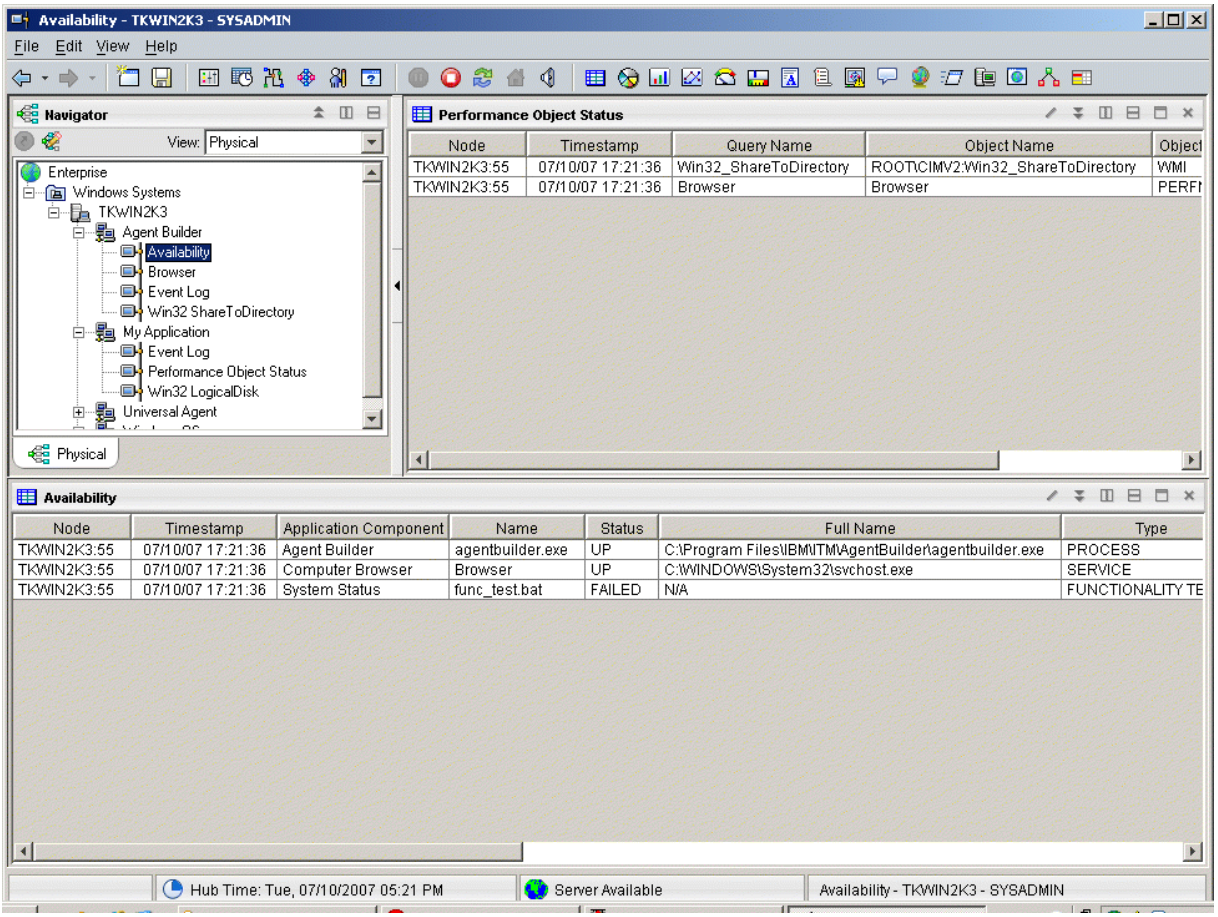


Figure 42. Availability node

– Performance Object Status, if the agent includes performance monitoring (not availability) data sources (Figure 43 on page 253)

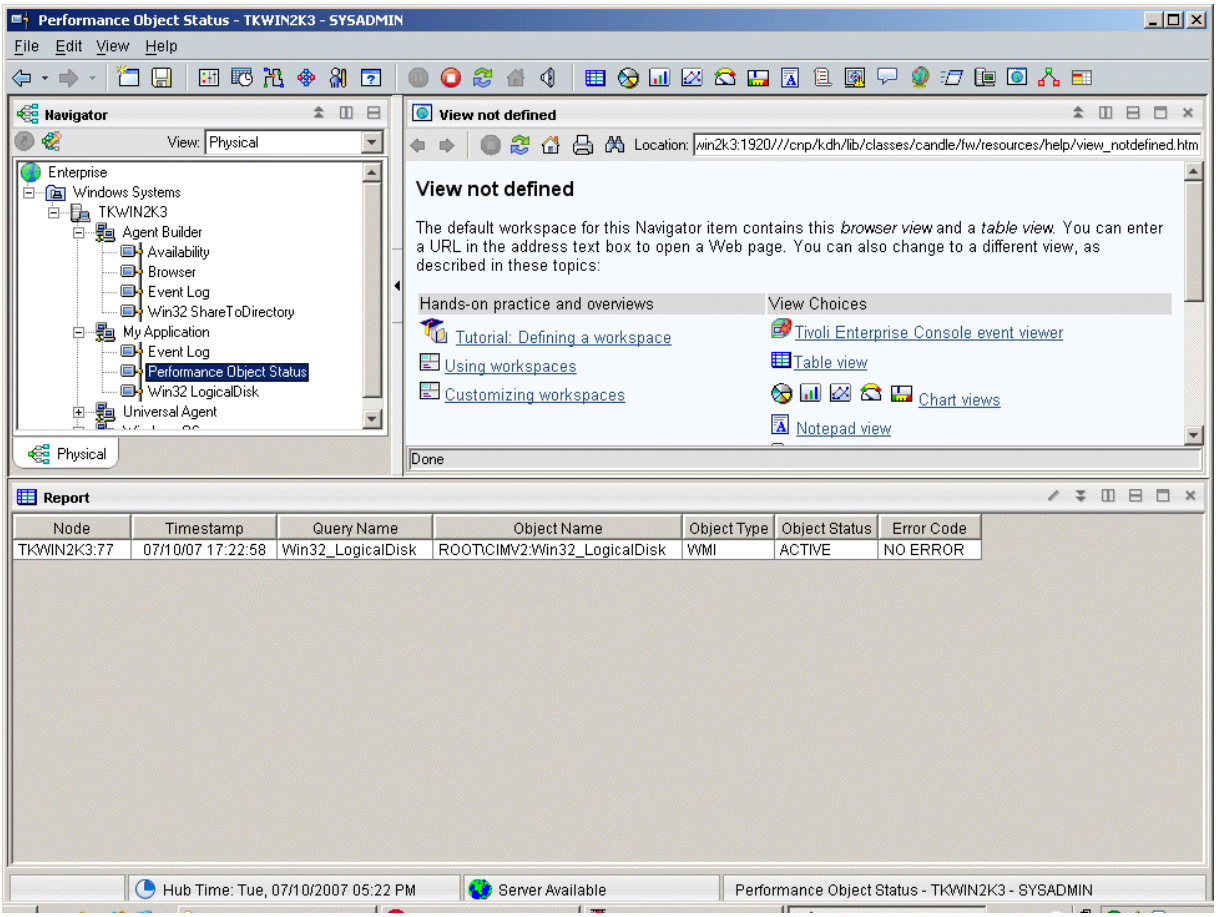


Figure 43. Performance Object Status node

– Event log, if the agent contains data sources producing log data (Figure 44 on page 254)

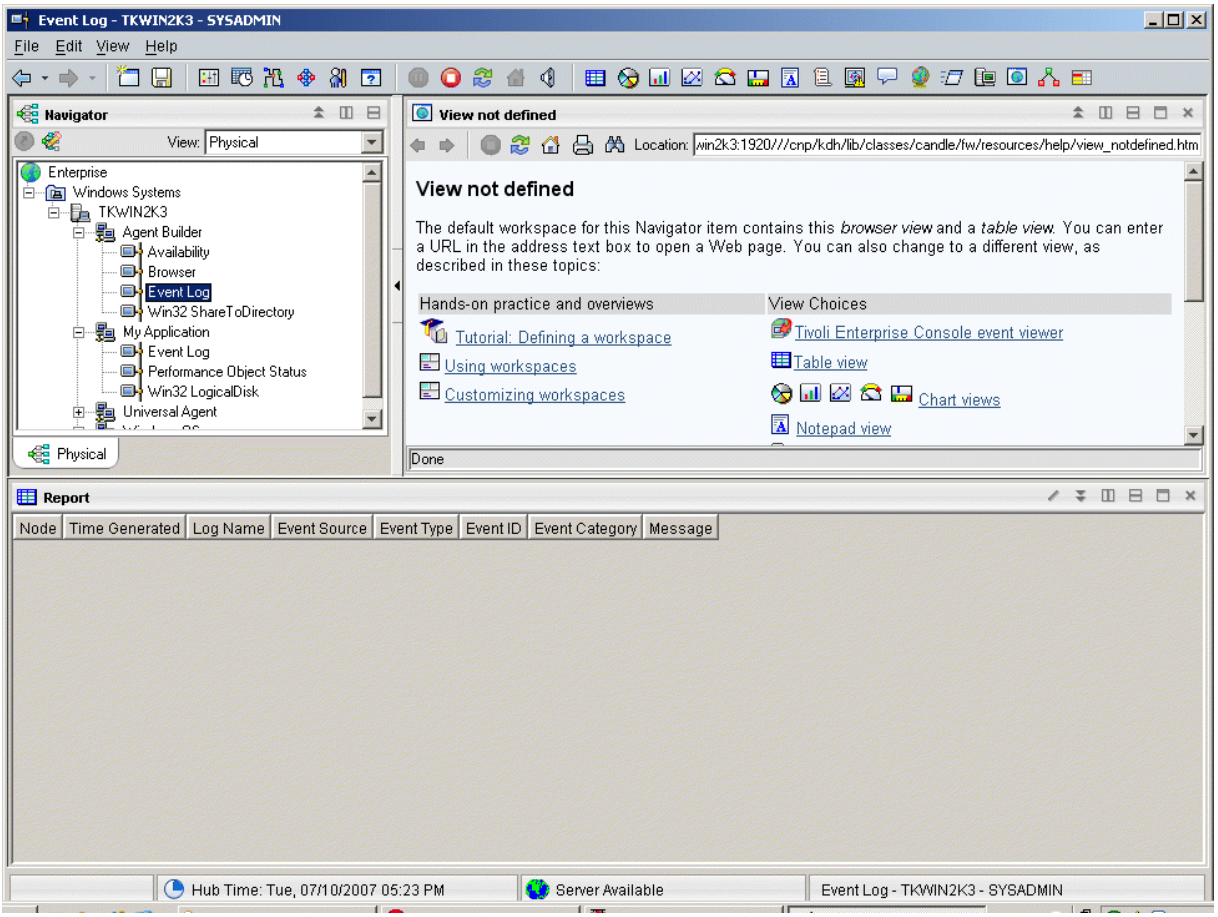


Figure 44. Event log node

See Appendix C, “Attributes reference,” on page 275 for descriptions of the attribute groups and attributes for Agent Builder.

Uninstalling an agent

You can remove an agent that the Agent Builder generated from a monitored host.

About this task

The uninstallation process uninstalls only the agent from the agent system. This process does not uninstall any other agent or any monitoring infrastructure.

In an IBM Tivoli Monitoring environment, you can use one of the following procedures to remove an agent that the Agent Builder generated:

- “Removing a Tivoli Monitoring agent by using the Tivoli Enterprise Portal” on page 255
- “Removing a Tivoli Monitoring agent without using the Tivoli Enterprise Portal” on page 255

After removing the agent using any of these procedures, clear it from the Tivoli Enterprise Portal using the following procedure: “Clearing a Tivoli Monitoring agent from the Tivoli Enterprise Portal” on page 255.

In an IBM Cloud Application Performance Management environment, use the following procedure: “Uninstalling an IBM Cloud Application Performance Management agent” on page 256.

Removing a Tivoli Monitoring agent by using the Tivoli Enterprise Portal

In an IBM Tivoli Monitoring environment, you can use the Tivoli Enterprise Portal to remove an agent.

Before you begin

Your operating system agent must be running in order to remove your created agent.

Procedure

To use the Tivoli Enterprise Portal to remove an agent, complete the following step:

- In the Tivoli Enterprise Portal navigation tree, right-click the agent and select **Remove**.

Removing a Tivoli Monitoring agent without using the Tivoli Enterprise Portal

If a Tivoli Enterprise Portal is not available in your IBM Tivoli Monitoring environment, you can use operating system scripts and commands to remove an agent.

Procedure

To remove an agent that the Agent Builder generated from the target system without using a Tivoli Enterprise Portal, you can complete any of the following steps:

- Windows

  On Windows systems, use the commands:

  ```
  cd ITM_INSTALL/TMAITM6
  kxx_uninstall.vbs ITM_INSTALL
  ```

  where *xx* is the product code for the agent

- Windows

  Alternatively. on Windows systems, you can use the `cscript.exe` command to run the uninstallation script. This command is the command-line interface parser for vbs scripts and does not display a window; instead, a message is displayed on the console:

  ```
  cd ITM_INSTALL/TMAITM6
  cscript.exe kxx_uninstall.vbs ITM_INSTALL
  ```

- Linux UNIX

  On Linux or UNIX systems. use the `uninstall.sh` file that is found in *ITM_INSTALL*`/bin`:

  ```
  uninstall.sh [-f] [-i] [-h ITM_INSTALL] [product platformCode]
  ```

Clearing a Tivoli Monitoring agent from the Tivoli Enterprise Portal

In an IBM Tivoli Monitoring environment, after you remove the agent, empty fields for information from the agent can remain in the Tivoli Enterprise Portal. To remove the fields, clear the agent from the Tivoli Enterprise Portal.

Procedure

1. Ensure that your Tivoli Enterprise Monitoring Server and Tivoli Enterprise Portal Server are up and running.
2. Log on to your Tivoli Enterprise Portal client.
3. From the Tivoli Enterprise Portal client Physical Navigator view, right-click **Enterprise** and select **Workspace** > **Managed System Status**.

   The Managed System Status workspace is displayed.

4. Select all of the IBM Tivoli Managed Systems for your agent.
5. Right-click and select **Clear off-line entry**, which clears all of the entries from that table.

Uninstalling an IBM Cloud Application Performance Management agent

You can uninstall your agent from any monitored system in an IBM Cloud Application Performance Management environment.

Procedure

1. On the system where the agent is installed, start a command line and change to the *install_dir*/bin directory, where *install_dir* the installation directory of the monitoring agents.
2. To uninstall a specific monitoring agent, enter the agent script name and the uninstall option where *name* is the agent script name:
   - On Windows systems, *name*-agent.bat uninstall
   - On Linux or AIX systems, ./*name*-agent.sh uninstall

Chapter 17. Importing application support files

If an agent is to be used in an IBM Tivoli Monitoring environment, custom situations, workspaces, Take Action commands, and queries can be included in the installation package.

About this task

To have a single installation image for situations, workspaces, and the agent, the situation, and workspace files must be in the same project as the agent. The Agent Builder provides a wizard to create the appropriate files in the agent project.

Definitions that are associated with an agent can also be included in the installation package. The content of these definitions is different for an agent that is used in an enterprise monitoring environment and in a system monitor environment. An enterprise monitoring agent image can include custom situations, workspaces, Take Action commands and queries. A system monitor agent image can include private situations, trap definitions, and agent configuration information.

To have a single installation package that includes the appropriate definitions and the agent itself, the files must be in the same project as the agent. The Agent Builder provides a wizard to create the appropriate files for an enterprise monitoring installation. The files for a system monitor agent environment are created by using the process that is described in the *Agent Autonomy* chapter in the *IBM Tivoli Monitoring Administrator's Guide*. The resulting files are copied into the root of the Eclipse project for the agent.

Exporting and importing files for Tivoli Enterprise Monitoring Agents

About this task

After you create situations, workspaces, queries, and Take Action commands in the Tivoli Enterprise Portal, you can export and import them into another Tivoli Monitoring Version 6.2 environment. For more information about creating situations and workspaces, see (Chapter 11, “Creating workspaces, Take Action commands, and situations,” on page 213). Use the following steps to extract the situations, workspaces, Take Action commands, and queries:

Procedure

1. From the **Project Explorer** tab, right-click the agent project folder.
2. Select **IBM Corporation** > **Import Application Support Files**.
3. Enter the host name of the Tivoli Enterprise Portal Server.
4. Enter the user name and password for the Tivoli Monitoring environment you are connecting to and click **Finish**.
5. If you defined situations for your agent, a dialog box is presented that lists the situations that are defined for the agent.
6. Select the situations that you want to export from the list and click **<<** to add them to the selected situations table and click **OK**.

   The import might take a few moments. When the task completes, you see the SQL files in the appropriate folders in the agent project.
7. If you defined Take Action commands for your agent, a dialog presents the Take Action commands defined. Choose the Take Action commands that you want to export from the list and click **>>** to add them to the Selected Take Actions table and click **OK**.

   The import might take a few moments. When the task completes, you see the SQL files in the appropriate folders in the agent project.

© Copyright IBM Corp. 2010, 2021

8. If you defined custom queries for your agent, a dialog presents the Queries defined. Select the queries that you want to export from the list and click **<<** to add them to the Selected Queries table and click **OK**.

   The import might take a few moments. When the task completes, you see the SQL files in the appropriate folders in the agent project. Workspaces are imported automatically.

What to do next

Re-create your custom agent, install your agent on the monitored host, and install the Tivoli Enterprise Portal support.

Exporting and importing files for Tivoli System Monitor Agents

About this task

The system monitor agent definitions are contained in three types of files:

- Private situations are defined in a file named *xx*`_situations.xml`, where *xx* is the two-character product code
- Trap configuration information is defined in a file named *xx*`_trapcnfg.xml`, where *xx* is the two-character product code
- For agents that require configuration, the configuration is defined in one file for each instance of the agent. When the agent is a single instance agent, the file is named *xx*`.cfg`. When the agent is a multi-instance agent, there is a file is present for each instance. The file names are *xx*`_`*instance name*`.cfg`, where *xx* is the two-character product code and *instance name* is the name of the agent instance.

Procedure

- Create the files by using the process that is described in the *Agent Autonomy* chapter in the *IBM Tivoli Monitoring Administrator's Guide*. Copy the files into the root of the project directory manually, or use the Eclipse import function to select the files to be imported: **File** > **Import** > **General** > **File System**.

  These files are included in the agent image and installed by the installer.

  When the agent is installed the installation:

  – Copies the included files into the appropriate locations.
  – Any private situations that are defined in the `pc_situations.xml` file that is run on the agent.
  – The trap definitions that are defined in the `pc_trapcnfg.xml` are used to forward traps that are based on the situations.
  – The agent is automatically configured and started if:
    - The agent is a single instance agent with no configuration defined as part of the agent.
    - The agent is a single instance agent with configuration defined as part of the agent and the image includes a `pc.cfg`file.
    - The agent is a multi-instance agent (all multi-instance agents require configuration): the installer starts one instance of the agent for each `pc_`*`inst`*`.cfg` file.

Chapter 18. Event filtering and summarization

An attribute group is defined to be *pure event* or *sampled*. Pure event attribute groups contain data rows that occur asynchronously. As each new row of data arrives, it is processed immediately by Tivoli Monitoring. Sampled attribute groups collect the current set of data rows each time the data is requested. The following attribute groups illustrate the difference:

- An `SNMPEvent` attribute group is created that represents all of the SNMP Traps and informs that are sent to the agent. Traps or informs arrive asynchronously as they are sent by the monitored systems. As each event arrives, it is passed to Tivoli Monitoring.
- A `Disk` attribute group is created to represent information about all of the disks on a system. The disk information is collected periodically. Each time disk information is collected, the agent returns a number of rows of data, one for each disk.

The difference between pure event and sampled attribute groups affects various aspects of Tivoli Monitoring. These aspects include: situations, warehouse data, and Tivoli Enterprise Portal views.

Each situation is assigned (or *distributed*) to one or more managed systems to be monitored for a specific condition of a set of conditions. When the determination of the event must be made based on observations that are made at specific intervals, the event is known as a *sampled event*. When the event is based on a spontaneous occurrence, the event is known as a *pure event*. Therefore, situations for sampled events have an interval that is associated with them, while situations for pure events do not. Another characteristic of sampled events is that the condition that caused the event can change, thus causing it to be no longer true. Pure events cannot change. Therefore, alerts that are raised for sampled events can change from true to false, while a pure event stays true when it occurs.

An example of a sampled event is `number of processes > 100`. An event becomes true when the number of processes exceeds 100 and later becomes false again when this count drops to 100 or less. A situation that monitors for `invalid logon attempt by user` is a pure event; the event occurs when an invalid logon attempt is detected, and does not become a False event. While you can create situations that are evaluated on a specific interval for sampled attribute groups, such evaluations are not possible for pure event attribute groups.

Similarly, for historical data, you can configure how frequently sampled data is collected. However, when you turn collection on for pure event data, you get each row as it happens.

The data that is displayed in the Tivoli Enterprise Portal for sampled data is the latest set of collected rows. The data that is displayed for pure event attribute groups is the contents of a local cache that is maintained by the agent. It does not necessarily match the data that is passed to Tivoli Monitoring for situation evaluation and historical collection.

Controlling duplicate events

Use the event filtering and summarization options to control how duplicate events are sent to Tivoli Monitoring.

Before you begin

For more information about event filtering and summarization, see Chapter 18, “Event filtering and summarization,” on page 259.

About this task

The Agent Builder defines attribute groups that represent event data as *pure event* in Tivoli Monitoring. These attribute groups include log file, AIX Binary Log, SNMP events, and JMX notifications. These attribute groups can produce multiple duplicate events. You can control how these duplicate events are sent to Tivoli Monitoring. You can activate these controls for log file, SNMP events, and JMX notifications

© Copyright IBM Corp. 2010, 2021

attribute groups in the **Event Information** tab under **Advanced Data Source Properties** in the **Advanced** window.

Whether an event is treated as a duplicate of other events is determined by the key attributes, you define in the attribute group. A duplicate event occurs when the values for all key attributes in the event match the values for the same key attributes in an existing event. When event filtering and summarization is enabled, the attributes for the `isSummary`, `occurrenceCount`, `summaryInterval`, and `eventThreshold` functions are added automatically.

Procedure

- In the **Event Filtering and Summarization Options** area, select one of the following options:
  - **No event filtering or summarization**: Sends all events without any event filtering or summarization. This option is the default option.
  - **Filter and summarize events**: Creates a summary record for each event with duplicates and each unique event that is based on the key attributes. Select also to choose the event filtering option. In the **Summarization Options** area, enter the summary interval. You can enter either a value in seconds or insert a configuration property.

    The event filtering options are:

    - **Only send summary events**: Sends only the summary records for the specified interval.
    - **Send all events**: Sends all events and summary records.
    - **Send first event**: For each event, sends only the first event that is received in the summary interval that is specified and no duplicate events. This option also sends the summary records.
    - **Event threshold**: Sends an event to Tivoli Monitoring when the number of duplicate events that are received in the interval is evenly divisible by the threshold. For example, if you set the event threshold to 5 and you receive less than five duplicates (including the first event) in the interval, no event is sent to Tivoli Monitoring. If you receive 5, 6, 7, 8, or 9 duplicates, one event is sent. If you receive 10 duplicates, 2 events are sent. In the **Event threshold** field, you can enter a number or insert a configuration property. This option also sends the summary records.

Viewing event filtering and summarization in the Tivoli Enterprise Portal

Examples of how data is treated depending on your event filtering and summarization choices.

The agent maintains a cache of the last events received. By default, this cache is 100 in size. If you enable agent event filtering and summarization, differences can occur between the number of events in the cache and the number sent to IBM Tivoli Monitoring. Additional events in the cache might not reach the designated threshold for sending. Or you might have fewer events in the cache if you selected the **Send all events** option. If the **Send all events** option is set, an event is sent each time a duplicate occurs. However, only one copy of the event is kept in the cache, and the occurrence count is incremented each time that the event occurs. To view the events that are sent to IBM Tivoli Monitoring, create a historical view. For information about creating historical views, see *Historical Reporting* in the Tivoli Enterprise Portal User's Guide. You can compare this view with the real-time cache view in the Tivoli Enterprise Portal. You can also use situations to make the same comparison.

