

Course Guide

IBM Agent Builder 6.3.4

Course code TV384 ERC 1.0



June 2017 edition

NOTICES

This information was developed for products and services offered in the USA.

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 grant you any license to these patents. You can send license inquiries, in writing, to:

*IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC1_1_9
Armonk, NY 1_0504-1_785
United States of America*

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 may 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 websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites 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.

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.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

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 "Copyright and trademark information" at www.ibm.com/legal/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.

Cell Broadband Engine is a trademark of Sony Computer Entertainment, Inc. in the United States, other countries, or both and is used under license therefrom.

Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

IT Infrastructure Library is a Registered Trade Mark of AXELOS Limited.

ITIL is a Registered Trade Mark of AXELOS Limited.

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

Linear Tape-Open, LTO, the LTO Logo, Ultrium, and the Ultrium logo are trademarks of HP, IBM Corp. and Quantum in the U.S. and other countries.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are 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.

© Copyright International Business Machines Corporation 2017.

This document may not be reproduced in whole or in part without the prior written permission of IBM.

US Government Users Restricted Rights - Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

About this coursexi
About the studentxii
Learning objectivesxiii
Course agendaxiv
Unit 1 Introduction to IBM Agent Builder	1-1
Learning objectives	1-2
Unit outline	1-2
Lesson 1 Introduction to Agent Builder	1-3
Abilities of Agent Builder agents	1-4
Create agents for IBM Tivoli Monitoring	1-5
Create agents for IBM Application Performance Management	1-6
Gather a wide range of data	1-7
Mix, match, and manipulate data	1-8
Establish custom runtime configuration	1-9
Lesson 2 Installing Agent Builder	1-10
Installation process	1-11
Silent installation and upgrading	1-12
Lesson 3 Introduction to the Agent Builder application	1-13
Eclipse technology	1-14
Starting Agent Builder	1-15
Creating workspaces	1-16
Application window	1-17
Directories and files	1-18
Wizards, browsers, and testers	1-19
Help	1-20
Agent Builder resources	1-21
Summary	1-22
Unit 2 Agent creation basics	2-1
Scenario business objectives	2-2
Scenario solution	2-3
Learning objectives	2-4
Unit outline	2-4
Lesson 1 Creating an agent in Agent Builder	2-5
Components of an IBM agent and monitoring environment	2-6
IBM agent components in Agent Builder	2-7
Agent creation process	2-9
Starting the agent creation wizard	2-10
Three basic components to define	2-11
Define the agent information	2-12

Define the agent information (continued)	2-13
Select the data source	2-15
Identify the data to gather	2-16
Set runtime configuration	2-17
Lesson 2 Monitoring Windows services	2-18
Introduction to availability monitoring	2-18
Selecting Windows services to monitor	2-20
Browsing for Windows services	2-21
Lesson 3 Troubleshooting in Agent Builder	2-22
Problems pane	2-23
Agent Builder trace files	2-24
Setting Agent Builder trace options	2-25
Classroom computer architecture	2-26
Student exercises	2-27
Lesson 4 Creating installers and installing an Agent Builder agent	2-28
Agent output and installation overview	2-29
Install agent locally	2-30
Installing ITM application support locally	2-31
Creating compressed installers	2-32
How to install by using installation scripts	2-33
Windows installation files	2-34
UNIX and Linux installation files	2-36
Lesson 5 Troubleshooting Agent Builder agents	2-38
Installation log files	2-39
Agent log files	2-40
Agent tracing	2-41
Summary	2-42

Unit 3.1 Customizing agents for IBM Application Performance Management	3-1
Scenario business objectives	3-2
Application Performance Management scenario solution	3-3
Learning objectives	3-4
Unit outline	3-5
Lesson 1 Modifying an agent for an IBM Application Performance Management environment	3-6
Defining the summary dashboard	3-7
Defining details dashboards	3-8
Defining the monitored resource	3-9
.	3-10
Lesson 2 Creating filtered data sources	3-11
Adding a filtered data source	3-12
Setting attribute severity	3-12
Lesson 3 Managing agents in an IBM Application Performance Management environment	3-13
Configuring, starting, stopping, and uninstalling agents	3-14
Confirming agents in the Performance Management Console	3-15
Adding an Agent to an application	3-16
Confirming the component summary dashboard	3-17
Confirming the component details dashboard	3-18

APM agent creation process summary	3-19
Student exercises	3-20
Summary	3-21
Unit 3.2 Customizing agents for IBM Tivoli Monitoring.....	3-22
Scenario business objectives	3-23
ITM scenario solution	3-24
Learning objectives	3-25
Unit outline	3-25
Lesson 1 Managing agents in an IBM Tivoli Monitoring environment	3-26
Configuring, starting, and stopping agents	3-27
Uninstalling ITM agents	3-28
Confirming a new agent in IBM Tivoli Monitoring	3-29
Confirm the data display in the Tivoli Enterprise Portal client	3-30
Student exercises	3-31
Lesson 2 Creating and importing IBM Tivoli Monitoring application support	3-32
Creating workspaces, queries, situations, and Take Actions	3-33
Creating importable workspaces	3-34
Publishing workspaces	3-35
Importing application support	3-36
Common problems with adding application support	3-37
Complete ITM agent creation process summary	3-38
Student exercise	3-39
Summary	3-40
Unit 4 Monitoring Windows resources	4-1
Scenario business objectives	4-2
Scenario solution	4-3
Learning objectives	4-4
Unit outline	4-5
Lesson 1 Monitoring Windows Management Instrumentation (WMI)	4-6
Introduction to performance monitors	4-7
Introduction to Windows Management Instrumentation (WMI)	4-8
WMI terminology	4-9
Adding WMI monitoring to an agent	4-10
Browsing WMI	4-12
Lesson 2 Monitoring Windows Performance Monitor	4-13
Introduction to Windows Performance Monitor (Perfmon)	4-14
Adding Perfmon monitoring to an agent	4-15
Browsing Perfmon objects	4-16
Monitoring object instances	4-17
Lesson 3 Monitoring Windows log events	4-18
Filtering events	4-19
Lesson 4 Common agent modifications	4-20
Selecting single data row or key attributes	4-21
Modifying data sources after the wizard	4-22
Availability	4-23
Windows event logs	4-24

Attributes that can be partially modified	4-25
Attributes that can be fully modified	4-26
Creating navigator groups	4-27
Steps for creating navigator groups	4-28
Joining attribute groups	4-29
Join results: Single-to-single join	4-31
Join results: Single-to-multiple-row join	4-32
Join results: Multiple-row-to-multiple-row join	4-33
Enabling self-describing agent	4-34
Enabling agent watchdog	4-35
Committing an agent	4-36
Commit process	4-37
Lesson 5 Testing an attribute group and full agent	4-38
Testing the full agent	4-39
Student exercises	4-40
Summary	4-41
Unit 5 Monitoring processes and command return codes	5-1
Scenario business objective	5-2
Scenario solution	5-3
Learning objectives	5-4
Unit outline	5-5
Lesson 1 Monitoring processes	5-6
Process monitoring	5-7
Adding a process	5-8
Browsing processes	5-9
Defining remote connections	5-10
Lesson 2 Monitoring command return codes	5-11
Adding a command return code data source	5-12
Defining commands	5-13
Defining return codes	5-14
Student exercises	5-16
Summary	5-17
Unit 6 Monitoring custom data sources	6-1
Scenario business objectives	6-2
Scenario solution: Script	6-3
Scenario solution: Socket	6-4
Scenario solution: Log file	6-5
Scenario solution: Java API	6-6
Learning objectives	6-7
Unit outline	6-8
Lesson 1 Monitoring script output	6-9
Command or script output	6-10
Adding script monitoring to an agent	6-11
Remote script command information	6-12
Remote script (SSH)	6-13
Testing a command	6-14

Lesson 2 Defining custom attributes	6-15
Scale	6-17
Range	6-18
Enumerations	6-19
Lesson 3 Creating derived attributes	6-20
Operators	6-21
Derived attribute functions	6-22
List of derived attributes functions	6-23
Function examples	6-26
Conditional expressions	6-27
Formula Editor	6-28
Lesson 4 Monitoring through a socket connection	6-29
Defining the socket attribute group and attributes	6-30
Defining error codes	6-31
Adding supplemental files	6-32
Configuring the listening port and remote connections	6-33
Sending data	6-34
More data samples	6-35
Notes on coding data	6-36
Formatting attributes	6-37
Take Actions	6-38
Lesson 5 Generating agent output from the command line	6-39
Agenttoolkit command	6-40
Command syntax	6-41
Student exercises	6-43
Lesson 6 Monitoring log files	6-44
Process overview	6-45
Specifying the log file to monitor	6-46
Dynamic file name support	6-47
Dynamic file name examples	6-48
Regular expressions for files names	6-49
Regular expressions	6-50
Defining record identification	6-51
Defining record identification rule	6-52
Defining how to process the log file	6-53
Defining global field identification	6-55
Testing log file settings	6-57
Defining basic attribute field identification	6-58
Defining advanced attribute field identification	6-59
Filtering records with attribute filters	6-60
Attribute filtering with regular expressions	6-61
Monitoring XML files	6-62
XML Browser	6-63
Testing XML log file parsing	6-64
Lesson 7 Custom runtime configuration	6-65
Creating custom runtime configuration	6-66
Inserting runtime configuration variables	6-68
Lesson 8 Java API	6-69

Process overview	6-70
Add an attribute group	6-71
Add an attribute to the attribute group	6-71
Define the client process	6-72
Define the configuration	6-73
Java API-generated client	6-74
Java API-generated client structure	6-75
Java API-generated main class	6-76
Java API-generated main base class	6-77
Java API attribute group classes	6-78
Sample edit to collectData section	6-79
Java API-generated code changes	6-80
Java API tips and tricks	6-81
Java API client lifecycle	6-82
Student exercises	6-83
Summary	6-84
Unit 7 Monitoring remote and optional resources	7-1
Scenario 1 business objectives	7-2
Scenario 1 solution	7-3
Scenario 2 business objectives	7-4
Scenario 2 solution	7-5
Learning objectives	7-6
Unit outline	7-7
Lesson 1 Data source-provided runtime configuration	7-8
Editing runtime configuration	7-9
Lesson 2 Subnodes	7-10
Usage styles	7-11
Creating a subnode	7-12
Setting subnode runtime configuration	7-13
Configuring an agent with subnodes	7-14
Lesson 3 Monitoring remote systems	7-15
Remote monitoring and agent runtime configuration	7-16
Enabling remote monitoring without subnodes	7-17
Enabling remote monitoring with subnodes	7-18
Lesson 4 Monitoring Simple Network Management Protocol	7-19
Adding SNMP monitoring to an agent	7-20
Browsing SNMP	7-21
SNMP runtime configuration	7-22
Lesson 5 Monitoring network availability with ping	7-24
Ping behaviors and reporting	7-25
Adding a Ping data source	7-27
Testing Ping data source	7-28
Ping attribute group and runtime configuration	7-29
Lesson 6 Monitoring HTTP URLs and objects	7-30
URLs file	7-31
Adding an HTTP data source	7-32
Testing the HTTP data source	7-33

HTTP attribute groups and runtime configuration	7-34
Configuring the agent for HTTP monitoring	7-35
Student exercises	7-36
Lesson 7 Monitoring the Common Information Model	7-37
Adding CIM monitoring to an agent	7-39
Browsing CIM	7-40
CIM runtime configuration	7-41
Lesson 8 Monitoring Java Management Extension	7-42
Browsing MBeans: Connecting to application server	7-43
Browsing MBeans: Connecting to WebSphere Community Edition	7-44
Browsing JMX MBeans	7-45
JMX monitors and operations	7-46
JMX runtime configuration for WebSphere Community Edition	7-47
Adding other JMX types to JMX runtime configuration	7-48
JMX reference materials	7-49
Lesson 9 Monitoring database content with JDBC	7-50
Adding a JDBC data source	7-51
Browsing JDBC data source	7-52
JDBC connection properties for remote browsing	7-53
Testing JDBC data source	7-54
Running stored procedures	7-55
JDBC attribute groups and runtime configuration	7-56
Student exercises	7-58
Summary	7-59

About this course

IBM Training

IBM

IBM Agent Builder 6.3.4

© Copyright IBM Corporation 2017
Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

Use IBM® Agent Builder to create, modify, debug, and install custom monitoring agents for use with the following products

- IBM Performance Management and IBM Application Performance Management, both on-premises and software as a service (SaaS)
- IBM Tivoli® Monitoring

In this class, you learn to create agents that monitor a vast array of data sources. You learn to add custom IBM Tivoli Monitoring application support, such as queries, situations, and workspaces. You learn to add IBM Application Performance Management dashboards and OSLC properties. You learn to deploy your custom agent in a multiplatform environment. This class includes extension hands-on experience.

The lab environment for this course uses the Windows and Linux® operating systems.

For information about other related courses, see the IT Service Management training paths website:

<https://www-03.ibm.com/services/learning/ites.wss/zz/en?pageType=page&c=C251250S17780K07>

Details	
Delivery method	Classroom or instructor-led online (ILO)
Course level	ERC 1.0
	This course is an update of TV383 IBM Agent Builder 6.3.1 ERC1.0
Product and version	IBM Agent Builder 6.3.4 IBM Tivoli Monitoring 6.3.0.7 IBM Application Performance Management 8.1.3.2
Duration	3 days
Skill level	Intermediate

About the student

This course is intended for agent developers, system administrators, and application administrators who need to create customized agents to monitor resources and integrate that monitoring into an IBM Tivoli Monitoring or IBM Application Performance Management environment. Before taking this course, make sure that you have the following skills:

- Administrator-level skills in Windows and Linux
- Administrator-level skills in either of the following IBM monitoring environments:
 - IBM Tivoli Monitoring 6.X, including creating queries, situations, Navigators, and workspaces
 - IBM Monitoring 8.x or IBM Application Performance Management base 8.x, SaaS or on-premises
- Experience installing, configuring, starting, and stopping IBM agents and application support in Windows and UNIX or Linux
- Basic understanding of potential data sources, including processes, Windows services, Windows Management Infrastructure (WMI), Windows Performance Monitor (Perfmon), Common Information Model (CIM), Simple Network Management Protocol (SNMP), Simple Network Management Protocol (SNMP) events, Java Database Connection (JDBC), Java Management Extensions (JMX), HyperText Transfer Protocol (HTTP), ICMP ping, log files, Windows Event log, command return codes, socket connections, and Java API

Learning objectives

IBM Training



Learning objectives

After completing this course, you should be able to perform the following tasks:

- Describe the IBM Agent Builder application and the kinds of agents you can create
- Describe the basic process of creating a custom agent with Agent Builder
- Troubleshoot an Agent Builder agent during the development process and after installation
- Create agents for both the IBM Tivoli Monitoring and IBM Application Performance Management environments
- Create and test agents that monitor the availability of resources, such as processes, Windows services, command return codes, and network devices
- Create and test agents that monitor events from log systems, log files, and SNMP
- Create and test agents that monitor data from server technologies
- Create and test agents that monitor data from custom technologies, such as scripts, log files, Java applications, and socket connections
- Create and test agents that include remote monitoring, custom attributes, derived attributes, Navigator groups, user-entered configuration information, and Tivoli Enterprise Portal components, such as queries, situations, and workspaces
- Create and test agents that use subnodes to optionally monitor local or remote data sources

Course agenda

IBM Training



Course outline

- Unit 1 Introduction to IBM Agent Builder
 - Lesson 1 Introduction to Agent Builder
 - Lesson 2 Installing Agent Builder
 - Lesson 3 Introduction to the Agent Builder application
- Unit 2 Agent creation basics
 - Lesson 1 Creating an agent in Agent Builder
 - Lesson 2 Monitoring Windows services
 - Lesson 3 Troubleshooting in Agent Builder
 - **Exercise 1 Creating an agent to monitor specific Windows services**
 - Lesson 4: Creating installers and installing an Agent Builder agent
 - Lesson 5: Troubleshooting an Agent Builder agent
- Unit 3.1 Customizing agents for IBM Application Performance Management
 - Lesson 1 Modifying agents for an IBM Application Performance Management environment
 - Lesson 2 Creating filtered data sources
 - Lesson 3 Managing agent in an IBM Application Performance Management environment
 - Lesson 4 Installing an agent with installation scripts
 - **Exercise 1 Modify and install the AB1 agent into an IBM Application Performance Management environment**

IBM Training



Course outline

- Unit 3.2 Customizing agents for IBM Tivoli Monitoring
 - Lesson 1 Managing agents in an IBM Tivoli Monitoring environment
 - **Exercise 1 Install the AB1 agent in an IBM Tivoli Monitoring environment**
 - Lesson 2 Creating and importing IBM Tivoli Monitoring application support
 - **Exercise 2 Create and import IBM Tivoli Monitoring application support**
 - **Exercise 3 Install and confirm IBM Tivoli Monitoring application support**
- Unit 4 Monitoring Windows resources
 - Lesson 1 Monitoring Windows Management Instrumentation (WMI)
 - Lesson 2 Monitoring Windows Performance Monitor
 - Lesson 3 Monitoring Windows log events
 - Lesson 4 Common agent modifications
 - Lesson 5 Testing an attribute group
 - **Exercise 1 Monitor Windows resources**
 - **Exercise 2 Install and confirm the updated AB1 agent**

Course outline

- Unit 7 Monitoring remote and optional resources
 - Lesson 1 Data source-provided runtime configuration
 - Lesson 2 Subnodes
 - Lesson 3 Monitoring remote systems
 - Lesson 4 Monitoring Simple Network Management Protocol
 - Lesson 5 Monitoring network availability with ping
 - Lesson 6 Monitoring HTTP URLs and objects
 - **Exercise 1 Remotely monitor many resources**
 - **Exercise 2 Install and confirm the updated AB1 agent**
 - Lesson 7 Monitoring using the Common Information Model
 - Lesson 8 Monitoring Java Management Extension
 - Lesson 9 Monitoring databases with JDBC
 - **Exercise 3 Allow data sources to be optional**
 - **Exercise 4 Install and confirm the updated AB2 agent**

The course contains the following units:

1. [Introduction to IBM Agent Builder](#)

This unit introduces you to Agent Builder by describing key abilities and functions that you use to create IBM agents for custom monitoring solutions.

This unit has no exercises.

2. [Agent creation basics](#)

The process for creating an agent is simple and straightforward. In this unit, you learn the basic process for creating an agent. The skills you gain in this unit apply to almost all agents. You can create many kinds of agents with Agent Builder. The agents can differ based on what they monitor, how they handle the data they gather, how you install and configure them, and how they display the data. How you create each agent differs based on these same criteria.

3.1 [Customizing agents for IBM Application Performance Management](#)

In Part 1 of this unit, you learn how to modify an agent so that you can deploy it within an IBM Application Performance Management environment. You also learn how to generate and install the agent by using the script installers.

3.2 [Customizing agents for IBM Tivoli Monitoring](#)

Part 2 of this unit gives you an overview of how to install an agent into an IBM Tivoli Monitoring environment and how to build custom application support for an IBM Tivoli Monitoring environment.

4. [Monitoring Windows resources](#)

In this unit, you expand your previous solution by adding several new Windows data sources. You are introduced to ways in which you can modify the target attributes your agent gathers. Last, you learn how various data sources organize the data they display and how you can modify that layout with Navigator groups.

5. [Monitoring processes and command return codes](#)

In this unit, you learn about availability monitoring of processes and command return codes. Unlike the Windows-only data sources, these data sources are run on multiple operating systems.

6. [Monitoring custom data sources](#)

This unit introduces you to custom data sources. Custom data sources require you to provide the instrumentation that is needed to gather the data that you want monitored. Custom data sources include monitoring with scripts, sockets, Java API, and parsing log files. It also introduces the runtime configuration feature, showing you how to create custom runtime configuration parameters. This unit finishes the installation topic by showing you how to generate Agent Builder output with the command-line interface (CLI).

7. [Monitoring remote and optional resources](#)

This unit teaches you how to use subnodes in your agent. You can use subnodes to enable a single installed agent to monitor multiple remote resources. You can also use them to create an agent with optional data sources that you can activate or not activate at run time. This unit also introduces several new data sources, including Simple Network Management Protocol (SNMP), HTTP server URLs, ICMP (Ping), Java Management Extension (JMX), Common Information Model (CIM), and Java Database Connectivity (JDBC). Each of these data sources requires runtime configuration, which you must refine for your agent.

Unit 1 Introduction to IBM Agent Builder

IBM Training



Introduction to IBM Agent Builder

© Copyright IBM Corporation 2017
Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

This unit introduces you to Agent Builder by describing key abilities and functions that you use to create IBM agents for custom monitoring solutions.

Learning objectives

After completing this unit, you should be able to perform the following tasks:

- Describe the main functions of IBM Agent Builder
- Install and start the Agent Builder application
- List the kinds of data sources from which Agent Builder agents can monitor

Unit outline

- Lesson 1: Introduction to Agent Builder
- Lesson 2: Installing Agent Builder
- Lesson 3: Introduction to the Agent Builder application

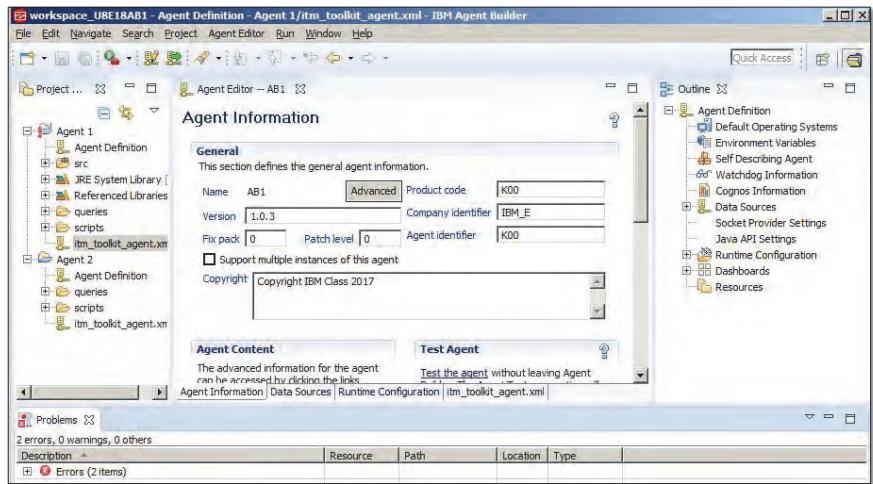
Lesson 1 Introduction to Agent Builder

IBM Training

IBM

Lesson 1: Introduction to Agent Builder

- GUI application used to create IBM agents for custom monitoring solutions
- Customer profiles
 - Customers need to manage components of their environment that are not covered by agents.
 - Business partners want to build solutions to extend the coverage offered by agents.



When an IBM provided agent does not meet your monitoring needs, you can create a custom agent with Agent Builder. It is a graphical user interface (GUI) application that you use to create monitoring agents for custom monitoring solutions, installation packages for the created agents, and application support extensions for existing agents.

Abilities of Agent Builder agents

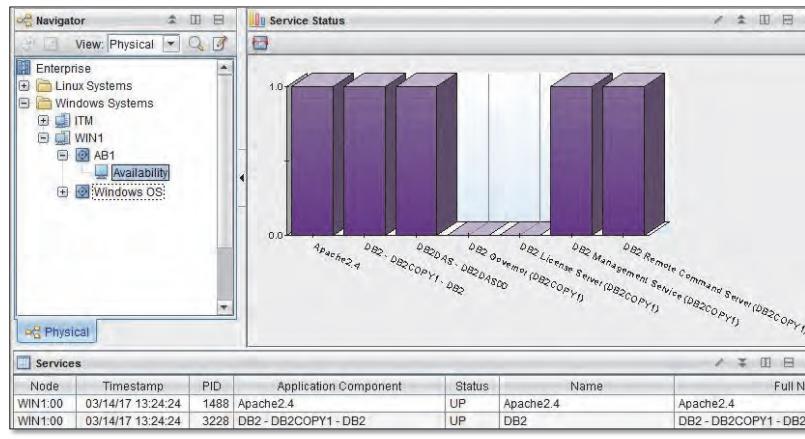
- Integrate with IBM Tivoli Monitoring and IBM Application Performance Management on premises and SaaS
- Gather and monitor wide range of data
- Mix, match, and manipulate data
- Establish agent runtime configuration

Abilities of Agent Builder agents

Each topic is covered in detail later in this lesson.

Create agents for IBM Tivoli Monitoring

- Group data sources in custom Tivoli Enterprise Portal Navigator nodes
- Create custom workspaces, situations, queries, and Take Actions for the agent
 - Packaged and installed with the agent
- Existing IBM Tivoli Monitoring agent must be installed on the agent host



Introduction to IBM Agent Builder

6

© Copyright IBM Corporation 2017

Create agents for IBM Tivoli Monitoring

You can deploy IBM Agent Builder agents in an IBM Tivoli Monitoring environment. You can create custom Tivoli Enterprise Portal (TEP) Navigator items to display multiple data sources under a common node. After you create, install, and test your agent, you can create custom application support components, such as workspaces, queries, Take Action commands, and situations for the solution. You can then import this custom application support into your agent in Agent Builder and include it in the installation image.

With another feature of the Agent Builder you can package and distribute custom application support for existing agents. With this feature, you can develop new situations, queries, and workspaces for an existing IBM Tivoli Monitoring V6.x agent. For example, suppose that you have insight into how the retail industry uses IBM DB2® for an in-store database. You can create customized workspaces and situations to give or sell to companies in this industry.

Create agents for IBM Application Performance Management

- Integrates with both IBM Application Performance Management on premises or SaaS
- Summary and details dashboards are packaged and installed with agent
- Existing IBM Application Performance Management agent must be installed on the agent host

The screenshot shows the IBM Application Performance Management interface. On the left, there's a sidebar with 'Application Dashboard', 'Applications' (with 'All My Applications' selected), 'Groups', and 'Components'. The main area shows a tree structure: 'All My Applications > My Application > Components > AB1 > WIN1:00'. Below this, there are three tabs: 'Status Overview' (selected), 'Events', and 'Attribute Details'. The 'Status Overview' tab displays 'WIN1 - AB1' with sections for 'HTTP Status' (UP) and 'Percent Processor Time' (0). To the right is a table titled 'Availability' listing various components:

Icon Component	Name	Status	Full Name	Type
Apache2.4	Apache2.4	UP	Apache2.4	SERV
DB2COPY1 - DB2	DB2	UP	DB2 - DB2COPY1 - DB2	SERV
DB2DAS00	DB2DAS00	UP	DB2DAS - DB2DAS00	SERV
DB2 Governor (DB2COPY1)	DB2GOVERNOR_DB2COPY1	DOWN	DB2 Governor (DB2COPY1)	SERV
DB2 License Server (DB2COPY1)	DB2LICD_DB2COPY1	DOWN	DB2 License Server (DB2COPY1)	SERV
DB2 Management Service (DB2...)	DB2MGMTSVC_DB2COPY1	UP	DB2 Management Service (DB2...	SERV
DB2 Remote Command Server (...)	DB2REMOTECMD_DB2COPY1	UP	DB2 Remote Command Server (...)	SERV

Introduction to IBM Agent Builder

7

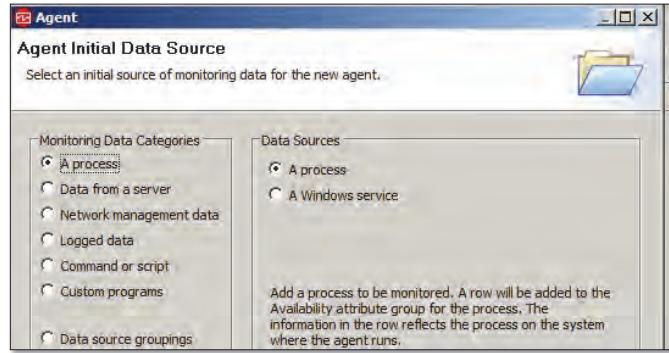
© Copyright IBM Corporation 2017

Create agents for IBM Application Performance Management

You can deploy IBM Agent Builder agents within an IBM Application Performance Management on-premises or SaaS environment. With Agent Builder, you can create web user interface definitions for the Application Performance dashboards in an IBM Application Performance Management environment.

Gather and monitor wide range of data

- Processes
 - Windows services
 - Windows Management Infrastructure (WMI)
 - Windows Performance Monitor (Perfmon)
 - Common Information Model (CIM)
 - Simple Network Management Protocol (SNMP)
 - Simple Network Management Protocol (SNMP) Events
 - Java Database Connection (JDBC)
 - Java Management Extensions (JMX)
 - HyperText Transfer Protocol (HTTP)
 - Simple Object Access Protocol (SOAP)
 - Network management data: ICMP Ping
 - Log files, including XML file parsing
 - AIX Binary Log
 - Windows Event log
- Command return codes
 - Output from a script (external scripts and commands)
 - Socket
 - Java API



Gather a wide range of data

Using the IBM Agent Builder, you can quickly create, modify, and test an agent. Agents collect and analyze data about the state and performance of different resources, such as disks, memory, processor, or applications.

The Builder creates an agent to monitor all the data sources that are listed in the slide. The screen capture shows that the data sources are organized into six categories:

- A process
- Data from a server
- Network management
- Logged data
- Command or script
- Custom programs

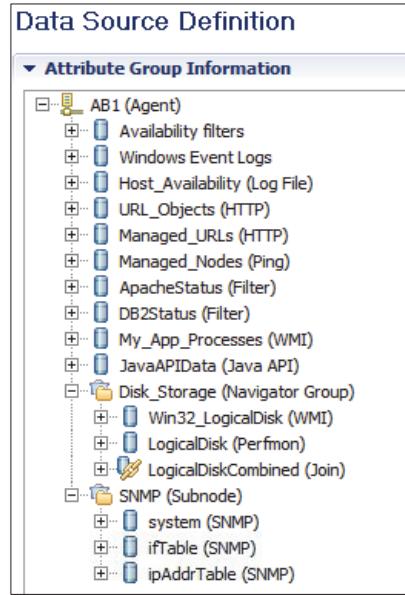
The remaining category, data source groups, offers options such as Navigator group and subnode, which you can use to group several data sources into one Navigator item.



Note: Each data source is defined later in this course.

Mix, match, and manipulate data

- Gather data from many sources at the same time
 - One or more groups of data from one data source
 - One or more data sources
 - Remote hosts
 - Multiple instances of a similar data source
- Manipulate data that is gathered
 - Combine data sources
 - Create new data based on data gathered, such as filters, sums, averages, and enumerations



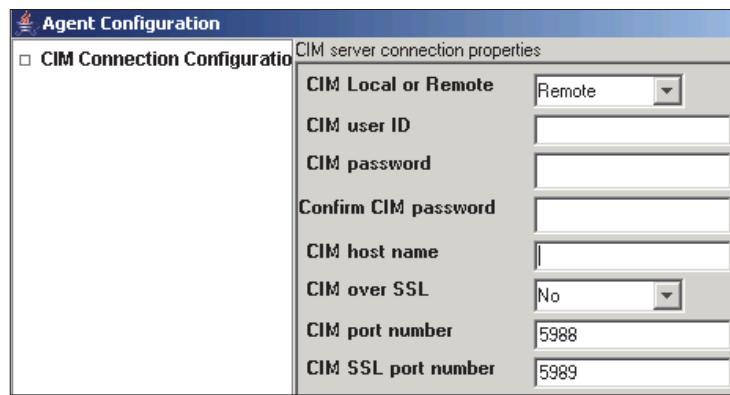
Mix, match, and manipulate data

A single agent can gather data from many sources at the same time. It can gather multiple groups of data from one data source, such as process, processor, and disk information from Microsoft® Windows® Management Instrumentation (WMI). A single agent can gather data from different data sources, such as WMI, SNMP, and a log file. Instead of monitoring local resources, you can configure a single agent to monitor resources on one or more remote hosts. Finally, you can configure a single agent to monitor multiple instances of the same thing. For example, two instances of WebSphere® on the same host or gathering the same SNMP information from 10 remote routers.

You can also configure an agent to manipulate the raw data it gathers to create new data. An agent can combine data from the same or different sources. For example, you might combine the disk information from WMI with the disk information from Windows Performance Monitor (Perfmon) to produce a single set of data about a host's disk performance.

Establish custom agent runtime configuration

- Agent parameters entered when configuring the agent
- Allows a unique agent configuration for each host
- Sample parameters
 - Host name
 - User ID and password
 - Directory or file name
 - Port number



Establish custom runtime configuration

After you install an agent on a specific host, use runtime configuration to enter custom parameters for the agent. You can configure each agent uniquely for the local host.

Lesson 2 Installing Agent Builder

IBM Training



Lesson 2: Installing Agent Builder

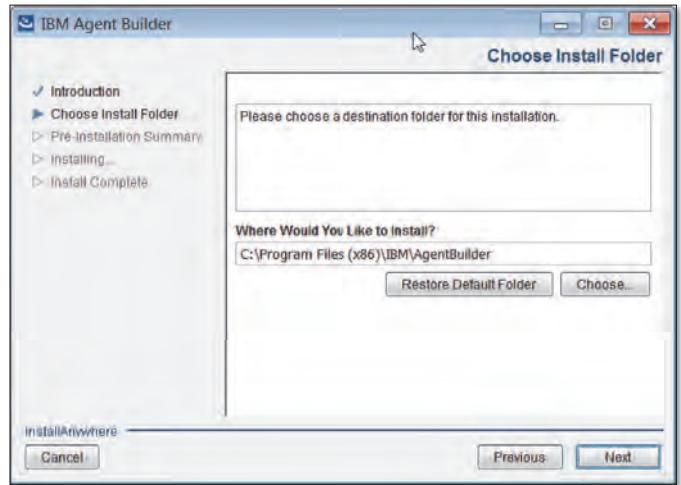
- Supported platforms
see the User's Guide for OS version requirements
 - AIX®
 - HP-UX
 - Linux (requires libstdc++.so.5 library)
 - On Red Hat Enterprise Linux, compat-libstdc++-33 package
 - On SUSE Enterprise Linux, libstdc++-33 package
 - Solaris
 - Windows
- Authorization
 - On Windows, install and run the Agent Builder from an Administrator ID
 - On Linux, install and run as any user. Run the Agent Builder from the same ID that you used to install. Testing can be limited to user permissions.
- Installed size: 500 MB (plus installation image, approximately 1 GB)

This lesson describes the supported operating systems, authorizations, and process for installing Agent Builder. After completing this lesson, you should be able to install the Agent Builder application.

You can install Agent Builder on Windows, Linux, and AIX®.

Installation process

- Run the platform-specific command:
 - setup.exe
 - setup.sh
- Invalid directory names
 - Do not install the Agent Builder into a path that contains these characters: !, #, %, and ;
- Install log location and format
 - In the root of the AB installation
 - Log file name: IBM_Agent_Builder_InstallLog.xml
 - XML format



Installation process

To install Agent Builder, run the platform-specific installer command and follow the prompts. The one parameter you can configure is the installation directory name.

Silent installation and upgrading Agent Builder

- Silent installation
 - **setup.[bat/sh] -i silent -f <path>\installer.properties**
 - installer.properties is included in the image. Edit to accept the license and provide installation location
 - In the root of the image
- Upgrading Agent Builder
 - Uninstall your existing Agent Builder
Uninstalling does not uninstall existing agent projects. The workspaces you created are not modified by uninstalling or installing Agent Builder
 - Install the new version of Agent Builder

Silent installation

You can install the product by using a silent installation method. The silent installation options file, `installer.properties`, is included on the installation media at the root of the installation directory. You must modify this file to meet your needs.

Upgrading Agent Builder

Before installing a new version of Agent Builder, uninstall the existing Agent Builder. Uninstalling Agent Builder does not affect your existing agent projects.

Lesson 3 Introduction to the Agent Builder application

IBM Training



Lesson 3: Introduction to the Agent Builder application

- Eclipse technology
- Starting Agent Builder
- Application Window
- Directories and files
- Agent wizard and browser
- Help
- Resources

This lesson describes the main features of the Agent Builder application. After completing this lesson, you should be able to describe the main functions of IBM Agent Builder.

Eclipse technology

- The Agent Builder is built on Eclipse technology
 - Bundles Eclipse 3.62
 - Each AB component and data source is a separate Eclipse plug-in and stored as a .jar file
- With Eclipse you can perform these tasks:
 - Customize your workspace
 - Set preferences
 - Some preferences are specific to Agent Builder

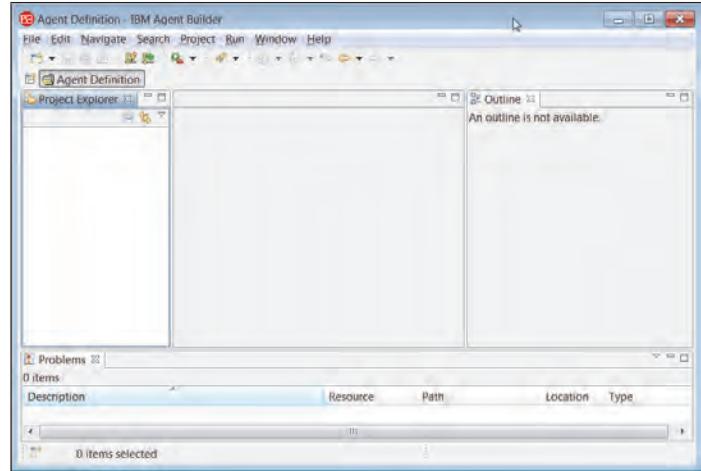
Eclipse technology

The IBM Agent Builder is an application for the Eclipse platform, an open source framework for the construction of powerful software development tools and rich desktop applications. Using the Eclipse plug-in framework to integrate technology on the desktop can save technology providers time and money for focusing effort on delivering differentiation and value for their offerings. An open source community of developers built Eclipse, which is provided royalty-free by the Eclipse Foundation. Eclipse supports multiple languages, multiple operating systems, multiple vendors. Written in the Java™ language, Eclipse includes extensive plug-in construction toolkits and examples. You can extend it and run it on a range of desktop operating systems, including Windows, Linux®, QNX™, and Macintosh® OS X.

To see full details about Eclipse and the Eclipse Foundation, go to <http://www.eclipse.org>.

Starting Agent Builder

- Windows:
 - Start > All Programs > IBM > Agent Builder
 - Install_Location\agentbuilder.exe
Default Install Location:
C:\Program Files (x86)\IBM\AgentBuilder
 - Agent Builder icon on desktop
- AIX or Linux:
 - Install_Location/agentbuilder
Default Install Location:
/opt/ibm/AgentBuilder

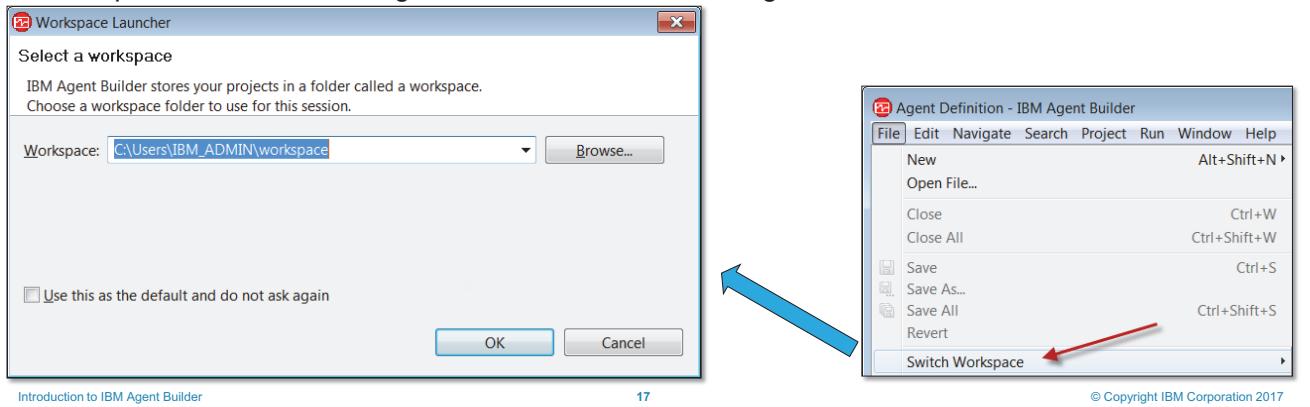


Starting Agent Builder

The slide shows how to start Agent Builder on Windows, AIX, and Linux.

Creating workspaces

- Master directory containing a set of Agent Builder projects (agents)
- You can have only one workspace open at a time
- Initial workspace is selected when you start Agent Builder
- A default can be identified
- Workspaces and contained agents are not affected when Agent Builder is uninstalled



Creating workspaces

In Agent Builder, each agent you define is part of a project. The project contains all components that make up the agent definition. Projects are organized into workspaces. A workspace is a group of projects, and corresponds to a specific master directory that contains all the projects in a separate directory. You can have an unlimited number of workspaces, but you can have only one workspace open at a time.

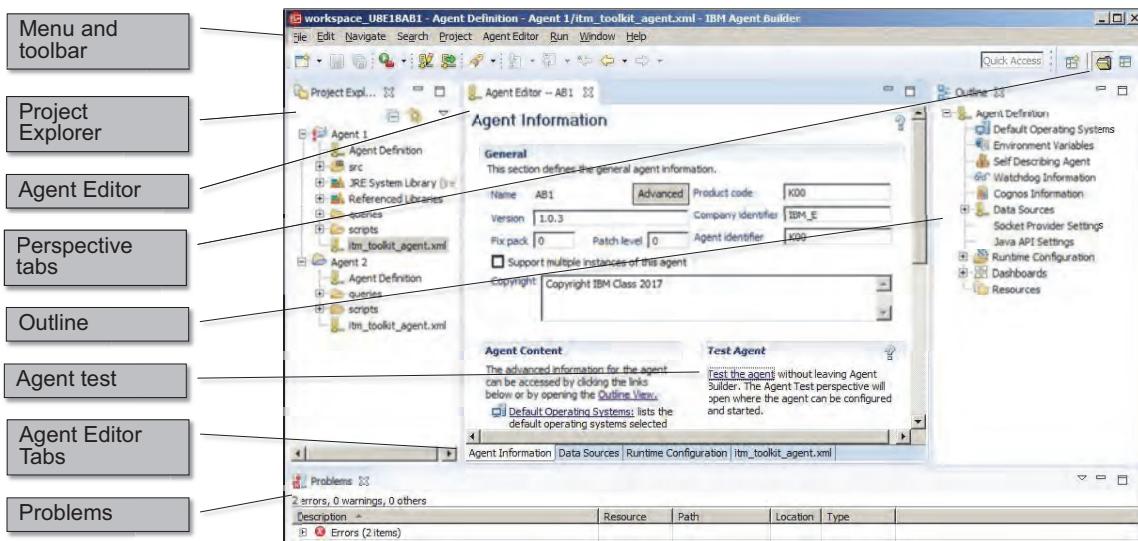
When you initially run the Agent Builder, it prompts you for the location of your workspace directory. You can designate any directory as your workspace, and you can set one as your default.

You can create new workspaces or open other workspaces at any time.



Hint: Keep your workspace locations outside the Agent Builder installation path. This practice makes it easy to remove Agent Builder and install a new version without affecting your existing workspaces.

Application window



Introduction to IBM Agent Builder

18

© Copyright IBM Corporation 2017

Application window

This slide shows the panes in the Agent Builder main interface.

The **Project Explorer** pane shows the projects (agents) in the current workspace and the components that make up each project. You can expand and contract folders. You can open chosen items by double-clicking.

The **Agent Editor** pane is the main interface into working with a defined agent. With it, you can edit the agent. The Agent Editor tabs give you access to the four main components of an agent definition: **Agent Information**, **Data Sources**, **Runtime Configuration**, and the **itm_toolkit_agent.xml** file. Click each tab to access its information.

Use the **Agent Test** pane to test the full agent in the Agent Builder without the need to deploy the agent into a development monitoring infrastructure.

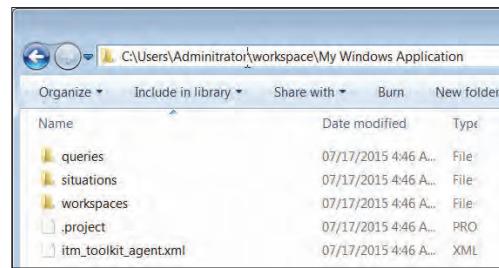
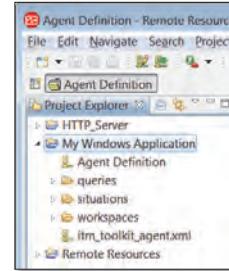
The **Problems** pane is a message area where Agent Builder shows any problems it identifies.

Use the **Perspective tabs** to switch between active perspectives, such as the Agent Definition and Agent Test perspectives.

The **Outline** pane offers an alternative method of editing an agent.

Directories and files

- Project (agent)
 - Defines a single agent
 - Is a directory in a workspace containing all the information (directories and files) that make up the agent
- Required components:
 - .project
 - itm_toolkit_agent.xml
 - Agent Information
 - Data Source Information
 - Runtime Configuration Information
- Optional components (directories)
 - Java code
 - Queries
 - Scripts
 - Situations
 - Workspaces



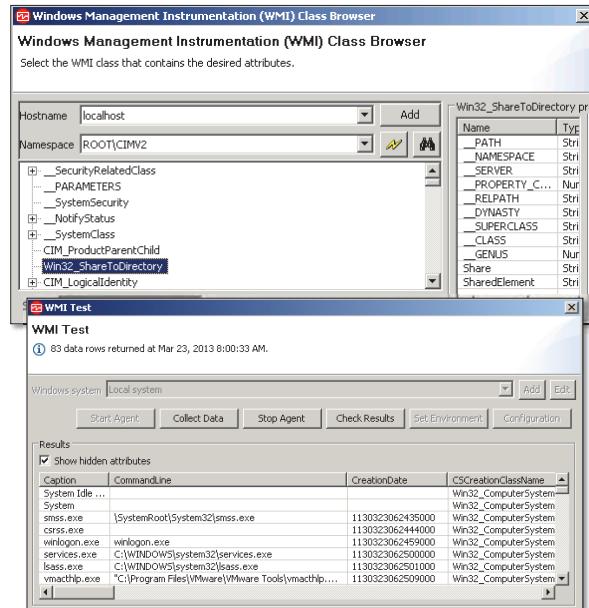
Directories and files

This slide shows the directories and files that relate to your agent. The workspace is a single directory with subdirectories for each agent project. The project directory contains all the information, directories and files, that make up an agent. Each project requires a **.project** file and an **itm_toolkit_agent.xml** file. The **itm_toolkit_agent.xml** file is the main agent definition component. It contains all the agent information that you enter in the **Agent Information**, **Data Sources**, and **Runtime Configuration** tabs. Other directories that you might add to a project as you expand its functions are Installer, Queries, Scripts, Situations, and Workspaces.

Notice the direct correlation between agent components that are shown in the Project Explorer and the actual directories and files in the workspace directory.