The following examples indicate how the same log data is treated depending on your choice, if any, of event filtering and summarization. The example agent was created to illustrate different behaviors. Each attribute group was defined to monitor the same log file. In each example, a historical view and a real-time (cache) view is shown. The names of the nodes in the Tivoli Enterprise Portal reflect the settings selected. By default, the historical view displays the newest events last. The default real-time view of the cache displays the newest events first. In these examples, the historical view shows the last 1 hour.

As new events arrive, you can see them in the cache view. As duplicates of an event arrive, the data is updated in the existing row. When a summary interval elapses, the existing events are converted to summary events and sent. New rows are then added for the next summary interval.

(Figure 45 on page 261) shows the historical view and cache view if you did not enable event filtering or summarization. Both views display the same data, but in reverse order. To display the corresponding events, the historical view is scrolled down and the real time (cache) view is scrolled up.

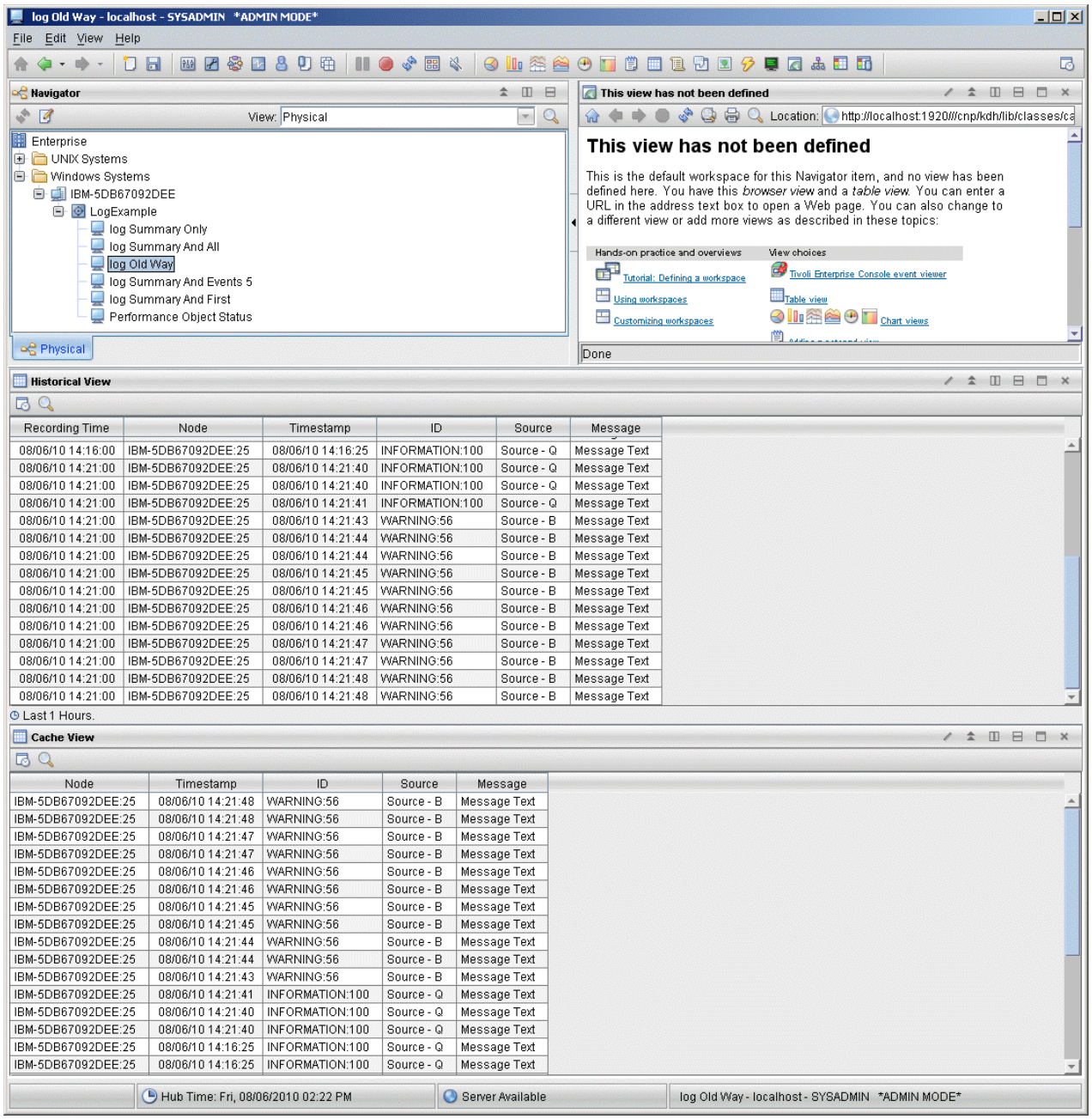


Figure 45. Historical view and cache view when event filtering or summarization is not enabled

(Figure 46 on page 262) shows the historical view and cache view if you selected the **Only send summary events** option in the **Event Information** tab. The summary events are displayed in both views, but the new events are only displayed in the real-time (cache) view.

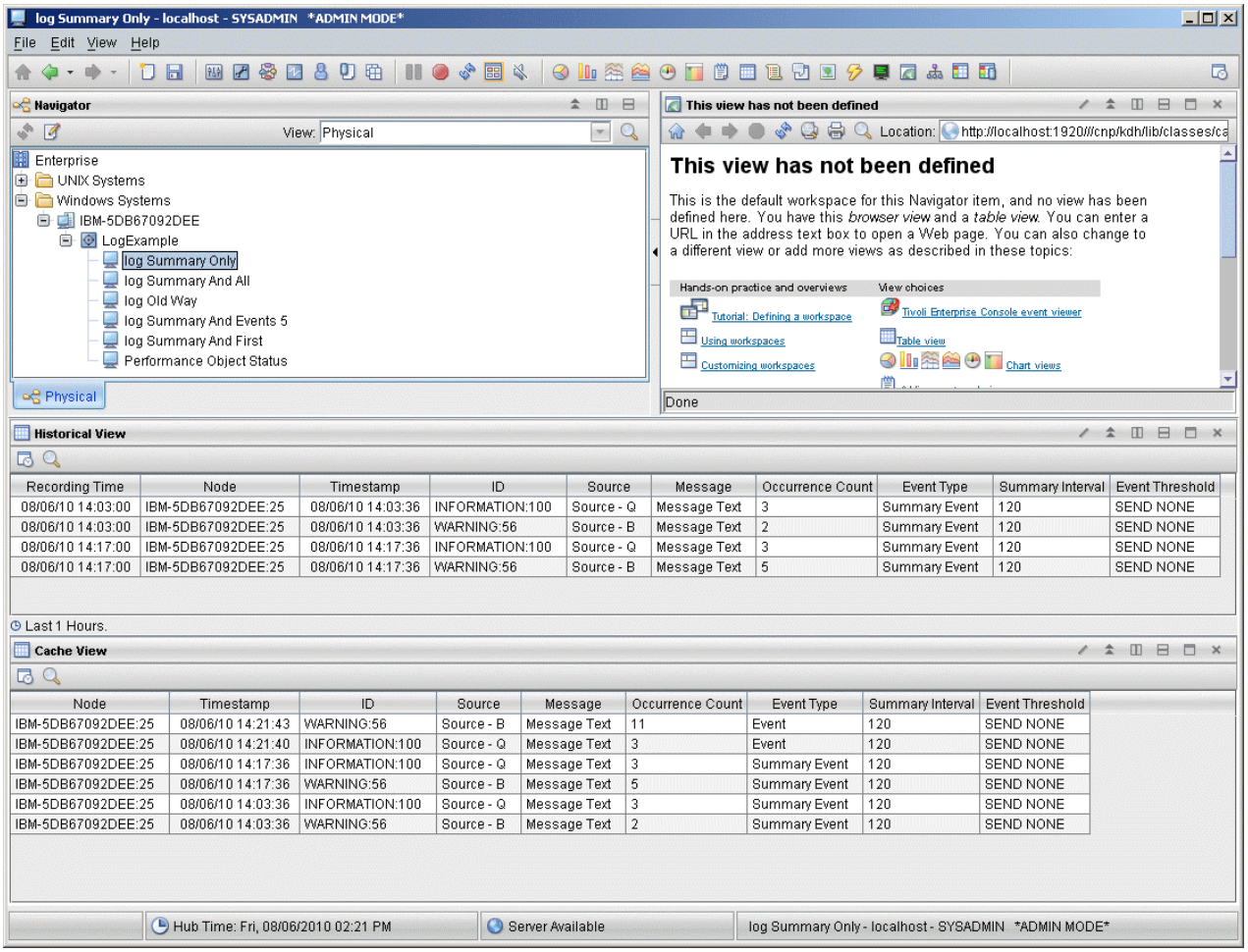


*Figure 46. Historical view and cache view when **Only send summary events** is selected*

(Figure 47 on page 263) shows the historical view and cache view if you selected the **Send all events** option in the **Event Information** tab. All of the events are shown in both views, but you also see the summary events that are created at the end of each interval. The real-time view changes when the interval elapses. The existing events are converted into summary records and then the new events are added. The addition of the other two available event attributes that are used to display the summary interval (120 seconds in this example) and the *SEND ALL* threshold.

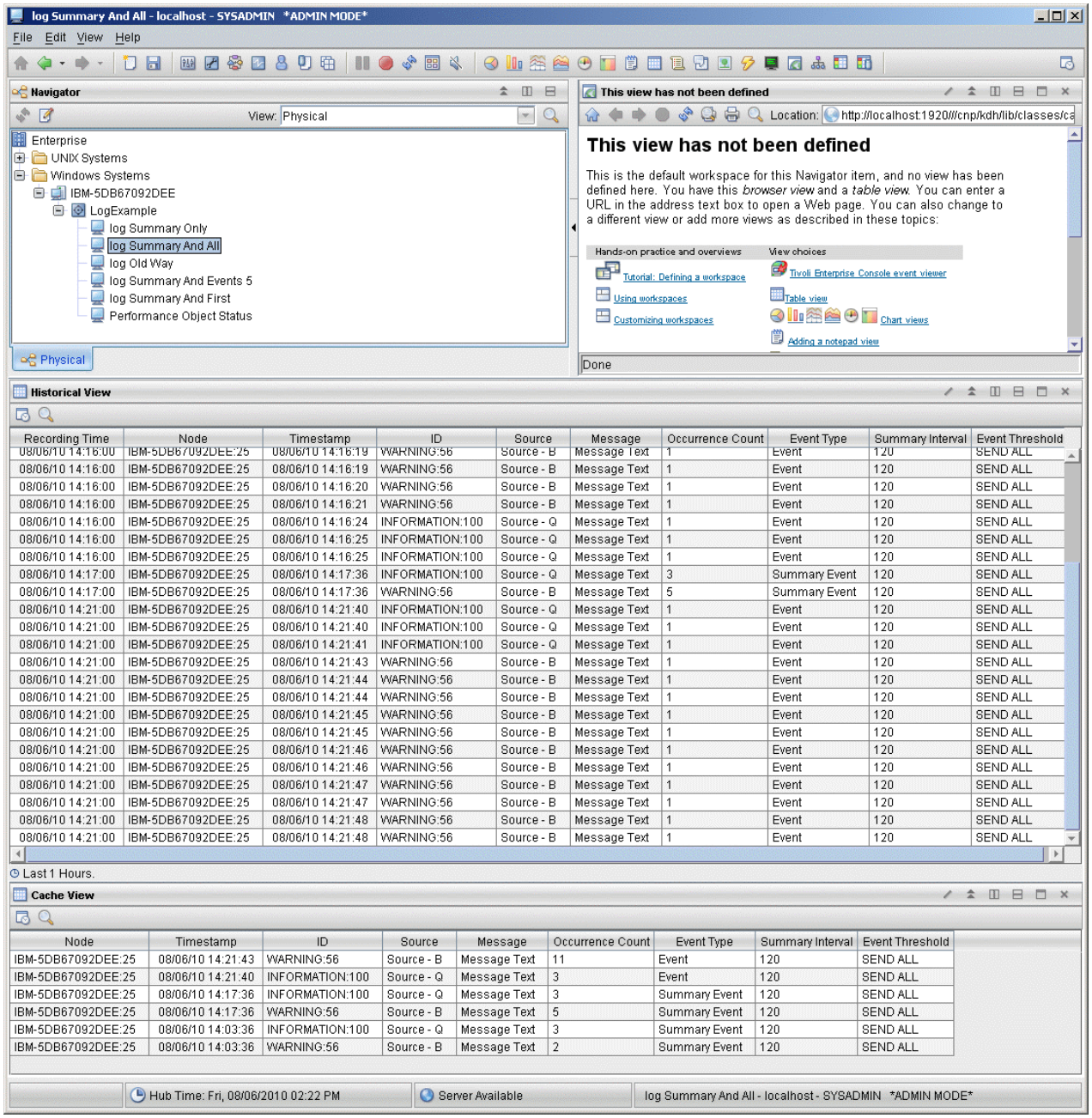


*Figure 47. Historical view and cache view when **Send all events** is selected*

(Figure 48 on page 264) shows the historical view and cache view if you selected the **Send first event** option in the **Event Information** tab. The summary events are displayed in both views, but all the new events are only displayed in the real-time (cache) view. For each event, the historical view displays only the first event that is received in the interval and no duplicate events.

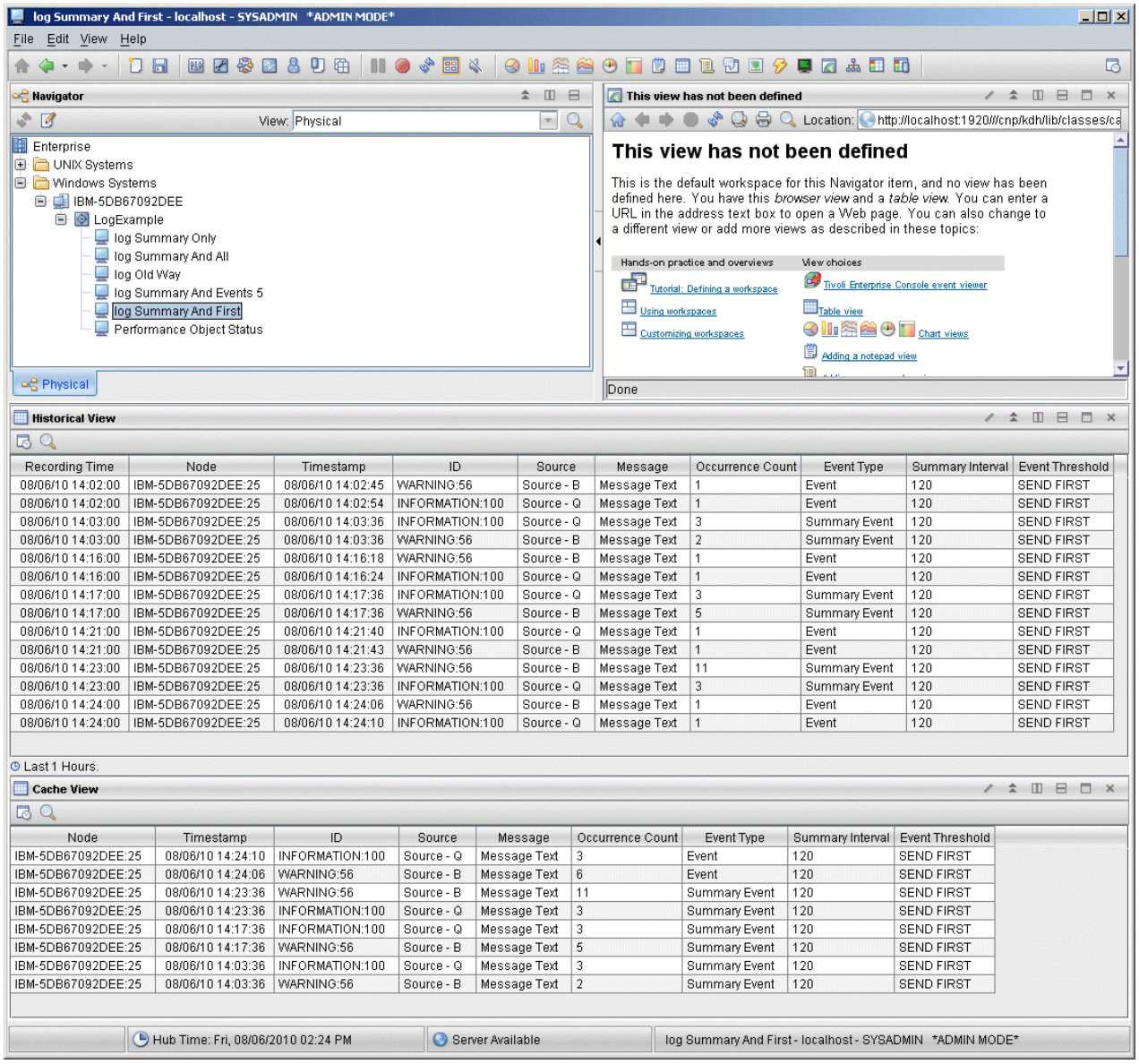


*Figure 48. Historical view and cache view when **Send first event** is selected*

(Figure 49 on page 265) shows the historical view and cache view if you selected the **Event threshold** option and entered a value of 5. The summary events are displayed in both views, but all the new events are only displayed in the real-time (cache) view. In this example, a threshold of 5 is specified. The historical view displays an event only when five duplicates of an event (including the first event) are received in the interval. If less than 5 are received, no event is displayed. If 6, 7, 8, or 9 duplicates are received in the interval, one event is displayed. If 10 duplicates are received, 2 events are displayed.

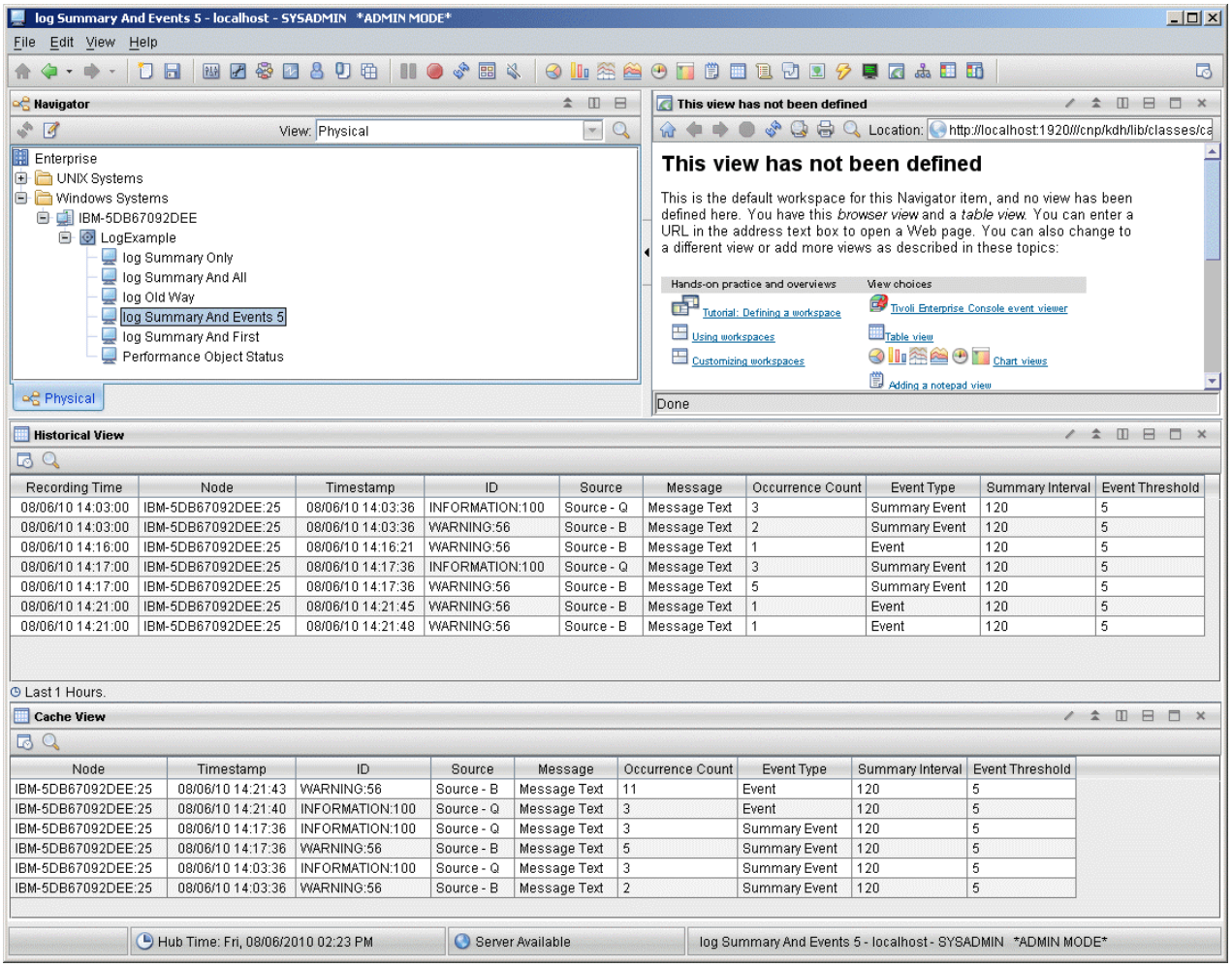


*Figure 49. Historical view and cache view when **Event threshold** is selected*

Related concepts

"Event filtering and summarization" on page 259

Chapter 19. Troubleshooting and support

Review the troubleshooting information for problems that you might experience with installing, configuring, or using IBM Agent Builder.

For help with troubleshooting issues while developing, installing, or using custom agents in the IBM Cloud Application Performance Management environment, see the IBM Cloud Application Performance Management Troubleshooting Guide.

For logging and message reference information and for help with troubleshooting issues related to the IBM Tivoli Monitoring environment, see the IBM Agent Builder Version 6.3.1 Troubleshooting Guide.

© Copyright IBM Corp. 2010, 2021

Appendix A. Sharing project files

Share an IBM Tivoli Monitoring agent project with someone.

Procedure

1. Obtain their files. You need the entire contents of the directory with the same name as the project in your workspace directory.
   For example, if your workspace directory is `c:\Documents and Settings\User1\workspace` and you want to share your project named `TestProject`. You must make the directory `c:\Documents and Settings\User1\workspace\TestProject` and all of its contents accessible to your system.
2. Select **File** > **Import**.
3. Open **IBM Tivoli Monitoring**.
4. Select **IBM Tivoli Monitoring Agent** and click **Next**.
5. Type the full path to the agent xml file or click **Browse** to browse to the file.
6. Click **Finish**.

Results

When the wizard completes, you see the new IBM Tivoli Monitoring agent project in your workspace.

Share a Solution Installer Project

Share a Solution Installer Project with someone

Procedure

1. Obtain their files. You must have the entire contents of the directory with the same name as the Solution Installer project in your workspace directory.
   For example, if your workspace directory is `c:\Documents and Settings\User1\workspace` and you want to share your Solution Installer project named `TestProject Installer`. You must make the directory `c:\Documents and Settings\User1\workspace\TestProject Installer` and all of its contents accessible to your system.
2. Click **File** > **Import**.
3. Open **General**.
4. Select **Existing Projects into Workspace**, and click **Next**.
5. Type the full path to the root directory of the Solution Installer project, or click **Browse** to browse to the root directory of the Solution Installer project. (In this example the `TestProject Installer` directory.) The Project in that directory is displayed in the Projects list and is selected by default.
6. Optional: Click **Copy projects into workspace**.
7. Click **Finish**.

© Copyright IBM Corp. 2010, 2021

Appendix B. Command-line options

Commands available from the Agent Builder command-line interface (CLI).

The Tivoli Monitoring Agent Builder contains a command-line interface (CLI) that you can use to generate the Tivoli Monitoring Agent without starting the Eclipse graphical user interface (GUI). You can generate the agent as part of a build, for example:

On Windows systems, you can use a batch file in the following directory to access the CLI:

```
install_location\agenttoolkit.bat
```

On UNIX and Linux systems, you can use a script in the following directory to access the CLI:

```
install_location/agenttoolkit.sh
```

The commands that are described in this documentation are formatted for Windows systems, which use a backslash (\) for directory paths.

For UNIX® or Linux® systems, use the same commands as for Windows systems, but with the following changes:

- Use a forward slash (/) for directory paths instead of a backslash (\).
- Use the `agenttoolkit.sh` script instead of the `agenttoolkit.bat` script.

Commands

Table 44 on page 271 lists the name and purpose statement for each command option for the text command:

Table 44. Command quick-reference table

Command	Purpose
generatelocal	Loads and validates the `itm_toolkit_agent.xml` file and generates the files that run the Tivoli Monitoring Agent. The installation is into a local Tivoli Monitoring environment.
generatemappingfile	Creates the mapping file for porting custom IBM Tivoli Monitoring v5.x resource models to IBM Tivoli Monitoring v6 agents.
generatezip	Generates a compressed file named *productcode*`.zip` or *productcode*`.tgz`.

The command descriptions that are referenced from the table describes how to run the commands by covering the following information:

Purpose
Lists the purpose of the command.

Format
Specifies the syntax that you type on the command line. The syntax contains the command name and a list of the parameters for the command. A definition of each parameter follows the command name.

Examples
The example for the command contains a brief description of the example and an example of the syntax.

Usage
Provides an explanation of the command and its purpose.

© Copyright IBM Corp. 2010, 2021

Comments
Provides commands or text that can give you more information.

Command - generatelocal

Use this command to load and validate XML and to generate files to run the Tivoli Monitoring Agent.

Purpose

Loads and validates the `itm_toolkit_agent.xml` file and generates the files for running the Tivoli Monitoring Agent. The installation is into a local Tivoli Monitoring environment.

Format

For Windows systems:

```
install_location\agenttoolkit.bat project_dir -generatelocal itm_install_dir
```

where:

install_location
Directory where the Agent Builder is installed

project_dir
Name of the directory that contains the `itm_toolkit_agent.xml` file

itm_install_dir
Location where Tivoli Monitoring is installed (for example `c:\IBM\ITM`)

Examples

In the following example for Windows, the agent definition in `C:\ABCAgent` is validated and the files that are required to run `ABCAgent` are generated in `C:\IBM\ITM`:

```
install_location\agenttoolkit.bat C:\ABCAgent -generatelocal C:\IBM\ITM
```

Command - generatemappingfile

Use this command to migrate custom IBM Tivoli Monitoring v5.x resource models to IBM Tivoli Monitoring v6 agents.

Purpose

This command creates the mapping file for migrating custom IBM Tivoli Monitoring v5.x resource models to IBM Tivoli Monitoring v6 agents.

Format

For Windows systems:

```
install_location\agenttoolkit.bat project_dir -generatemappingfile output_dir
   itm5_interp_list
```

Where:

install_location
Directory where the Agent Builder is installed

project_dir
Name of the directory that contains `itm_toolkit_agent.xml`

output_dir
Name of the directory where the mapping file is written

itm5_interp_list
Comma-separated list of the ITM 5x operating systems on which the custom resource model ran. The following values are allowed:

- `aix4-r1`
- `hpux10`
- `linux-ix86`
- `linux-ppc`
- `linux-s390`
- `os2-ix86`
- `os400`
- `solaris2`
- `solaris2-ix86`
- `w32-ix86`

Examples

For Windows systems

```
install_location\agenttoolkit.bat c:\ABCAgent -generatemappingfile c:\output
   linux-ix86,linux-ppc,linux-s390
```

Command - `generatezip`

Use this command to load and validate XML and to generate a compressed file that can be used to install the agent on another system.

Purpose

Loads and validates the `itm_toolkit_agent.xml` file and generates a compressed file named *`productcode`*`.zip` or *`productcode`*`.tgz`. The generated compressed file can be used to install the agent on another system. Depending on your environment, both file types can be generated.

Format

For Windows systems:

```
install_location\agenttoolkit.bat project_dir -generatezip output_dir
```

Where:

project_dir
Name of a directory that contains the `itm_toolkit_agent.xml` file

output_dir
Name of a directory where the compressed file is written

Examples

In the following example for Windows, the agent definition in `C:\ABCAgent` is validated and a compressed file that contains the required files for running `ABCAgent` is generated in `C:\Output`:

```
install_location\agenttoolkit.bat\ C:\ABCAgent -generatezip C:\Output
```

Appendix C. Attributes reference

Contains descriptions of the attributes for each attribute generated group included in the Agent Builder.

Availability node

The Availability attribute group contains availability data for the application.

The table provides a common format for representing application availability, which includes relevant information for three aspects of an application: services (Windows only), processes, and command return codes.

The following list contains information about each attribute in the Availability attribute group:

Node attribute - This attribute is a key attribute

Description
The managed system name of the agent

Type
String

Names

Attribute name
Node

Column name
ORIGINNODE

Timestamp attribute

Description
The local time at the agent when the data was collected

Type
Time

Names

Attribute name
Timestamp

Column name
TIMESTAMP

Application Component attribute - This attribute is a key attribute

Description
The descriptive name of a part of the application

Type
String

Names

Attribute name
Application_Component

Column name
COMPONENT

Name attribute

© Copyright IBM Corp. 2010, 2021

Description
The name of the process, service, or functional test. This name matches the executable name of the process, the service short name, or the name of the process that is used to test the application.

Type
String

Names

Attribute name
Name

Column name
NAME

Status attribute

Description
The status of the application component.

- For processes, the values are `UP`, `DOWN`, `WARNING`, or `PROCESS_DATA_NOT_AVAILABLE`. `PROCESS_DATA_NOT_AVAILABLE` is displayed for a process when the matching process is running but the resource use information cannot be collected for that process.
- For services, the values are `UP`, `DOWN`, or `UNKNOWN`. `UNKNOWN` is displayed when the service is not installed.
- For command return codes, the values are `PASSED` or `FAILED`.

Type
String

Names

Attribute name
Status

Column name
STATUS

Full Name attribute

Description
The full name of the process which includes information that is process-dependent. The name might include the full path if the process was started that way. The name can also include a partial path or even a path that is changed by the process.

Type
String

Names

Attribute name
Full_Name

Column name
FULLNAME

Type attribute

Description
Identifies the type of the application component. Components are processes, services, or command return codes.

Type
Integer (gauge)

Names

Attribute name
Type

Column name
TYPE

Virtual Size attribute

Description
The virtual size (in MB) of the process

Type
Integer (gauge)

Names

Attribute name
Virtual_Size

Column name
VIRTSIZE

Page Faults Per Sec attribute

Description
The rate of page faults for the process that is measured in faults per second. This value contains only valid data for processes.

Type
Integer (gauge)

Names

Attribute name
Page_Faults_Per_Sec

Column name
PAGEFAULTS

Working Set Size attribute

Description
The working set size of the process in MB. This value contains only valid data for processes.

Type
Integer (gauge)

Names

Attribute name
Working_Set_Size

Column name
WORKSET

Thread Count attribute

Description
The number of threads that are currently allocated by this process. This value contains only valid data for processes.

Type
Integer (gauge)

Names

Attribute name
Thread_Count

Column name
THREADS

PID attribute

Description
The process id that is associated with the process. This value contains only valid data for processes.

Type
Integer (gauge)

Names

Attribute name
PID

Column name
PID

Percent Privileged Time attribute

Description
The percentage of the available processor time that is being used by this process for privileged operation

Type
Integer (gauge)

Names

Attribute name
Percent_Privileged_Time

Column name
PERCPRIV

Percent User Mode Time attribute

Description
The percentage of the available processor time that is being used by this process for user mode operation

Type
Integer (gauge)

Names

Attribute name
Percent_User_Mode_Time

Column name
PERCUSER

Percent Processor Time attribute

Description
The percentage of the elapsed time that this process used the processor to run instructions

Type
Integer (gauge)

Names

Attribute name
Percent_Processor_Time

Column name
PERCPROC

Command Line attribute

Description
The program name and any arguments that are specified on the command line when the process was started. This attribute has the value *N/A* if you are running a Service or Functionality test.

Type
String

Names

Attribute name
Command_Line

Column name
CMDLINE

Functionality Test Status attribute

Description
The return code of the functionality test. When the monitored application is running correctly, SUCCESS is returned. NOT_RUNNING is returned when the application is not running correctly. N/A is returned when the row does not represent a functionality test.

Type
Integer with enumerated values. The strings are displayed in the Tivoli Enterprise Portal, the warehouse, and queries return the numbers. The defined values are: N/A(1), SUCCESS (0), GENERAL_ERROR (2), WARNING (3), NOT_RUNNING (4), DEPENDENT_NOT_RUNNING (5), ALREADY_RUNNING (6), PREREQ_NOT_RUNNING (7), TIMED_OUT (8), DOESNT_EXIST (9), UNKNOWN (10), DEPENDENT_STILL_RUNNING (11), or INSUFFICIENT_USER_AUTHORITY (12). Any other values display the numeric value in the Tivoli Enterprise Portal.

Names

Attribute name
Functionality_Test_Status

Column name
FUNCSTATUS

Functionality Test Message attribute

Description
The text message that corresponds to the Functionality Test Status. This attribute is valid only for command return codes.

Type
String

Names

Attribute name
Functionality_Test_Message

Column name
FUNCMSG

Performance Object Status node

Use the Performance Object Status attribute group to see the status of all of the attribute groups that make up the agent. Each of the attribute groups is represented by a row in this table or other type of view. The status of an attribute group reflects the result of the last data collection attempt, or data reception event, for the attribute group. When you check the status information, you can see whether the agent is operating correctly. When your agent does not collect data, but receives it (event data), attributes that relate to sampled data do not contain useful data. Only the first seven attributes that are listed are relevant for event data.

Historical group
This attribute group is eligible for use with Tivoli Data Warehouse.

Attribute descriptions
The following list contains information about each attribute in the Performance Object Status attribute group:

Node attribute: This attribute is a key attribute.

Description
The managed system name of the agent.

Type
String

Warehouse name
NODE

Timestamp attribute

Description
The local time at the agent when the data was collected.

Type
String

Warehouse name
TIMESTAMP

Query Name attribute: This attribute is a key attribute.

Description
The name of the attribute group.

Type
String

Warehouse name
QUERY_NAME or ATTRGRP

Object Name attribute

Description
The name of the performance object.

Type
String

Warehouse name
OBJECT_NAME or OBJNAME

Object Type attribute

Description
The type of the performance object.

Type
Integer with enumerated values. The strings are displayed in the Tivoli Enterprise Portal. The warehouse and queries return the values that are shown in parentheses. The following values are defined:

- WMI (0)
- PERFMON (1)
- WMI ASSOCIATION GROUP (2)
- JMX (3)
- SNMP (4)
- SHELL COMMAND (5)
- JOINED GROUPS (6)
- CIMOM (7)
- CUSTOM (8)
- ROLLUP DATA (9)
- WMI REMOTE DATA (10)
- LOG FILE (11)
- JDBC (12)
- CONFIG DISCOVERY (13)
- NT EVENT LOG (14)
- FILTER (15)
- SNMP EVENT (16)
- PING (17)
- DIRECTOR DATA (18)
- DIRECTOR EVENT (19)
- SSH REMOTE SHELL COMMAND (20)

Any other value is the value that is returned by the agent in the Tivoli Enterprise Portal.

Warehouse name
OBJECT_TYPE or OBJTYPE

Object Status attribute

Description
The status of the performance object.

Type
Integer with enumerated values. The strings are displayed in the Tivoli Enterprise Portal. The warehouse and queries return the values that are shown in parentheses. The following values are defined:

- ACTIVE (0)
- INACTIVE (1)

Any other value is the value that is returned by the agent in the Tivoli Enterprise Portal.

Warehouse name
OBJECT_STATUS or OBJSTTS

Error Code attribute

Description
The error code that is associated with the query.

Type

Integer with enumerated values. The strings are displayed in the Tivoli Enterprise Portal. The warehouse and queries return the values that are shown in parentheses. The following values are defined:

- NO ERROR (0)
- GENERAL ERROR (1)
- OBJECT NOT FOUND (2)
- COUNTER NOT FOUND (3)
- NAMESPACE ERROR (4)
- OBJECT CURRENTLY UNAVAILABLE (5)
- COM LIBRARY INIT FAILURE (6)
- SECURITY INIT FAILURE (7)
- PROXY SECURITY FAILURE (9)
- NO INSTANCES RETURNED (10)
- ASSOCIATOR QUERY FAILED (11)
- REFERENCE QUERY FAILED (12)
- NO RESPONSE RECEIVED (13)
- CANNOT FIND JOINED QUERY (14)
- CANNOT FIND JOIN ATTRIBUTE IN QUERY 1 RESULTS (15)
- CANNOT FIND JOIN ATTRIBUTE IN QUERY 2 RESULTS (16)
- QUERY 1 NOT A SINGLETON (17)
- QUERY 2 NOT A SINGLETON (18)
- NO INSTANCES RETURNED IN QUERY 1 (19)
- NO INSTANCES RETURNED IN QUERY 2 (20)
- CANNOT FIND ROLLUP QUERY (21)
- CANNOT FIND ROLLUP ATTRIBUTE (22)
- FILE OFFLINE (23)
- NO HOSTNAME (24)
- MISSING LIBRARY (25)
- ATTRIBUTE COUNT MISMATCH (26)
- ATTRIBUTE NAME MISMATCH (27)
- COMMON DATA PROVIDER NOT STARTED (28)
- CALLBACK REGISTRATION ERROR (29)
- MDL LOAD ERROR (30)
- AUTHENTICATION FAILED (31)
- CANNOT RESOLVE HOST NAME (32)
- SUBNODE UNAVAILABLE (33)
- SUBNODE NOT FOUND IN CONFIG (34)
- ATTRIBUTE ERROR (35)
- CLASSPATH ERROR (36)
- CONNECTION FAILURE (37)
- FILTER SYNTAX ERROR (38)
- FILE NAME MISSING (39)
- SQL QUERY ERROR (40)

- SQL FILTER QUERY ERROR (41)
- SQL DB QUERY ERROR (42)
- SQL DB FILTER QUERY ERROR (43)
- PORT OPEN FAILED (44)
- ACCESS DENIED (45)
- TIMEOUT (46)
- NOT IMPLEMENTED (47)
- REQUESTED A BAD VALUE (48)
- RESPONSE TOO BIG (49)
- GENERAL RESPONSE ERROR (50)
- SCRIPT NONZERO RETURN (51)
- SCRIPT NOT FOUND (52)
- SCRIPT LAUNCH ERROR (53)
- CONF FILE DOES NOT EXIST (54)
- CONF FILE ACCESS DENIED (55)
- INVALID CONF FILE (56)
- EIF INITIALIZATION FAILED (57)
- CANNOT OPEN FORMAT FILE (58)
- FORMAT FILE SYNTAX ERROR (59)
- REMOTE HOST UNAVAILABLE (60)
- EVENT LOG DOES NOT EXIST (61)
- PING FILE DOES NOT EXIST (62)
- NO PING DEVICE FILES (63)
- PING DEVICE LIST FILE MISSING (64)
- SNMP MISSING PASSWORD (65)
- DISABLED (66)
- URLS FILE NOT FOUND (67)
- XML PARSE ERROR (68)
- NOT INITIALIZED (69)
- ICMP SOCKETS FAILED (70)

Any other value is the value that is returned by the agent in the Tivoli Enterprise Portal.

Warehouse name
ERROR_CODE or ERRCODE

Last Collection Start attribute

Description
The most recent time a data collection of this group started.

Type
Time stamp with enumerated values. The strings are displayed in the Tivoli Enterprise Portal. The warehouse and queries return the values that are shown in parentheses. The following values are defined:

- NOT COLLECTED (0691231190000000)
- NOT COLLECTED (0000000000000001)

Any other value is the value that is returned by the agent in the Tivoli Enterprise Portal.

Warehouse name
LAST_COLLECTION_START or COLSTRT

Last Collection Finished attribute

Description
The most recent time a data collection of this group finished.

Type
Time stamp with enumerated values. The strings are displayed in the Tivoli Enterprise Portal. The warehouse and queries return the values that are shown in parentheses. The following values are defined:

- NOT COLLECTED (0691231190000000)
- NOT COLLECTED (0000000000000001)

Any other value is the value that is returned by the agent in the Tivoli Enterprise Portal.

Warehouse name
LAST_COLLECTION_FINISHED or COLFINI

Last Collection Duration attribute

Description
The duration of the most recently completed data collection of this group in seconds.

Type
Real number (32-bit counter) with two decimal places of precision

Warehouse name
LAST_COLLECTION_DURATION or COLDURA

Average Collection Duration attribute

Description
The average duration of all data collections of this group in seconds.

Type
Real number (32-bit counter) with two decimal places of precision with enumerated values. The strings are displayed in the Tivoli Enterprise Portal. The warehouse and queries return the values that are shown in parentheses. The following values are defined:

- NO DATA (-100)

Any other value is the value that is returned by the agent in the Tivoli Enterprise Portal.

Warehouse name
AVERAGE_COLLECTION_DURATION or COLAVGD

Refresh Interval attribute

Description
The interval at which this group is refreshed in seconds.

Type
Integer (32-bit counter)

Warehouse name
REFRESH_INTERVAL or REFRINT

Number of Collections attribute

Description
The number of times this group is collected since agent start.

Type
Integer (32-bit counter)

Warehouse name
NUMBER_OF_COLLECTIONS or NUMCOLL

Cache Hits attribute

Description
The number of times an external data request for this group is satisfied from the cache.

Type
Integer (32-bit counter)

Warehouse name
CACHE_HITS or CACHEHT

Cache Misses attribute

Description
The number of times an external data request for this group was not available in the cache.

Type
Integer (32-bit counter)

Warehouse name
CACHE_MISSES or CACHEMS

Cache Hit Percent attribute

Description
The percentage of external data requests for this group that are satisfied from the cache.

Type
Real number (32-bit counter) with two decimal places of precision

Warehouse name
CACHE_HIT_PERCENT or CACHPCT

Intervals Skipped attribute

Description
The number of times a background data collection was skipped because the previous collection was still running when the next one was due to start.

Type
Integer (32-bit counter)

Warehouse name
INTERVALS_SKIPPED or INTSKIP

Thread Pool Status attribute group

The Thread Pool Status attribute group contains information that reflects the status of the internal thread pool that is used to collect data asynchronously.

The following comprises a list of the attributes for this attribute group. The name in bold text shows how the attribute is displayed in the Tivoli Enterprise Portal.

The following list contains information about each attribute in the Thread Pool Status attribute group:

Node attribute - This attribute is a key attribute

Description
The managed system name of the agent

Type
String

Names

Attribute name
Node

Column name
ORIGINNODE

Timestamp attribute

Description
The time that is collected from the agent system when the data row was built and sent from the agent to the Tivoli Enterprise Monitoring Server. Or stored for historical purposes. It represents the local time zone of the agent system.

Type
Time

Names

Attribute name
Timestamp

Column name
TIMESTAMP

Thread Pool Size attribute

Description
The number of threads currently existing in the thread pool.

Type
Integer

Names

Attribute name
Thread_Pool_Size

Column name
THPSIZE

Thread Pool Max Size attribute

Description
The maximum number of threads that are allowed to exist in the thread pool.

Type
Integer

Names

Attribute name
Thread_Pool_Max_Size

Column name
TPMAXSZ

Thread Pool Active Threads attribute

Description
The number of threads in the thread pool currently active doing work.

Type
Integer

Names

Attribute name
Thread_Pool_Active_Threads

Column name
TPACTTH

Thread Pool Avg Active Threads attribute

Description
The average number of threads in the thread pool simultaneously active doing work.

Type
Integer

Names

Attribute name
Thread_Pool_Avg_Active_Threads

Column name
TPAVGAT

Thread Pool Min Active Threads attribute

Description
The minimum number of threads in the thread pool simultaneously active doing work.

Type
Integer

Names

Attribute name
Thread_Pool_Min_Active_Threads

Column name
TPMINAT

Thread Pool Max Active Threads attribute

Description
The peak number of threads in the thread pool simultaneously active doing work.

Type
Integer

Names

Attribute name
Thread_Pool_Max_Active_Threads

Column name
TPMAXAT

Thread Pool Queue Length attribute

Description
The number of jobs currently waiting in the thread pool queue.

Type
Integer

Names

Attribute name
Thread_Pool_Queue_Length

Column name
TPQLGTH

Thread Pool Avg Queue Length attribute

Description
The average length of the thread pool queue during this run.

Type
Integer

Names

Attribute name
Thread_Pool_Avg_Queue_Length

Column name
TPAVGQL

Thread Pool Min Queue Length attribute

Description
The minimum length the thread pool queue reached.

Type
Integer

Names

Attribute name
Thread_Pool_Min_Queue_Length

Column name
TPMINQL

Thread Pool Max Queue Length attribute

Description
The peak length the thread pool queue reached.

Type
Integer

Names

Attribute name
Thread_Pool_Max_Queue_Length

Column name
TPMAXQL

Thread Pool Avg Job Wait attribute

Description
The average time a job spends waiting on the thread pool queue.

Type
Integer

Names

Attribute name
Thread_Pool_Avg_Job_Wait

Column name
TPAVJBW

Thread Pool Total Jobs attribute

Description
The number of jobs that are completed by all threads in the pool since agent start.

Type
Integer

Names

Attribute name
Thread_Pool_Total_Jobs

Column name
TPTJOBS

Event log attribute node

The Event log attribute group contains any recent event log entries that pertain to the application.

By default, the agent displays only events that occur after the agent is started. Events are removed from the Event Log view 1 hour after they occur.

The following list contains information about each attribute in the Event Log attribute group:

Node attribute - This attribute is a key attribute

Description
The managed system name of the agent

Type
String

Names

Attribute name
Node

Column name
ORIGINNODE

Log Name attribute

Description
The event log - Application, System, Security, or an application-specific log

Type
String

Names

Attribute name
Log_Name

Column name
LOGNAME

Event Source attribute

Description
The event source that is defined by the application

Type
String

Names

Attribute name
Event_Source

Column name
EVTSOURCE

Event Type attribute

Description
Event Type - Error(0), Warning(1), Informational(2), Audit_Success(3), Audit_Failure(4), Unknown(5)

Type
Integer

Names

Attribute name
Event_Type

Column name
EVTTYPE

Event ID attribute

Description
The ID of the event

Type
Integer

Names

Attribute name
Event_ID

Column name
EVTID

Event Category attribute

Description
The category of the event

Type
String

Names

Attribute name
Event_Category

Column name
EVTCATEG

Message attribute

Description
The event message

Type
String

Names

Attribute name
Message

Column name
MESSAGE

Time Generated attribute

Description
The time the event was generated

Type
Time

Names

Attribute name
Time_Generated

Column name
TIMESTAMP

Log File Summary

The attributes of this attribute group are included in summary attribute groups when that option is selected in the advanced properties of the data source.

A Summary node is created for each Log File data source when **Include attribute in summary attribute group** is selected in the advanced properties of the data source. The name of the summary node is the name of the data source with Summary added to the end.

The following list contains information about each of the default attributes in the Log File Summary attribute group. These attributes are always included in summary attribute groups. If you select **Include attribute in summary attribute group**, see step "9" on page 108 in ("Monitoring a log file" on page 104), then the summary attribute group for that log attribute group also contains each of the attributes you selected. The values are a copy of the corresponding attribute in the log file attribute group.

All of the added attributes together become a key and the summary table includes one row per unique set of keys. The row indicates how many log records are received during the interval where all of the provided keys matched the value reported in the corresponding attributes.

Node attribute - This attribute is a key attribute

Description
The managed system name of the agent

Type
String

Names

Attribute name
Node

Column name
ORIGINNODE

Timestamp attribute

Description
The local time at the agent when the data was collected

Type
Time

Names

Attribute name
Timestamp

Column name
TIMESTAMP

Interval Unit attribute

Description
The number of seconds between summary attribute generation

Type
Integer (gauge)

Names

Attribute name
_Interval_Unit

Column name
IU

Interval attribute

Description
Offset of the current interval within the next larger unit of time (for example, minutes within an hour)

Type
Integer (gauge)

Names

Attribute name
_Interval

Column name
INV

Occurrences attribute

Description
The number of occurrences that are recorded during the interval

Type
Integer (gauge)

Names

Attribute name
_Occurrences

Column name
OCC

LocalTimeStamp attribute

Description
The time that the summary data was generated

Type
Timestamp

Names

Attribute name
_LocalTimeStamp

Column name
LTS

DateTime attribute

Description
The time that the summary data was generated

Type
String

Names

Attribute name
_Date_Time

Column name
DT

Interval Unit Name attribute

Description
The word description of the interval unit

Type
String

Names

Attribute name
_Interval_Unit_Name

Column name
IUN

AIX Binary Log attribute group

The AIX Binary Log attribute group displays events from the AIX Binary Log as selected by the provided `errpt` command string.