Wizards, browsers, and testers

- GUI uses windows, dialog boxes, and wizards
- Wizards
 - Provide steps for creation and modification processes
 - Ensure that required components built correctly
- Browsers
 - Browse a sample data source and select from the browser the data to be gathered by the agent
 - Creates attribute group and attribute definitions that correctly define the target data to be gathered by the agent
- Testers
 - Test individual data sources or the full agent without installing agent
 - For some custom data sources, creates attribute group and attribute definitions



Introduction to IBM Agent Builder

20

© Copyright IBM Corporation 2017

Wizards, browsers, and testers

Agent Builder provides wizards, browsers, and testers to facilitate the agent creation process.

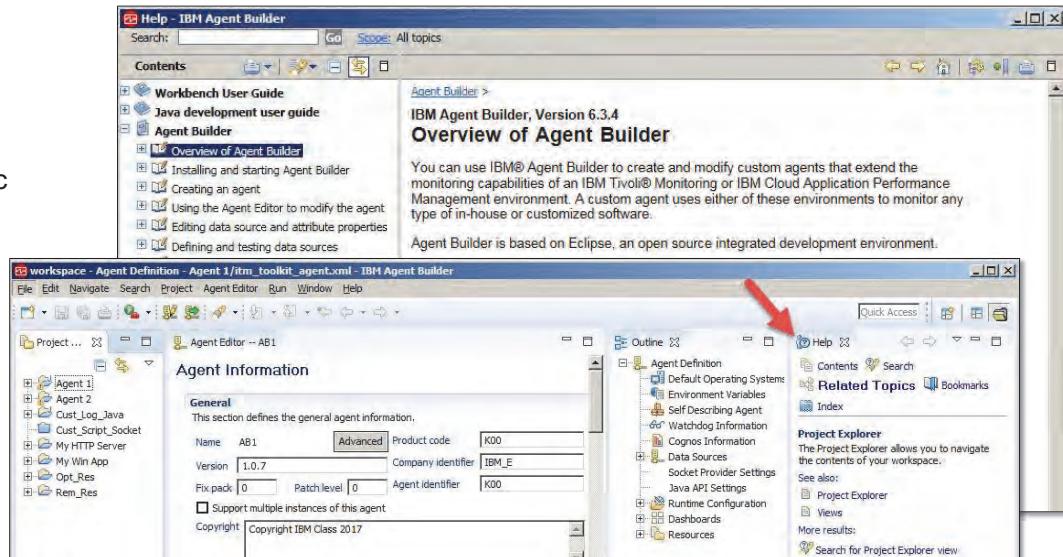
The Agent wizard guides you through the creation and modification process, such as entering agent information, identifying data sources, and setting runtime configuration. The wizard ensures that all components are built correctly.

Agent Builder provides several data source browsers. With them, you can sample an actual data source so that you can easily identify the data to gather. You can use testers to test individual data sources or the full agent to ensure that the agent can generate the data that you want without having to install the agent.

In most instances, the browsers and testers also help with creating and defining the attribute groups and attributes that you need for the data you select to gather.

Help

- Help menu
 - Help content
 - Search
 - Dynamic help
- Wizard dynamic help



Help

Agent Builder provides a full online help system. From the **Help** menu, you can select to view the full help content, search help, or view dynamic help, which shows context-sensitive help.

The Agent wizard also has dynamic, context-sensitive help access through the question mark icon.

Agent Builder resources

- Agent Builder 6.3.4 software and User's Guide
<https://www.ibm.com/developerworks/community/wikis/home?lang=en#!/wiki/Tivoli%20Monitoring/page/Agent%20Builder>
- Agent Builder 6.3.3 Knowledge Center
https://www.ibm.com/support/knowledgecenter/SSMKFH/com.ibm.apmaas.doc/install/agent_builder_guide.htm
- IBM Tivoli Monitoring 6.3 Knowledge Center
https://www.ibm.com/support/knowledgecenter/SSTFXA_6.3.0.2/com.ibm.itm.doc_6.3/welcome.htm
- IBM Performance Management 8.1.3 Knowledge Center
https://www.ibm.com/support/knowledgecenter/SSHNR_8.1.3/com.ibm.pm.doc/welcome.htm

Agent Builder resources

The slide contains links to helpful resources.

Summary

Now that you have completed this unit, you should be able to perform the following tasks:

- Describe the main functions of IBM Agent Builder
- Install and start the Agent Builder application
- List the kinds of data sources from which Agent Builder agents can monitor

Unit 2 Agent creation basics

Agent creation basics

© Copyright IBM Corporation 2017
Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

The process for creating an agent is simple and straightforward. In this unit, you learn the basic process for creating an agent. The skills you gain in this unit apply to almost all agents. You can create many kinds of agents with Agent Builder. The agents can differ based on what they monitor, how they handle the data they gather, how you install and configure them, and how they display the data. How you create each agent differs based on these same criteria.

Scenario business objectives

- A company has a web application that provides order processing and client management
 - The application runs on multiple HTTP servers
 - The application pulls data from DB2 database
 - The installations are Windows-only
- You must create an agent that monitors the HTTP server and DB2 Windows services of this web application

Scenario business objectives

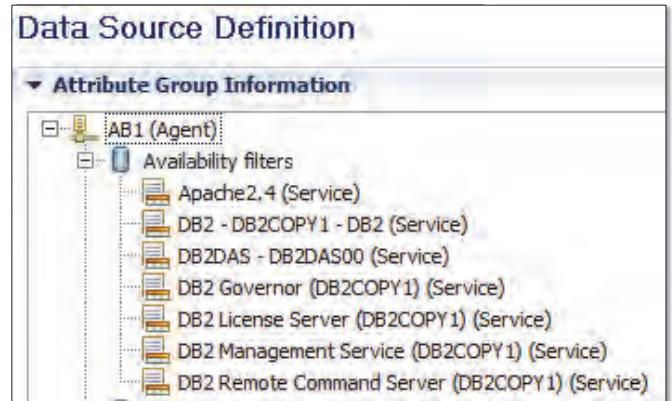
Your company has a key web application that provides order processing and client management. The application runs on multiple HTTP servers and stores its critical data in the local DB2 database. All parts of this application, including the web servers and the DB2 server, run only on Windows servers.

Scenario solution

- Agent called AB1
- Availability monitoring of the Apache and DB2 Windows services

Note: In later units, you modify this agent for and install into one of these locations:

- An IBM Tivoli Monitoring environment or
- An IBM Application Performance Management environment



Scenario solution

In this unit, you create an agent, My Windows Application, with a display name of MyWApp_TEP. It monitors the HTTP server and DB2 services of any target host.

In the next unit, you install this agent into, and modify it for, an IBM Tivoli Monitoring environment. In Unit 4, you expand this agent to monitor other aspects of this application. In Unit 5, you modify it for, and install it, into an IBM Application Performance Management environment.

Learning objectives

After completing this unit, you should be able to perform the following tasks:

- Describe the basic process of creating an agent
- Create an agent that monitors the availability of Apache and DB2 Windows services
- Navigate and describe the Agent Builder editor interface
- Describe troubleshooting techniques for Agent Builder
- Install an Agent Builder agent with a quick local installation and with installation scripts
- Describe troubleshooting techniques for your installed agent

Unit outline

- Lesson 1: Creating an agent in Agent Builder
- Lesson 2: Monitoring Windows services
- Lesson 3: Troubleshooting in Agent Builder
- **Exercise 1: Creating an agent to monitor specific Windows services**
- Lesson 4: Creating installers and installing an Agent Builder agent
- Lesson 5: Troubleshooting an Agent Builder agent

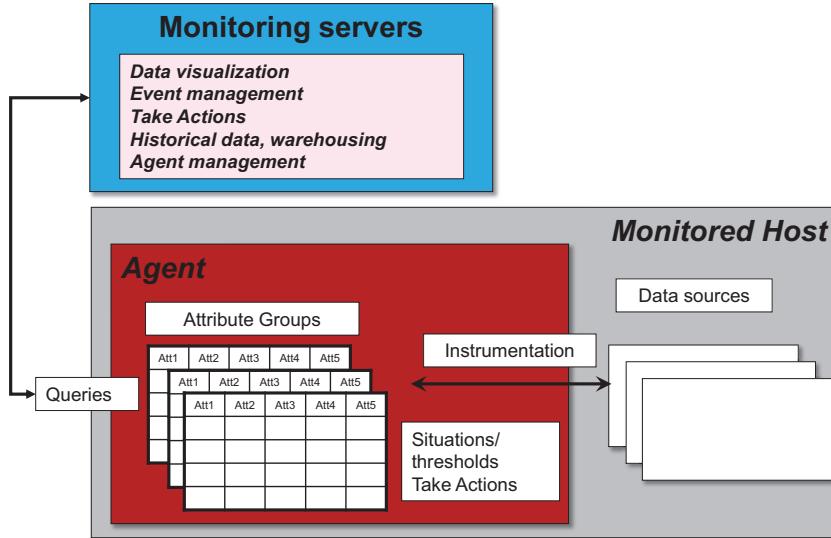
Lesson 1 Creating an agent in Agent Builder

IBM Training

IBM

Lesson 1: Creating an agent in Agent Builder

What is an IBM agent?



Agent creation basics

6

© Copyright IBM Corporation 2017

This lesson teaches the basic process for creating an agent, including the components that you must define.

After completing this lesson, you should be able to do the following tasks:

- Describe the parts of an Agent Builder agent
- Describe the basic process of creating an agent

Components of an IBM agent and monitoring environment

An IBM agent is made up of the following components:

- Data visualization
- Event management
- Take Actions
- Historical data and data warehousing
- Agent Management
- Attribute groups and attributes
- Data source instrumentation
- Queries
- Situations and thresholds to generate event notifications
- Take Actions

The monitoring environment into which you install an IBM agent can provide the following services.

- Data visualization
- Event management
- Take Actions
- Historical data and data warehousing
- Agent Management

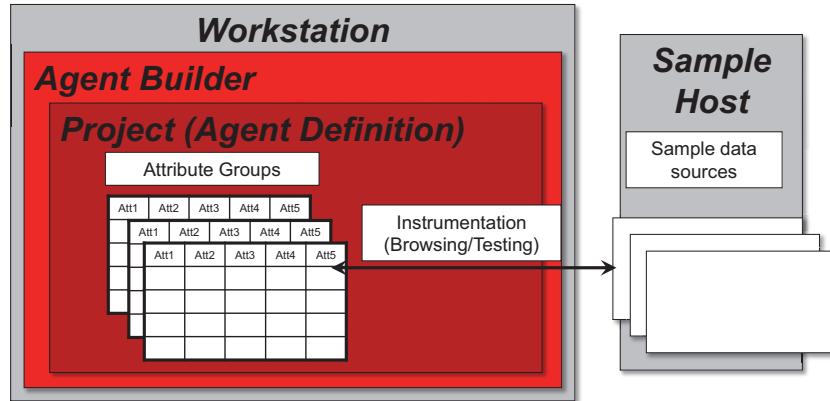
When you create an IBM agent, you might find it necessary to prepare the IBM agent for these monitoring environment services.

IBM agent components in Agent Builder

- Created by user in Agent Builder**
- Target data sources and attribute groups
 - Custom instrumentation
 - Runtime configuration
 - APM dashboards (APM)
 - OSLC definitions (APM)

- Created outside Agent Builder and imported into agent (ITM)**
- Workspaces
 - Queries other than default
 - Situations
 - Take Actions

- Generated automatically by Agent Builder**
- Queries
 - Warehouse and Cognos data models
 - Non-custom instrumentation
 - Installers



Attribute groups and attributes

Every monitored data source has an attribute group to define what data it monitors and to hold the monitored data. With the Browsers within Agent Builder, you can extract from a sample data source the metadata, column names, types, formats, and so on, needed to build the attribute groups. For custom attributes, you manually define each one.

Data source instrumentation

For most data sources, Agent Builder provides the instrumentation that allows the agent to interact with established information and management systems to gather monitored data, handle error codes, and pass Take Actions. But, for some of the data sources, you need to provide part of the instrumentation:

- For log file monitoring, you must configure the agent to correctly parse the log file.
- For the script data source, you must provide the script that generates the data and error codes.
- For the Java API data source, you must provide the Java code that generates the data and error codes, and interacts with the monitored application.
- For the socket data source, you must configure the monitored application to create a socket connection with the agent to pass data, error codes, and actions by using the correct XML format.

Components that Agent Builder creates

Agent Builder can automatically create the following components:

- Queries: retrieves data from the agent
- Data warehouse instrumentation: enables warehousing of the data
- Cognos data model: enables creation of Tivoli Common Reporting reports for the data
- Agent installer package: enables installation of the agent and application support

Components that you create in Tivoli Enterprise Portal and import into Agent Builder

You create the following items in a test or development IBM Tivoli Monitoring environment and imported them into the agent in Agent Builder:

- Data visualization, workspaces
- Queries, beyond the default attribute group query
- Situations
- Take Actions

Creating and importing these components requires you to install the agent into a test or development environment. Because you can create these components in a production environment after you deploy the agent, it is not required that you create them in a test or development environment and import them into the agent. But, creating them in a test or development environment and importing them into the agent can help with version control as you update and modify the agent.

How you create these components and import them into your Agent Builder agent is discussed in Unit 4.

Components that you create for IBM Application Performance Management environments

To install an agent into an IBM Application Performance Management environment, you must create the following items in Agent Builder:

- Summary and detailed dashboards
- Open Service for Lifecycle Collaboration (OSLC) definitions

The summary dashboard is the initial data display for the status of the monitored component. The details dashboard shows a table of data for each monitored attribute group. Use the Open Service for Lifecycle Collaboration definitions to identify the agent to the monitoring environment and help with agent management.

Creation of these components is taught in Unit 3.

Agent creation process

1. Create the agent in Agent Builder
 - Agent information
 - Data source information
 - Runtime configuration
2. Configure agent for monitoring environment
3. Test and update the agent
4. Create and install production agent

Note: Steps 1-3 tend to be iterative. Only when the agent is finalized is the agent output and installed into production.

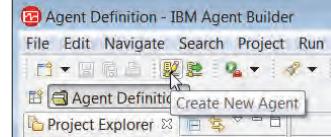
Agent creation process

This slide provides an overview of the four basic steps of the process of creating and deploying an Agent Builder agent. Steps 1 - 3 are iterative. You repeat them until the agent is tested and finalized. Only then do you deploy an agent into a production environment.

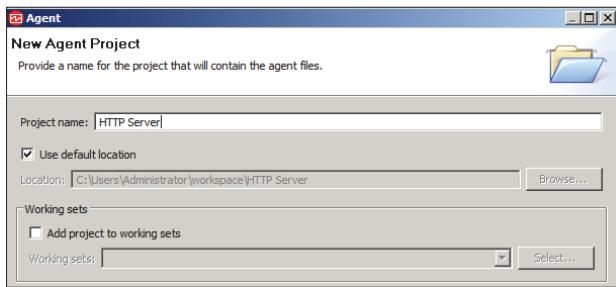
This unit focuses on Step 1.

Starting the agent creation wizard

1. Start the Agent wizard



2. Name the project



Starting the agent creation wizard

The easiest way to create an agent or to add new data sources is with the Agent wizard.



Note: Although it is possible for you to create an agent or add new data sources to an agent without the Agent wizard, always use the Agent wizard for these processes. Using the Agent wizard ensures that you configure the properties correctly.

You can start the Agent wizard from the menus or from the Create New Agent button.

Each agent must have a project name, which is the name of the project in Agent Builder. You can also copy the name into the **Display Name** field in the **Agent Information** window.

Three basic components to define

- Agent information
- Data sources
- Runtime configuration

Three basic components to define

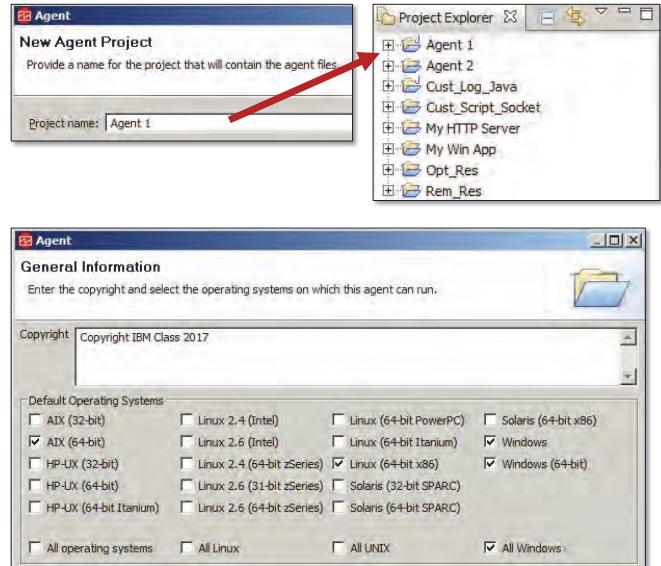
When creating an agent, you must set three basic categories of information:

- Agent information
- Data sources
- Runtime configuration

Each is described in the following pages.

Define the agent information

- Project name: name of agent project in Agent Builder, workspace folder containing agent components
- Copyright: copyright statement
- Default operating systems: where this agent can be installed
 - Individual data sources initially match these, but can be different
 - Local data sources can be a subset
 - Remote data sources not restricted



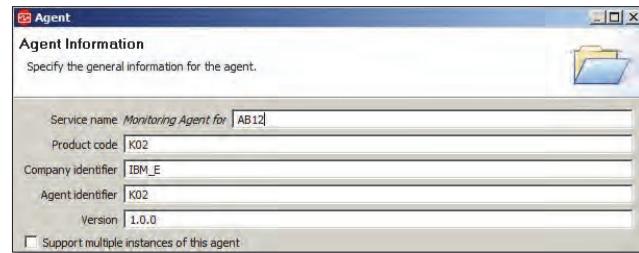
Define the agent information

The slide shows the Agent Information window and describes key points about several properties. Each property is defined here:

- Project name is the name of the agent project in Agent Builder. It is the name that is displayed in Project Explorer and the directory name in the workspace directory. It contains all components of the agent project.
- **Copyright** is the copyright statement that you want to use for your new agents. This statement is determined by your corporate legal requirements for copyrights. This copyright statement is inserted into all of the files that are generated for the agent and you can edit it when you need to. After you supply the statement the first time, the IBM Agent Builder stores your copyright statement. It then enters it in the copyright area each time that you start the wizard to create a new agent. For example, you might use the following format and contents: *Copyright ABC Corp 2017. All rights reserved.* If you are unsure what information you want to supply about the copyright, contact your enterprise legal department.
- **Default Operating Systems** identifies the operating systems on which you can install this agent. Some data sources can have target operating systems as well. Initially, they match the agent's default operating systems, but they can be different. Local data sources are limited by the agent's default operating systems, but can be a subset of them. Remote data sources are not restricted by the default operating systems.

Define the agent information (continued)

- Agent name: Name of the agent in most interfaces. Normally describes what this agent monitors
 - Service name always starts with Monitoring Agent for
 - Project name and agent name are the same by default, but can be different
- Product code: K00-K99, K{0-2}{A-Z}, and K{4-9}{A-Z}
- Company Identifier: uniquely identifies the organization creating the agent
- Agent Identifier: alphabetic string that uniquely identifies the agent
 - Product code used by default
- Combined company and agent IDs
 - Must be unique
 - Cannot exceed 11 characters



- Version: 3-digit number for version, release, and modification
- Multiple instances:
 - Allows multiple instances of agent on same host
 - Mostly superseded by subnodes

Define the agent information (continued)

The slide shows the Agent Information window and describes key points about several properties. Each property is defined here:

- **Agent name** is the name of the agent as shown in most interfaces. It is the root phrase that you type here and normally describes what this agent monitors. The service name always starts with Monitoring Agent for and ends with the agent name. Which form of the name that is displayed differs from utility to utility. By default, the agent name initially takes on the project name.
- **Product code** is the product code for your new agent. A range of product codes is reserved for use with the Agent Builder and Agent Builder accepts only values within this range. The allowed values are K00-K99, K{0-2}{A-Z}, and K{4-9}{A-Z}. These values are for your company's use only and are not intended for agents that are shared or sold.

If you are creating an agent to share with others, you must send a note to toolkit@us.ibm.com to reserve a product code. Your request for a product code must include a description of the agent you are building. A product code is then assigned, registered, and returned to you. When you receive the three-letter product code, you also receive instructions that describe how to modify Agent Builder to use the new product code.

- **Company identifier** is an alphabetic string that uniquely identifies the organization that develops the agent. *IBM* is reserved. A good candidate for this field can come from your company's URL. For example, from a **mycompany.com** URL, use the text *mycompany*.
- **Agent identifier** is an alphabetic string that uniquely identifies the agent that you develop. The Product code is copied into the agent identifier field, but you can make them different.

The combined length of the **Agent identifier** field and the **Company identifier** field cannot exceed 11 characters.

- **Version** is a three-digit number that identifies the agent version in the format VRM, where:

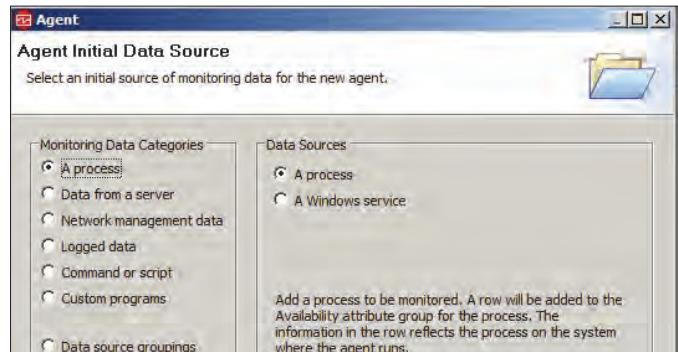
- V = Version
- R = Release
- M = Modification

For a patch level, use this field when you must release a fix for an agent, but you do not want to update the version.

- **Support multiple instances of this agent** indicates that you want to monitor applications that you might configure to run multiple instances of themselves on a computer at the same time. When building agents for these types of applications, separate the management of each of the application instances. That way, the operations user knows when one of the application instances has problems. A similar condition happens if you are building an agent that uses the JMX or SNMP data provider. Because a data monitor like JMX or SNMP can monitor remote systems, it is possible to install the agent on one computer and monitor several other systems at the same time. You must represent each of these remote systems separately in the monitoring environment, and each must have different configuration values for communication. To make this setup, select the **Support multiple instances** check box, which causes the Agent Builder to create a *template* agent. After the agent is installed, you can create and configure an instance of the agent for each instance of the monitored application. This option is rarely used as its functions have been mostly superseded by subnodes.

Select the data source

- A process
 - Processes
 - Windows services
- Data from a server
 - Common Information Model (CIM)
 - Windows Management Infrastructure (WMI)
 - Windows Performance Monitor (Perfmon)
 - Common Information Model (CIM)
 - Simple Network Management Protocol (SNMP)
 - Simple Network Management Protocol (SNMP) Events
 - Java Database Connection (JDBC)
 - Java Management Extensions (JMX)
 - HyperText Transfer Protocol (HTTP)
 - Simple Object Access Protocol (SOAP)
- Network management data
 - Network management data: ICMP Ping
- Logged data
 - Log files, including XML file parsing
 - AIX Binary Log
 - Windows Event log



- Command or script
 - Command return codes
 - Output from a script (external scripts and commands)
- Custom programs
 - Socket
 - Java API

Select the data source

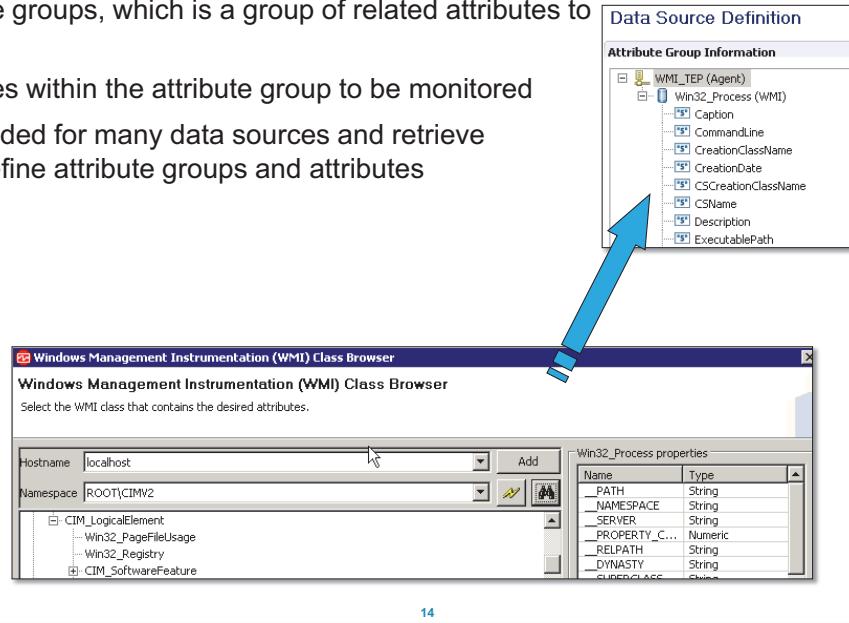
After you set the agent information, the next step is to select the data source you want to monitor. You can divide this step into two substeps, selecting a type of data source and specifying the data to gather.

The different types of data sources are divided into several categories. Which data sources are contained within which category and the window you use to select the type of data source is shown in the slide.

Notice that the **Data source groupings** category is not really a type of data source, but two methods of grouping data sources.

Identify the data to gather

- Define attribute groups, which is a group of related attributes to be monitored
- Define attributes within the attribute group to be monitored
- Browsers provided for many data sources and retrieve metadata to define attribute groups and attributes



Identify the data to gather

After you select the type of data source you want to gather, you must identify the data you want to gather.

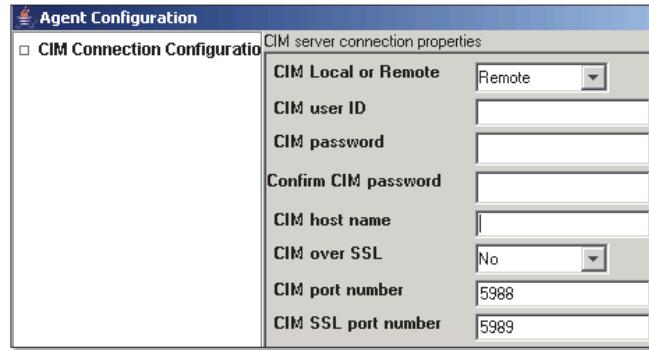
The Agent Builder terms that are used to define what data to gather, and the structures that hold the gathered data, are **attribute group** and **attribute**. An attribute group is analogous to a database table and defines the chunk of related data to retrieve. Attributes are the column definitions and define the specific data to retrieve. The retrieved data is shown as rows in the attribute group.

How this process works differs from data source to data source and is described in detail when you learn about each data source. For some data sources, you identify which rows are gathered. For other data sources, you define the attribute group and attributes.

Browsers are provided to make it easier to identify and select the target data source.

Set runtime configuration

- Agent properties entered when agent is configured
- Some data sources have no runtime configuration; some require it
 - Windows services do not
- Examples
 - Application directory
 - User name and password
 - Host name or IP address of target host



Set runtime configuration

Runtime configuration is agent configuration that you can set at agent installation or configuration. With it, you can customize the agent for the specific host. Some data sources have no runtime configuration, while others require it.

The slide provides several examples of common runtime configuration parameters.



Note: Runtime configuration is not used in the first agents you create in this course. Therefore, it is described in more detail later in the course.

Lesson 2 Monitoring Windows services

IBM Training

IBM

Lesson 2: Monitoring Windows services

Introduction to availability monitoring

- Types
 - Windows services
 - Processes
 - Command return codes
- All data returned to Availability display for all types
- Cannot manipulate the attributes returned
- Some attributes are not valid on some types

Availability		
Application Component	Name	Status
Apache2.4	Apache2.4	UP
DB2 - DB2COPY1 - DB2	DB2	UP
DB2DAS - DB2DAS00	DB2DAS00	UP
DB2 Governor (DB2COPY1)	DB2GOVERNOR_DB2COPY1	DOWN
DB2 License Server (DB2COPY1)	DB2LICD_DB2COPY1	DOWN

Node	Timestamp	Application Component	Name	Status	Full Name	Type
Virtual Size	Page Faults per Sec	Working Set Size	Thread Count	PID	Percent Privileged Time	Percent User Mode Time
Percent Processor Time	Command Line	Functionality Test Status	Functionality Test Message			

This lesson teaches you how to add availability monitoring of Windows services to your agent.

After completing this lesson, you should be able to do the following tasks:

- List and describe availability monitors
- Create an agent that monitors the availability of HTTP Server and DB2 Windows services

Introduction to availability monitoring

Availability is a category of monitoring that an Agent Builder agent does. Its main goal is to identify the current availability of resources, such as Windows services and multiplatform processes. With the command return code monitor, you can add tests to run against a computer or application to determine its health in a more detailed way. For example, you can ping a port or run a provided management script that indicates whether an application is available or able to do useful work.

This unit focuses on monitoring Windows services, and [Unit 5, “Monitoring processes and command return codes”](#) covers the other availability data sources.

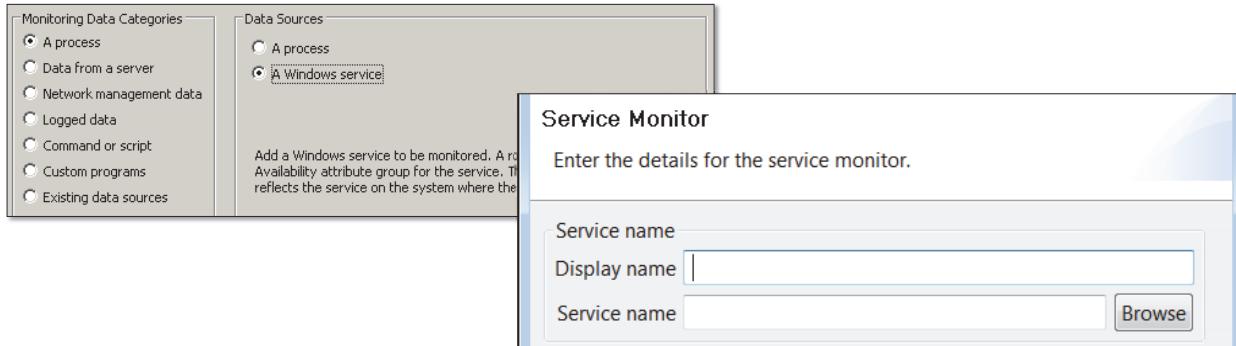
Availability monitoring differs from most of the other types of data source monitoring. All availability data sources such as Windows services, processes, and command return codes on an agent return their data to a single Availability attribute group and Navigator item. This behavior differs from performance monitors, in which each data source has its own attribute group and Navigator item.

The availability data that is gathered is predefined and cannot be changed. So, although you can change the filters to monitor, you cannot edit what data is returned, allowing a common view of the availability for all applications in your environment. The data is consistent across all Agent Builder agents and many IBM agents, allowing for a common availability model, similar Tivoli Enterprise Portal views, and even Tivoli Common Reporting. After you learn about availability for one agent, it is the same for all others.

Because some availability filter types return different data, not all columns in the data table apply to all filter types. For example, all availability filters return application component, name, and status information, but only the command return code availability filter returns functional test status and functionality test message attributes.

Selecting Windows services to monitor

- Service Monitor window
 - Display name is the name listed in the Tivoli Enterprise Portal navigator
 - Service is the service to monitor
- You can enter the names or you can browse and have either name autopopulate



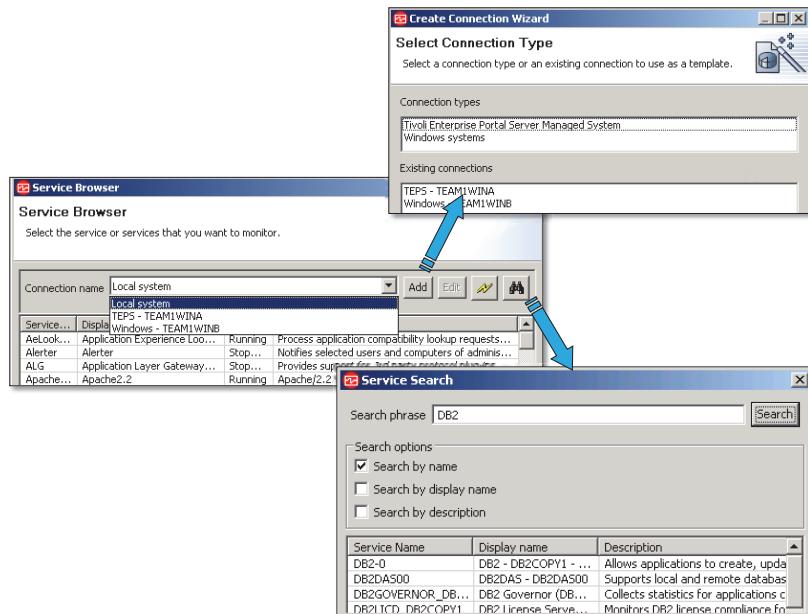
Selecting Windows services to monitor

To specify a Windows service to monitor, all you need provide is a display name and the service name. The display name is the name that appears in the Tivoli Enterprise Portal Navigator and the Performance Management console. The service name is the correct name for the Windows service you want to monitor.

You can enter these values manually, or you can use the browser feature to locate and select one or more Windows services. Selecting a service with the browser returns a display and service name and overwrites anything that is entered before browsing. You can change the display name without affecting the service that is monitored. Selecting more than one service does not return a display name and service name. It returns a message that indicates that you selected multiple items.

Browsing for Windows services

- Standard actions
 - Local or remote browsing
 - Refresh
 - Find: substring match and is not case-sensitive
 - Sort by column
 - Select one or more processes
- Remote browsing
 - Add host name
 - Login required
 - Lists services on remote system
- Selecting services
 - Shift+click for consecutive
 - Ctrl+click for nonconsecutive



Agent creation basics

18

© Copyright IBM Corporation 2017

Browsing for Windows services

With the **Service Browser**, you can locate and select one or more Windows services you want the agent to monitor.

The agent wizard provides similar browsing utilities for most data sources. You can complete many standard actions in these browsers, such as the following examples:

- Local or remote browsing
- Refreshing the view
- Locating a data source with the find feature
- Specifying the data that you want to monitor

With the Find feature, you can search for specified text, but you cannot use any variable search criteria.

You can browse Windows services on the local host or on remote hosts. Remote services browsing is through a standard Windows remote connection and requires the remote host name and a login. The main reason for browsing a remote host is because the local computer does not have the service that you want the agent to monitor.



Note: Recognize that browsing a remote host is not related to the host that is ultimately monitored by the agent. The browsing process results only in a Windows service that is identified for the agent to monitor, not a host. Determining which host that the agent is monitoring is a function of where you install the agent and how you configure the agent after it is installed.

Lesson 3 Troubleshooting in Agent Builder

IBM Training



Lesson 3: Troubleshooting in Agent Builder

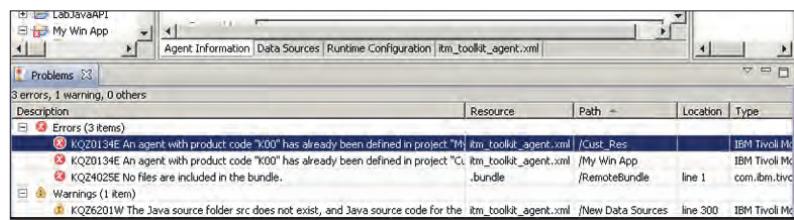
- Agent definition issues
- Agent Builder application issues

This lesson introduces troubleshooting Agent Builder and your Agent Builder agent. This lesson focuses solely on the various log files that provide you troubleshooting information.

After completing this lesson, you should be able to describe troubleshooting techniques for Agent Builder and your agent.

Problems pane

- Identifies problems within the agent definition
- Typically inhibits generating agent
- Path indicates which project
- Line indicates location of error in itm_toolkit.agent.xml
- Double-clicking might take you directly to the error
- Saving causes reevaluation of agent for errors



Problems pane

During the save process, Agent Builder evaluates the agent definition and identified problems are listed in the Problems pane.

Agent Builder trace files

- Trace files are stored in the Agent Builder workspace directory
 - The workspace directory is specified when starting Agent Builder
- Default locations:
`workspace_directory/.metadata`
`workspace_directory/.metadata/tivoli/KQZ/logs`
 - Location can be changed per Agent Builder plug-in
 - The metadata directory might not be visible
 - On Linux and AIX, use
`ls -al`
 - On Windows, ensure that the folder is configured to show hidden files and folders

Agent Builder trace files

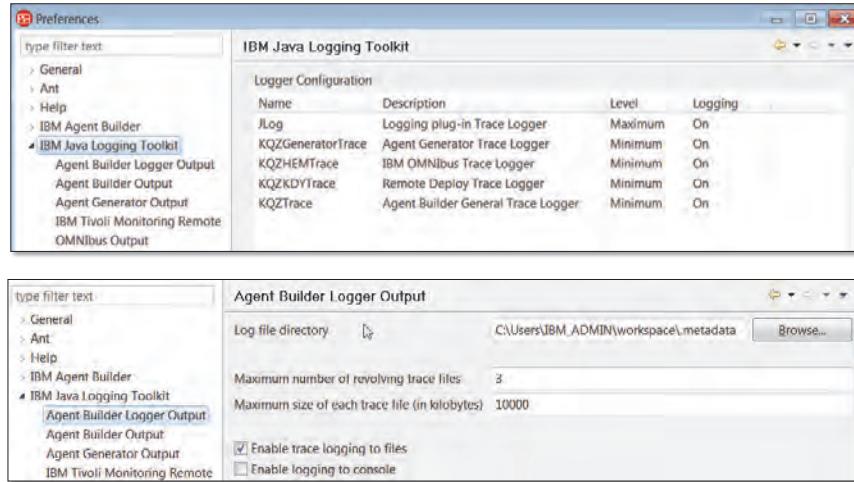
To troubleshoot problems within Agent Builder, consult the Agent Builder logs.

The default locations of Agent Builder trace logs are listed in the slide. You can change these locations per the Agent Builder plug-in. Notice that the default location places the log files within the current workspace directory.

The **metadata** directory is a hidden directory by default. Methods of making hidden directories visible are listed.

Setting Agent Builder trace options

- Trace options are set in the Agent Builder user interface on a per plug-in basis
- Trace settings take effect at run time (except for the console logging option)
- Access the tracing options by clicking **Windows > Preferences** and selecting IBM Java Logging Toolkit



Setting Agent Builder trace options

You can configure logging from within the Agent Builder application.

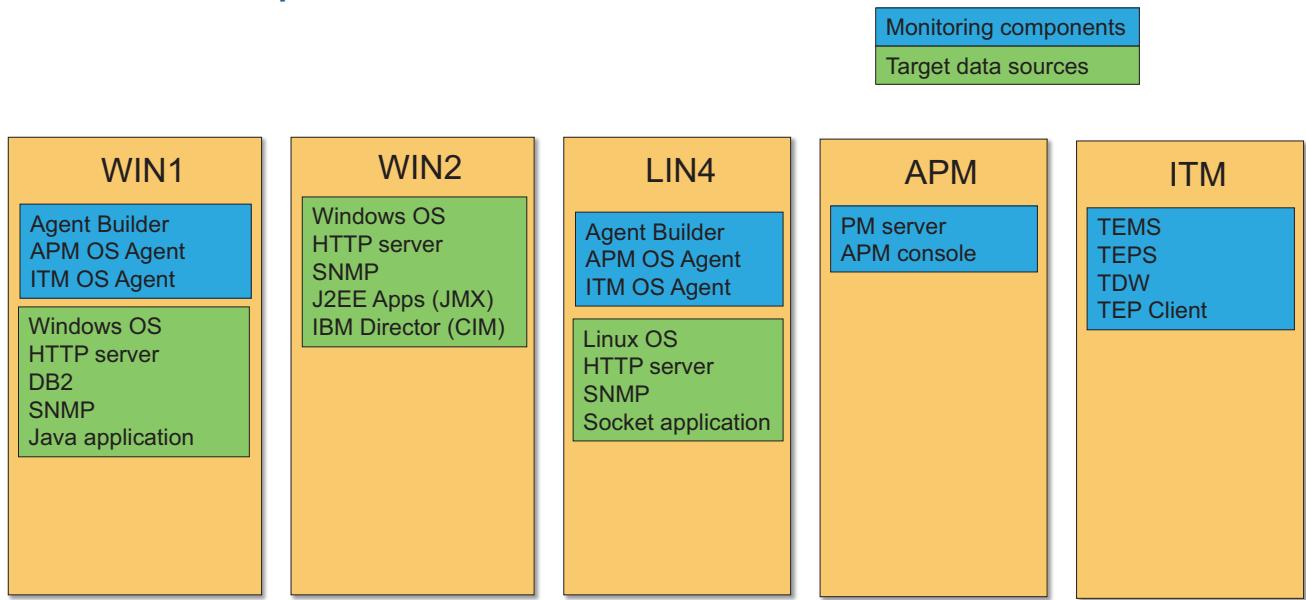
Select IBM Java Logging Toolkit to configure the following property for each plug-in:

- Name for each log file
- Log level
- Enable or disable logging

Select each plug-in from the Navigator to set the following parameters:

- Log file directory
- Number of revolving log files
- Maximum size of each log file
- Enable or disable logging
- Enable or disable logging to the console

Classroom computer architecture



Agent creation basics

23

© Copyright IBM Corporation 2017

Classroom computer architecture

This slide shows the computer environment that you use to complete the exercises in this course. You work with three monitored systems (WIN1, WIN2, and LIN4) and one monitoring server (APM or ITM).



Important: You must choose to deploy your agents into either an IBM Application Performance Management or IBM Tivoli Monitoring environment. You cannot do both. You use the monitoring server (APM or ITM) that is appropriate for your choice.

The items in the dark shaded box (blue) are monitoring components. The items in the lighter shaded box (green) are software that is installed on each box and represent different sources of monitored data.

Student exercises



Exercise 1: Create an agent to monitor specific Windows services

Student exercises

Open your Student Exercises book and complete the exercises for this unit.

Lesson 4 Creating installers and installing an Agent Builder agent

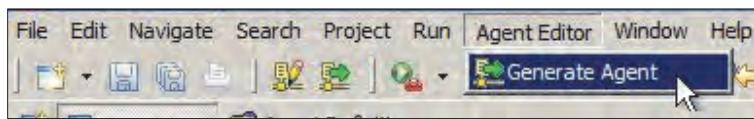
IBM Training

IBM

Lesson 4: Creating installers and installing an Agent Builder agent

Agent output and installation overview

- Types
 - Install locally: Install agent and ITM application support directly on local computer
 - Output agent installer: Create script-based installers for installing agent or ITM application support Archived in single zip and tar files
- Can be done from within from Agent Builder graphical user interface or command-line interface
- In Agent Builder, performed with Agent Generator wizard



- A standard agent must be installed on the target

Note: Performing these actions from the CLI will be taught in a later unit.

Agent creation basics

25

© Copyright IBM Corporation 2017

Instructor:

This lesson introduces the various ways in which you can install an agent and provides detailed instruction on installing it directly into an IBM Tivoli Monitoring environment.

After completing this lesson, you should be able to do the following tasks:

- Describe the types of Agent Builder agent outputs and installations
- Complete a quick installation of your agent onto the Agent Builder host
- Output the script-based installers from the Agent Builder GUI
- Install the agents and application support by using the script-based installers

Agent output and installation overview

Install an agent and its application support in one of two ways:

- **Local install:** This option is for installing and testing an agent and its IBM Tivoli Monitoring application support onto the Agent Builder host. Use it when Agent Builder is running on a host where you need to install the agent or agent application support.
- **Output agent image:** This method generates script-based installers and archives them into a compressed archive. It is the most commonly used method of installation. You use it when you install the agent or application support on a system other than your Agent Builder workstation.

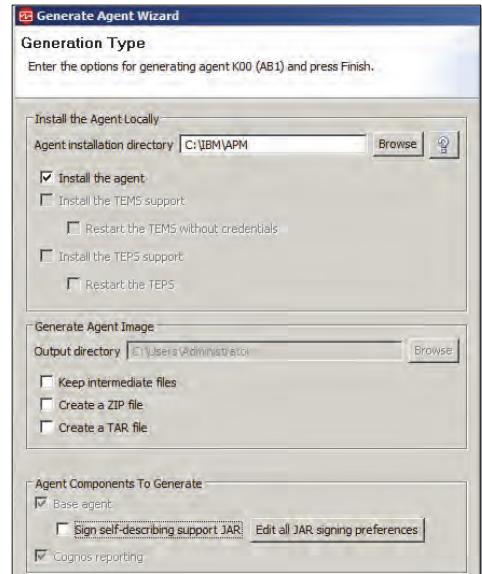
You can do the local installation and output of the agent image from within the Agent Builder application and from a command-line interface (CLI) process.



Important: Agent Builder agents require you to install a standard APM agent on the target host before you can install Agent Builder agents. The standard APM agent provides infrastructure that the Agent Builder agent uses.

Install agent locally

- Select installation directory
 - Must contain installation of pay-per agent
 - Options will appear based on what is found in target directory
- Select **Install the agent**
- Optional: Set and configure custom key stores for security signing



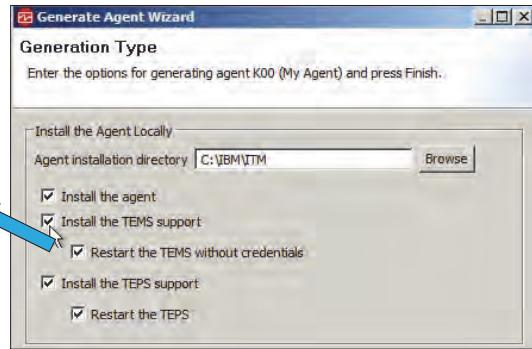
Install agent locally

As with all the output methods, you use the Agent Generation wizard to do a direct installation. You can start the Agent Generator from the menus or the toolbar. The direct installation can install only application support into local Tivoli Enterprise Monitoring Server and Tivoli Enterprise Portal Server.

Enter the path to an existing IBM Tivoli Monitoring installation on the host. If a Tivoli Enterprise Monitoring Server or Tivoli Enterprise Portal Server is found, the options to install the application support appear. After you click **OK**, Agent Builder installs the agent on the local host.

Install ITM application support locally

- Options will show, if TEMS or TEPS found in installation directory
- Select to install TEMS or TEPS support



- Select to restart Tivoli Enterprise Monitoring Server or not
In 6.2 FP1+, application support can be installed and activated without restarting Tivoli Enterprise Monitoring Server
Requires login
- Select to restart Tivoli Enterprise Portal Server or not
Must be restarted to activate application support

Important: With self-describing agents, Tivoli Enterprise Monitoring Server and Tivoli Enterprise Portal Server application support installations are not required

Installing ITM application support locally

Enter the path to an existing IBM Tivoli Monitoring installation on the host. If a Tivoli Enterprise Monitoring Server or Tivoli Enterprise Portal Server is found, the options to install the application support show.