The following list contains information about each attribute in the AIX Binary Log Attribute Group:

Note: The Agent Builder prevents removing, reordering, or changing the size of the `Identifier`, `ErrptTimestamp`, `Type`, `Class`, `ResourceName`, and `Description` attributes. The agent parses the data that comes back from an `errpt` command that is based on columns within the line of text. These columns are defined by the order and size of the `Identifier`, `ErrptTimestamp`, `Type`, `Class`, `ResourceName`, and `Description` attributes. Removing, reordering, or changing the size of these attributes, changes the attribute that the various columns go into. The resulting row as seen in Tivoli Monitoring is then incorrect.

You can, however, rename these attributes.

Node attribute - This attribute is a key attribute

Description
The managed system name of the agent

Type
String

Names

Attribute name
Node

Column name
ORIGINNODE

Identifier attribute - This attribute is a key attribute

Description
The event identifier reported by `errpt`

Type
String

Names

Attribute name
Identifier

Column name
IDENTIFIER

ErrptTimestamp attribute

Description
The time the event is recorded as reported by `errpt`.

Note: This attribute is hidden at run time. This attribute contains a raw value. Other attributes that are derived from this attribute display the value in a more usable form. This attribute is available from within Agent Builder for that purpose, but by default it is not visible in the Tivoli Monitoring environment at run time. If you want to make it visible, select the attribute in the **Data Source Definition** page in the Agent Editor and select **Display attribute in the Tivoli Enterprise Portal**.

Type
String

Names

Attribute name
ErrptTimestamp

Column name
ERRPTTIMES

Type

Description
The single character event type reported by `errpt`, one of `I(NFO)`, `P(END/ERF/ERM)`, `T(EMP)`, and `U(NKN)`

Type
String

Names

Attribute name
Type

Column name
TYPE

Class attribute - This attribute is a key attribute

Description
The event class reported by `errpt`, one of `Hardware`, `Software`, `Operator`, and `Undertermined`. These values are enumerated. The raw values for use with situations are H, S, O, and U.

Type
String

Names

Attribute name
Class

Column name
CLASS

ResourceName

Description
The resource name reported by `errpt`, identifies the origin of the error record

Type
String

Names

Attribute name
ResourceName

Column name
RESOURCENA

Description attribute

Description
The description reported by `errpt`, typically a short text message that describes the nature of the error

Type
String

Names

Attribute name
Description

Column name
DESCRIPTIO

LogFile attribute

Description
The full name of the binary `errpt` log including the path.

Note: This attribute is hidden at run time. This attribute contains a raw value. Other attributes that are derived from this attribute display the value in a more useable form. This attribute is available from within Agent Builder for that purpose, but by default it is not visible in the Tivoli Monitoring environment at run time. If you want to make it visible, select the attribute in the **Data Source Definition** page in the Agent Editor and select **Display attribute in the Tivoli Enterprise Portal**.

Type
String

Names

Attribute name
LogFile

Column name
LOGFILE

System attribute

Description
The host name of the system where the error was collected

Type
String

Names

Attribute name
System

Column name
SYSTEM

LogName attribute

Description
The base name of the binary `errpt` log from which the record was collected

Type
String

Names

Attribute name
LogName

Column name
LOGNAME

LogPath attribute

Description
The directory name that contains the binary `errpt` log from which the record was collected

Type
String

Names

Attribute name
LogPath

Column name
LOGPATH

EntryTime attribute

Description
The time the event is recorded as reported by `errpt` in Tivoli Timestamp format. This time is not necessarily identical to the time when the agent received the event, as recorded in the **`Timestamp`** field.

Type
Time stamp

Names

Attribute name
EntryTime

Column name
ENTRYTIME

Monitor and Notification attribute groups

Definitions for the Monitor and Notification attribute groups.

The first 4 are specific to monitors and the last is for notifications (all are related to JMX).

Each one is listed with an indication whether it is event-based or not. For non-event based attribute groups, data is collected when needed. For event-based attribute groups, the agent maintains a cache of the last 100 events received. These events are used to respond to requests from the Tivoli Enterprise Portal. The events are forwarded immediately for analysis by situations and warehousing.

Counter Notifications

The Counter Notifications attribute group is a non-event based attribute group that sends events that are received by all counter monitors.

The following list contains information about each attribute in the Counter Notifications attribute group:

Node attribute - This attribute is a key attribute

Description
The managed system name of the agent

Type
String

Names

Attribute name
Node

Column name
ORIGINNODE

Timestamp attribute

Description
The local time at the agent when the data was collected

Type
Time

Names

Attribute name
Timestamp

Column name
TIMESTAMP

Notification Type attribute

Description
The type of notification received. Describes how the observed attribute of the MBean triggered the notification.

Type
String

Names

Attribute name
Notification_Type

Column name
NOTIFICATI

Monitor ID attribute

Description
Monitor ID of the monitor who generated this notification

Type
Integer

Names

Attribute name
Monitor_ID

Column name
MONITOR_ID

Observed MBean attribute

Description
The MBean whose attribute is being monitored

Type
String

Names

Attribute name
Observed_MBean

Column name
OBSERVED_M

Observed Attribute attribute

Description
Name of the attribute that is monitored in the Observed MBean

Type
String

Names

Attribute name
Observed_Attribute

Column name
OBSERVED_A

Threshold attribute

Description
The current threshold of the monitor

Type
String

Names

Attribute name
Threshold

Column name
THRESHOLD

Offset attribute

Description
The value added to the threshold each time the attribute exceeds the threshold. This value forms a new threshold.

Type
String

Names

Attribute name
Offset

Column name
OFFSET

Modulus attribute

Description
The maximum value of the attribute. When it reaches this value, it rolls over and begins counting again from zero.

Type
Integer

Names

Attribute name
Modulus

Column name
MODULUS

Counter Value attribute

Description
Value of the counter that triggered the notification

Type
Integer

Names

Attribute name
Counter_Value

Column name
COUNTER_VA

Notification Time Stamp attribute

Description
Time that the notification was triggered

Type
Time

Names

Attribute name
Notification_Time_Stamp

Column name
NOTIFICAT0

Notification Message attribute

Description
The message in the notification

Type
String

Names

Attribute name
Notification_Message

Column name
NOTIFICAT1

Gauge Notifications

The Gauge Notifications attribute group is a non-event based attribute group that sends events that are received by all gauge monitors.

The following list contains information about each attribute in the Gauge Notifications attribute group:

Node attribute - This attribute is a key attribute

Description
The managed system name of the agent

Type
String

Names

Attribute name
Node

Column name
ORIGINNODE

Timestamp attribute

Description
The local time at the agent when the data was collected

Type
Time

Names

Attribute name
Timestamp

Column name
TIMESTAMP

Notification Type attribute

Description
The type of notification received. Describes how the observed attribute of the MBean triggered the notification.

Type
String

Names

Attribute name
Notification_Type

Column name
NOTIFICATI

Monitor ID attribute

Description
Monitor ID of the monitor who generated this notification

Type
Integer

Names

Attribute name
Monitor_ID

Column name
MONITOR_ID

Observed MBean attribute

Description
The MBean whose attribute is being monitored

Type
String

Names

Attribute name
Observed_MBean

Column name
OBSERVED_M

Observed Attribute attribute

Description
Name of the attribute that is monitored in the Observed MBean

Type
String

Names

Attribute name
Observed_Attribute

Column name
OBSERVED_A

Low Threshold attribute

Description
The threshold that the monitor is watching for the observed attribute to cross

Type
String

Names

Attribute name
Low_Threshold

Column name
LOW_THRESH

High Threshold attribute

Description
The threshold that the monitor is watching for the observed attribute to cross

Type
String

Names

Attribute name
High_Threshold

Column name
HIGH_THRES

Gauge Value attribute

Description
Value of the gauge that triggered the notification

Type
String

Names

Attribute name
Gauge_Value

Column name
MODULUSGAUGE_VALU

Notification Time Stamp attribute

Description
Time that the notification was triggered

Type
Time

Names

Attribute name
Notification_Time_Stamp

Column name
NOTIFICAT0

Notification Message attribute

Description
The message in the notification

Type
String

Names

Attribute name
Notification_Message

Column name
NOTIFICAT1

Registered Monitors

The Registered Monitors attribute group is an event-based attribute group that shows a list of all JMX Monitors that are created by the agent.

The following list contains information about each attribute in the Registered Monitors attribute group:

Node attribute - This attribute is a key attribute

Description
The managed system name of the agent

Type
String

Names

Attribute name
Node

Column name
ORIGINNODE

Timestamp attribute

Description
The local time at the agent when the data was collected

Type
Time

Names

Attribute name
Timestamp

Column name
TIMESTAMP

Monitor ID attribute - This attribute is a key attribute

Description
The unique integer identifier for a monitor

Type
Integer

Names

Attribute name
Monitor_ID

Column name
MONITOR_ID

Monitor Parameters attribute

Description
The parameters that are used to create the monitor

Type
String

Names

Attribute name
Monitor_Parameters

Column name
MONITOR_PA

Monitor Name attribute

Description
The JMX Object Name of the monitor MBean

Type
String

Names

Attribute name
Monitor_Name

Column name
MONITOR_NA

String Notifications

The String Notifications attribute group is a non-event based attribute group that sends events that are received by all string monitors.

The following list contains information about each attribute in the String Notifications attribute group:

Node attribute - This attribute is a key attribute

Description
The managed system name of the agent

Type
String

Names

Attribute name
Node

Column name
ORIGINNODE

Timestamp attribute

Description
The local time at the agent when the data was collected

Type
Time

Names

Attribute name
Timestamp

Column name
TIMESTAMP

Notification Type attribute

Description
The type of notification received. Describes how the observed attribute of the MBean triggered the notification.

Type
String

Names

Attribute name
Notification_Type

Column name
NOTIFICATI

Monitor ID attribute - This attribute is a key attribute

Description
The unique integer identifier for a monitor

Type
Integer

Names

Attribute name
Monitor_ID

Column name
MONITOR_ID

Observed MBean attribute

Description
The MBean whose attribute is being monitored

Type
String

Names

Attribute name
Observed_MBean

Column name
OBSERVED_M

Observed Attribute attribute

Description
Name of the attribute that is monitored in the Observed MBean

Type
String

Names

Attribute name
Observed_Attribute

Column name
OBSERVED_A

Compare String attribute

Description
The string that is used in the comparison operation

Type
String

Names

Attribute name
Compare_String

Column name
COMPARE_ST

String Value attribute

Description
Value of the attribute that triggered the notification

Type
String

Names

Attribute name
String_Value

Column name
STRING_VAL

Notification Time Stamp attribute

Description
Time that the notification was triggered

Type
Time

Names

Attribute name
Notification_Time_Stamp

Column name
NOTIFICAT0

Notification Message attribute

Description
The message in the notification

Type
String

Names

Attribute name
Notification_Message

Column name
NOTIFICAT1

SNMP Event attribute groups

SNMP event attribute groups are used to receive traps and informs. These attribute groups are event-based attribute groups

The following list contains information about each attribute in the SNMP Event Attribute Groups:

Note: You can change the default display name of these attributes. These display names are distinct from the internal ID for each attribute.

Enterprise_OID
The enterprise OID that generated the trap.

Source_Address
Host name or IP address of the SNMP agent that sent the trap.

Generic_Trap
Generic trap number that is extracted from the received trap. Possible values:

```
0 ColdStart
1 WarmStart
2 LinkDown
3 LinkUp
4 Authentication Failure
5 EGPNeighborLoss
```

Specific_Trap
Enterprise-specific trap number that is extracted from the received trap. Applies only when Generic_Trap = 6.

Alert_Name
Trap name as specified in the definition in the trap configuration file.

Category
Trap category as specified in the definition in the trap configuration file.

Description
Trap description as specified in the definition in the trap configuration file. The maximum description length is 256 characters.

Enterprise_Name
Trap Enterprise name as specified in the trap configuration file and determined through the trap object identifier.

Source_Status
Status of the agent that originated the trap after the trap is sent as specified in the trap definition in the trap configuration file.

Source_Type
Type of the agent that originated the trap as specified in the trap definition in the trap configuration file.

Event_Variables
Variable binding (VarBind) data that is received in the trap protocol data unit (PDU). The string is constructed as:

```
{OID[type]=value}{OID[type]=value}{oid[type]=value}...
```

Where:

`oid`
MIB variable object identifier

`type`
SMI data type

`value`
Variable value

`{}`
Each triplet is surrounded by braces ({}).

Note: The attributes Alert Name, Category, Description, Enterprise_Name, Source_Status, and Source_Type provide more information. In the **SNMP MIB Browser** window, select the **Include attributes that show information defined in the trap configuration file** check box to include these attributes.

JMX Event attribute groups

JMX event attribute groups are used to receive notifications from an MBean server.

These attribute groups are non-event based attribute groups and are generated with the following attributes that can be edited by the agent developer.

The following list contains information about each attribute in the JMX Event Attribute Groups:

Node attribute - This attribute is a key attribute

Description
The managed system name of the agent

Type
String

Names

Attribute name
Node

Column name
ORIGINNODE

Timestamp attribute

Description
The local time at the agent when the data was collected

Type
Time

Names

Attribute name
Timestamp

Column name
TIMESTAMP

Type attribute

Description
The notification type

Type
String

Names

Attribute name
Type

Column name
TYPE

Source attribute

Description
The MBean that caused the notification to be sent

Type
String

Names

Attribute name
Source

Column name
SOURCE

Sequence Number attribute

Description
The sequence number from the notification object

Type
String

Names

Attribute name
Sequence_Number

Column name
SEQUENCE_N

Message attribute

Description
The notification message

Type
String

Names

Attribute name
Message

Column name
MESSAGE

User Data attribute

Description
The user data object from the notification

Type
String

Names

Attribute name
User_Data

Column name
USER_DATA

Ping attribute group

The Ping attribute group contains the results of ICMP pings that are sent to lists of devices.

The following list contains information about each attribute in the Ping Attribute Group:

Node attribute - This attribute is a key attribute

Description
The managed system name of the agent.

Type
String

Names

Attribute name
Node

Column name
ORIGINNODE

Timestamp attribute

Description
The time that is collected from the agent system when the data row was built and sent from the agent to the Tivoli Enterprise Monitoring Server. Or stored for historical purposes. It represents the local time zone of the agent system.

Type
Time

Names

Attribute name
Timestamp

Column name
TIMESTAMP

Address attribute - This attribute is a key attribute

Description
The IP address of the host that is monitored.

Type
String with enumerated value. The value UNKNOWN_ADDRESS is displayed if the IP address is unknown. The warehouse and queries return 0.0.0.0 for this enumeration. Any other IP address values are displayed as is.

Names

Attribute name
Address

Column name
PNGADDR

Device Entry attribute - This attribute is a key attribute

Description
The entry in the device list file for this node.

Type
String

Names

Attribute name
Device_Entry

Column name
PINGDEVC

Current Response Time attribute

Description
The current network response time for ICMP requests for the managed node in milliseconds.

Type
Integer with enumerated values. The strings are displayed in the Tivoli Enterprise Portal. The warehouse and queries return the numbers. The defined values are `TIMEOUT(-1)` and `SEND_FAILURE(-2)`. Any other values show the numeric value.

Names

Attribute name
Current_Response_Time

Column name
PINGRSTM

Name attribute

Description
The host name of the managed node. If the node address cannot be resolved through DNS, then the dotted decimal IP address is shown.

Type
String with enumerated value. The value UNKNOWN_HOSTNAME is displayed if the host name is unknown. The warehouse and queries return 0.0.0.0 for this enumeration. Any other host name values are displayed as is.

Names

Attribute name
Name

Column name
PNGNAME

Node Description attribute

Description
The description of the managed node.

Type
String

Names

Attribute name
Node_Description

Column name
PNGDESC

Node Status attribute

Description
The current operating status of the managed node.

Type
Integer with enumerated values. The strings are displayed in the Tivoli Enterprise Portal. The warehouse and queries return the numbers. The defined values are `INVALID(-2)`, `UNKNOWN(-1)`, `INACTIVE(0)`, and `ACTIVE(1)`.

Names

Attribute name
Node_Status

Column name
PNGSTAT

Node Type attribute

Description
The type of the managed node. If the node is online, it is an IP Node. If it is offline, the type is Unknown.

Type
Integer with enumerated values. The strings are displayed in the Tivoli Enterprise Portal. The warehouse and queries return the numbers. The defined values are `UNKNOWN(0)` and `IP NODE(1)`.

Names

Attribute name
Node_Type

Column name
PNGTYPE

Status Timestamp

Description
The date and time the node was last checked.

Type
Time

Names

Attribute name
Status_Timestamp

Column name
PNGTMSP

HTTP attribute groups

The two HTTP attribute groups, `Managed URLs` and `URL Objects`, are used to receive information from URLs and the objects within theses URLs.

For information about the syntax that is used in the Managed URLs and URL Objects tables, see ("Specific fields for HTTP attributes" on page 139).

Managed URLs

The following list contains information about each attribute in the Managed URL Attribute Group:

Node attribute - This attribute is a key attribute

Description
The managed system name of the agent

Type
String

Names

Attribute name
Node

Column name
ORIGINNODE

Timestamp attribute

Description
The local time at the agent when the data was collected

Type
Time

Names

Attribute name
Timestamp

Column name
TIMESTAMP

URL attribute - This attribute is a key attribute

Description
The URL that is being monitored.

Type
String

Names

Attribute name
URL

Column name
HTTPURL

Response Time attribute

Description
The amount of time it took to download the response in milliseconds.

Type
Integer with enumerated value. The string is displayed in the Tivoli Enterprise Portal, the warehouse, and queries return the number. The defined value is `TIMEOUT (-1)`.

Names

Attribute name
Response_Time

Column name
HTTPURL

Page Size attribute

Description
The size of the page that is returned by the HTTP request.

Type
Integer with enumerated value. The string is displayed in the Tivoli Enterprise Portal, the warehouse, and queries return the number. The defined value is `NO_RESPONSE_RECEIVED(-1)`.

Names

Attribute name
Page_Size

Column name
PAGESZ

Page Objects attribute

Description
The total number of objects that are associated with the monitored page.

Type
Integer with enumerated value. The string is displayed in the Tivoli Enterprise Portal, the warehouse, and queries return the number. The defined value is `NOT_COLLECTED(-1)`.

Names

Attribute name
Page_Objects

Column name
PGOBJS

Total Object Size attribute

Description
The size of the page that is returned by the HTTP request.

Type
Integer with enumerated value. The string is displayed in the Tivoli Enterprise Portal, the warehouse, and queries return the number. The defined value is `NOT_COLLECTED(-1)`.

Names

Attribute name
Total_Object_Size

Column name
TOTOSZ

Page Title attribute

Description
The page title of the received URL page.

Type
String

Names

Attribute name
Page_Title

Column name
PAGETTL

Server Type attribute

Description
The type of server that is used at the target URL website.

Type
String

Names

Attribute name
Server_Type

Column name
SRVTYP

Response Code attribute

Description
The response code of the HTTP request.

Type
Integer with enumerated value. The string is displayed in the Tivoli Enterprise Portal, the warehouse, and queries return the number. The defined value is `NO_RESPONSE_RECEIVED(-1)`.

Names

Attribute name
Response_Code

Column name
CODE

Status attribute

Description
The current managed URL status (OK or status description).

Type
String

Names

Attribute name
Status

Column name
STATUS

URL Alias attribute

Description
The user-specified alias for the URL.

Type
String

Names

Attribute name
URL_Alias

Column name
ALIAS

User Data attribute

Description
The user data that is specified with the URL.

Type
String

Names

Attribute name
User_Data

Column name
USER

URL Objects

The following list contains information about each attribute in the URL Objects Attribute Group:

Node attribute - This attribute is a key attribute

Description
The managed system name of the agent

Type
String

Names

Attribute name
Node

Column name
ORIGINNODE

Timestamp attribute

Description
The local time at the agent when the data was collected

Type
Time

Names

Attribute name
Timestamp

Column name
TIMESTAMP

URL attribute - This attribute is a key attribute

Description
The URL that is being monitored.

Type
String

Names

Attribute name
URL

Column name
HTTPURL

Object Name attribute

Description
The name of the page object within the target URL.

Type
String

Names

Attribute name
Object_Name

Column name
ONAME

Object Size attribute

Description
The size (bytes) of the page object within the target URL.

Type
Integer with enumerated values. The strings are displayed in the Tivoli Enterprise Portal. The warehouse and queries return the numbers. The defined values are `NOT_COLLECTED (-1)`, `OBJECT_NOT_FOUND (-2)`. Any other values show the numeric value.

Names

Attribute name
Object_Size

Column name
SIZE

Object Response Time attribute

Description
The amount of time it took to download the object in milliseconds.

Type
Integer with enumerated values. The strings are displayed in the Tivoli Enterprise Portal. The warehouse and queries return the numbers. The defined values are `NOT_COLLECTED (-1)`, `NO_RESPONSE_RECEIVED (-2)`, `STATUS_CODE_ERROR (-3)`. Any other values show the numeric value.

Names

Attribute name
Object_Response_Time

Column name
ORTIME

Discovery attribute groups

An attribute group that represents the set of subnode instances that are defined for a subnode type

When you create a subnode type, an attribute group is created that represents the set of subnode instances that are defined for that subnode type. Each of these attribute groups includes the same set of attributes.

The following list contains information about each attribute in a Discovery attribute group. The name in bold text shows how the attribute is displayed in the Tivoli Enterprise Portal:

Node attribute - This attribute is a key attribute

Description
The managed system name of the agent

Type
String

Names

Attribute name
Node

Column name
ORIGINNODE

Timestamp attribute

Description
The time from the agent system when the data row was built and sent to the Tivoli Enterprise Monitoring Server (or stored for historical purposes). It represents the local time zone of the agent system.

Type
Time

Names

Attribute name
Timestamp

Column name
TIMESTAMP

Subnode MSN attribute

Description
The Managed System Name of the subnode agent.

Type
String

Names

Attribute name
Subnode_MSN

Column name
SN_MSN

Subnode Affinity attribute

Description
The affinity for the subnode agent.

Type
String

Names

Attribute name
Subnode_Affinity

Column name
SN_AFFIN

Subnode Type attribute

Description
The node type of this subnode.

Type
String

Names

Attribute name
Subnode_Type

Column name
SN_TYPE

Subnode Resource Name attribute

Description
The resource name of the subnode agent.

Type
String

Names

Attribute name
Subnode_Resource_Name

Column name
SN_RES

Subnode Version attribute

Description
The version of the subnode agent.

Type

Names

Attribute name
Subnode_Version

Column name
SN_VER

Take Action Status attribute group

The Take Action Status attribute group contains the status of actions that the agent processed.

This attribute group is event-based and contains information about each attribute in the Take Action Status attribute group:

Node attribute - This attribute is a key attribute

Description
The managed system name of the agent.

Type
String

Names

Attribute name
Node

Column name
ORIGINNODE

Timestamp attribute

Description
The time that is collected from the agent system, when the data row was built and sent from the agent to the Tivoli Enterprise Monitoring Server. Or stored for historical purposes. It represents the local time zone of the agent system.

Type
Time

Names

Attribute name
Timestamp

Column name
TIMESTAMP

Action Name attribute

Description
The name of the action that was run

Type
String

Names

Attribute name
Action_Name

Column name
TSKNAME

Action Status attribute

Description
The status of the action.

Type
Integer with enumerated values. The values are: OK (0), NOT_APPLICABLE (1), GENERAL_ERROR (2), WARNING (3), NOT_RUNNING (4), DEPENDENT_NOT_RUNNING (5), ALREADY_RUNNING (6), PREREQ_NOT_RUNNING (7), TIMED_OUT (8), DOESNT_EXIST (9), UNKNOWN (10), DEPENDENT_STILL_RUNNING (11), INSUFFICIENT_USER_AUTHORITY (12)

Names

Attribute name
Action_Status

Column name
TSKSTAT

Action Application Return Code attribute

Description
The return code of the application the action started.

Type
Integer

Names

Attribute name
Action_App_Return_Code

Column name
TSKAPRC

Action Message attribute

Description
The message that is associated with the return code of the action.

Type
String

Names

Attribute name
Action_Message

Column name
TSKMSGE

Action Instance attribute

Description
The instance that is associated with the output produced by running the action. If the action is a system command, the instance is the line number of the output of the command.

Type
String

Names

Attribute name
Action_Instance

Column name
TSKINST

Action Results attribute

Description
The output that is produced by running the action.

Type
String

Names

Attribute name
Action_Results

Column name
TSKOUTP

Action Command attribute

Description
The command that was run by the action.

Type
String

Names

Attribute name
Action_Command

Column name
TSKCMND

Action Node attribute

Description
The node where the action ran.

Type
String

Names

Attribute name
Action_Node

Column name
TSKORGN

Action Subnode attribute

Description
The subnode where the action ran.

Type
String

Names

Attribute name
Action_Subnode

Column name
TSKSBND

Action ID attribute

Description
The ID of the action.

Type
Integer

Names

Attribute name
Action_ID

Column name
TSKID

Action Type attribute

Description
The type of the action.

Type
Integer with enumerated values. The strings are displayed in the Tivoli Enterprise Portal, the warehouse, and queries return the numbers. The defined values are: UNKNOWN (0), AUTOMATION (1).

Names

Attribute name
Action_Type

Column name
TSKTYPE

Action Owner attribute

Description
The name of the situation or user that initiated the action.

Type
String

Names

Attribute name
Action_Owner

Column name
TSKOWNR

Log File Status attribute group

The Log File Status attribute group contains information that reflects the status of log files this agent is monitoring.

The Log File Status attribute group is included if you have a log attribute group and the agent is at the default minimum Tivoli Monitoring version of 6.2.1 or later. The Log File Status attribute group includes two attributes that are defined as 64-bit numbers so that they can handle large files. 64-bit numeric attribute support is provided by Tivoli Monitoring version 6.2.1 or later.

The following list contains information about each attribute in the Log File Status attribute group:

Node attribute - This attribute is a key attribute

Description
The managed system name of the agent.

Type
String

Names

Attribute name
Node

Column name
ORIGINNODE

Timestamp attribute

Description
The value is the time that is collected from the agent system, when the data row was built and sent from the agent to the Tivoli Enterprise Monitoring Server. Or stored for historical purposes. It represents the local time zone of the agent system.

Type
Time

Names

Attribute name
Timestamp

Column name
TIMESTAMP

Table Name attribute - This attribute is a key attribute

Description
The name of the table in which this log is being monitored

Type
String

Names

Attribute name
Table_Name

Column name
TBLNAME

File Name attribute - This attribute is a key attribute

Description
The name of the file that is being monitored

Type
String

Names

Attribute name
File_Name

Column name
FILNAME

RegEx Pattern attribute - This attribute is a key attribute

Description
The regular expression pattern (if any) that caused this file to be monitored

Type
String

Names

Attribute name
RegEx_Pattern

Column name
REPATRN

File Type attribute

Description
The type of this file (regular file or pipe)

Type
Integer with enumerated values. The strings are displayed in the Tivoli Enterprise Portal. The defined values are UNKNOWN(0), REGULAR FILE(1), PIPE(2)

Names

Attribute name
File_Type

Column name
FILTYPE

File Status attribute

Description
The status of the file that is being monitored

Type
Integer with enumerated values. The strings are displayed in the Tivoli Enterprise Portal. The defined values are: OK(0), PERMISSION DENIED(1), FILE DOES NOT EXIST(2), INTERRUPTED SYSTEM CALL(4), I/O ERROR(5), NO SUCH DEVICE(6), BAD FILE NUMBER(9), OUT OF MEMORY(12), ACCESS DENIED(13), RESOURCE BUSY(16), NOT A DIRECTORY(20), IS A DIRECTORY(21), INVALID ARGUMENT(22), FILE TABLE OVERFLOW(23), TOO MANY OPEN FILES(24), TEXT FILE BUSY(26), FILE TOO LARGE(27), NO SPACE LEFT ON DEVICE(28), ILLEGAL SEEK ON PIPE(29), READ-ONLY FILE SYSTEM(30), TOO MANY LINKS(31), BROKEN PIPE(32)

Names

Attribute name
File_Status

Column name
FILSTAT

Num Records Matched attribute

Description
The number of processed records from this log which matched one of the specified patterns

Type
Integer

Names

Attribute name
Num_Records_Matched

Column name
RECMTCH

Num Records Not Matched attribute

Description
The number of processed records sent to the UnmatchLog; did not match any patterns

Type
Integer

Names

Attribute name
Num_Records_Not_Matched

Column name
RECUNMT

Num Records Processed attribute

Description
The number of records that are processed from this log since agent start (including ones that are not matches/events)

Type
Integer

Names

Attribute name
Num_Records_Processed

Column name
RECPROC

Current File Position attribute

Description
The current position in bytes into the monitored file. Data up to this position is processed, data after this position is not processed. Not applicable to pipes.

Type
Integer

Names

Attribute name
Current_File_Position

Column name
OFFSET

Current® File Size attribute

Description
The current size of the monitored file. Not applicable to pipes.

Type
Integer

Names

Attribute name
Current_File_Size

Column name
FILESIZE

Last Modification Time attribute

Description
The time when the monitored file was last written to. Not applicable to pipes.

Type
Timestamp

Names

Attribute name
Last_Modification_Time

Column name
LASTMOD

Codepage attribute

Description
The language codepage of the monitored file

Type
String

Names

Attribute name
Codepage

Column name
CODEPG

Log File RegEx Statistics attribute group

The Log File RegEx Statistics attribute group contains information that shows the statistics of the log file regular expression search expressions.

Regular expressions can be used to filter records or to define records. This attribute group shows information about both types. When the Result Type attribute contains either `INCLUDE` or `EXCLUDE`, the filter is use to filter records. If the Result Type attribute contains `BEGIN` or `END`, the filter is used to define records. The CPU measurements are approximations that are based on the granularity of the data that is exposed by the operating system. These measurements can result in values of `0.00` when a regular expression takes a small time to evaluate. Use the CPU times to determine the relative cost of regular expressions and to optimize the behavior of specific regular expressions.

The Log File RegEx Statistics attribute group is included if you have a log attribute group and the agent is at Tivoli Monitoring version of 6.2.1 or later. The minimum Tivoli Monitoring Version is selected on the **Agent Information** page. For more information, see (“Naming and configuring the agent” on page 13). The Log File RegEx Statistics attribute group includes attributes that are defined as 64-bit numbers so that they can handle long durations. Support for 64-bit numeric attributes is provided by Tivoli Monitoring version 6.2.1 or later.

The following list contains information about each attribute in the Log File RegEx Statistics attribute group:

Node attribute - This attribute is a key attribute

Description
The managed system name of the agent.

Type
String

Names

Attribute name
Node

Column name
ORIGINNODE

Timestamp attribute

Description
The local time at the agent when the data was collected.

Type
Time

Names

Attribute name
Timestamp

Column name
TIMESTAMP

Table Name attribute - This attribute is a key attribute

Description
The name of the log file attribute group.

Type
String

Names

Attribute name
Table_Name

Column name
TBLNAME

Attribute Name attribute - This attribute is a key attribute

Description
The name of the attribute to which this filter is applied.

Type
String

Names

Attribute name
Attribute_Name

Column name
ATRNAME

Filter Number

Description
The sequence number, starting at zero, of the filter that is being used for the attribute.

Type
Integer (Numeric Property)

Names

Attribute name
Filter_Number

Column name
FLTRNUM

Result Type attribute

Description
The result type can be INCLUDE or EXCLUDE to accept or reject the attribute if the filter matches. The result type can be BEGIN or END to specify the start or end of a record for multi-line records.

Type
Integer with enumerated values. The strings are displayed in the Tivoli Enterprise Portal. If the filter is used to filter records, the defined values are `INCLUDE(1)` or `EXCLUDE(2)`. If the filter is used to define records, the defined values are `BEGIN(3)` or `END(4)`.

Names

Attribute name
Result_Type

Column name
RSTTYPE

Average Processor Time attribute

Description
The average number of processor seconds used to process the filter for this attribute. The average processor time is the total processor seconds divided by the filter count.

Type
Integer (Gauge)

Names

Attribute name
Average_Processor_Time

Column name
CPUTAVG

Processor Time attribute

Description
The total number of processor seconds used to process the filter for this attribute. The processor time is cumulative and is truncated, not rounded. Similar to the Linux `/proc/<pid>/task/`*`thread`*`/stat file`.

Type
Integer (Counter)

Names

Attribute name
Processor_Time

Column name
CPUTIME

Max Processor Time attribute

Description
The maximum number of processor seconds used for a single filter processing. It is possible that the maximum is zero if the filter was never used or if each of the filter processing took less than 0.01 seconds.

Type
Integer (Gauge)

Names

Attribute name
Max_Processor_Time

Column name
CPUTMAX

Min Processor Time attribute

Description
The minimum number of processor seconds used for a single filter processing. It is possible that the minimum is zero if a filter processing took less than 0.01 seconds.

Type
Integer (Gauge)

Names

Attribute name
Min_Processor_Time

Column name
CPUTMIN

Filter Count attribute

Description
The number of times the filter is run. Used with the total processor time to compute the average processor time.

Type
Integer (Counter)

Names

Attribute name
Filter_Count

Column name
COUNT

Filter Count Matched attribute

Description
The number of times the filter is run and the attribute matched.

Type
Integer (Counter)

Names

Attribute name
Filter_Count_Matched

Column name
COUNTMA

Filter Count Unmatched attribute

Description
The number of times the filter is run and the attribute did not match.

Type
Integer (Counter)

Names

Attribute name
Filter_Count_Unmatched

Column name
COUNTUN

RegEx Pattern attribute - This attribute is a key attribute

Description
The regular expression that is used for the match.

Type
String

Names

Attribute name
RegEx_Pattern

Column name
REGXPAT

Last Matched Time attribute

Description
The last time the filter was used and the result matched.

Type
Time

Names

Attribute name
Last_Matched_Time

Column name
LASTMAT

Last Unmatched Time attribute

Description
The last time the filter was used and the result was unmatched.

Type
Time

Names

Attribute name
Last_Unmatched_Time

Column name
LASTUMA

Appendix D. Creating application support extensions for existing agents

For the IBM Tivoli Monitoring environment, you can build an installable package to distribute custom workspaces, situations, queries, and Take Action commands that you created, as an application support extension for an existing agent.

Before you begin

For more information about how to create custom situations, workspaces, Take Action commands, and queries, see (Chapter 11, “Creating workspaces, Take Action commands, and situations,” on page 213).

About this task

Important: This task is not how you add application support to an agent that you are building. To add application support to an agent that you are building see (Chapter 17, “Importing application support files,” on page 257).

Procedure

1. From the Agent Builder, select **File** > **New** > **Other**.
2. Select **Agent Builder Application Support Extension** under **Agent Builder**.
3. Click **Next** to get to the welcome page for the **IBM Tivoli Monitoring Application Support Extension** wizard.
4. Click **Next** on the welcome page.
5. Enter a name for the project and click **Finish**

Creating an Application Support Extension project

Create an Application Support Extension project by using Agent builder.

Procedure

1. From the Agent Builder, select **File** > **New** > **Other**.
2. Select **Agent Builder Application Support Extension** under **Agent Builder**.
3. Click **Next** to get to the welcome page for the **IBM Tivoli Monitoring Application Support Extension Wizard**.
4. Click **Next** on the welcome page.
5. Enter a name for the project and click **Finish**

Adding support files to a project

Add your support files to an Application Support Extension project

Before you begin

Create an Application Support Extension project. For more information, see “Creating an Application Support Extension project” on page 331.

Procedure

1. Right-click an Application Support Extension project and select **IBM Tivoli** > **Import Application Support Extensions**

© Copyright IBM Corp. 2010, 2021

2. In the **Import Information** window, select the name of the host where the Tivoli Enterprise Portal Server is located or click **Add** to add one.
3. In the **Application** field, enter the agent product code.
4. Enter the affinity of the agent for which you are creating custom application support.

   The agent affinity is a Tivoli Monitoring internal identifier that associates workspaces, queries, and other items, with the agent. It must be unique in the Tivoli Monitoring installation. Click **Browse** to open the **Node Types** window and select this information from a list rather than typing it.
5. When you are satisfied with the import information, click **Finish**.
6. In the **Situations** window, select the situations that you want to import from the Available Situations list.

   Click **<<** to add them to the Selected Situations list and click **OK**. A new folder is created under the project, and it contains the necessary files to install the workspaces, situations, and queries.
7. In the **Queries** window, select the queries that you want to import from the Available Queries list.

   Click **<<** to add them to the Selected Queries list and click **OK**.
8. In the **Take Actions** window, Choose the Take Action commands that you want to import from the Available Take Actions list.

   Click **<<** to add them to the Selected Take Actions list and click **OK**. The support files for the agent are put in the project under the appropriate folder.

What to do next

You can repeat this process for as many different agents as you want. The Agent Builder creates a single installation image from all of the support files in the Application Support Extension project.

Generating the Application Support Extension installation image

Generate an Application Support Extension installation image.

Procedure

1. Right-click on the Application Support Extension project and select **IBM Tivoli** > **Create Application Support Extension Install Image**.
2. In the **Application Support Extension Information** window, enter the directory where the image is to be placed.
3. Your Application Support Extension must have its own product code. Enter the registered product code for your new agent. You can use one of the product codes that are reserved for use with the Agent Builder. The allowed values are `K00-K99`, `K{0-2}{A-Z}`, and `K{4-9}{A-Z}`.

   Note: These values are for internal use only and are not intended for agents that are to be shared or sold. If you are creating an agent to be shared with others, you must send a note to toolkit@us.ibm.com to reserve a product code. The request for a product code must include a description of the agent to be built. A product code is then assigned, registered, and returned to you. When you receive the three-letter product code, you are told how to enable the Agent Builder to use the assigned product code.
4. Enter the name of the Application Support Extension.
5. Enter a description of the Application Support Extension.
6. Enter a version for the Application Support Extension in the VVRRMMFF format where `vv` = version number; `rr` = release number; `mm` = modification number (fix pack number); and `ff` = interim fix number.
7. Click **Finish**.

Installing your Application Support Extension

Install your Application Support Extension

Procedure

1. Transfer your image to your Tivoli Enterprise Monitoring Server and Tivoli Enterprise Portal Server servers.
2. To install the Tivoli Enterprise Monitoring Server support, run one of the following commands:
   - On Windows: `install`*KXX*`TEMSSupport.bat`
   - On UNIX: `install`*KXX*`TEMSSupport.sh`

   The format for the command is as follows:

   ```
   installKXXTEMSSupport[.bat | .sh] <ITM Install Directory> [-s tems_host]
    [-u tems_user]  \[-p tems_password]
   ```
3. To install the Tivoli Enterprise Portal Server support, run one of the following commands:
   - On Windows: `install`*KXX*`TEPSSupport.bat`
   - On UNIX: `install`*KXX*`TEPSSupport.sh`

   The format for the command is as follows:

   ```
   installKXXTEPSSupport[.bat | .sh] <ITM Install Directory> [-r]
   ```

   where `-r` indicates that the Tivoli Enterprise Portal Server must be restarted after installation

Converting a Solution Install Project to an Application Support Extension project

Convert an existing **Solution Install Project** to an Application Support Extension project

About this task

If you have an existing **Solution Install Project** that you want to convert to an Application Support Extension project, complete the following steps:

Note: In the **Solution Install Project** only Support files are migrated.

Procedure

1. Right-click on the **Solution Install project** and select **IBM Tivoli** > **Convert Solution Install Project**.
2. Enter the name of a new Application Support Extension project or select an existing one from the list
3. Click **Finish**.

Appendix E. Cognos data model generation

Agent Builder can generate a Cognos data model for each agent. Use the data model to import agent information into the Cognos Framework Manager for report creation.

This Cognos data model can be opened and viewed in the Framework Manager, which builds a model package to be published into Tivoli Common Reporting. The data model can also be customized or modified within the Framework Manager before publication.

When a report is created, Agent Builder also allows for a final report package to be imported into the Agent Builder project. This feature enables future agent projects to be generated with the reports that are already part of the agent package. The reports that are packaged as part of the agent installation image can be imported into Tivoli Common Reporting in your production environment.

Note: In this documentation, note the following convention:

- K*xx* or k*xx* refers to the product code given to the agent, for example, k99.
- *dbType* refers to the database that is being used by the Tivoli Data Warehouse, for example, DB2.

Prerequisites to generating a Cognos data model

Complete these tasks before you generate a Cognos data model

About this task

Note:

- These steps must be completed only one time, as all future data models generated with Agent Builder will use this environment.
- It is advisable to create an isolated development environment for agent testing and report creation.

Procedure

1. Install and configure a (“Tivoli Data Warehouse” on page 335).
2. Create tables and Procedures in the Tivoli Data Warehouse.
   a) “Create tables and Procedures in the Tivoli Data Warehouse” on page 335.
   b) “Populating the Tivoli Data Warehouse with the Tivoli Reporting and Analytics Model” on page 338.
3. Install and configure (“Tivoli Common Reporting” on page 338).
4. Install and configure the (“Framework Manager” on page 339).

Tivoli Data Warehouse

About Tivoli Data Warehouse.

To create reports, you need a Tivoli Data Warehouse, a Warehouse Proxy agent, and a Summarization and Pruning agent, to be installed and configured in your environment. For more information, see the *IBM Tivoli Monitoring Installation and Setup Guide*.

Create tables and Procedures in the Tivoli Data Warehouse

Create or alter the `ManagedSystem` Table and Stored Procedure in the Tivoli Data Warehouse

About this task

The generated Cognos data model includes a `ManagedSystem` table which is used to define a `ManagedSystem` dimension. The `ManagedSystem` dimension allows reports to be created that can

© Copyright IBM Corp. 2010, 2021

correlate managed systems. For example, if the agent is a subnode agent, the dimension can be used to determine the subnodes that exist for a specific agent instance.

The `ManagedSystem` table is not created by the Tivoli Data Warehouse. Therefore, when an agent is generated in Agent Builder, SQL scripts are generated for each database platform that will:

- Create the `ManagedSystem` table. Use this script if the table does not exist in the Tivoli Data Warehouse.
- Edit the `ManagedSystem` table. Use this script if the table exists in the Tivoli Data Warehouse. Other reporting products can create the `ManagedSystem` table, but they do not create it with all of the required columns.
- Create a stored procedure that populates the `ManagedSystem` table from tables in the Tivoli Data Warehouse.

Run these scripts one time only.

Running DB2 Scripts to Create tables and Procedures in the Tivoli Data Warehouse

For a DB2 database, use these scripts to create tables in the Tivoli Data Warehouse

Before you begin

The scripts for DB2 are in the following directory:

```
reports/db2/Kxx/reports/cognos_reports/itmkxx/db_scripts
```

Procedure

1. The generated scripts (`create_table.sql`, `alter_table.sql`, and `create_procedure.sql`) all use *itmuser* as the Tivoli Data Warehouse user ID. If *itmuser* is not the Tivoli Data Warehouse user ID in your environment, change all occurrences of *itmuser* to the correct user ID.
2. Connect to the Tivoli Data Warehouse as the Tivoli Data Warehouse User:

   ```
   db2 connect to <Tivoli Data Warehouse alias name> user
   <Tivoli Data Warehouse user id> using  <password>
   ```

3. Determine whether the ManagedSystem table exists:

   ```
   db2 "select count(*) from sysibm.systables where name = 'MANAGEDSYSTEM'
   and creator=upper ('<Tivoli Data Warehouse user id>')"
   ```

4. Create or alter the table.
   - If the query returns `1`, the table exists. Run the alter script:

     ```
     db2 -tvf alter_table.sql
     ```

   - If the query returns `0`, the table does not exist. Run the create script:

     ```
     db2 -tvf create_table.sql
     ```

5. Run the script to create the stored procedure:

   ```
   db2 -td@ -f create_procedure.sql
   ```

Running Oracle Scripts to Create tables and Procedures in the Tivoli Data Warehouse

For an Oracle database, use these scripts to create tables in the Tivoli Data Warehouse

Before you begin

The scripts for Oracle are in the following directory:

```
reports/oracle/Kxx/reports/cognos_reports/itmkxx/db_scripts
```

Procedure

1. The generated scripts (`create_table.sql`, `alter_table.sql`, and `create_procedure.sql`) all use *itmuser* as the Tivoli Data Warehouse user ID. If *itmuser* is not the Tivoli Data Warehouse user ID in your environment, change all occurrences of *itmuser* to the correct user ID.
2. Start `sqlplus`:

   ```
   sqlplus <IBM Tivoli Monitoring user ID>/<password>@
   <Tivoli Data Warehouse SID>
   ```

3. Determine whether the ManagedSystem table exists:

   ```
   select count(*) from user_tables where table_name = 'MANAGEDSYSTEM';
   ```

4. Create or alter the table.
   - If the query returns `1`, the table exists. Run the alter script:

     ```
     @<path to alter_table.sql>;
     ```

   - If the query returns `0`, the table does not exist. Run the create script:

     ```
     @<path to create_table.sql>;
     ```

5. Run the script to create the stored procedure:

   ```
   @<path to create_procedure.sql>;
   ```

Running SQL Server 2005 and 2008 Scripts to Create tables and Procedures in the Tivoli Data Warehouse

Before you begin

The scripts for SQL Server are in the following directory:

```
reports/mssql/Kxx/reports/cognos_reports/itmkxx/db_scripts
```

Procedure

1. The generated scripts (`create_table.sql`, `alter_table.sql`, and `create_procedure.sql`) all use *itmuser* as the Tivoli Data Warehouse user ID. If *itmuser* is not the Tivoli Data Warehouse user ID in your environment, change all occurrences of *itmuser* to the correct user ID.
2. Determine whether the `ManagedSystem` table exists:

   ```
   osql -S <Server> -U <Tivoli Data Warehouse user ID> -P <password> -d
   <Tivoli Data Warehouse database name> -Q "Select count(*)
   from INFORMATION_SCHEMA.TABLES where table_name = 'ManagedSystem'"
   ```

3. Create or alter the table.
   - If the query returns `1`, the table exists. Run the alter script:

     ```
     osql -S <Server> -U <Tivoli Data Warehouse user ID> -P <password> -d
     <Tivoli Data Warehouse database name> -I -n -i <path to alter_table.sql>
     ```

   - If the query returns `0`, the table does not exist. Run the create script:

     ```
     osql -S <Server> -U <Tivoli Data Warehouse user ID> -P <password> -d
     <Tivoli Data Warehouse database name> -I -n -i <path to create_table.sql>
     ```

4. Run the script to create the stored procedure:

```
osql -S <Server> -U <Tivoli Data Warehouse user ID> -P
<password> -d <Tivoli Data Warehouse database name>
-I -n -i <path to create_procedure.sql>
```

Populating the Tivoli Data Warehouse with the Tivoli Reporting and Analytics Model

Use provided database scripts to populate the Tivoli Data Warehouse

About this task

Tivoli Reporting and Analytics Model (TRAM) contains the base-set of knowledge that is common to all reporting packages. TRAM is installed by a set of scripts unique to each database. The necessary scripts for populating each supported database are included in the agent installation image, within the reports directory. Use the following procedure to create Tivoli Reporting and Analytics Model Common Dimensions in Tivoli Data Warehouse.

Procedure

1. Browse to the Tivoli Reporting and Analytics Model database scripts.
2. Extract the agent package.
   - On Windows systems, agent package is k*xx*.zip.
   - On Linux and UNIX systems, agent package is k*xx*.tgz.
3. Go to the appropriate database scripts.
   - DB2 scripts are in the Agent package at:

     ```
     reports/db2/Kxx/reports/cognos_reports/itmkxx/db_scripts
     ```

   - Oracle scripts are in the Agent package at:

     ```
     reports/oracle/Kxx/reports/cognos_reports/itmkxx/db_scripts
     ```

   - Microsoft SQL Server scripts are in the Agent package at:

     ```
     reports/mssql/Kxx/reports/cognos_reports/itmkxx/db_scripts
     ```

4. Run the database scripts to generate the common dimensions within the Tivoli Data Warehouse. Each script set provides a readme file for usage instructions.
5. Verify that the scripts added the following tables to the Tivoli Data Warehouse:

   ```
   "Computer System", WEEKDAY_LOOKUP, MONTH_LOOKUP, TIMEZONE_DIMENSION, TIME_DIMENSION
   ```

Tivoli Common Reporting

Tivoli Common Reporting contains the Cognos Business Intelligence engine, which contains elements to assist with the creation of agent reports.