After you click **OK**, Agent Builder installs all appropriate agent components on the local host, including the agent, the Tivoli Enterprise Monitoring Server, Tivoli Enterprise Portal Server, and desktop application support for Tivoli Enterprise Portal.

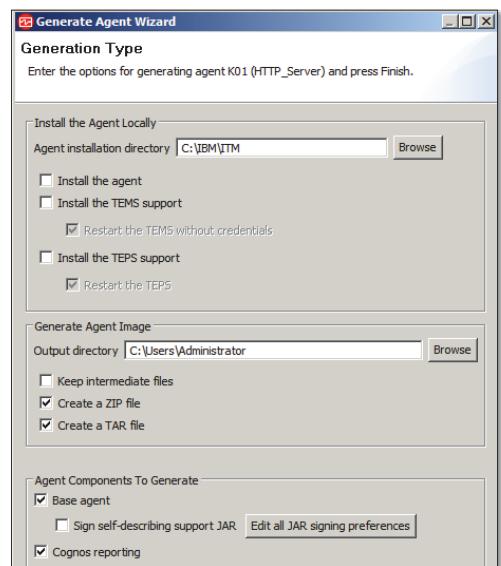
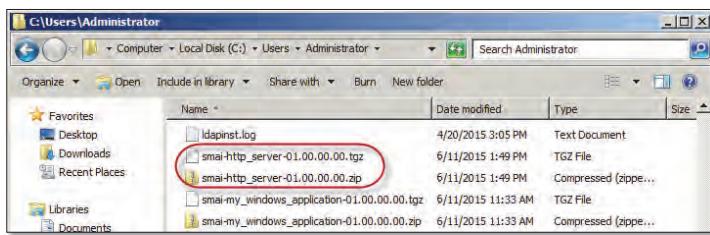
In Agent Builder 6.2 FP1 or 6.2.1, if you are installing application support for Tivoli Enterprise Monitoring Server, you are prompted to log in. If you log in, the application support is installed without restarting Tivoli Enterprise Monitoring Server and Tivoli Enterprise Portal Server. You can skip the login by selecting **Cancel**, which causes the Tivoli Enterprise Monitoring Server and Tivoli Enterprise Portal Server to restart after the application support is installed. You must restart Tivoli Enterprise Servers before version 6.2.1 to activate new application support.



Important: Agent Builder agents are self-describing by default. Therefore, application support installations on the Tivoli Enterprise Monitoring Server and Tivoli Enterprise Portal Server are not required. The agent can upload the application support when it connects to the Tivoli Enterprise Monitoring Server. In all labs in this course, the application support installations are done to speed up the process, but they are not required.

Creating compressed installers

- Creates a compressed installer that is used to install agent or IBM Tivoli Monitoring application support
- Select output format: ZIP or TAR file
 - Format specific to extractor, not target OS
 - Both contain the same installers for all platforms
 - File name: smai-*displayname*-version#.tgz/zip
- Optional: Select components to generate



Creating compressed installers

Besides local installations, the script-based installers are created with the Agent Generation wizard. You create multi-platform installers and then compress them into two archives: one a .zip file and the other a .tgz file. Both archives, .zip and .tgz contain the installers for all operating systems.

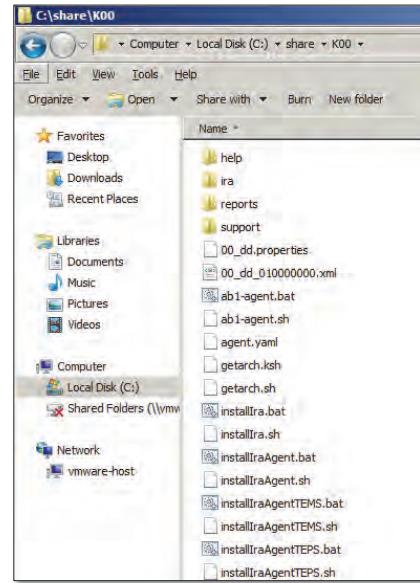
How to install by using installation scripts

1. Copy an archive file to target system
Notice that each archive contains installers for all operating systems
2. Unpack with platform-specific tool
3. Run the appropriate installation scripts
You receive messages indicating a successful or unsuccessful installation

```
C:\Users\Administrator>cd K01_zip
C:\Users\Administrator>K01_zip>installIra.bat C:\IBM\ITM -h ITM -u sysadmin -p object00
Installing agent into C:\IBM\ITM
Installing K01 ...
Install of K01 Agent successful.
Validating user...
KUIC00007I: User sysadmin logged into server on https://ITM:3661.
The requested service has already been started.
More help is available by typing NET HELPMSG 2182.

Install of K01 TEMS Support successful.
Online help for this agent will not be available until the Help Server is restarted, which
involves restarting the TEPS.

Install of K01 TEPS Support successful.
C:\Users\Administrator\K01_zip>
```



How to install by using installation scripts

The slide describes the process for installing the agent or application support with the script-based installers. The slide also shows the extracted .zip and .tar files and the installation commands that you run.

Windows installation files

- **installIraAgent.bat** installs the agent
`installIraAgent.bat <install_dir>`
where *install_dir* is the ITM installation directory in ITM or the agent installation directory in APM
- **installIraAgentTEMS.bat** installs support and seeds Tivoli Enterprise Managing Server
`installIraAgentTEMS.bat <install_dir> [-h Hub_TEMS -u username -p password]`
If login information not provided, restarts Tivoli Enterprise Managing Server
- **installIraAgentTEPS.bat** installs Tivoli Enterprise Portal Server, TEP client, and Eclipse Help Server support
`installIraAgentTEPS.bat <install_dir>`
Restarts Tivoli Enterprise Portal Server and Eclipse Help Server (if installed)
- **installIra.bat** runs the other three sequentially
`installIra.bat <install_dir> [-h Hub_TEMS -u username -p password]`

Windows installation files

There are four Windows installation files:

installIraAgent.bat installs the actual agent.

`installIraAgent.bat <install_dir>`



Important: The `installIraAgent` installation script is the only installation script that an IBM Application Performance Management environment uses.

installIraAgentTEMS.bat installs Tivoli Enterprise Managing Server support and seeds Tivoli Enterprise Managing Server.

`installIraAgentTEMS.bat <install_dir> [-h Hub_TEMS -u username -p password]`

If you do not provide login information, the file causes Tivoli Enterprise Managing Server to restart

installIraAgentTEPS.bat installs Tivoli Enterprise Portal Server, TEP client, and Eclipse Help Server support.

`installIraAgentTEPS.bat <install_dir>`

- This file causes Tivoli Enterprise Portal Server and Eclipse Help Server, if installed, to restart
- You must restart the TEP client manually to activate new support
- This file installs Tivoli Enterprise Portal Server, TEP client, Help support conditionally, based on the presence of the components, use the file on a Tivoli Enterprise Portal-only system

installIra.bat runs the other three installation files sequentially.

```
installIra.bat <install_dir> [-h Hub_TEMS -u username -p password]
```

For more information, see Chapter 23, “Testing and debugging your agent,” in the *IBM Agent Builder User’s Guide*.

UNIX and Linux installation files

- **installIraAgent.sh** installs the agent

```
installIraAgent.sh <install_dir> [-a architecture]
```

where *install_dir* is the ITM installation directory in IBM Tivoli Monitoring or the agent installation directory in IBM Application Performance Management

- **installIraAgentTEMS.sh** installs support and seeds Tivoli Enterprise Managing Server

```
installIraAgentTEMS.sh <install_dir> [-a architecture]  
[-h Hub_TEMS -u username -p password]
```

If login information not provided, restarts Tivoli Enterprise Managing Server

- **installIraAgentTEPS.sh** installs Tivoli Enterprise Portal Server, TEP client, and Eclipse Help Server support

```
installIraAgentTEPS.sh <install_dir>
```

Restarts Tivoli Enterprise Portal Server and Eclipse Help Server

- **installIra.sh** runs the other three sequentially

```
installIra.sh <install_dir> [-a architecture -h Hub_TEMS -u username -p password]
```

UNIX and Linux installation files

There are also four UNIX and Linux installation files:

installIraAgent.sh installs the actual agent.

```
installIraAgent.sh <install_dir> [-a architecture]
```



Important: The `installIraAgent` script is the only installation script to use in an IBM Application Performance Management environment

installIraAgentTEMS.sh installs Tivoli Enterprise Managing Server support and seeds Tivoli Enterprise Managing Server.

```
installIraAgentTEMS.sh <install_dir> [-a architecture] [-h Hub_TEMS -u username -p password]
```

If you do not provide login information, it causes Tivoli Enterprise Managing Server to restart.

installIraAgentTEPS.sh installs Tivoli Enterprise Portal Server, TEP client, and Eclipse Help Server support.

```
installIraAgentTEPS.sh <install_dir>
```

- Causes Tivoli Enterprise Portal Server and Eclipse Help Server to restart
- You must restart the TEP client manually to activate new support
- Installs Tivoli Enterprise Portal Server, TEP client, Help support conditionally, based on the presence of the components, use on a Tivoli Enterprise Portal-only system

installIra.sh runs the other three installation files sequentially.

```
installIra.sh <install_dir> [-a architecture -h Hub_TEMS -u username -p password]
```

For more information, see the “Testing and debugging your agent” chapter in the *IBM Agent Builder User’s Guide*.

Lesson 5 Troubleshooting Agent Builder agents

IBM Training



Lesson 5: Troubleshooting Agent Builder agents

- Installation log files
- Agent log files
- Agent tracing

This lesson introduces troubleshooting the installation of your agent and your running agent. This lesson focuses solely on the various log files that can provide troubleshooting information.

Installation log files

- Logs created installing an agent
 - K<pc>install.log or k<pc>installIraAgent.log*
 - Default location:
 - Windows: <install_dir>\TMAITM6[_x64]\logs
 - UNIX and Linux: <install_dir>/logs
- Logs created installing application support
 - <install_dir>\logs\K<pc>installITEMS.log or k<pc>installIraAgentITEMS.log*
 - <install_dir>\logs\K<pc>installTEPS.log or <pc>installIraAgentTEPS.log*
- Logs created uninstalling an agent
 - <install_dir>\logs\k<pc>uninstall.log

*Created only with local installation from Agent Builder on Windows

Installation log files

To troubleshoot problems with installing the Agent Builder agent or its application support, consult the log files that this slide lists.

A log file is created for each installer script to log problems with that installer. Additionally, the Tivoli Enterprise Monitoring Server or Tivoli Enterprise Portal Server each create a log file while installing the agent's application support.

The default location of the log file is shown for each operating system.



Hint: Local installations from the Agent Builder application on Windows create files with IraAgent string in their name.

Agent log files

- First place to look when troubleshooting an agent
- Agent Builder agent logging works the same as any other ITM or APM agent
- Logs created by agent
 - Windows: <hostname>_<pc>_k<pc>agent_<timestamp>-01.log
 - Linux or UNIX: <hostname>_<pc>_<timestamp>-01.log
 - <pc>_asfActivity_<timestamp>.log
- Default location
 - Windows: <install_dir>\TMAITM6[_x64]\logs
 - Linux or UNIX: <install_dir>/logs/

Agent log files

To troubleshoot problems with the agent after you install it, consult the agent log files. The slide lists the log file locations and names.

Notice that Agent Builder agent logs work the same as any IBM agent. They are stored in the same directories with the same naming conventions, and you configure them in the same ways.

Agent tracing

- More detail is available in the logs if tracing is turned on
- Trace level can be set per modules

Problem area	KBB_RAS1 setting
No error tracing	KBB_RAS1=none
General error tracing	KBB_RAS1=ERROR
Intensive error tracing	KBB_RAS1=ERROR (COMP:kqz ALL)
Agent interaction with management server	ERROR (UNIT:genericagent ALL)
Data collection	KBB_RAS1=ERROR (UNIT: ALL)
Availability data provider	ERROR (UNIT:availability ALL) (UNIT:winavailability ALL)
CIM data provider	ERROR (UNIT:cim ALL)
Custom data provider	ERROR (UNIT:custom ALL) (UNIT:cps_socket ALL) (UNIT:cpc1 ALL)

Problem area	KBB_RAS1 setting
ICMP Ping data provider	ERROR (UNIT:ping ALL)
Log file data provider	ERROR (UNIT:logmonitor ALL)
Perfmon data provider	ERROR (UNIT:queryclass ALL)
Script data provider	ERROR (UNIT:Shell ALL) (UNIT:commandwithtimeout ALL)
SNMP data provider	ERROR (UNIT:snmp ALL)
Windows Event Log data provider	ERROR (UNIT:eventlog ALL) (UNIT:winlog ALL)
WMI data provider	ERROR (UNIT:wmi ALL)

For more information, see the Troubleshooting chapter of User's Guide

Agent tracing

This slide lists logging parameters unique to Agent Builder agents.

As with the logging parameters for any IBM agent, these parameters are set in the agent environment file, pcENV, where pc is the product code. You can manually edit the configuration file to set trace logging or access it from the MTMS application.

To activate these changes, restart the agent.

Summary

Now that you have completed this unit, you should be able to perform the following tasks:

- Describe the basic process of creating an agent
- Create an agent that monitors the availability of Apache and DB2 Windows services
- Navigate and describe the Agent Builder editor interface
- Describe troubleshooting techniques for Agent Builder
- Install an Agent Builder agent with a quick local installation and with installation scripts
- Describe troubleshooting techniques for your installed agent

Unit 3.1 Customizing agents for IBM Application Performance Management

IBM Training



Customizing agents for IBM Application Performance Management

© Copyright IBM Corporation 2017
Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

In Part 1 of this unit, you learn how to modify an agent so that you can deploy it within an IBM Application Performance Management environment. You also learn how to generate and install the agent by using the script installers.

Scenario business objectives

- Existing AB1 agent
 - A company has a web application that provides order processing and client management
 - The application runs on multiple HTTP servers
 - The application pulls data from a DB2 database
 - The installations are Windows only
 - You must create an agent that, when installed on application servers, monitors key systems related to this application
- Needed solutions
 - The agent must integrate with an IBM Application Performance Management environment and when installed on the application servers provide summary and detailed dashboard to monitor the HTTP server and DB2 Windows services

Scenario business objectives

The scenario business objective for this unit is to modify the AB1 agent that you create in previous units so that you can install it into an IBM Application Performance Management environment.

Application Performance Management scenario solution

- Define attributes to generate the data needed for the PM console dashboards and OSLC resources
- Define dashboards and the monitored resources
- Create an application within the Performance Management Console called My Application
- Add the AB1 agent to this application

Application Component	Name	Status	Full Name	Type
Apache2.4	Apache2.4	UP	Apache2.4	SERV
DB2 - DB2COPY1 - DB2	DB2	UP	DB2 - DB2COPY1 - DB2	SERV
DB2AS - DB2AS00	DB2AS00	UP	DB2AS - DB2AS00	SERV
DB2 Governor (DB2COPY1)	DB2GOVERNOR_DB2COPY1	DOWN	DB2 Governor (DB2COPY1)	SERV
DB2 License Server (DB2COPY1)	DB2LIC_DB2COPY1	DOWN	DB2 License Server (DB2COPY1)	SERV
DB2 Management Service (DB2...)	DB2MGMTSVC_DB2COPY1	UP	DB2 Management Service (DB2...)	SERV
DB2 Remote Command Server (...)	DB2REMOTECMD_DB2COPY1	UP	DB2 Remote Command Server (...)	SERV

Customizing agents

3

© Copyright IBM Corporation 2017

Application Performance Management scenario solution

The scenario solution requires that you define a summary dashboard to show the overall status of the component it monitors and a details dashboard to show data from select data sources.

The scenario solution also requires that you define the OSLC resources for the agent so that the Performance Management server can identify and manage it.

Because the current agent does not contain the properties that the summary dashboard definitions need, the solution requires you to derive custom attributes for it.

Finally, to complete the agent deployment, you create the application within the Performance Management console and add the agent to that application definition.

Learning objectives

After completing this unit, you should be able to perform the following tasks:

- Modify an agent for an IBM Application Performance Management environment
- Create filters for existing data sources
- Manage an Agent Builder agent in an IBM Application Performance Management environment
- Manage an Agent Builder agent in an IBM Tivoli Monitoring environment
- Add custom IBM Tivoli Monitoring workspaces and situations to your agent

Learning objectives

The learning objectives that are presented in this slide include the skills that are needed to implement the scenario solution.

Up to this point in the class, you worked with the quick installation of your agent. This unit introduces the script installers that you use to install the agent on a computer other than the one on which Agent Builder is running. Use the script installers in both the IBM Application Performance Management, the focus of this unit, and the IBM Tivoli Monitoring environments.

Unit outline

- Lesson 1: Modifying agents for an IBM Application Performance Management environment
- Lesson 2: Creating filtered data sources
- Lesson 3: Managing agents in an IBM Application Performance Management environment
- **Exercise 1: Modifying the AB1 agent for IBM Application Performance Management**

This slide gives you an outline of the lesson material in this unit.

Lesson 1 Modifying an agent for an IBM Application Performance Management environment

IBM Training



Lesson 1: Modifying an agent for an IBM Application Performance Management environment

- Integrates with both IBM Application Performance Management on premises or SaaS
- Agent Builder tasks
 - Build the custom agent, including attributes for dashboards and resource definitions
 - Define the dashboards
 - Define the resource properties
- Output the agent to script installers
- IBM Application Performance Management tasks
 - Install, configure, and start the agent
 - Add the agent to an application

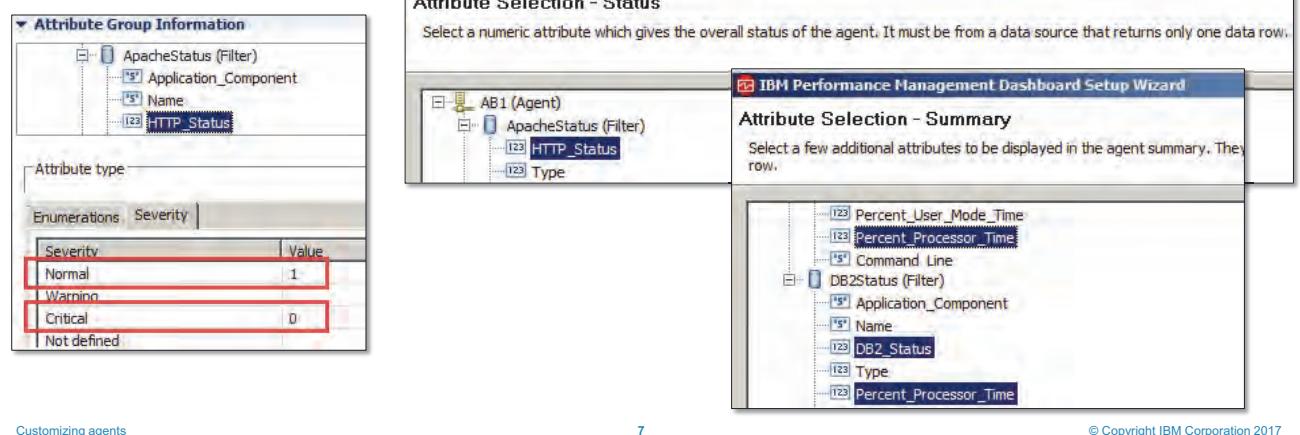
You can deploy Agent Builder agents within both the on-premises and SaaS versions of IBM Application Performance Management.

This slide shows the basic process that you use to modify an agent for an IBM Application Performance Management environment and to deploy the agent in that environment.

Use Agent Builder to configure the agent dashboards for the Performance Management Console and set Open Services Lifecycle Collaboration properties that are used to identify the agent to the APM environment.

Defining the summary dashboard

- Single-row attribute group with one numeric attribute with Severity values
- Up to five additional numeric attributes from single-row attribute groups can be added
- You might have to create attributes specific to these values
- Use the Dashboard wizard



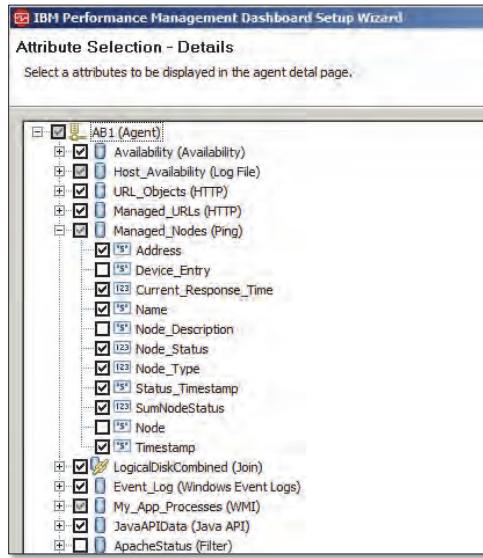
Defining the summary dashboard

You can set five attributes for display in the summary dashboard. Each of the attributes must come from an attribute group that gathers only a single row of data. One attribute defines the overall status of the monitored component. The attribute must also define values for status severity.

Use the Dashboard wizard that is found in the Outline pane on the right to set the properties that define the summary dashboard.

Defining details dashboard

Select the attribute groups and attributes to be displayed in the details dashboard

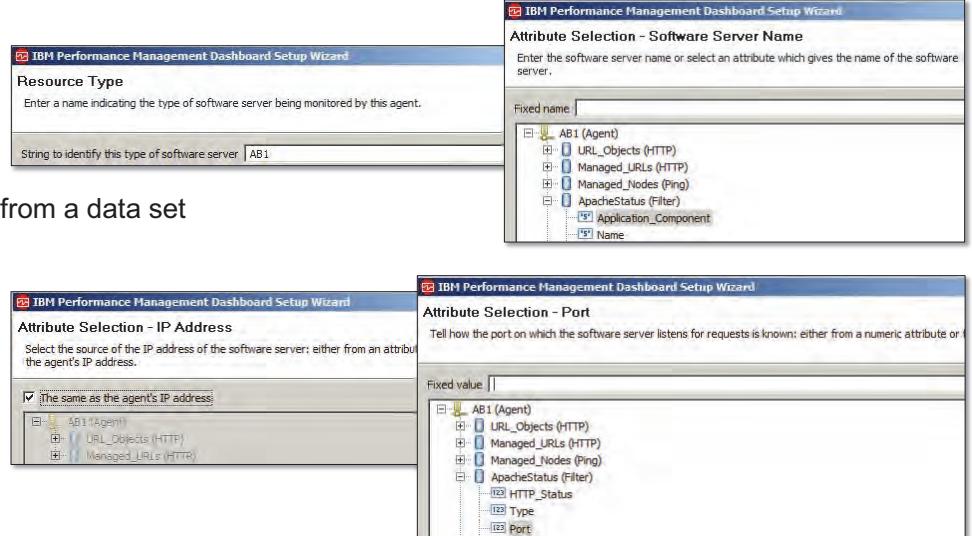


Defining details dashboards

From this window, you select attribute groups and attributes that you want displayed in the Detailed dashboard. Select entire attribute groups or select attributes within an attribute group.

Defining the monitored resource

- Building resource definitions for the agent
- Expects values for
 - Resource type
 - Software server name
 - IP address
 - Port
- At least one must come from a data set
 - You might have to create attributes specific to these values
- Use the Dashboard wizard



Defining the monitored resource

The Performance Management server uses Open Services Lifecycle Collaboration (OSLC) to identify and manage agents as they connect to the server.

Within Agent Builder, you must set the following properties so that the monitored resource is properly identified and managed:

- Resource (software server) type
- Software server name
- IP address
- Port

Use the Dashboard wizard that is found in the Agent Builder Outline pane to set these properties.