Tivoli Common Reporting must be installed and configured with a data source that connects to the Tivoli Data Warehouse.

Installing Tivoli Common Reporting

You must install Tivoli Common Reporting. Versions 1.3, 2.1, 2.1.1 or later are supported. For information about installing Tivoli Common Reporting, see Installing Tivoli Common Reporting.

Configuring Tivoli Common Reporting

You must configure Tivoli Common Reporting. For information about configuring Tivoli Common Reporting, see Configuring IBM Tivoli Common Reporting.

Create a data source between the Tivoli Data Warehouse and Tivoli Common Reporting. For more information, see Configuring database connection. Click the appropriate database type. Note the name that is given to the data source. The default is **TDW**.

Note: The data source name must match the name in the **Data source** field of the **Cognos Information** page. For more information about the **Cognos Information** page, see "Cognos information" on page 30.

Framework Manager

Framework Manager is an application that ships with the Tivoli Common Reporting application, but must be installed and configured separately.

Framework Manager is used to view and modify data models and to publish data models to Tivoli Common Reporting

Installing Framework Manager

You must install Framework Manager. Versions 8.4, 8.4.1 or later are supported.

The Framework Manager ships with Tivoli Common Reporting, but must be manually installed. Tivoli Common Reporting 1.3 ships with Framework Manager 8.4. Tivoli Common Reporting 2.1 and 2.1.1 ships with Framework Manager 8.4.1. For information about installing Framework Manager, see Installing Framework Manager in the *Tivoli Common Reporting User's Guide*.

Configuring Framework Manager

You must configure Framework Manager. For information about configuring Framework Manager, see Configuring Framework Manager in the *Tivoli Common Reporting User's Guide*.

Creating reports

Use the Framework Manager to publish the agent model, and Report Studio to begin creating reports.

Before you begin

When the agent is completed, it must be installed into the Tivoli Monitoring environment. In addition, historical collection for the agent must be configured and the agent be run for at least one warehouse upload interval. Summarization must be configured, and the summarization setting choices that are made in Tivoli Monitoring must be identical to the summarization choices made in the Agent Builder. The Summarization and Pruning agent must run at least one time after the agent's data is uploaded to the warehouse.

1. Install, configure, and start your agent.
2. Create and distribute to the agent a historical collection for each attribute group you want to create a report for.

   Note: The warehouse upload interval defaults to daily. However, you might want to shorten this interval.

   For information about configuring historical collection, see Managing historical data in the *IBM Tivoli Monitoring Administrator's Guide*.
3. In Tivoli Monitoring, configure summarization for all of the attribute groups you created historical collections for in Step 2.

   Note: When you configure historical collection and summarization, you must wait enough time for data to end up in the summary tables.

Note: By default, the Summarization and Pruning agent is configured to run one time a day at 2 a.m. You might want to change this setting. For example, you can configure it to run hourly. For information about configuring the Tivoli Data Warehouse, see Setting up data warehousing in the *IBM Tivoli Monitoring Installation and Setup Guide*.

About this task

Generating an agent in Agent Builder creates an entire Framework Manager project, which includes the data model and the Framework Manager project file. Framework Manager can open the project file directly, which opens the data model for modification, customization, or publication.

Procedure

Note: The generated data model for the agent contains all of the summary time dimensions for each attribute group: hourly, daily, weekly, monthly, quarterly, and yearly. The dimensions exist only in the Tivoli Data Warehouse for the agent if summarization and pruning is configured for the agent. And also if the dimensions are selected, and if the Summarization and Pruning agent, created, and populated the tables. Reports can be defined and published into Tivoli Common Reporting that use dimensions that do not exist. Such reports do not function until the summary tables are created by the Summarization and Pruning agent.

1. Open the Agent Data Model in Framework Manager:
   a) Open the Framework Manager.
   b) From the **Welcome** page, click **Open a project**.

      Tip: If you are in Framework Manager, click **Open** from the **File** menu.
   c) Browse to the Agent data model.
      - For DB2:

        ```
        reports/db2/Kxx/model/
        ```
      - For Oracle:

        ```
        reports/oracle/Kxx/model/
        ```
      - For Microsoft SQL Server:

        ```
        reports/mssql/Kxx/model/
        ```
   d) Select the agent project file, K*xx*`.cpf`.

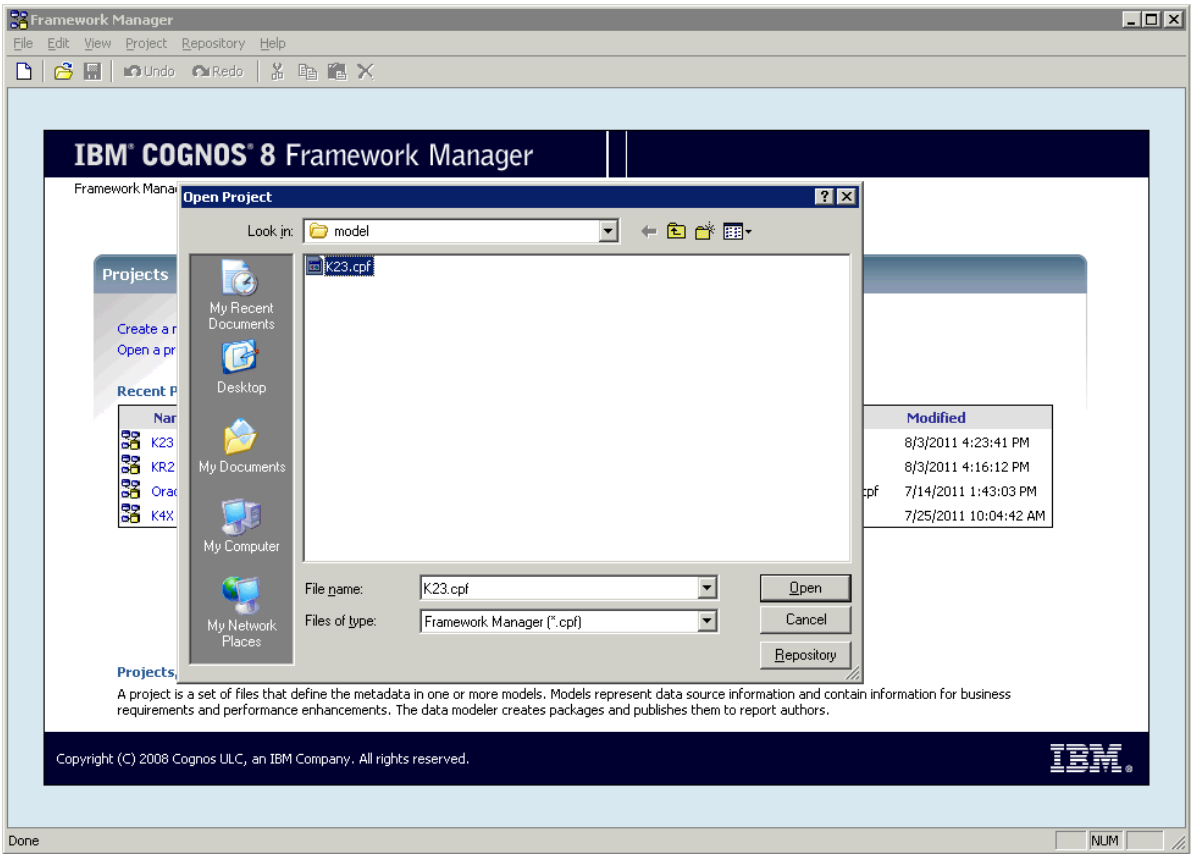


Figure 50. Selecting agent project file

Note: When an agent project is opened in Framework Manager, the agent name is listed under the Recent Projects.

2. Populate the Managed System Table. For more information, see "Populating the ManagedSystem Table" on page 345
3. Use the Framework Manager to publish the Agent Model to Tivoli Common Reporting
   a) Open the Framework Manager.
   b) Open the Agent project.
   c) Expand **Packages** in the navigation tree.
   d) Right-click the agent package and select **Publish Packages**.

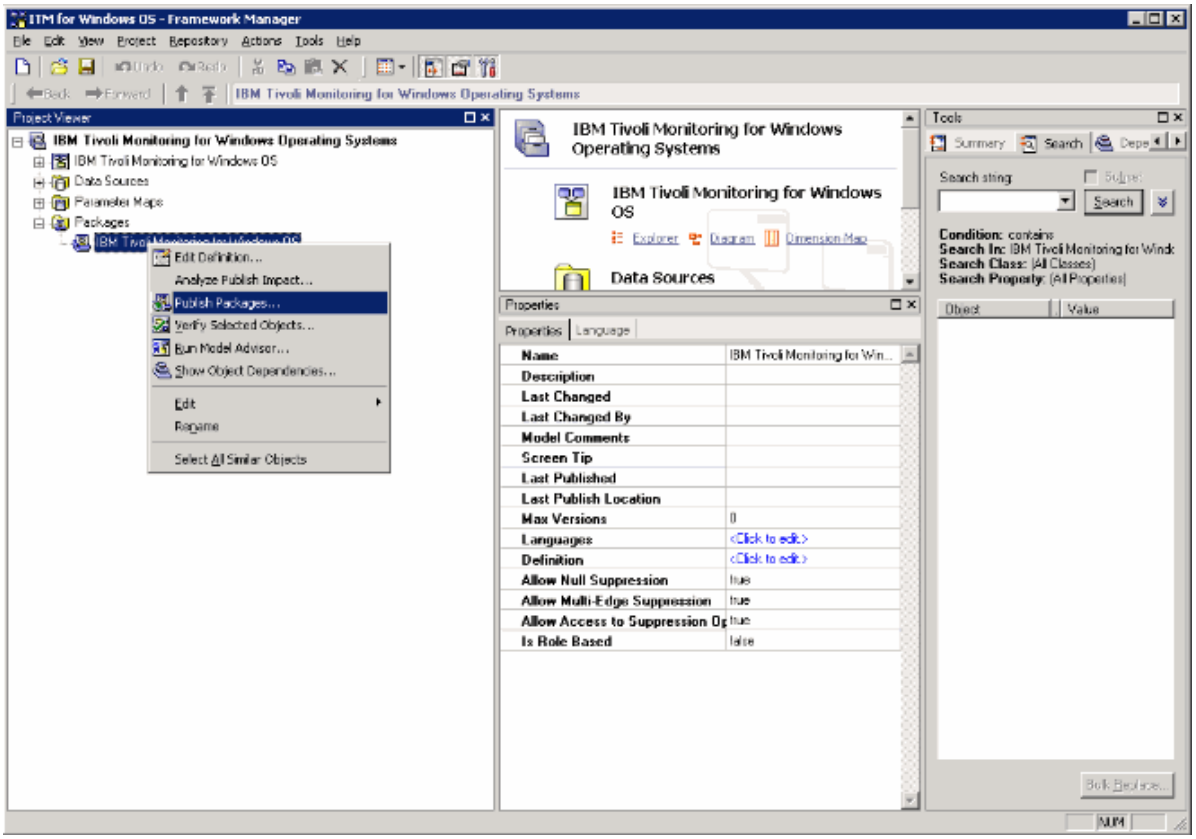


Figure 51. Selecting Publish Packages

4. Use Report Studio to create new reports or templates.
   a) Log on to Tivoli Common Reporting.
   b) Browse to Public Folders, expand **Reporting** in the navigation panel, and select **Common Reporting**.

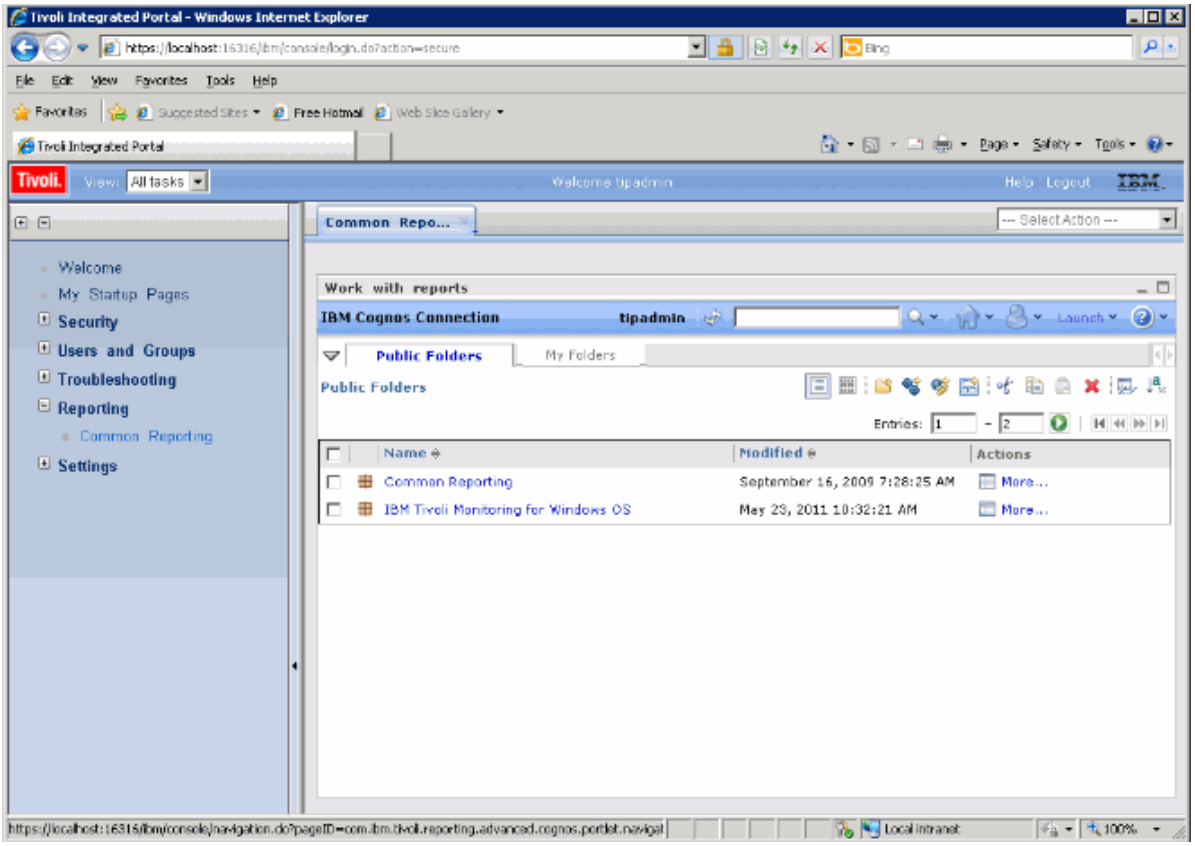


Figure 52. Selecting Common Reporting

c) Select your Tivoli Monitoring agent from the list provided.

d) Open the report creation tool, by clicking the Launch menu and selecting **Report Studio** or **Query Studio**.

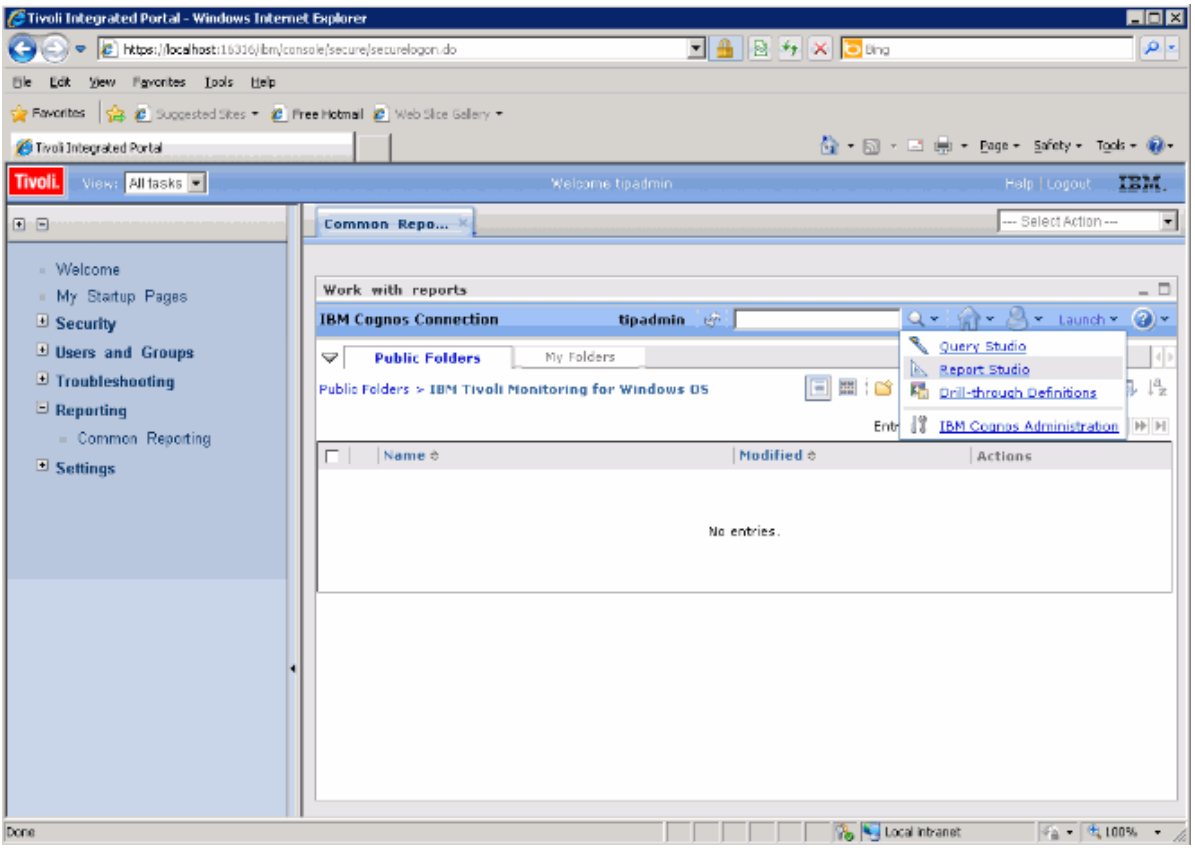


Figure 53. Selecting Report Studio

What to do next

You can use the Report Studio to create new reports or templates, or you can modify an existing report or template.

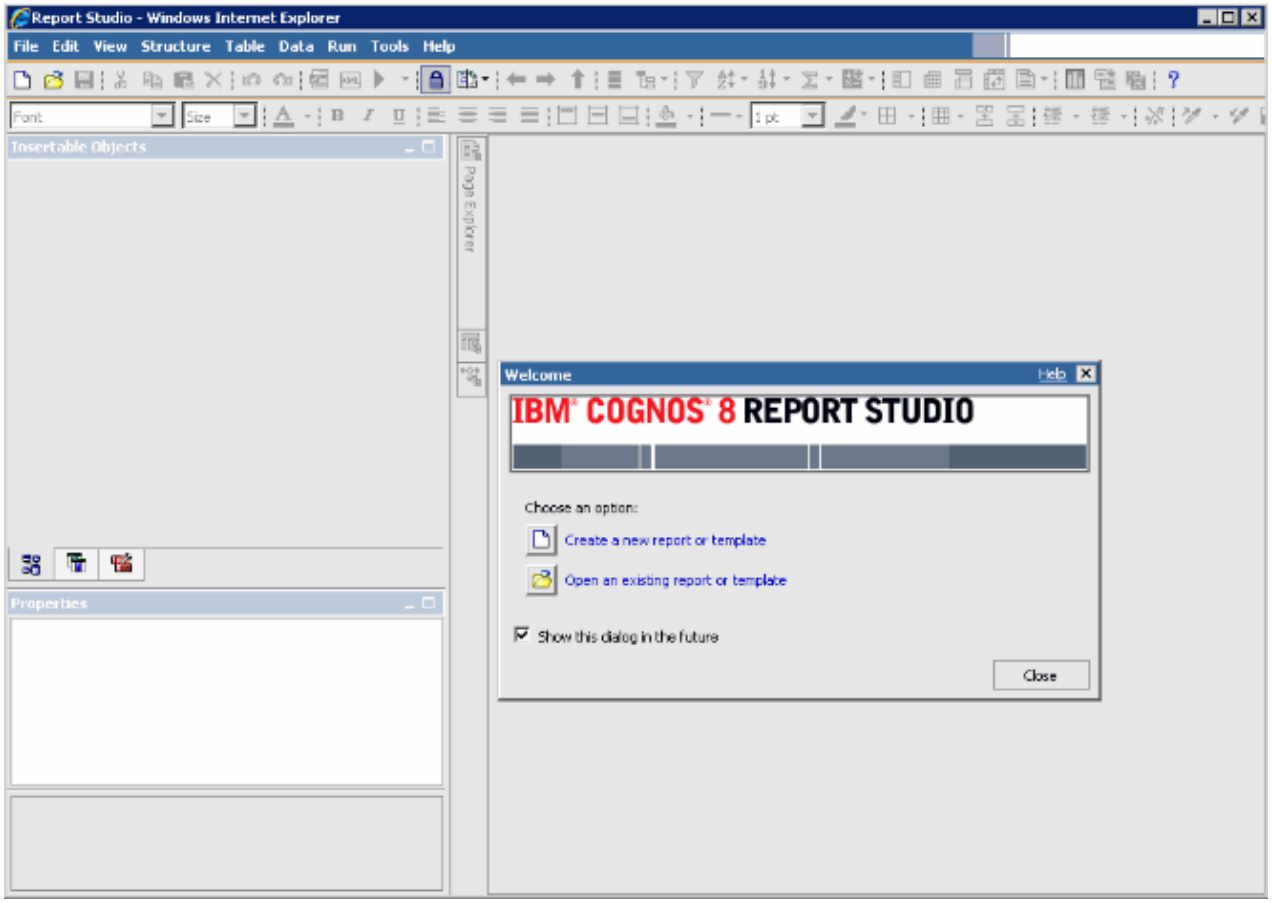


Figure 54. Report Studio

For more information, see the Tivoli Common Reporting topic collection on IBM Knowledge Center.

Populating the ManagedSystem Table

The `ManagedSystem` table is populated by using the `kqz_populate_msn` stored procedure.

For more information, see "Running the DB2 stored procedure" on page 347. This procedure must be run periodically so that the `ManagedSystem` table contains the current list of managed system names.

The stored procedure reads the following historical tables in the Tivoli Data Warehouse if they exist:

- The agent's Performance Object status table
- The agent's availability table. Agents that monitor processes or services have an availability table.
- The agent's discovery tables. Subnode agents create discovery tables.

Historical collection must be started on a particular set of attribute groups. A set of scripts is generated that creates and starts historical collection for these attribute groups. If you do not want to use the scripts, the list of attribute groups is listed in the comment header block of the script.

Sample scripts are created that show which tables must have historical collection enabled:

- `reports/configuretdw.sh`
- `reports/configuretdw.bat`

The following table describes the required arguments:

Note: You must specify either `-n` or `-m`, but not both.

Table 45. Required arguments	
Argument	**Description**
`-h` *candle_home*	The Tivoli Monitoring installation path.
`-u` *teps_user*	The Tivoli Enterprise Portal Server user to log in as when you create the historical collections.
`-n` *tems_name*	The Tivoli Enterprise Monitoring Server where the collections must be started. More than one Tivoli Enterprise Monitoring Server can be specified by using a space separated list. If you specify more than one Tivoli Enterprise Monitoring Server, put the list in quotation marks. For example, `-n "tems1 tems2"`
`-m` *managed_system_group_or_managed_system*	The managed system group or managed system name against which the collection must be started. More than one managed system group or managed system can be specified by using a space separated list. If you specify more than one managed system group or managed system, put the list in quotation marks. For example, `-m "msg1 msg2"`

The following table describes the optional arguments:

Table 46. Optional arguments	
Argument	**Description**
`-s` *teps_host*	The host name or IP address of the Tivoli Enterprise Portal Server. If not specified, the default is localhost.
`-p` *teps_password*	The password for the Tivoli Enterprise Portal Server user that is specified with the `-u` option. If not specified, the script prompts for the password
`-c` *historical_collection_interval*	The historical collection interval to use when you start the historical collections. If not specified, the default is 1h (1 hour). The valid values are: 15m, 30m, 1h, 12hor 1d, where m is minutes, h is hours and d is days.
`-r` *pruning_interval*	The pruning interval to use for the historical data. The historical data must be pruned so that the tables do not continue to grow in size. If not specified, the default is 2d(2 days). Use d for days, m for months, y for years.