The resource, software server, type uniquely defines the type of thing that is monitored. Each resource type has the same monitoring infrastructure: what is monitored, how it is shown, threshold settings, historical data, and so on. Examples of resource types for standard agents include Linux operating system, WebSphere Application Server, and DB2 server.

Functionally for the implementer and system administrator, resource types have the following characteristics:

- Can deploy independently from other resource types
- Can add to applications independently from other resource types
- Have their own summary dashboard, details dashboard, and thresholds

In most instances, the entire agent represents a single resource, software server, type. In the Dashboard wizard, the agent name is offered as the resource type name, but you can manually set it to something else. With subnodes, it is possible that each subnode represents a different resource type. In the Dashboard wizard, the subnode name is offered as the resource type, but you can manually set it to something else.

Use software server name, IP address, and port values to identify the specific monitored software server. Commonly, use the target host name for the software server name. You must pull these property values from existing attributes. You must define those attributes before entering the OLSC wizard.

Lesson 2 Creating filtered data sources

IBM Training

IBM

Lesson 2: Creating filtered data sources

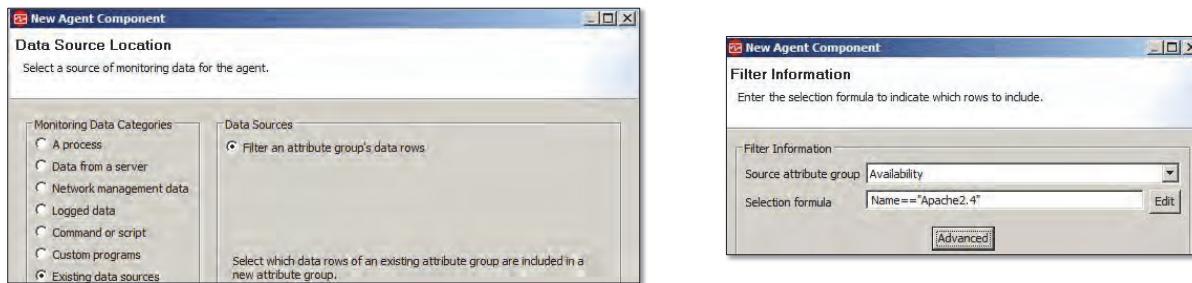
- Summary and resource attributes must come from single-row data sets
- Attribute filters filter an existing data set to create a unique, smaller data set
- Very helpful for creating single-row data sets for APM Summary dashboards and resource definitions

The screenshot shows the APM interface with the following components:

- Data Tables:** Two tables are displayed, each showing one data row returned at different times.
 - The first table has columns: Application_Component, Name, HTTP_Status, Full_Name, Type, Virtual_Size, Page_F. It shows one row for Apache2.4.
 - The second table has columns: Application_Component, Name, DB2_Status, Full_Name, Type, Virtual_Size, Page_F. It shows one row for DB2 - DB2COPY1 - DB2.
- Status Overview:** A summary section for component AB1, showing:
 - HTTP Status: UP
 - Percent Processor Time: 0
 - DB2 Status: UP
 - Percent Processor Time: 0

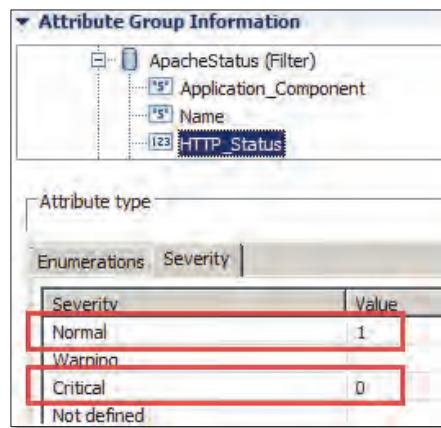
Adding a filtered data source

- Add new data source
- Select **Existing data sources > Filter an attribute group's data rows**
- Select the source attribute group and enter the selection formula
- For Summary dashboards, select an overall status attribute and set severity values



Setting attribute severity

- Sets a severity level to specific numeric values
 - Applies only to attributes of numeric type
 - Required for overall status attribute in APM Summary dashboards
- Creates color coding in the monitored environment displays:
 - Normal: Green
 - Warning: Yellow
 - Critical: Red
 - Not defined: White



Lesson 3 Managing agents in an IBM Application Performance Management environment

IBM Training



Lesson 3: Managing agents in an IBM Application Performance Management environment

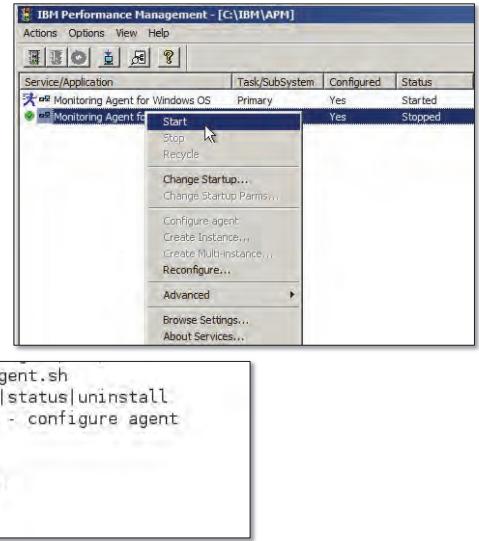
- Configuring, starting, stopping, and uninstalling agents
- Confirming agents in the Performance Management Console

This lesson covers tasks that are related to managing an agent in the IBM Application Performance Management environment.

Configuring, starting, stopping, and uninstalling agents

Use same platform-specific process as standard APM agent

- IBM Application Performance Management utility (Windows only)
- Command line
 - `name-agent.sh config`
 - `name-agent.sh start`
 - `name-agent.sh stop`
 - `name-agent.sh status`
 - `name-agent.sh uninstall`
- Default command locations
 - Windows
`<install_dir>\BIN\`
 - Linux or UNIX:
`<install_dir>/bin/`



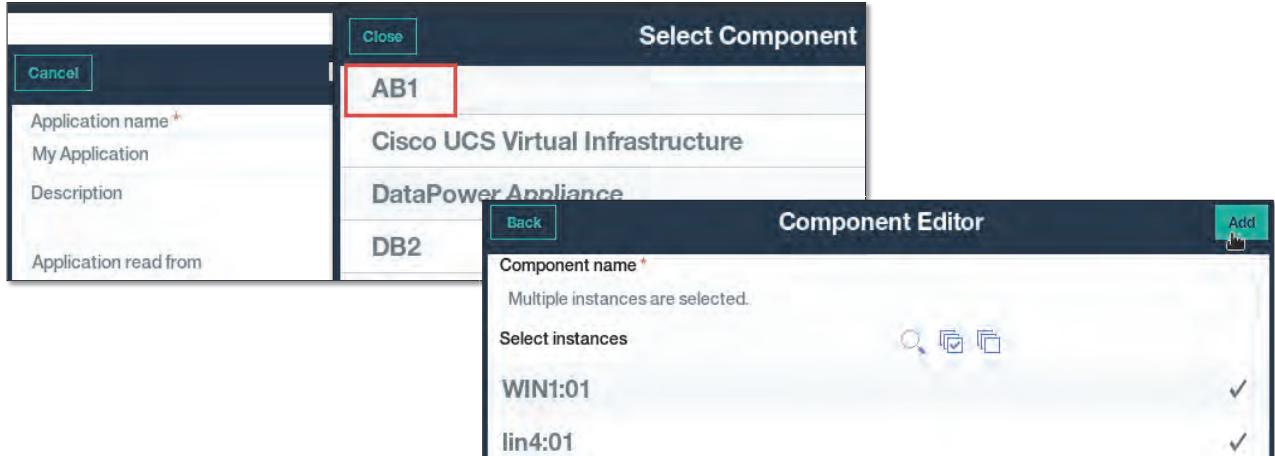
After you install the agent, you must configure and start the agent, as you would with any APM agent. Using the same command, you can stop, show status of, and uninstall the agent.

Confirming agents in the Performance Management Console

- Add an agent to application
- Confirm component summary dashboard
- Confirm details dashboard

Adding an agent to an application

After it starts, the agent automatically becomes available to be added to an application in the Performance Management console



To view the monitored data and events that are generated by an APM agent in the Performance Management console, add the agent component to an application. After an Agent Builder agent starts, it is visible in the Add Application section in the Performance Management console, and you can assign it to an application.

Confirming the component summary dashboard

- Summary dashboard is displayed in Components under the Application and Groups
- Contains summary widgets that display the overall status of the monitored component
Must come from single-row attribute groups
- One attribute is defined as the overall status for the component and requires severity definitions

The screenshot shows the 'Status Overview' section of the component summary dashboard for 'AB1'. The left sidebar lists 'Applications' (All My Applications, My Application, My Components) and 'Groups' (Components, AB1). The main area displays a summary for 'WIN1 - AB1' with two rows of metrics: 'HTTP Status' (UP) and 'Percent Processor Time' (0). Below this, there are four summary icons: red (0), yellow (0), green (2), and blue (0).

Customizing agents

17

© Copyright IBM Corporation 2017

Confirming the component summary dashboard

In the Performance Management Console, the summary dashboard is visible in the Components dashboard under the application and Groups. It shows up to five summary attributes that indicate the overall status of the monitored component. Each of these attributes must come from single-row attribute groups. One attribute is defined as the overall status for the component and requires that you define status severities.

Confirming the component details dashboard

- Clicking the summary widget navigates to the details tab
- Details shows a widget for select data sets from the agent

Availability										
Application Component	Name	Status	Full Name	Type	Virtual S...	Page Fa...	Working...	Thread ...	PID	
Apache2.4	Apache2.4	UP	Apache2.4	SERVICE	45	0	9	7	1	
DB2 - DB2COPY1 - DB2	DB2	UP	DB2 - DB2COPY1 - DB2	SERVICE	731	0	176	38	3	
Managed_URLs										
URL	Respons..	Page Size	Page Ob..	Total Obj..	Page Title	Server Type	Respons..	Status		
http://win1/	38	1501	8	92572	Any Bank home page	Apache/2.4.25 (Win64) OpenSSL/1.0.2-fips PHP/7.2.10	200	OK		
http://win1/AnyBank.htm	31	861	2	92949	Any Bank Banking page	Apache/2.4.25 (Win64) OpenSSL/1.0.2-fips PHP/7.2.10	200	OK		
Managed_Nodes										
Address	Device Entry	Current ...	Name	Node Description	Node St...	Node Type	Sta...			
192.168.1.103	WIN1	1	win1.ibm.edu	PingList	Active	IP_Node	3/1			
192.168.1.103	192.168.1.103	1	win1.ibm.edu	PingList	Active	IP_Node	3/1			
Event_Log										
Log Name	Event Source	Event Ty...	Event ID	Event Category	Message	Time Generat...				
Application	DB2	Warning	5	None	2017-03-14-18.05.21.099000 Inserted 1 row(s).	3/14/17, 6:05 F				
Application	DB2	Informa...	1	None	DB2STOP : SQL1064N DB2STOP...	3/14/17, 6:05 F				

Customizing agents

© Copyright IBM Corporation 2017

Confirming the component details dashboard

After you install the agent, you must configure and start the agent, as you would with any APM agent.

APM agent creation process summary

1. Create the agent in Agent Builder
 - Agent information
 - Data source information
 - Runtime configuration
2. Test and update the agent
 - Test in Agent Builder
 - Individual data source
 - Full agent
 - Define dashboards and OSLC resources
 - Install and confirm the agent in test environment
 - Revise and retest as needed
3. Create and install production agent
 - Output final script installers
 - Commit agent project
 - Install production agents

APM agent creation process summary

This slide summarizes the process that this unit teaches to create and deploy a final Agent Builder agent in an IBM Application Performance Management environment.

Student exercises



APM Exercise 1: Modify and install the AB1 agent into an IBM Application Performance Management environment

Student exercises

Open your Course Exercises book and complete the exercises for this unit.



Important: This lab is the only standard lab that deploys an agent in the IBM Application Performance Management environment. After this lab, deploy all agents in subsequent labs in the IBM Tivoli Monitoring environment.

Summary

Now that you have completed this unit, you should be able to perform the following tasks:

- Modify an agent for an IBM Application Performance Management environment
- Create filters for existing data sources
- Manage an Agent Builder agent in an IBM Application Performance Management environment

Unit 3.2 Customizing agents for IBM Tivoli Monitoring

IBM Training

IBM

Customizing agents for IBM Tivoli Monitoring

© Copyright IBM Corporation 2017
Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

Part 2 of this unit gives you an overview of how to install an agent into an IBM Tivoli Monitoring environment and how to build custom application support for an IBM Tivoli Monitoring environment.

Scenario business objectives

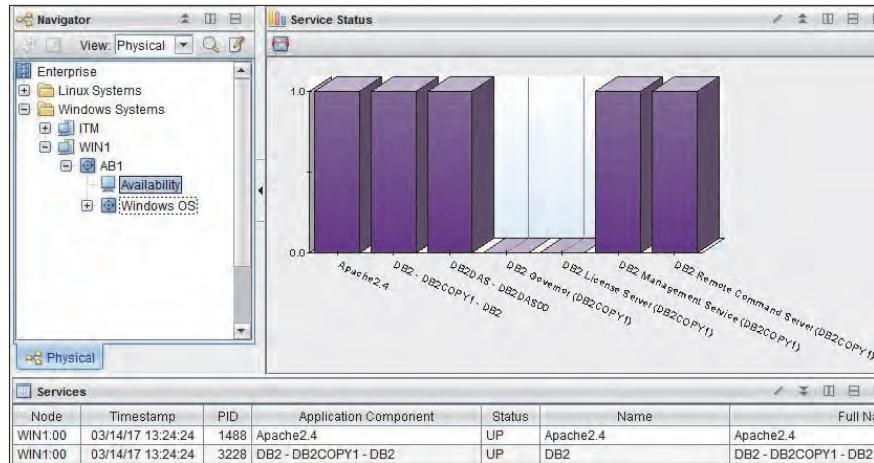
- Existing AB1 agent
 - A company has a web application that provides order processing and client management
 - The application runs on multiple HTTP servers
 - The application pulls data from a DB2 database
 - The installations are Windows only
 - You must create an agent that, when installed on application servers, monitors key systems related to this application
- Needed solution
 - The agent must integrate with an IBM Tivoli Monitoring environment and when installed on the application servers provide custom workspaces and situations to monitor the HTTP server and DB2 Windows services

Scenario business objectives

The agent must integrate with an IBM Tivoli Monitoring environment and when you install it on the application servers provide custom workspaces and situations to monitor the HTTP server and DB2 Windows services.

IBM Tivoli Monitoring scenario solution

- Installable into IBM Tivoli Monitoring environment
- Default workspace that displays availability data in tabular and a bar chart
- Custom situation that generates an event when either service is down
- Install the solution, including the custom workspace and situation



In Unit 2, you created an agent that monitors the HTTP Server and DB2 Windows services of this order processing and client management application. No additional work is needed to allow IBM Agent Builder agents to integrate with an IBM Tivoli Monitoring environment.

The solution includes a custom default workspace that quickly shows the current availability of all monitored services and a detailed display of all availability data collected. The solution includes a custom situation that creates an event whenever any of the monitored services is not running.

Finally, you must package the agent and all of the custom application support into an installer so that you can install the application support and agents in the production environment.

Learning objectives

After completing this unit, you should be able to perform the following tasks:

- Manage an Agent Builder agent in an IBM Tivoli Monitoring environment
- Add custom IBM Tivoli Monitoring workspaces and situations to your agent

Unit outline

- Lesson 1: Managing agents in an IBM Tivoli Monitoring environment
- **Exercise 1: Installing an agent in an IBM Tivoli Monitoring environment**
- Lesson 2: Creating and importing ITM application support
- **Exercise 2: Creating and importing custom ITM application support**
- **Exercise 3: Installing and confirming custom ITM application support**

Lesson 1 Managing agents in an IBM Tivoli Monitoring environment

IBM Training



Lesson 1: Managing agents in an IBM Tivoli Monitoring environment

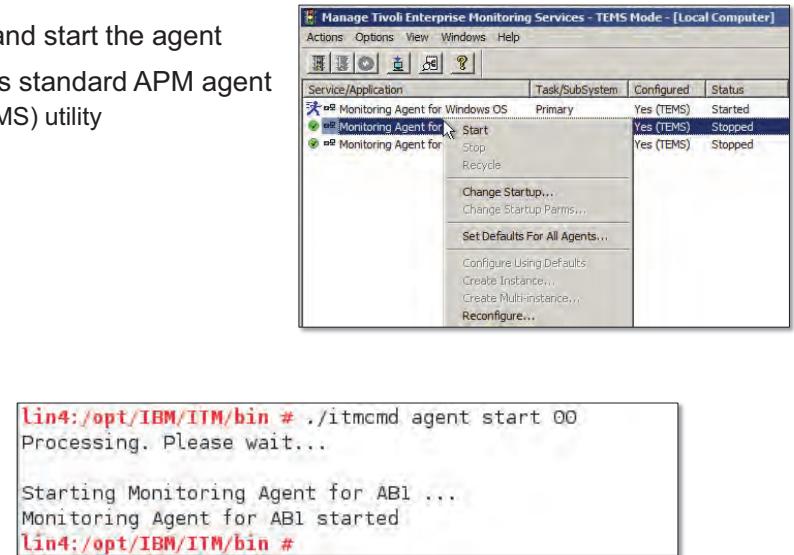
- Configuring, starting, and stopping agents
- Uninstalling an Agent Builder ITM agent
- Confirming agents in the Tivoli Enterprise Portal

Configuring, starting, and stopping agents

- After installation, you must configure and start the agent
- Use same platform-specific process as standard APM agent
 - Manage Tivoli Monitoring Services (MTMS) utility
 - Command

```
itmcmd config -A pc
itmcmd agent start pc
itmcmd agent stop pc
```

where pc is the agent product code
 - Default command locations
 - Windows
<install_dir>\BIN\
 - Linux or UNIX:
<install_dir>/bin/



Configuring, starting, and stopping agents

After installing the agent, you must configure and start it. You complete these steps by using the standard, platform-specific processes that you use to configure and install any IBM agents.

The slide shows the Manage Tivoli Monitoring System (MTMS) interface, which you might often use to configure and start an agent. Command-line options are also available.

Uninstalling ITM agents

- Uninstalling:
 - Windows

```
{agent_home}\TMAITM6\k[xx]_uninstall.vbs {{agent_home}}\}
```

where agent_home is the installation directory and xx is the product code for the agent
 - UNIX or Linux

```
{agent_home}/bin/uninstall.sh [-f] [-i] [-h {agent_home}] [xx]
```

where agent_home is the installation directory and xx is the product code for the agent
- Remove from Tivoli Enterprise Portal
 - Right-click the Navigator nodes under agent and select **Clear off-line entry**
 - For multiple agents, Managed System Status workspace, select all of the managed systems, right-click, and select **Clear off-line entry**

Uninstalling ITM agents

Uninstallation scripts are provided for uninstalling your Agent Builder agents. In Windows, a Visual Basic script is created for each agent. To uninstall an agent, run the script specific to that agent. In UNIX and Linux, you use the standard Tivoli Monitoring **uninstall** command and provide the product code specific to your agent.

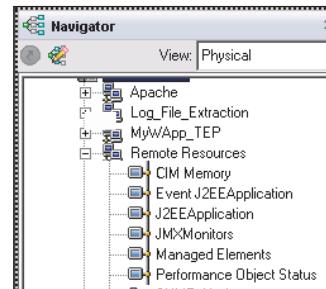
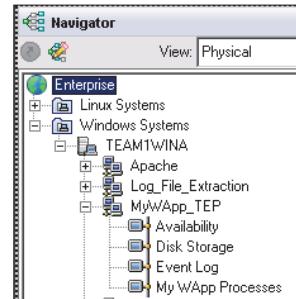
After uninstalling an agent, remove the agent from the Tivoli Monitoring managed systems list and the Navigator in the Tivoli Enterprise Portal. Steps for each of these processes are the same as you do for any IBM agent. For single managed systems, right-click the nodes under the agent in the Navigator and select **Clear off-line entry**.

For multiple agents, go to the Managed System Status workspace, select all of the IBM Tivoli Managed Systems for your agent, right-click, and select **Clear off-line entry**.

Confirming a new agent in IBM Tivoli Monitoring

Confirm the Navigator Items

- Open Tivoli Enterprise Portal client and confirm Navigator items
- For availability monitors, all data is displayed under one Availability node
- For performance monitors
 - Each attribute group gets own node
 - Performance Object Status displays status of all performance monitors



Confirming a new agent in IBM Tivoli Monitoring

After you install your agent, complete the testing process by confirming the new agent.

Confirm the navigator items

The first confirmation step is to confirm the agent display in the Tivoli Enterprise Portal. What Navigator items you can expect depends on the data source the agent is monitoring.

All availability monitor data is grouped into one Navigator item: Availability.

All other data sources have one Navigator item for each attribute group that you are monitoring, unless they are being grouped with a Navigator group. This process is described in [Unit 4, "Monitoring Windows resources"](#) on page 4-1.

Any time your agent monitors performance (nonavailability) data sources, you also get a single Performance Object Status Navigator item. This Navigator item provides status information about each of the performance data source monitors.

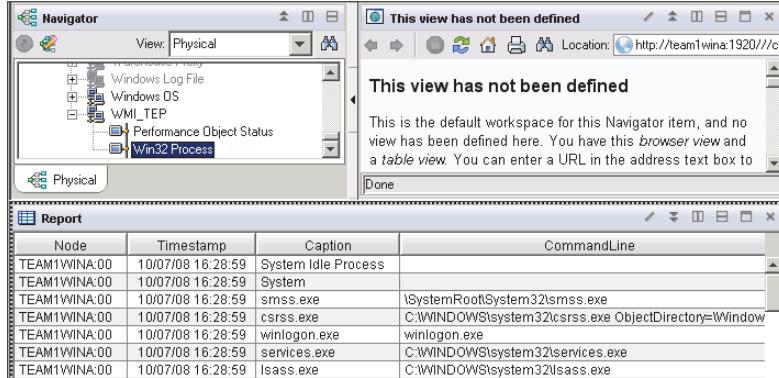
If you combine availability and performance data sources, you do not receive the Performance Object Status Navigator item. Instead, the attribute group that contains the information is associated with the Availability Navigator item.



Note: For any agent installation of Tivoli Enterprise Portal and TEP client application support, stop TEP client before installing or restart after installing.

Confirm the data display in the Tivoli Enterprise Portal client

- No default workspace for Agent Builder Navigator items
 - Notice lack of workspace name in upper left
- If single attribute group assigned Navigator item, shows single table of data
- If more than one attribute group assigned Navigator item, shows blank workspace with a query error
 - To see data, add a view for each attribute group or query
 - Save workspace



Confirm the data display in the Tivoli Enterprise Portal client

After you confirm that you have the Navigator items that you expect, you must confirm that the agent is gathering the data that you expect. Click each Navigator item and view the data.

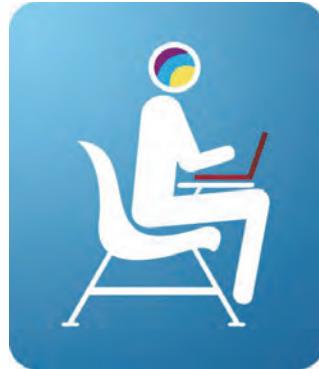
The default workspace for Agent Builder Navigator items is a single table view. In some instances, the expected initial workspace shows no data. This situation is true of any workspace that has more than one data source that is associated with it, such as the Availability workspace when you also have performance data sources. In this case, both the Availability and Performance Object Status attribute groups are associated with the Availability Navigator item. Also, a Navigator group with two or more attribute groups that are associated with it do not show data until you manually create the views.