After historical collection is started, the `kqz_populate_msn` stored procedure must be run periodically. The stored procedure is run periodically so that the `ManagedSystem` table contains the most current list of managed systems in the Tivoli Monitoring environment.

Running the DB2 stored procedure

Run a stored procedure on DB2.

About this task

Perform the following steps to run the stored procedure on DB2:

Procedure

1. Connect to the Tivoli Data Warehouse database as the warehouse user:

```
connect to <Tivoli Data Warehouse database alias> user
<Tivoli Data Warehouse user id> using <password>
```

2. Run the stored procedure:

```
db2 "call <Tivoli Data Warehouse schema>.kqz_populate_msn
('<three letter product code for the agent>')"
```

Running the Oracle stored procedure

Run a stored procedure on Oracle.

About this task

Perform the following steps to run the stored procedure on Oracle:

Procedure

1. Start `sqlplus`:

```
sqlplus <Tivoli Data Warehouse user id>/<password>@
<Oracle SID>
```

2. Run the stored procedure:

```
execute kqz_populate_msn('<three letter product code for the agent>');
```

Running the stored procedure on SQL Server 2005 and 2008

Run a stored procedure on SQL Server.

About this task

Perform the following steps to run the stored procedure on SQL Server 2005 and 2008:

Procedure

Run the stored procedure:

```
osql -S <server> -U <Tivoli Data Warehouse id> -P
<Tivoli Data Warehouse password> -d
<Tivoli Data Warehouse database name> -Q "EXEC
[<Tivoli Data Warehouse schema>].[kqz_populate_msn]
@pv_productcode = N'<three letter product code>'"
```

Exporting reports and data models from Tivoli Common Reporting

Export reports and data models from Tivoli Common Reporting.

Procedure

1. Log in to the Tivoli Common Reporting.
2. Go to Public Folders, and under **Reporting** in the navigation panel select **Common Reporting**.
3. In the Work with reports section, click the **Launch** menu and select **IBM Cognos Administration**.
4. Click the **Configuration** tab.
5. Click **Content Administration**.

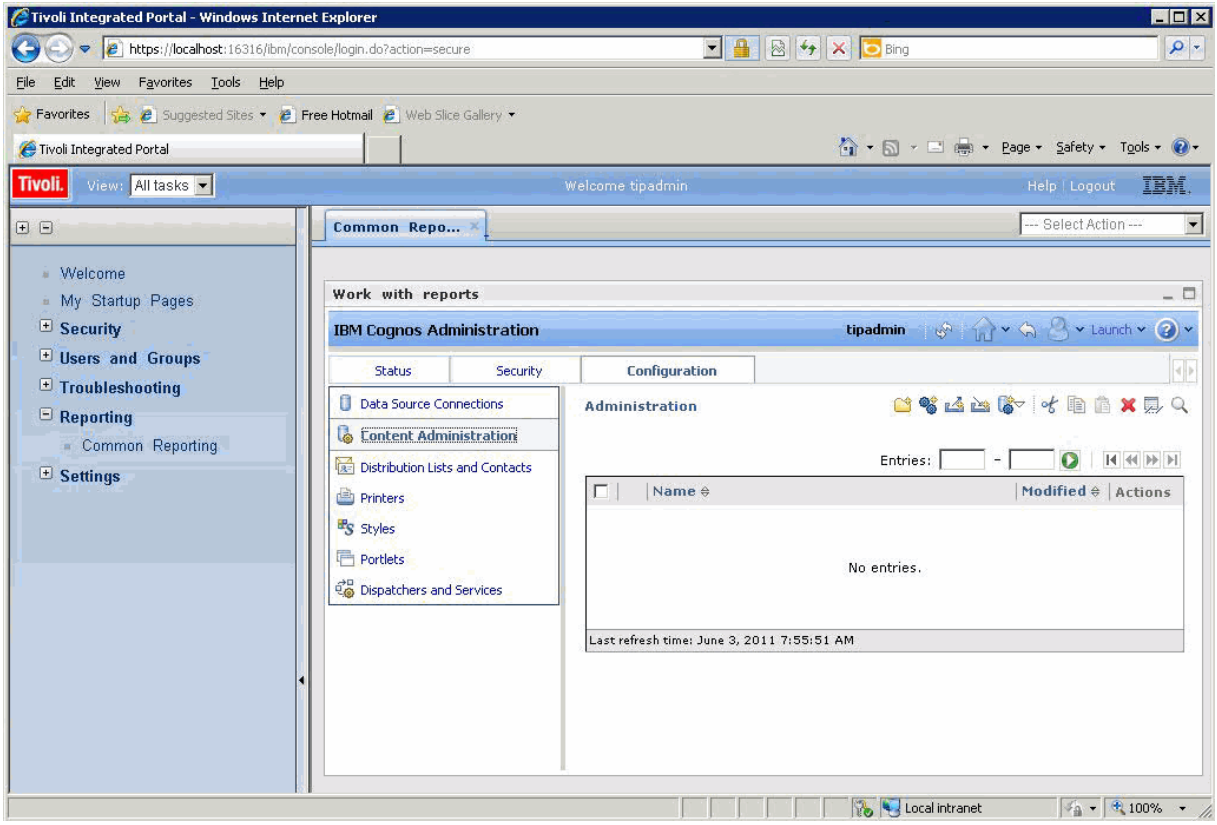


Figure 55. The Content Administration tab

6. Click the **New Export** icon to export a new package.
7. Name the package. Optionally, you can add a screen tip and description.
8. Select **Select public folders and directory content** .
9. In the Public Folders dialog, click the **Add** link.
10. Move your agent package to **Selected entries**.
11. On the last page of the wizard, select **Save Only**. When the wizard completes, the report package is listed on the Content Administration tab.
12. On the Content Administration tab, click the green arrow (Run) to create the compressed `.zip` file.

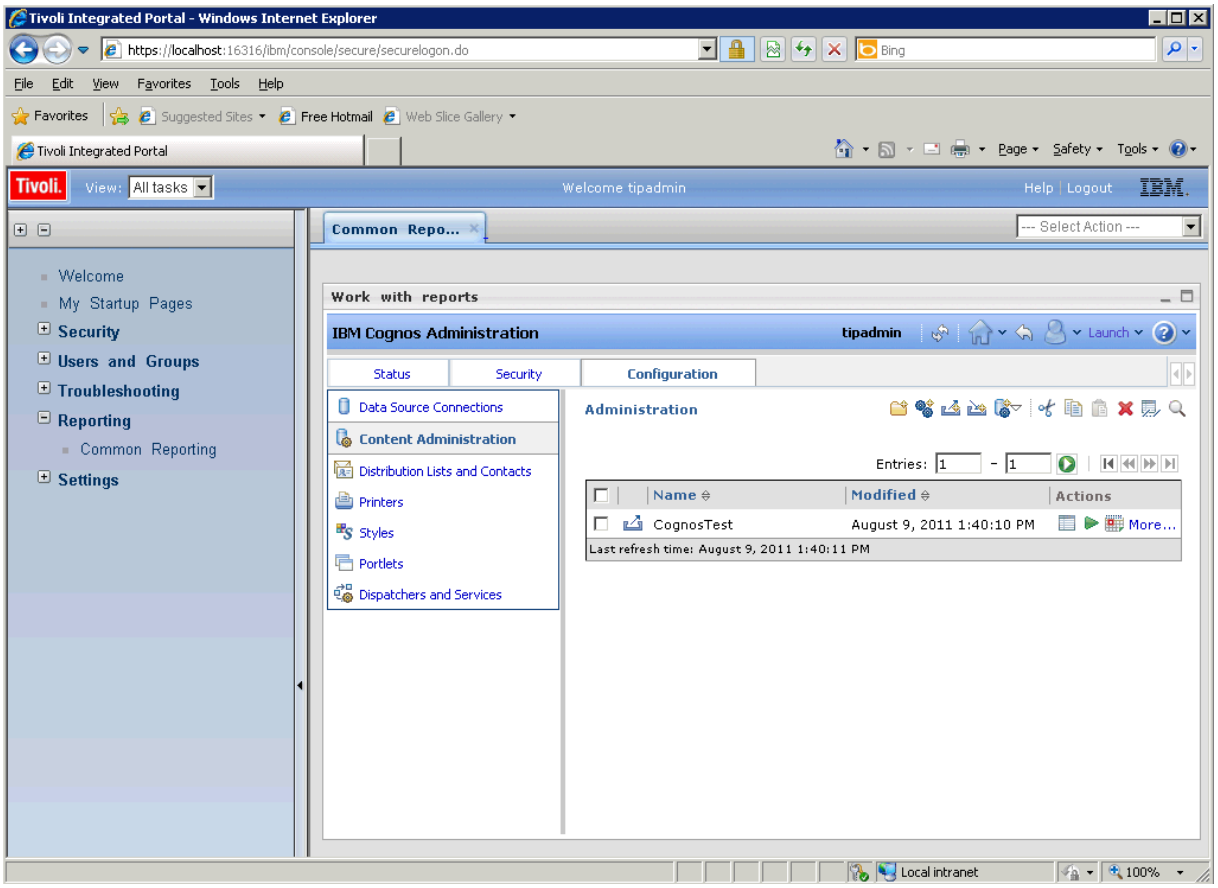


Figure 56. The Content Administration tab with agent package listed

Results

The compressed `.zip` file that is created by the export process is placed in the deployment directory.

- The directory path for Tivoli Common Reporting version 1.3 is:

```
C:\IBM\tivoli\tip\products\tcr\Cognos\c8\deployment
```

- The directory path for Tivoli Common Reporting version 2.1 or later is:

```
C:\IBM\tivoli\tipv2Components\TCRComponent\cognos\deployment
```

What to do next

For more information about exporting reports, see Exporting Cognos report packages in the *Tivoli Common Reporting User's Guide*.

Importing reports into Agent Builder

When the report package is exported from Tivoli Common Reporting, it can be imported into the Agent Builder project. The report package can then be included in the agent installation image.

Procedure

1. Right-click on the agent project in the Agent Builder.
2. Select **IBM** > **Import Report Package**.
3. In the **Import Report Package** window, select the **Database Type** on which the report package was created.
4. Enter the fully qualified path to the report package, or click **Browse** to select it.

5. Click **OK**.
6. The report package is now shown in the agent project under the `reports/`*`dbtype`* directory.

   Note: If you create report packages that are database-specific you must import each package into the Agent Builder.

Installing reports from an agent package into Tivoli Common Reporting

Import a report package from your agent to Tivoli Common Reporting

Procedure

1. Follow the steps in the wizard to import a new package from your agent image.

   In the agent image, the reports are found in: `reports/`*`dbType`*`/K`*`xx`*`/reports/cognos_reports/itmk`*`xx`*`/packages`
2. Copy the reports compressed `zip` file into the Tivoli Common Reporting deployment directory.
   - The directory path for Tivoli Common Reporting version 1.3 is: `C:\IBM\tivoli\tip\products\tcr\Cognos\c8\deployment`
   - The directory path for Tivoli Common Reporting version 2.1 or later is: `C:\IBM\tivoli\tipv2Components\TCRComponent\cognos\deployment`
3. Log in to the Tivoli Common Reporting.
4. Go to Public Folders, and under **Reporting** in the navigation panel select **Common Reporting**.
5. In the Work with reports section, click the **Launch** menu and select **IBM Cognos Administration**.
6. Go to the **Configuration** tab, and open the **Content Administration** section.
7. Click **New Import** to create a package import.
8. Select the agent's reports package.
9. Select the public folders that you want to import.
10. Select save.
11. Click the green (run) arrow to import.

Results

For more information, see Logging in to the reporting interface in the *Tivoli Common Reporting User's Guide*.

Appendix F. ICU regular expressions

A description of the specifics of the ICU regular expression implementation.

This reference content is extracted from the *ICU User Guide*. The content describes the specifics of the ICU regular expression implementation. This information is essential if you are using the Agent Builder regular expression feature because different programming languages implement regular expressions in slightly different ways.

Table 47. Regular expression metacharacters

Character	Description
\a	Match a BELL, \u0007
\A	Match at the beginning of the input. Differs from ^ in that \A does not match after a new line within the input.
\b, outside of a [Set]	Match if the current position is a word boundary. Boundaries occur at the transitions between word (\w) and non-word (\W) characters, with combining marks ignored. For more information about word boundaries, see ICU Boundary Analysis.
\b, within a [Set]	Match a BACKSPACE, \u0008.
\B	Match if the current position is not a word boundary.
\cX	Match a Ctrl-X character.
\d	Match any character with the Unicode General Category of Nd (Number, Decimal Digit.)
\D	Match any character that is not a decimal digit.
\e	Match an ESCAPE, \u001B.
\E	Terminates a \Q ... \E quoted sequence.
\f	Match a FORM FEED, \u000C.
\G	Match if the current position is at the end of the previous match.
\n	Match a LINE FEED, \u000A.
\N{UNICODE CHARACTER NAME}	Match the named character.
\p{UNICODE PROPERTY NAME}	Match any character with the specified Unicode Property.
\P{UNICODE PROPERTY NAME}	Match any character not having the specified Unicode Property.
\Q	Place quotation marks around all following characters until \E.
\r	Match a CARRIAGE RETURN, \u000D.
\s	Match a white space character. White space is defined as [\t\n\f\r\p{Z}].

© Copyright IBM Corp. 2010, 2021

Table 47. Regular expression metacharacters (continued)	
Character	**Description**
\S	Match a non-white space character.
\t	Match a HORIZONTAL TABULATION, \u0009.
\uhhhh	Match the character with the hex value hhhh.
\Uhhhhhhhh	Match the character with the hex value hhhhhhhh. Exactly eight hex digits must be provided, even though the largest Unicode code point is \U0010ffff.
\w	Match a word character. Word characters are [\p{Ll}\p{Lu}\p{Lt}\p{Lo}\p{Nd}].
\W	Match a non-word character.
\x{hhhh}	Match the character with hex value hhhh. From one to 6 hex digits can be supplied.
\xhh	Match the character with 2 digit hex value hh.
\X	Match a Grapheme Cluster.
\Z\	Match if the current position is at the end of input, but before the final line terminator, if one exists.
\z	Match if the current position is at the end of input.
\n	Back Reference. Match whatever the nth capturing group matched. n must be a number > 1 and < total number of capture groups in the pattern. **Note:** Octal escapes, such as \012, are not supported in ICU regular expressions.
[pattern]	Match any 1 character from the set. See UnicodeSet for a full description of what can appear in the pattern
.	Match any character.
^	Match at the beginning of a line.
$	Match at the end of a line.
\	Place quotation marks around the following character. Characters that must have surrounding quotation marks to be treated as literals are * ? + [( ) { } ^ $ | \ . /

Table 48. Regular expression operators	
Operator	**Description**
|	Alternation. A|B matches either A or B.
*	Match 0 or more times. Match as many times as possible.
+	Match 1 or more times. Match as many times as possible.

Table 48. Regular expression operators (continued)

Operator	Description
?	Match zero or 1 time. Prefer one.
{n}	Match exactly n times
{n,}	Match at least n times. Match as many times as possible.
{n,m}	Match between n and m times. Match as many times as possible, but not more than m.
*?	Match 0 or more times. Match as few times as possible.
+?	Match 1 or more times. Match as few times as possible.
??	Match zero or 1 time. Prefer zero.
{n}?	Match exactly n times
{n,}?	Match at least n times, but no more than required for an overall pattern match
{n,m}?	Match between n and m times. Match as few times as possible, but not less than n.
*+	Match 0 or more times. Match as many times as possible when first encountered, do not retry with fewer even if overall match fails (Possessive Match)
++	Match 1 or more times. Possessive match.
?+	Match zero or 1 time. Possessive match.
{n}+	Match exactly n times
{n,}+	Match at least n times. Possessive Match.
{n,m}+	Match between n and m times. Possessive Match.
(...)	Capturing parentheses. Range of input that matched the parenthesized subexpression is available after the match.
(?: ...)	Non-capturing parentheses. Groups the included pattern, but does not provide capturing of matching text. More efficient than capturing parentheses.
(?> ...)	Atomic-match parentheses. First match of the parenthesized subexpression is the only one tried. If it does not lead to an overall pattern match, back up the search for a match to a position before the "(?>".
(?# ...)	Free-format comment (?# comment).
(?= ...)	Look-ahead assertion. True if the parenthesized pattern matches at the current input position, but does not advance the input position.

Table 48. Regular expression operators (continued)	
Operator	**Description**
`(?! ... )`	Negative look-ahead assertion. True if the parenthesized pattern does not match at the current input position. Does not advance the input position.
`(?<= ... )`	Look-behind assertion. True if the parenthesized pattern matches text that precedes the current input position. The last character of the match is the input character just before the current position. Does not alter the input position. The length of possible strings that is matched by the look-behind pattern must not be unbounded (no * or + operators.)
`(?<!...)`	Negative Look-behind assertion. True if the parenthesized pattern does not match text that precedes preceding the current input position. The last character of the match is the input character just before the current position. Does not alter the input position. The length of possible strings that is matched by the look-behind pattern must not be unbounded (no * or + operators.)
`(?ismx-ismx: ... )`	Flag settings. Evaluate the parenthesized expression with the specified flags enabled or disabled.
`(?ismx-ismx)`	Flag settings. Change the flag settings. Changes apply to the portion of the pattern that follows the setting. For example, `(?i)` changes to a not case-sensitive match.

Replacement text

The replacement text for find-and-replace operations can contain references to capture-group text from the find. References are of the form $n, where n is the number of the capture group.

Table 49. Replacement text characters	
Character	**Description**
$n	The text of the positional capture group n is substituted for `$n.` n must be `>= 0`, and not greater than the number of capture groups. A $ not followed by a digit has no special meaning, and is displayed in the substitution text as itself, a $.
\	Treat this character as a literal, suppressing any special meaning. Backslash escaping in substitution text is required only for `'$'` and `'\'`, but can be used on any other character without adverse effects.

Table 49. Replacement text characters (continued)	
Character	**Description**
$@n	The text of capture group n is substituted for the regular expression that matched capture group n. n must be >= 0, and not greater than the number of capture groups. A $@ not followed by a digit has no special meaning, and is displayed in the substitution text as itself, a $@.
$♯n	The text of the matched capture group n is substituted for $♯n. n must be >= 0, and not greater than the number of matched capture groups. A $♯ not followed by a digit has no special meaning, and is displayed in the substitution text as itself, a $♯.

Flag options

The following flags control various aspects of regular expression matching. The flag values can be specified at the time that an expression is compiled into a RegexPattern object. Or, they can be specified within the pattern itself using the (?ismx-ismx) pattern options.

Table 50. Flag options		
Flag (pattern)	**Flag (API constant)**	**Description**
i	UREGEX_CASE_INSENSITIVE	If set, matching take place in a case-insensitive manner.
x	UREGEX_COMMENTS	If set, white space and ♯comments can be used within patterns.
s	UREGEX_DOTALL	If set, a "." in a pattern matches a line terminator in the input text. By default, it does not. A carriage-return / line-feed pair in text behaves as a single-line terminator, and matches a single "." in a RE pattern
m	UREGEX_MULTILINE	Control the behavior of "^" and "$" in a pattern. By default these patterns match only at the start and end, respectively, of the input text. If this flag is set, "^" and "$" also match at the start and end of each line within the input text.

Appendix G. Creating Non-agent file bundles

You can create file bundles that can be placed in the Tivoli Monitoring depot. These file bundles can then be deployed to target systems in your environment.

About this task

With this function, you can remotely configure products for which there is no remote configuration option. To use this function, you place pre-populated configuration files into the depot and send them out to the wanted systems.

Procedure

1. From the Agent Builder, select **File** > **New** > **Other**.
2. Under **Agent Builder**, select **Non-Agent Remote Deploy Bundle**.
3. Click **Next**.
4. In the **Project name** field, enter a name for your project.
5. Click **Next**.
6. Complete the information in the **Remote Deploy Bundle Information** window:
   a) In the **Bundle identifier** field, type an identifier that is a unique alphanumeric string of 3 - 31 characters. This string can contain a hyphen. The string must start with a letter, but it cannot start with a K or a hyphen.
   b) In the **Bundle description** field, type a description of the bundle.
   c) In the **Version** field, type a version for the bundle in the VVRRMMFFF format. Where `vv`= version number; `rr`= release number; `mm`= modification number (fix pack number); and `fff` = interim fix number.
7. In the **Operating Systems** area, select the operating systems to which the bundle can be deployed.
8. Click **Finish** to create a project in the workspace and open the **Remote Deploy Bundle Editor**.

Remote Deploy Bundle Editor

The Remote Deploy Bundle Editor is used to generate commands to help deploy your file bundle.

The Remote Deploy Bundle Editor provides information about the bundle for a project.

The **Bundle Identification Information** section contains the following information:

Bundle identifier
Unique ID for the bundle

Bundle description
Description for the bundle

Bundle version
Version of the bundle

Build
Build identifier for the bundle. Enter a build number here. If no build number is specified, a number is generated from the date and time when the bundle is generated.

Create copy commands for the files in the bundle check box
Click the check box to generate a set of default copy commands that run when the bundle is deployed. The files are copied to the location specified in the **Copy location** text box. The default location is *INSTALLDIR*. Specify this remote deployment variable from the command-line deployment by setting KDY.*INSTALLDIR*=...

The **Operating Systems** section shows the operating systems to which the bundle can be deployed.

© Copyright IBM Corp. 2010, 2021

The **Commands** section shows the commands to run when the bundle is deployed.

Prerequisite Bundles section shows the bundles that must be present for this bundle to work.

Use the Remote Deploy Bundle Editor to opt for a set of default copy commands that copy the files in your bundle to a set location. If this option is selected, then a copy command is generated for each file in your bundle project. The default copy location is *INSTALLDIR*. A special remote deployment variable that, if not set on the deployment command line, defaults to *CANDLEHOME*. To change the location that is specified by *INSTALLDIR*, specify the **KDY.INSTALLDIR** property when you run the **addSystem** command.

The same directory structure that is specified in your bundle project is replicated in *INSTALLDIR*. For example, if there is a folder named `config` in your bundle project with a file named `myprod.config`, then the generated copy command copies the file to *INSTALLDIR*`/config/myprod.config`when the bundle is deployed.

Adding commands to the bundle

You can specify more commands to run during the deployment.

About this task

You can specify more commands to run during the deployment by using the **Remote Deploy Bundle Editor**.

Procedure

1. To specify more commands to run during the deployment, click **Add** in the **Commands** section of the **Remote Deploy Bundle Editor**.
2. In the **Command** window, select the type of command **Preinstall**, **Install**, **Post-Install**, or **Uninstall**and then specify the command to run.

   You must specify the fully qualified path to the command you want to run. For convenience, remote deployment provides a set of predefined variables. To reference the variable for a command, surround the variable with vertical bars, for example, `|DEPLOYDIR|`. For more information about predefined variables for commands, see (Table 51 on page 358).

   Table 51. Predefined Variables for Commands

Variable	Description
DEPLOYDIR	The temporary directory on the endpoint where the bundle is stored during the deployment. For instance, if you want to run `myscript.sh`, a script that is included in your bundle, you specify the following command: `|DEPLOYDIR|/myscript.sh`
INSTALLDIR	Either *CANDLEHOME* or the value of *KDY.INSTALLDIR* if specified on the **addSystem** command.
CANDLEHOME	The Tivoli Monitoring installation directory.

3. Finally, select the **Operating Systems** on which the command is to run.

Adding prerequisites to the bundle

Use the **Remote Deploy Bundle Editor** to specify prerequisites for the bundle.