This situation occurs because Agent Builder Navigator items have no default workspaces. The default Tivoli Enterprise Portal behavior creates the workspaces that are shown based on the single query that is installed with the agent application support.

If no workspace is created for a Navigator item, the Tivoli Enterprise Portal checks to see what queries are associated with that Navigator item. If only one query is associated, the Tivoli Enterprise Portal creates a temporary workspace that contains a single table view of that data. But, if the Navigator item has more than one query that is associated with it, the Tivoli Enterprise Portal does not show any data. You must manually create a view for each query, if you want to see and confirm the data.

This reason also explains why the workspace name is not shown in the upper left title bar because no workspace exists in the Tivoli Enterprise Portal database for that Navigator item.

Student exercises



IBM Tivoli Monitoring Exercise 1: Installing the AB1 agent in an IBM Tivoli Monitoring environment

Student exercises

Open your Course Exercises book and complete the exercises for this unit.

Lesson 2 Creating and importing IBM Tivoli Monitoring application support

IBM Training



Lesson 2: Creating and importing IBM Tivoli Monitoring application support

- Default application support with Agent Builder agents
 - Single query for each attribute group
 - Navigator items for availability monitors, for each performance monitor, and possibly Performance Object Status
 - An on-the-fly workspace for Navigator items with one query
- You can add the following items to your agent:
 - Queries
 - Workspaces
 - Situations
 - Take Actions

In this lesson, you learn to add application support, such as queries, workspaces, situations, and Take Actions, to your Agent Builder agent.

Before describing the types of Tivoli Monitoring components you can create for an agent, it is helpful to restate exactly what application support is provided with an Agent Builder agent by default. Agent Builder agents come with only the following application support components:

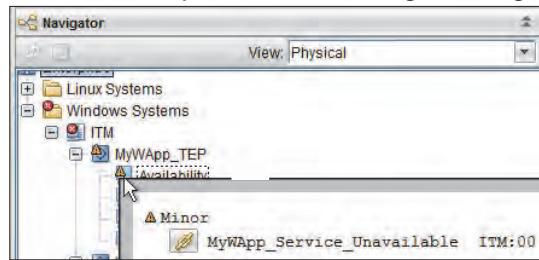
- Single query for each attribute group
- Navigator items for availability monitors, for each performance monitor, and possibly Performance Object Status
- A dynamic workspace for Navigator items with one query

As stated, you can add Tivoli Monitoring queries, workspaces, situations, and Take Actions to your Agent Builder agent.

Because the Solution Install image is the only installation method that can include your custom application support, create them after you finalize the base agent.

Creating workspaces, queries, situations, and Take Actions

- Items must be created in Tivoli Enterprise Portal and imported back into agent in Agent Builder
- Immediately available for importing
 - Queries
 - Situations
 - Take Actions
- You must tag and publish workspaces



Note: Creating these application support components is outside the scope of this class and is considered a prerequisite skill

Creating workspaces, queries, situations, and Take Actions

Any custom Tivoli Monitoring components that you create for your Agent Builder agent are created in the Tivoli Enterprise Portal and then imported into your agent definition in Agent Builder.

Situations, queries, and Take Actions that are associated with your agent are immediately available for importing. You must tag and publish workspaces before you import them into your agent in Agent Builder.

Creating importable workspaces

- Steps to creating importable workspaces
 - Enable tagging of workspaces
 - Publish workspaces
- Enabling tagging of workspaces
 - Stop Tivoli Enterprise Portal client
 - Edit the **set _CMD** line of the **the path_to_file\cnp.(bat or sh)** file by inserting the following option:
-Dcnp.candle.mode="\$_KCJ_\$"
 - **path_to_file** locations:
 - Windows: {itm_install}\CNP
 - UNIX and Linux:
{itm_install}/li263/cj/bin/
 - Restart Tivoli Enterprise Portal

```
cnp.bat - Notepad
File Edit Format View Help
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
-DCnp.http.url.host=TEAM1WINA -Dvbroker.agent.enableLocator=false
-Dnv_inst_flag=%NV_INST_FLAG% -Dnvwc.cwd=%NVWC_WORKING_DIR%
-Dnvwc.java=%NVWC_JAVA% candle.fw.pres.CMWApplet %1 %2 %3 %4 %5 %6
%7 %8 %9 %10
```

Creating importable workspaces

If you want to import workspaces into Agent Builder, you must set up the following two conditions:

- You must enable the Tivoli Enterprise Portal client to tag workspaces for import into Agent Builder.
- You must publish the workspaces in workspace administration mode.

Enabling tagging of workspaces

The Tivoli Enterprise Portal client must tag workspaces with a special code that identifies them as importable into Agent Builder. To enable this tagging, modify the Tivoli Enterprise Portal client that you are using to publish the workspaces. Set the Tivoli Enterprise Portal client startup script option that is shown in the slide.

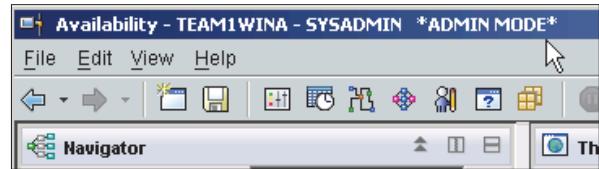


Note: Make this change one time. You do not have to repeat it for subsequent workspaces you want to import into Agent Builder.

Creating importable workspaces (continued)

Publishing workspaces

1. Create your custom workspace
2. Turn on Admin mode
3. Publish your custom workspace by resaving it
4. Turn off Admin mode



Tips

- Consider setting workspaces to unmodifiable and undeletable in Properties.
- Do not save workspaces other than the ones you want imported
- Turn off Admin mode when you are finished

Publishing workspaces

After you enable the tagging of workspaces for import into Agent Builder, you must then publish the workspaces that you want to import. Publishing of workspaces requires activating the Tivoli Enterprise Portal workspace administration mode (Admin mode) and then saving the workspace.



Note: Publishing of workspaces is a standard Tivoli Enterprise Portal administrator process that makes new or modified workspaces available to Tivoli Enterprise Portal users.

While Admin mode is active, any workspaces that you save are published and tagged for import into Agent Builder. Saving other workspaces while in Admin Mode can change global workspaces. In this case, workspaces that are not part of your Agent Builder solution might be marked for import into your agent. Therefore, when publishing workspaces, enter and exit the Tivoli Enterprise Portal Admin Mode for each workspace you are publishing.

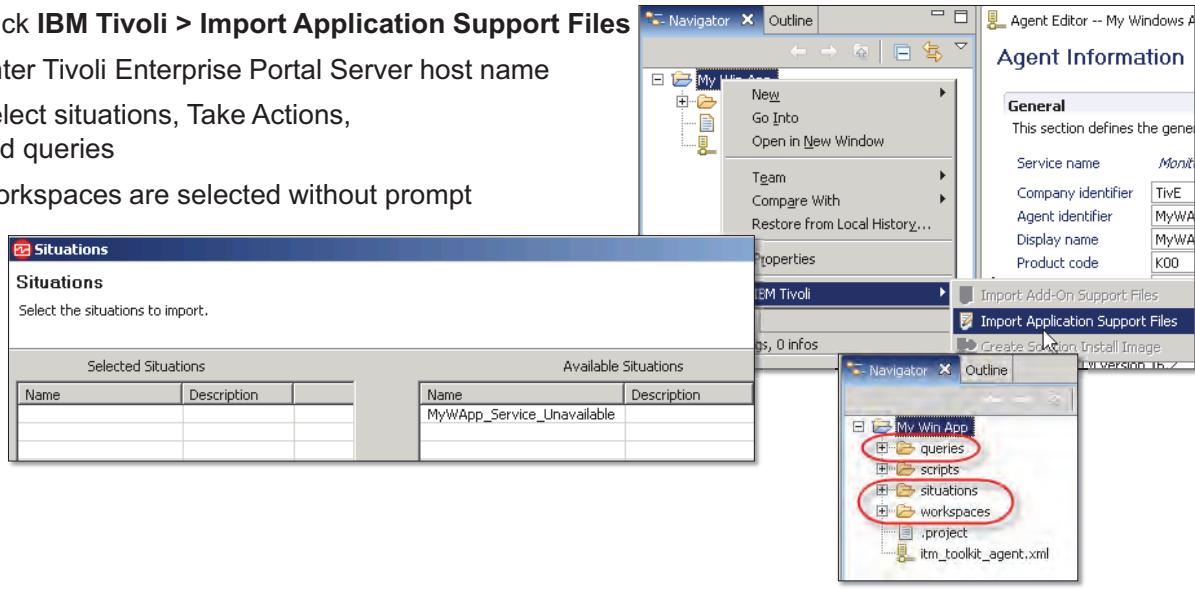
When you finish creating your workspaces, do not forget to turn off Admin Mode.



Note: Detailed steps for turning Admin Mode on and off are included in the student exercise for this unit. For more information about the Tivoli Enterprise Portal Admin Mode, see the *IBM Tivoli Monitoring Administrator's Guide*.

Importing application support

- Click **IBM Tivoli > Import Application Support Files**
- Enter Tivoli Enterprise Portal Server host name
- Select situations, Take Actions, and queries
- Workspaces are selected without prompt



Importing application support

The slide lists the process for importing custom application support into your agent in Agent Builder.

For situations, Take Actions, and queries, you are given a list from which you specify the items you want to import. Workspaces are identified and imported without prompts.

New items become visible in Agent Builder Navigator under your agent project where the imported application support definitions are stored.

Common problems with adding application support

- The Agent Builder cannot reestablish a connection with the Tivoli Enterprise Portal Server to extract workspaces
 - Avoid restarting the Tivoli Enterprise Portal Server while extracting situations, workspaces, and so on
 - Make sure that the Tivoli Enterprise Portal Server is running
 - Restart the Agent Builder
- Modifying company identifier, agent identifier, or product code invalidates any imported application support, and they must be re-created

Common problems with adding application support

This slide describes common problems and their solutions that users might experience when creating custom application support for Agent Builder agents.

Complete ITM agent creation process summary

1. Create the agent in Agent Builder
 - Agent information
 - Data source information
 - Runtime configuration
2. Test agent
 - Test in Agent Builder
 - Individual data source
 - Full agent
 - Install and confirm the agent in test environment
 - Revise and retest as needed
3. Add application support
 - Create in Tivoli Enterprise Portal, including queries, workspaces, situations, and Take Actions
 - Import application support into agent in Agent Builder
 - Retest the agent and application support
4. Create and install production agent
 - Output final script installers
 - Commit agent
 - Install agents and application support (Tivoli Enterprise Monitoring Server, Tivoli Enterprise Portal Server, and Tivoli Enterprise Portal Server desktop clients) on production servers and desktops

Complete ITM agent creation process summary

This slide summarizes the process that you use to create and deploy a final Agent Builder agent in an IBM Tivoli Monitoring environment.

Student exercises



IBM Tivoli Monitoring application Exercise 2: Create and import IBM Tivoli Monitoring application support
IBM Tivoli Monitoring application Exercise 3: Install and confirm IBM Tivoli Monitoring application support

Student exercise

Open your Course Exercises book and complete the exercises for this unit.

Summary

Now that you have completed this unit, you should be able to perform the following tasks:

- Manage an Agent Builder agent in an IBM Tivoli Monitoring environment
- Add custom IBM Tivoli Monitoring workspaces and situations to your agent

Unit 4 Monitoring Windows resources

IBM Training

IBM

Monitoring Windows resources

© Copyright IBM Corporation 2017
Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

In this unit, you expand your previous solution by adding several new Windows data sources. You are introduced to ways in which you can modify the target attributes your agent gathers. Last, you learn how various data sources organize the data they display and how you can modify that layout with Navigator groups.

Scenario business objectives

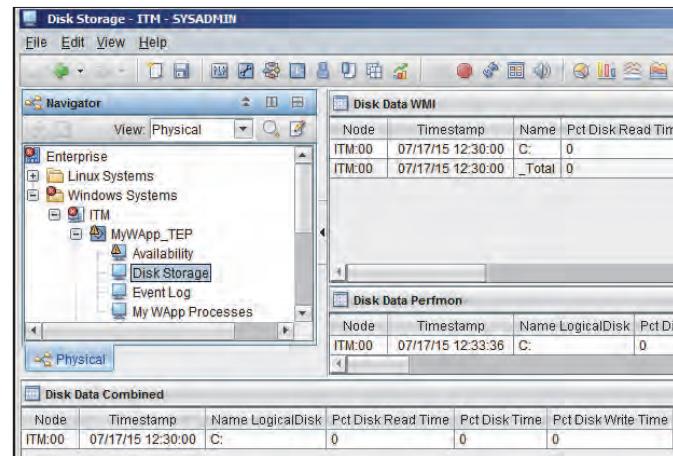
- A company has a web application that provides order processing and client management
 - The application runs on multiple HTTP servers
 - The application pulls data from a DB2 database
 - The installations are Windows only
- You must create an agent that, when installed on application servers, monitors key systems related to this application

Scenario business objectives

The training in this unit focuses on the knowledge and skills that are needed to meet this scenario.

Scenario solution

- *HTTP server and DB2 database Windows service availability monitoring (completed)*
 - Extend the AB1 agent
- Disk storage data gathered from Windows Perfmon and WMI
 - Custom Navigator group
 - Joined attribute group of both data sets
- HTTP server and DB2 database application events from Windows Event Log
- HTTP server and DB2 database process data in from WMI



Scenario solution

In a previous unit, you created the AB1 agent that monitors the availability of the HTTP server and DB2 Windows services in a simpler version of this scenario. The solution that you explore in this unit is to take that agent and expand its abilities to meet the new requirements.

Windows Performance Monitor (Perfmon) and Windows Management Instrumentation (WMI) are good sources for both logical and physical disk information. Normally, the default Tivoli Enterprise Portal Navigator display for each attribute group is its own Navigator item. Because the data is so closely related, you associate both these data sources with a single Navigator group, creating one Navigator item in Tivoli Enterprise Portal for both attribute groups.

Because the data from Perfmon and WMI describes different information about the same disk storage, you can combine the actual data into a new single attribute group.

The Windows Event Log is a source for HTTP server and DB2 event information.

Finally, Windows Management Instrumentation is a good source for process information.



Note: This scenario and the content in this unit focuses on Windows data source to limit the topics that are introduced in the unit. There are no limitations on the number or types of data sources that you can include in an agent.

Learning objectives

After completing this unit, you should be able to perform the following tasks:

- Add monitoring of Windows Management Instrumentation (WMI) data sources to an agent
- Add monitoring of Windows Performance Monitor (Perfmon) data sources to an agent
- Add monitoring of Windows Log Event data to an agent
- Edit data source attributes
- Create a Navigator Group that contains data from multiple sources
- Join two attribute groups into a combined attribute group
- Test a new data source attribute group and a full agent in Agent Builder

Learning objectives

The learning objectives that are shown in this slide include the skills that you need to implement the scenario solution.

Unit outline

- Lesson 1: Monitoring Windows Management Instrumentation
- Lesson 2: Monitoring Windows Performance Monitor
- Lesson 3: Monitoring Windows Log Events
- Lesson 4: Common Agent Modifications
- Lesson 5: Testing an attribute group and full agent
- **Exercise 1: Monitoring Windows resources**
- **Exercise 2: Install and confirm the updated AB1 agent**

Unit outline

This slide gives you an outline of the lesson material in this unit.

In this unit, you expand your previous solution by adding several new Windows data sources. You are introduced to ways in which you can modify the target attributes your agent gathers. Last, you learn how various data sources organize the data they show in the Tivoli Enterprise Portal Navigator and how you can modify that layout with Navigator groups.

Lesson 1 Monitoring Windows Management Instrumentation (WMI)

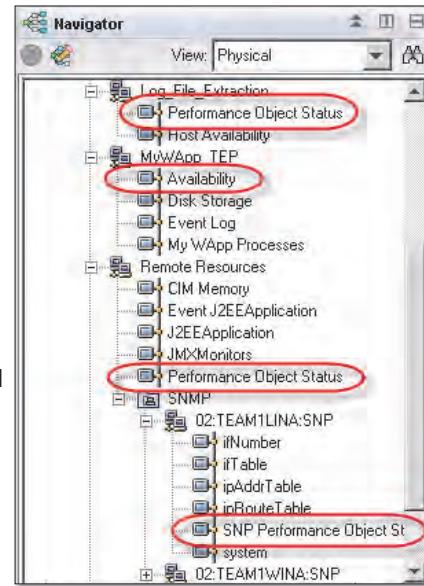
IBM Training

IBM

Lesson 1: Monitoring Windows Management Instrumentation (WMI)

Introduction to performance monitors

- Monitors: all data sources except availability data sources: processes, services, and command return codes
- Differences from availability
 - Each data source has own attribute group
 - You can add, edit, and remove attributes
 - Each attribute group has its own Navigator item in IBM Tivoli Monitoring or table in Application Performance Management details page
Unless you change
 - Performance Object Status attribute group and Navigator item in IBM Tivoli Monitoring but not in Application Performance Management



In this lesson, you learn how to gather data from Windows Management Instrumentation (WMI). After completing this lesson, you should be able to add monitoring of Windows Management Instrumentation (WMI) data sources to an agent.

Introduction to performance monitors

In the previous unit, you were introduced to the availability monitors category of data sources. In this unit, Windows Management Instrumentation introduces **performance monitors**, a new category of data sources. The performance monitors are WMI, Perfmon, CIM, SNMP, JMX, script, and log file.

Performance monitors differ from availability monitors in the following ways:

- Each data source has its own attribute group.
- You can add, edit, and remove attributes.
- Each data source gets its own data display. In Application Performance Management, each can have its own data widget in the Details dashboard. In IBM Tivoli Monitoring, each gets its own Navigator item in Tivoli Enterprise Portal. You can configure multiple data sources together under a single Navigator item.
- Performance Object Status is an attribute group that monitors the availability of each performance data source. It can be displayed in both Application Performance Management and IBM Tivoli Monitoring. In the Tivoli Enterprise Portal, if there is no availability monitor, you have a Performance Object Status Navigator item. If the agent has availability monitors, there is an Availability Navigator item, and the Performance Object Status data is displayed in the Availability Navigator item.

Introduction to Windows Management Instrumentation (WMI)

- Microsoft implementation of Web-Based Enterprise Management (WBEM)
- Software can monitor and control managed Windows resources throughout the network
Examples: hard disks, file systems, settings of operating system, processes, services, shares, registry settings, networking components, event logs, users, and groups
- Implemented or can be implemented in most Windows operating systems
Note: Monitoring the Common Information Model (CIM) data source is similar to WMI and is covered in Unit 6.

Introduction to Windows Management Instrumentation (WMI)

WMI is the acronym for Windows Management Instrumentation. WMI is the Microsoft implementation of Web-Based Enterprise Management (WBEM), a management technology that enables software to monitor and control managed resources throughout the network. Such managed resources include hard disks, file systems, settings of operating systems, processes, services, shares, registry settings, networking components, event logs, users, and groups.

WMI is built into Windows 2000 and higher. You can install it on any other 32-bit Windows computer.

WMI terminology

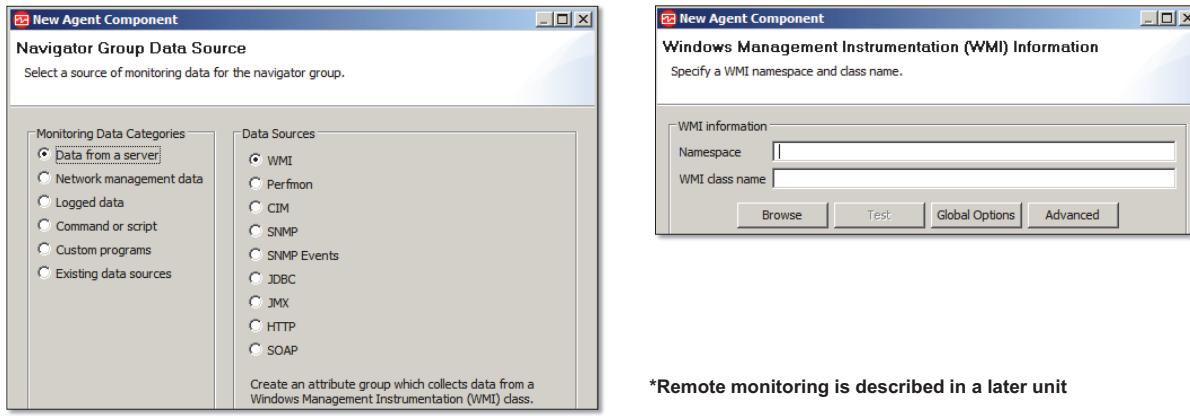
- Namespaces
 - Information about resources used to monitor and control
 - Container grouping related child objects (class)
 - Contains a hierarchy of classes and associations that define either a machine's object or the relationship between two or more objects
 - Most common namespace is CIMV2
- Translating WMI terminology to IBM agent terminology
 - Namespace + Class = attribute group
 - Property = agent attribute

WMI terminology

A WMI namespace is a container that groups child objects that relate to one another in some way. A namespace contains a hierarchy of classes and associations that define either a computer's object or the relationship between two or more objects. The namespace most commonly in use is CIMV2 (Common Information Model Version 2).

Adding WMI monitoring to an agent

- Identify WMI as the data source
- Specify the namespace and class to be monitored; all properties initially imported
- Manually enter the information or use the browser
- Global Options enables runtime configuration for remote monitoring*



*Remote monitoring is described in a later unit

Adding WMI monitoring to an agent

As with any data source, you can start the agent wizard in several ways, but you must always identify the correct data source from the Initial Agent Data Source window. The namespace and WMI class name are the properties that the agent needs to determine what to monitor. As stated earlier, the namespace identifies the container that holds the information that you want, and the class name defines the data that you want to gather.

You can manually enter the namespace and WMI class name, or you can use the WMI browser.



Important: If you manually enter the WMI information, you must manually define each attribute to retrieve. When you use the browser to identify a WMI data source, it automatically defines the attribute group and attributes it needs based on information that WMI provides. All of the browsers under *Data from a server* function this way.

Also, with the WMI browser you can browse local or remote hosts for the namespace, class name, and properties you want to monitor.



Note: Browsing a remote host does not determine which host that the agent monitors. Browsing results only in identifying the information to gather. Where an agent is installed and how it is configured determines what host it monitors.

In most cases, an agent monitors data sources on the host where the agent is installed. It is possible to create an agent that, when installed on one host, pulls WMI data from a remote host. You use Global Options to enable this ability. Remote monitoring is covered in [Unit 7, “Monitoring remote and optional resources”](#).

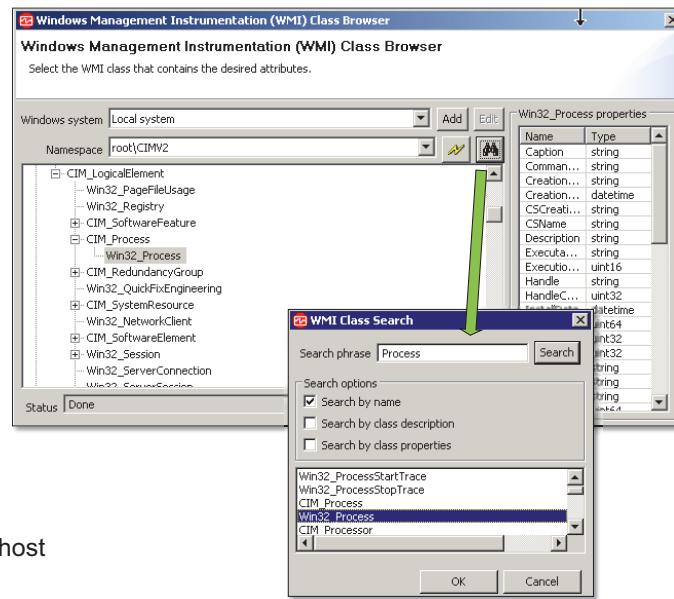


Note: To collect metrics through WMI, a Windows operating system must host the agent, even if your agent is gathering WMI data from a remote Windows host. If you are monitoring a remote host, you must enable remote registry administration on the remote computer.