Procedure

1. To add a prerequisite, click **Add** in the **Prerequisite Bundles** section of the **Remote Deploy Bundle Editor**, **Bundle Information** page.
2. In the **New Prerequisite** window, enter the bundle identifier on which this bundle depends and the minimum version required.
3. Select the operating systems for which this prerequisite is required.
4. Click **OK** to complete and exit.

Adding files to the bundle

Add files to a file bundle by using the **Remote Deploy Bundle Editor**.

Procedure

1. To add files to the remote deployment bundle, do one of the following procedures:
   - In the Bundle Editor, click **Add files to the bundle**.
   - Right-click the project in the Navigator tree, then click **IBM Tivoli Monitoring Remote Deploy** > **Add Files to Bundle**

   Both of these actions display the **Import Bundle Files** window:
2. Specify individual files or directories that contain files in **File Information** area.
3. Click **Finish**.

   The files or directories that are specified are copied into the project directory. The directory structure in the project is maintained when you build the remote deployment bundle. If you want Agent Builder to generate default copy commands, ensure that the files are in the correct directory structure for deployment.

Generating the bundle

Use Agent Builder to generate a bundle for remote deployment of an agent.

Procedure

1. To generate the remote deployment bundle, use one of the following procedures to display the **Generate Final Remote Deploy Bundle** window
   - In the **Remote Deploy Bundle Editor**, click **generate the final Remote deploy bundle**.
   - Right-click the project in the Navigator tree, then click **IBM Tivoli Monitoring Remote Deploy** > **Generate Remote Deploy Bundle**
2. You can now generate the bundle in two ways:
   - If there is a Tivoli Enterprise Monitoring Server on the system where you are running the Agent Builder, click **Install the Remote Deploy bundle into a local TEMS depot**.

     The Agent Builder attempts to determine the Tivoli Monitoring installation location and enter it into the **Directory** field. If *CANDLE_HOME* is not set, the default location of `C:\IBM\ITM` or `/opt/IBM/ITM` is used. Ensure that the installation location is correct before you continue.

     You must provide Tivoli Enterprise Monitoring Server login information to install the bundle.
   - To generate the bundle to a directory on your system, click **Generate the Remote Deploy bundle in a local directory**

After the process is complete, you must transfer this directory to a Tivoli Enterprise Monitoring Server system and use the `tacmd addbundles` command to add the bundle to the depot.

What to do next

When you deploy the bundle, you must use the `tacmd addSystem` command. For example:

```
tacmd addsystem -t MONITORINGCOLLECTION -n Primary:ITMAGT:NT
```

Where `-t` (type) is the Product Code as returned by the `tacmd viewDepot` command:

```
>tacmd viewDepot
Product Code : MONITORINGCOLLECTION
Version : 010000003
Description : MonitoringCollectionScripts
Host Type : WINNT
Host Version : WINNT
Prerequisites:
```

Note: You cannot deploy remotely from the Tivoli Enterprise Portal Desktop or Browser. Deploy remotely from the Tivoli Enterprise Portal Desktop or Browser results in the KFWITM219E message.

See the Tivoli Monitoring documentation for more details.

Creating deployable bundles for Tivoli Netcool/OMNIbus probes

You can use the Agent Builder to create package and configuration bundles that can be used to deploy Tivoli Netcool/OMNIbus probes to remote computers.

About this task

To support the remote deployment of probes, you can also create Tivoli Netcool/OMNIbus bundles that can be deployed to the remote computers before you deploy the probes.

Procedure

1. From the Agent Builder, select **File** > **New** > **Other**.
2. Under **IBM Tivoli OMNIbus Wizards**, select **Package Bundle**.
3. Click **Next**.

What to do next

Next, use the **OMNIbus Install Bundle** wizard to create the bundles. For information about using this wizard, see the Tivoli Netcool/OMNIbus documentation.

Appendix H. Dynamic file name support

Use dynamic file name support to specify a file name pattern instead of an actual file name.

Some application programs create an output file name that is subject to change. The name changes based on specific criteria such as the current day, month, year, or a file name that includes an incrementing sequence number. In these cases, you can specify the file name pattern instead of the actual file name. There are two pattern formats that are recognized when you specify the file name pattern:

- Regular Expressions (preferred).
- IBM Tivoli Universal Agent dynamic file name syntax (deprecated).

Regular expression file name patterns

To specify file name patterns, you can use regular expressions according to the International Components for Unicode (ICU) syntax that is documented in (Appendix F, “ICU regular expressions,” on page 351). To use this capability, you must select the **File names match regular expression** check box on the **Advanced Log File Attribute Group Information** page. When you specify regular expression patterns, you must also select an option from the **When Multiple Files Match** list on the **Advanced Log File Attribute Group Information**page to specify the guidelines for selecting the most current matching file.

Note: Regular expressions is the preferred method to specify file name patterns.

For more information about how to configure advanced log file attribute group properties, see (“Monitoring a log file” on page 104), Step (“6” on page 105). For example, if you specified a file name pattern:

```
d:\program files\logs\tivoli.*
```

This pattern searches for file names that start with `tivoli` in the `d:\program files\logs` directory. Regular expressions can be specified only for the file name portion, and not the path name.

Dynamic file name syntax

With the dynamic file name syntax, only one file at a time can be monitored. The File Data Provider inspects all files in the designated path location, seeking files that match the defined pattern. The File Data Provider always monitors the most current matching file that is based on whichever matching file name has the highest number or date-time value. The appropriate file to monitor is determined by file name, instead of by file creation or other criteria.

Patterns can be specified for file names with any number of parts. For example, `Log{###}` matches on one-part file names such as `Log010` or `Log456`. In multi-part file names, pattern characters can be specified in any part of the file name or in multiple parts. For example, `aaa.bbb{???}.ccc` is a valid pattern, and `aaa.bbb{???}.ccc{###}` is also valid.

Note: Regular expressions rather than dynamic file name syntax is the preferred method to specify file name patterns, for more about regular expressions, see “Regular expression file name patterns” on page 361

The following examples illustrate file name pattern specification:

`{#########}.abc`
Matches numeric file names of length 8 and the file extension *`.abc`*, such as `10252006.abc` or `10262006.abc`. File `10262006.abc` is monitored because `10262006` is greater than `10252006`.

`{#########}.*`
Matches numeric file names of length 8 and ignores the file extension. Examples include `20061025.log`, `20061101.log`, and `10252006.abc`. File `20061101.log` is monitored because `20061101` is the largest number.

© Copyright IBM Corp. 2010, 2021

{######??}.abc
Matches numeric file names of length 8 and file extension `.abc`, and ignores the last two positions in the name portion. Examples include `02110199.abc`, `02110200.abc`, and `021101AZ.abc`. File `02110200.abc` is monitored because `021102` is the largest number.

Console.{######}
Matches file names that contain *Console* in the name portion and a six-digit number in the extension portion. Examples include `Console.000133`, `Console.000201`, and `Console.000134`. File `Console.000201`is monitored.

IN{######}.log
Matches file names that start with `IN` followed by six numerals and the file extension `.log`. Examples include `IN021001.log`, `IN021002.log`, and `IN021004.log`. File `IN021004.log` is monitored.

PS{###}FTP.txt
Matches file names that start with PS followed by three numerals, followed by FTP, and the extension `.txt`. Examples include `PS001FTP.txt`, `PS005FTP.txt`, and `PS010FTP.txt`. File `PS010FTP.txt`is monitored.

Follow these guidelines to establish file name patterns:

- Use braces {} to enclose pattern characters in a file name. The presence of pattern characters inside braces indicates that a file name pattern is being used.
- Use an asterisk (*) as a wildcard to ignore file extensions or any trailing characters in the file name. For example, `Myapp{###}.log*` specifies that any file name that starts with `Myapp`, followed by three digits, and followed by `.log`, is a match, regardless of what comes after.

  The asterisk must be specified after the curly braces `({ })` and cannot be used at the beginning of a file name. When you use the asterisk in a file name extension, the asterisk must be used by itself.

  Examples of correct wildcard `(*)`usage:

  err{??}.*

  error{$}.*

  Examples of incorrect wildcard `(*)` usage:

  error.20*
  No curly braces precede the asterisk `(*)`.

  error*.{###}
  The asterisk is not used at the end of the file name.

  error.*
  No curly braces precede the asterisk `(*)`.
- If a specific file extension is defined, then only files with the same extension are considered.
- Use a number sign to indicate each numeric element of a file name.
- Use a question mark to exclude each element of the naming convention that does not serve as search criteria in determining the appropriate file name.
- Use a dollar sign `($)` to represent either any character or no character. For example, if you want to match on two files named `Log` and `LogA`, specify `Log{$}`. The dollar sign has several usage restrictions. When you use one or more dollar signs to prefix a file name as in `{$$$$$$}_abc.log`, the number of dollar signs must exactly match the number of characters in that position in the file name. Also, you cannot specify dollar signs in multiple locations in a file name pattern, for example, `{$$$}b{$$$}.log` does not match `abc.log`. Given these dollar sign restrictions, use regular expression file name patterns if there are an indeterminate number of characters in the file names.
- The total number of number signs and question marks that are enclosed in braces is significant. It must match the portion of file name exactly. For example, the pattern `AA{####}` instructs the File Data Provider to look for files such as `AA0001`. File names, such as `AA001` or `AA00001`, are not considered.
- The exact file name pattern, the constant, and the numeric parts, must match the file name exactly. For example, the pattern `AA{###}` instructs the File Data Provider to check file `AA101`. File names, such as `XAA101`, `AA222X` and `AA55555`, are not considered.

- Use the reserved pattern string {TIVOLILOGTIME}to substitute for the hex timestamp and file sequence number in a Tivoli Monitoring agent or server log file. This pattern string is useful when you do self-monitoring of Tivoli Monitoring components. For example, if you want to monitor the latest monitoring server log in the /opt/IBM/ITM/logsdirectory, can specify a file name pattern:

  ```
  /opt/IBM/ITM/logs/Host1_ms_{TIVOLILOGTIME}.log
  ```

  If Host1_ms_452053c0-01.log, Host1_ms_451f11f4-01.log, Host1_ms_45205946-01.log, and Host1_ms_451f11f4-02.log are present in the /logsdirectory, the Host1_ms_45205946-01.log file is selected for monitoring.

To precisely specify a file name that consists of date components (year, month, and day), use the capital letters Y, M, and D. These letters must be specified within braces; otherwise they are treated as literal characters in the file name.

See the following examples:

{YYYYMMDD}.log
Specifies file names such as 20060930.log or 20061015.log.

{MMDDYY}.log
Specifies file names such as 101106.log or 110106.log.

{DDMMYYYY}.log
Specifies file names such as 01092006.log or 15082006.log.

{DDMMMYY}.log
Specifies file names such as 24Jan07 or 13Sep06.

{MM-DD-YY}.log
Specifies file names such as 11-02-06 or 04-29-07. The (-)separator character is ignored in the date field and does not require a question mark pattern character to skip over it.

MY{YYDDD}.log
Specifies file names such as MY06202.log, MY06010.log, or MY04350.log.

Complex cases exist, where a date field is embedded within a longer file name, and the date patterns in the previous examples are not sufficient. For complex cases, create patterns that mix number signs and question marks and still perform numeric comparisons that select the most current file for monitoring. For example, the pattern ABC{?####?##?##?##?##?##?}XYZ.TXT can be used for file names such as ABC 2006-04-20 11_22_33 XYZ.TXT. In this example, you are interested in only the #- marked digits and question marks serve as placeholders that ignore other characters in the file name.

The File Data Provider periodically checks for new files that match the defined file pattern in the target path location. When a newer file that matches the pattern is detected, the File Data Provider automatically switches application monitoring to the new file. The File Data Provider searches for the best matching file when:

- The File Data Provider first starts.
- The currently monitored file no longer exists because of possible renaming or deleting.
- The existing file contents, changed because of possible rewriting.
- The check interval expired. The default interval is 10 minutes. You can change the interval to a longer or shorter interval value by specifying the environment variable

  ```
  KUMP_DP_FILE_SWITCH_CHECK_INTERVAL=number-of-seconds
  ```

Appendix I. SNMP trap configuration

Description of the configuration file that is used by the SNMP Data Provider to render trap information in a more easily readable form. The file is also used to assign categories, severities, status, and source IDs to traps.

Also contains instructions for modifying the default file or substituting your own configuration file.

SNMP trap configuration file, `trapcnfg`

At startup, the SNMP Data Provider reads a configuration file named `trapcnfg`. One purpose of this file is to translate SNMP trap information into a more readable form. Another is to assign categories, severities, status, and source IDs to specific traps, since these categories are not defined by SNMP.

You can modify the `trapcnfg` file to suit your site-specific needs by adding new trap or enterprise definitions or changing the existing ones. You can also use your own configuration file.

Use the HP OpenView `trapd.conf` file

The `trapcnfg` file is similar in format, but not identical, to the HP OpenView Network Node Manager trap configuration file `trapd.conf`. You can copy the OpenView file and reuse many of the definition statements if necessary.

Types of records

`trapcnfg` contains three types of records or record blocks:

comments
: Comment records begin with a number sign (#).

enterprise definitions
: Enterprise definitions consist of two blank-delimited tokens, where the first token is a name and the second is an object identifier (OID) surrounded by curly brackets ({ }).

trap definitions
: Trap definitions consist of eight blank-delimited tokens. Trap definitions are block records because each definition might consist of multiple records.

The first type is self-explanatory. (Figure 57 on page 366) shows examples of the second and third types.

The first example in Figure 57 on page 366 shows an enterprise definition record which defines enterprise OID 1.3.6.1.4.1.311.1.1.3.1.1 as being Microsoft Windows NT.

The second example shows a trap definition record that defines `trapName MSNTCOLD` as being associated with enterprise OID `1.3.6.1.4.1.311.1.1.3.1.1`, generic trap number 0, and specific trap number 0. Notice that the severity is in decimal form whereas the category is in textual form. Severities are translated into their textual form before they are displayed. The next record in the type 3 record block is the short description, which the Agent Builder does not use. The Agent Builder uses the long description that is enclosed within the delimiters SDESC and EDESC.

© Copyright IBM Corp. 2010, 2021

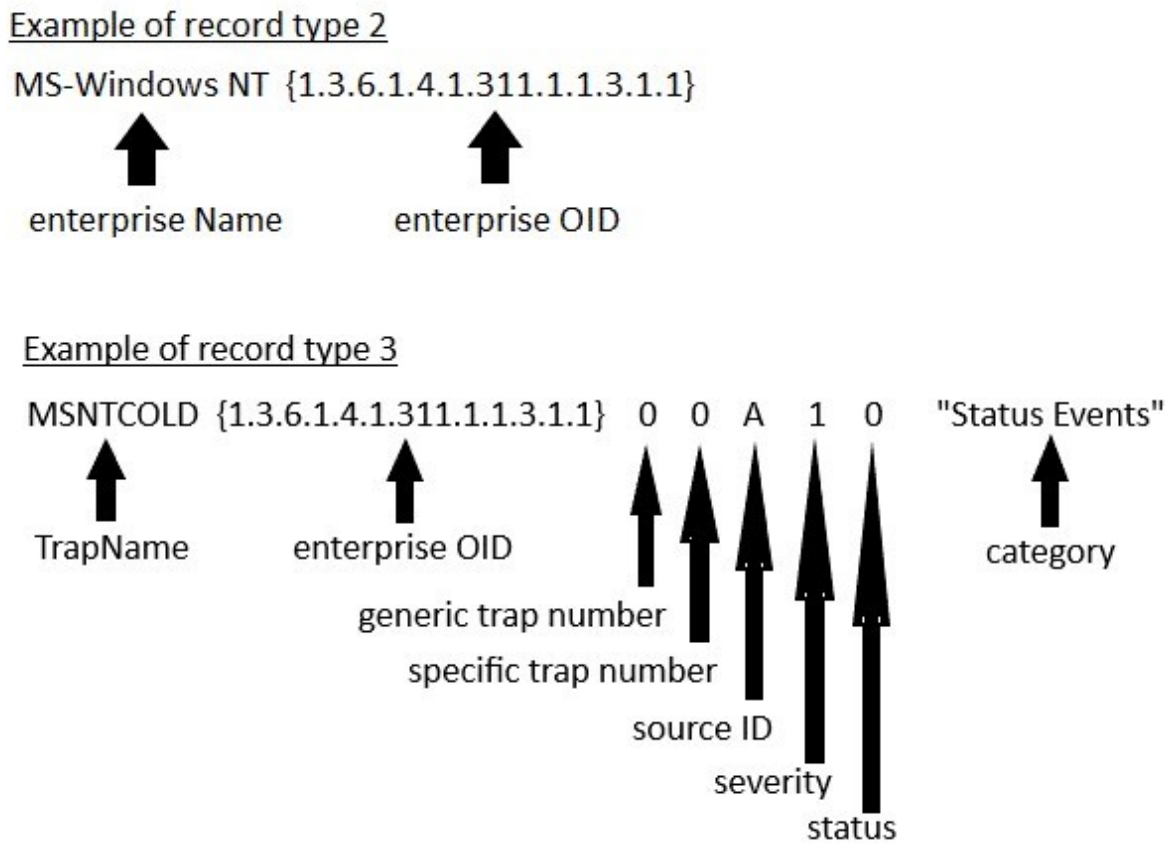


Figure 57. Examples of configuration record types 2 and 3

Defaults for the `trapcnfg` file

Tables that list the defaults that are supported by the SNMP Data Provider.

Supported categories

(Table 52 on page 366) shows the categories that are supported by the Agent Builder.

Table 52. Categories supported by the SNMP Data Provider

Category	Textual representation
0	`Threshold Events`
1	`Network Topology Events`

Table 52. Categories supported by the SNMP Data Provider (continued)	
Category	**Textual representation**
2	Error Events
3	Status Events
4	Node Configuration Events
5	Application Alert Events
6	All Category Events
7	Log Only Events
8	Map Events
9	Ignore Events

(Table 53 on page 367) lists the severities that are supported by the Agent Builder.

Table 53. Severities supported by the SNMP Data Provider	
Severity	**Textual representation**
0	Clear
1	Indeterminate
2	Warning
3	Minor Error
4	Critical
5	Major Error

Supported statuses

(Table 54 on page 367) shows the statuses that are defined in the Agent Builder configuration file.

Table 54. Statuses supported by the SNMP Data Provider	
Status	**Textual representation**
0	Unchanged
1	Unknown
2	Up
3	Marginal
4	Down
5	Unmanaged
6	Acknowledge
7	User1
8	User2

Supported source IDs

(Table 55 on page 368) lists the source IDs supported by `trapcnfg`.

Table 55. Source IDs supported by the SNMP Data Provider	
Source ID	**Description**
a	Application
A	Agent
C	Xnmcollect
d	Demo
D	Data Collector
E	Nvevents
I	Ipmap
L	LoadMIB
m	Shpmon
M	IP topology
n	netmon related
N	netmon-generated traps
O	OSI SA
P	Non-IP traps
r	Tralertd
s	Spappld
S	Security Agent
t	Xnmtrap
T	Trapd
V	Vendor related
?	Unknown

Appendix J. Take Action commands reference

An overview of Take Action commands, references about Take Action commands, and descriptions of special Take Action commands.

About Take Action commands

Take Action commands can be included in an Agent Builder monitoring agent. Take Action commands can be run from the portal client or included in a situation or a policy. When included in a situation, the command runs when the situation becomes true. A Take Action command in a situation is also known as reflex automation. When you enable a Take Action command in a situation, you automate a response to system conditions. For example, you can use a Take Action command to send a command to restart a process on the managed system. You can also use a Take Action command to send a text message to a cell phone.

Advanced automation uses policies to run actions, schedule work, and automate manual tasks. A policy comprises a series of automated steps that are called activities that are connected to create a workflow. After an activity is completed, the Tivoli Enterprise Portal receives return code feedback, and advanced automation logic responds with subsequent activities prescribed by the feedback.

A basic Take Action command displays the return code of the operation in a message box or a log file that is displayed after action completion. After you close this window, no further information is available for this action.

More information about Take Action commands

For more information about working with Take Action commands, see the *Tivoli Enterprise Portal User's Guide*.

For a list and description of the Take Action commands for this monitoring agent, see ("Special Take Action commands" on page 369). See also the information in that section for each individual command.

Special Take Action commands

An Agent Builder monitoring agent can recognize and do special processing for a set of Take Action commands:

- SSHEXEC

For more information about creating these commands and including them in an Agent Builder monitoring agent project, see (Chapter 11, "Creating workspaces, Take Action commands, and situations," on page 213).

SSHEXEC action

Before you begin

For more information about Take Action commands, see (Appendix J, "Take Action commands reference," on page 369).

About this task

The SSHEXEC action is recognized for a monitored application that has at least one SSH Script attribute group. It indicates that the command that follows the SSHEXEC keyword is remotely started on the SSH target system. The command is started with the credentials and privileges of the user that is configured to monitor the SSH target system. The command is run on the remote system that is represented by the Managed System Name.

© Copyright IBM Corp. 2010, 2021

Procedure

To include the Take Action command in a situation or workflow policy, use the following syntax for the system command:

```
SSHEXEC [Command]
```

For example:

```
SSHEXEC [ls &path]
```

Note: You can customize the command or portions of the command during invocation of the Take Action by using the Take Action arguments option with the *Command*.

Note: If the *Command* includes multiple arguments, then consider including the bracket parenthesis to enable invocation of the Take Action command with the **tacmd** command-line interface.

Accessibility features

Accessibility features assist users who have a disability, such as restricted mobility or limited vision, to use information technology content successfully.

Accessibility features

The web-based interface of IBM Cloud Application Performance Management is the Cloud APM console. The console includes the following major accessibility features:

- Enables users to use assistive technologies, such as screen-reader software and digital speech synthesizer, to hear what is displayed on the screen. Consult the product documentation of the assistive technology for details on using those technologies with this product.
- Enables users to operate specific or equivalent features using only the keyboard.
- Communicates all information independently of color.[1]

The Cloud APM console uses the latest W3C Standard, WAI-ARIA 1.0 (http://www.w3.org/TR/wai-aria/), to ensure compliance with US Section 508 (http://www.access-board.gov/guidelines-and-standards/communications-and-it/about-the-section-508-standards/section-508-standards), and Web Content Accessibility Guidelines (WCAG) 2.0 . To take advantage of accessibility features, use the latest release of your screen reader in combination with the latest web browser that is supported by this product.

The Cloud APM console online product documentation in IBM Knowledge Center is enabled for accessibility. The accessibility features of IBM Knowledge Center are described at IBM Knowledge Center release notes .

Keyboard navigation

This product uses standard navigation keys.

Interface information

The Cloud APM console web user interface does not rely on cascading style sheets to render content properly and to provide a usable experience. However, the product documentation does rely on cascading style sheets. IBM Knowledge Center provides an equivalent way for low-vision users to use their custom display settings, including high-contrast mode. You can control font size by using the device or browser settings.

The Cloud APM console web user interface includes WAI-ARIA navigational landmarks that you can use to quickly navigate to functional areas in the application.

The Cloud APM console user interface does not have content that flashes 2 - 55 times per second.

Related accessibility information

In addition to standard IBM help desk and support websites, IBM has established a TTY telephone service for use by deaf or hard of hearing customers to access sales and support services:

TTY service 800-IBM-3383 (800-426-3383) (within North America)

IBM and accessibility

For more information about the commitment that IBM has to accessibility, see IBM Accessibility (www.ibm.com/able).

[1] Exceptions include some **Agent Configuration** pages of the Performance Management console.

© Copyright IBM Corp. 2010, 2021

Notices

This information was developed for products and services offered in the U.S.A. IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785 U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan, Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement might not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
2Z4A/101
11400 Burnet Road
Austin, TX 78758 U.S.A.

© Copyright IBM Corp. 2010, 2021

Such information may be available, subject to appropriate terms and conditions, including in some cases payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurement may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. You may copy, modify, and distribute these sample programs in any form without payment to IBM for the purposes of developing, using, marketing, or distributing application programs conforming to IBM‘s application programming interfaces.

If you are viewing this information in softcopy form, the photographs and color illustrations might not display.

Trademarks

IBM, the IBM logo, and ibm.com® are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at http://www.ibm.com/legal/us/en/copytrade.shtml.

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

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

Microsoft and Windows are registered trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Other company, product, and service names may be trademarks or service marks of others.