Browsing WMI

- Standard actions
 - Local or remote browsing
 - Refresh
 - Find
 - Select Namespace
 - Select one class and examine the properties that are shown on right
 - Select **properties** (optional)
- Remote browsing
 - Add host name
 - Login required
 - Lists namespaces on remote system

Note: Remote browsing does not identify what host is monitored by the agent



Browsing WMI

With the Windows Management Instrumentation browser, you can locate and specify the class of data you want the agent to monitor.

You can complete many standard actions in the WMI browser, such as:

- Local or remote browsing
- Refreshing the view
- Locating data in the namespace with the Find feature
- Specifying the data that you want to monitor

You can select only one namespace and class. Many classes have properties that are classes themselves. If you select a class that has child classes, the browser creates more attribute groups for each child class.

You can specify the properties that you want to monitor for the class you select. If you do not specify properties, the agent gathers all of them.

You can open any local or remote WMI namespace. Remote WMI browsing is through a standard Windows remote connection and requires the remote host name and a login.

Lesson 2 Monitoring Windows Performance Monitor

IBM Training



Lesson 2: Monitoring Windows Performance Monitor

Introduction to Windows Performance Monitor (Perfmon)

- Perfmon provided in most versions of Windows
- Provides performance and usage data on system resources, such as system, processor, memory, disk drives, network cards, and print queues
- Translating Perfmon terminology to IBM agent terminology
 - Object = attribute group
 - Instance = one of multiple-device objects = attribute group
 - Counter = attribute

This lesson introduces you to gathering data from Windows Performance Monitor (Perfmon). After completing this lesson, you should be able to add monitoring of Windows Perfmon data sources to an agent.

Introduction to Windows Performance Monitor (Perfmon)

The Windows Performance Monitor provides performance and usage data on system resources, such as computer, processor, memory, disk drives, network cards, and print queues.

The Windows Perfmon utility groups performance metrics by performance objects. Each performance object contains a number of counters, each of which holds a performance metric.

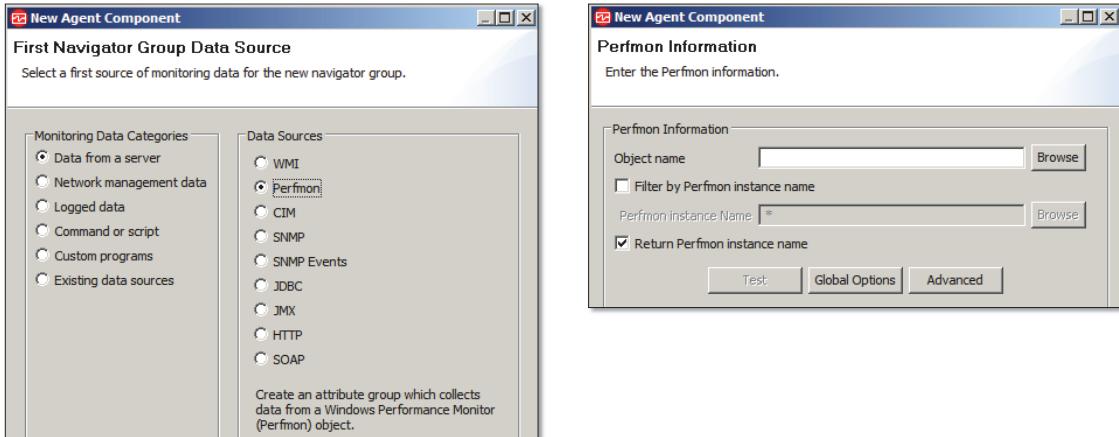
If a performance object can report on multiple devices or entities on the host computer, each of its counters has more than one instance, one for each device or entity.

For example, on a host computer with more than one physical disk, the counters that provide metrics for physical disk performance have multiple instances, one for each disk.

The slide shows how each of these terms relates to the IBM agent terms.

Adding Perfmon monitoring to an agent

- Identify Perfmon as the data source
- Enter Perfmon object name manually or use browser



Note: Instance options are described soon. Global Options is described in Unit 6.

Adding Perfmon monitoring to an agent

To gather data from Perfmon, the only property the agent requires is the Perfmon object name. You can manually enter the object name or use a browser to locate and select it.

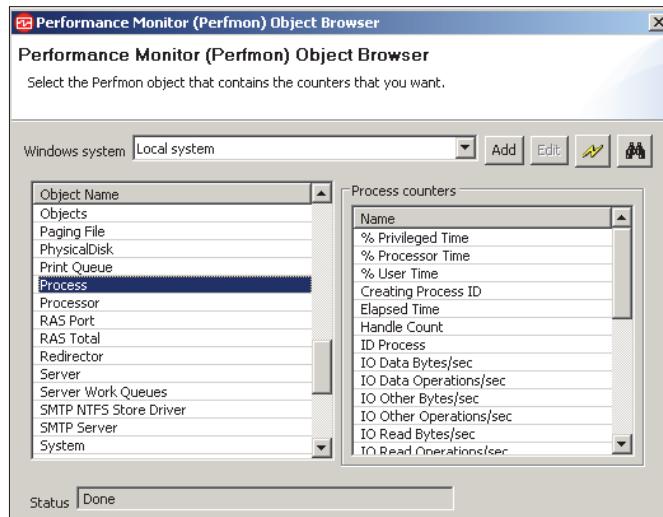
As with WMI because the objects in Perfmon are well defined, the agent can automatically define an attribute group and attributes to hold the data it monitors. No additional definition work is needed, but you can modify the default information after you create it in Agent Builder.



Note: The instance options are described in [“Monitoring object instances”](#) on page 4-17. Use **Global Options**, as described in the previous lesson, to enable remote monitoring. It is covered in [“Remote monitoring and agent runtime configuration”](#) on page 7-16.

Browsing Perfmon objects

- Standard actions
 - Local or remote browsing
 - Refresh
 - Find
 - Select one object, counter definitions are shown on right
- Remote browsing
 - Add host name
 - Login required
 - Lists Perfmon objects on remote system



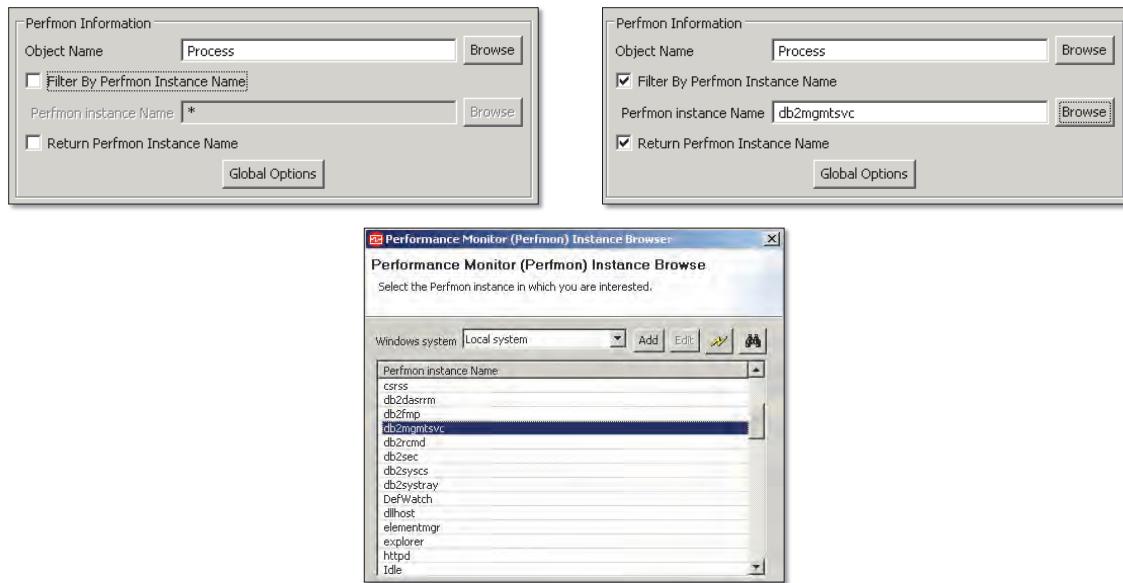
Browsing Perfmon objects

The Perfmon browser works similarly to the WMI browser, with the exception that there is only one Perfmon repository, unlike the WMI namespaces. You can do the following actions:

- Browse the local or remote Perfmon repository
- Refresh the screen
- Search for object names with the Find feature select the object name and attributes you want the agent to gather

You do remote Perfmon browsing through a standard Windows remote connection. You must have the remote host name and a login.

Monitoring object instances



Monitoring object instances

Some Perfmon objects can report on multiple devices or entities on the host computer that results in multiple instances of the object. If your object has multiple instances, you can use the instance properties that are shown here.

You can use the first property, *Filter By Perfmon Instance Name*, to override the default behavior of returning data on all instances and limiting the data to one or more instances. Enter a specific instance name or a pattern to match for multiple instances. An instance browser is available to facilitate this process.

You can also configure the agent to return the instance name.

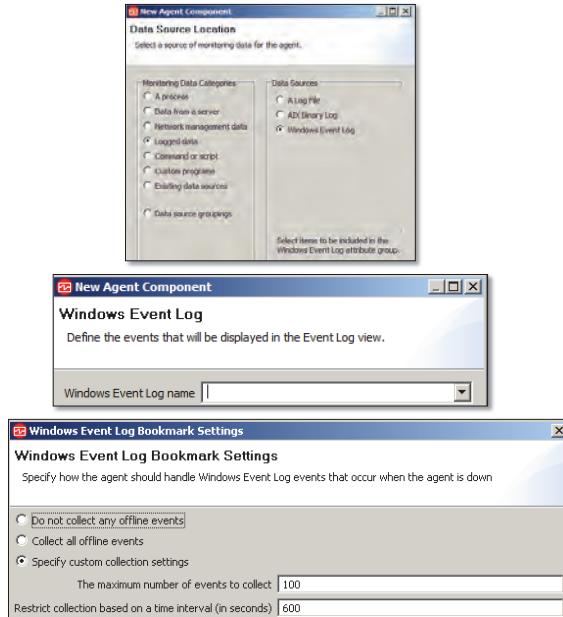
Lesson 3 Monitoring Windows log events

IBM Training

IBM

Lesson 3: Monitoring Windows log events

- Identify Windows event log as the data source
- Enter Windows event log name manually or use browser
 - Application
 - HardwareEvents
 - Internet Explorer
 - Key Management Service
 - Security
 - System
 - ThinkPrint Diagnostics
 - Windows PowerShell
- Offline event handling
 - On restarting, agents can now process Windows Event Log events that were generated while the agent was shut down



Monitoring Windows resources

15

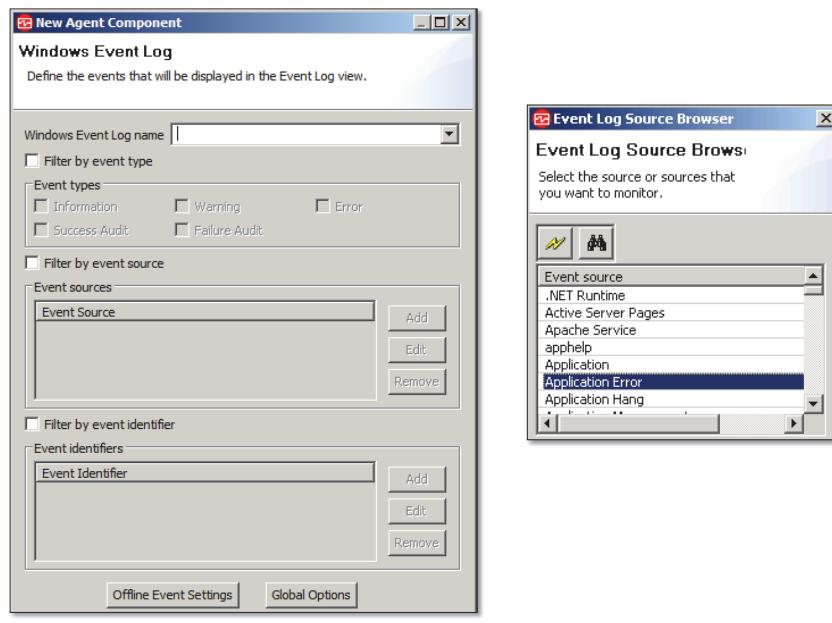
© Copyright IBM Corporation 2017

This lesson introduces you to gathering data from Windows Event Logs. You can configure an agent to gather event data from the application, security, and system logs. The only property the agent requires is the Windows event log name. You can manually enter the log name or use a browser to select it.

After completing this lesson, you should be able to add monitoring of Windows Log Event data to an agent.

Filtering events

- Filter by event type
 - One or more
- Filter by event source
 - One or more
 - Browser to list
 - Browser can add multiples at one time
- Event identifier
 - One or more
 - Enter event ID number



Monitoring Windows resources

16

© Copyright IBM Corporation 2017

Filtering events

You must identify which events to gather by filtering for specific event source names or the event IDs. You can add one or more filters for source names and IDs.

The default agent behavior is to return only new events after the agent starts. To configure the agent to send every event in the Windows event log, set the following environment variable to YES:

`CDP_NT_EVENT_LOG_GET_ALL_ENTRIES_FIRST_TIME`

In Windows, you set this variable in the **KXXENV** file. On UNIX and Linux systems, you can set these variables in the agent's **\$CANDLEHOME/config/XX.ini** file. For more information, see Chapter 35, "Troubleshooting," in the *IBM Agent Builder User's Guide*.

Lesson 4 Common agent modifications

IBM Training



Lesson 4: Common agent modifications

- Selecting single data row or key attributes
- Modifying base attribute groups and attributes
- Creating navigator groups
- Joining attribute groups
- Enabling self-describing
- Enabling agent watchdog
- Committing an agent

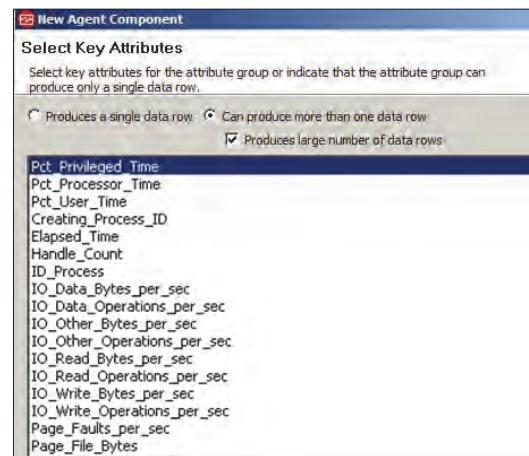
This lesson introduces some common modifications that you might make to an agent.

After completing this lesson, you should be able to do the following tasks:

- Edit data source attributes
- Create a Navigator group that contains data from multiple sources
- Join two attribute groups into a combined attribute group

Selecting single data row or key attributes

- Indicate whether will return single or multiple rows of data
 - Set automatically for some data sources
- Multiple rows of data require one or more key attributes to uniquely identify each row
 - Critical for historical data aggregation
- Example
 - Perfmon PhysicalDisk returns data on all disk drives
 - Perfmon PhysicalDisk filtered for the C: instance returns only one row of data



Selecting single data row or key attributes

For some data sources, you must specify whether they return a single row of data that is overwritten each time, or whether they return multiple instances of data. If the data source does return multiple rows, you must select one or more attributes as keys that uniquely identify each row of data.



Note: For some data sources, Agent Builder configures these properties automatically.

Completing these steps ensures that the summarization and pruning agent properly aggregates the attribute group data.

Correct classification also ensures that the Agent Builder correctly calculates rates and deltas. Most of the attribute types display raw values for the current sample set: counter, gauge, display string, enumeration, and time stamp. But some of the attribute types use the current value and previous value to calculate the number that is shown: delta, percent change, rate of change. For these calculations, key attributes are used to match the row from the previous sample set to the row in the current sample set. If the key attributes are incorrectly marked or are incomplete and do not identify a unique row, these calculations might not be done correctly.

Modifying data sources after the wizard

Data source	Modifiable Attributes	
Availability Windows Services Process Command return code Windows Log File AIX Log	No, set by agent	
WMI Perfmon SNMP CIM JMX	JDBC HTTP SOAP	Yes. Initially set by data source, but some attribute properties can be modified
Log Files Scripts Java API Socket	Yes. Undefined, must be manually defined	

Modifying data sources after the wizard

After using the Agent wizard, you can continue to modify the agent data sources. This table breaks the different data sources into three categories that are based on their editing similarities. Each grouping is described in the following slides.

Attributes that cannot be modified

For event logs and availability monitoring, Windows services, processes, and command return codes, you are limited in the number of modifications you can make.

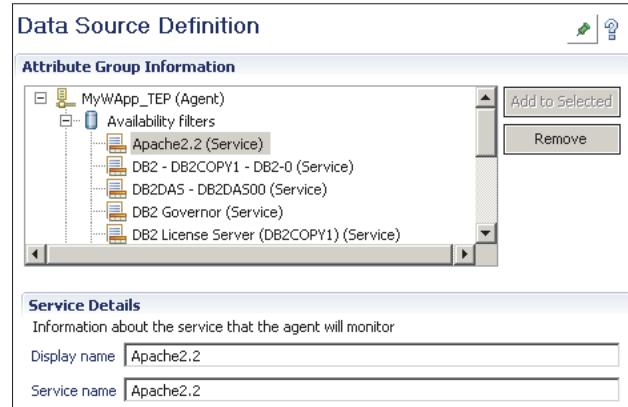
Attributes that cannot be modified: Availability data

- All filters return the same data to the Availability node

Node	Timestamp	Application Component	Name	Status	Full Name	Type
Virtual Size	Page Faults per Sec	Working Set Size	Thread Count	PID	Percent Privileged Time	Percent User Mode Time
Percent Processor Time	Command Line	Functionality Test Status	Functionality Test Message			

Cannot manipulate the attribute group, attributes, or Navigator item

- Can add and remove filters (targets) and change display name

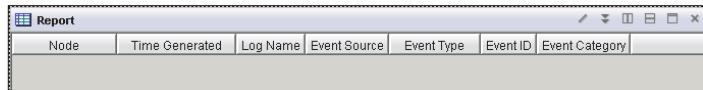


Availability

All availability filters return the same data, which is shown in this slide, to the Availability node. You cannot change the data that returns or the Navigator item with which the data is associated. You can add and remove filters, services, processes, and commands, and you can modify the display name of the filter.

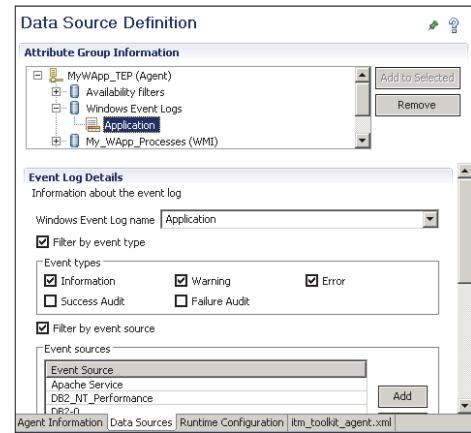
Attributes that cannot be modified: Windows event logs

- All logs return the same data to the Event Log node



Cannot manipulate the attribute group, attributes, or Navigator item

- Can add and remove event logs
- Can modify the event filters

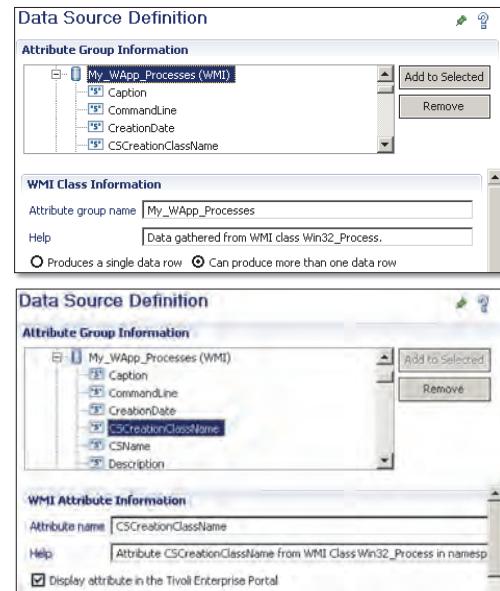


Windows event logs

All Windows event logs return the same data, which is shown in this slide, to the Event Log node. You cannot change the data that returns or the Navigator item with which the data is associated. You can add and remove event logs, and you can modify event filters for a specific log.

Attributes that can be partially modified

- For data from a server
 - The attribute group and attributes are automatically defined by the data source
 - Each attribute group has its own Navigator item
- You can do the following tasks:
 - Add (data source), remove, edit, and combine attribute groups
 - Remove, edit, don't display, and create derived attributes
 - Associate two or more attribute groups to a single Navigator item



Attributes that can be partially modified

The WMI, Perfmon, SNMP, CIM, and JMX attribute groups and attributes are automatically defined because data source browsers pull the metadata from the sample. But, unlike the Availability and Windows Event Log data sources, you can modify the attribute group and attribute information for these data sources. You can add, remove, edit, and combine the attribute groups. Furthermore, you can complete the following actions for the attributes of these data sources:

- Remove them
- Edit them
- Prevent them from showing in the Tivoli Enterprise Portal or Performance Management console
- Create derived attributes for them

Attributes that can be fully modified

- For log files, script, Java API, and socket
 - The attribute group and attributes can be automatically defined by the Agent Builder Test feature, but customization is almost always required
 - Each attribute group has its own Navigator item
- You can completely redefine the attribute group and attribute definitions

Note: Monitoring log files and scripts is described in Unit 6.

Attributes that can be fully modified

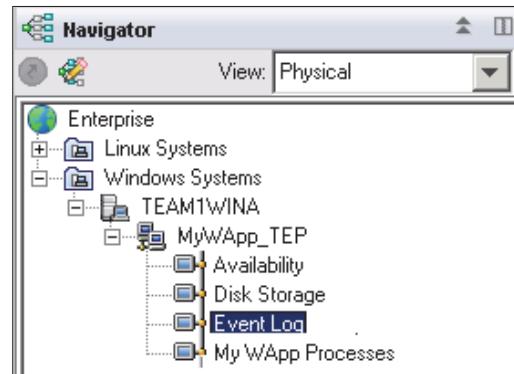
Unlike the other data sources, Agent wizard cannot easily determine the attribute group and attribute properties for monitoring log files and scripts. Although a test feature exists that can facilitate the process of defining the attribute, some customization is almost always needed. For this same reason, the amount of modification you can do after the Agent wizard is greater too.



Note: [Unit 6, “Monitoring custom data sources”](#) describes monitoring log files and scripts.

Creating navigator groups

- Defaults
 - All availability data sources appear under Availability Navigator item
 - Windows Event Log has own Navigator item
 - Each performance attribute group appears as own Navigator item
- New Navigator group
 - Create Navigator group to populate with multiple performance data sources
 - Can move existing data sources into Navigator group
 - No data shown in initial workspace; you must manually create valid views
 - Has no function in IBM Application Performance Management



Creating navigator groups

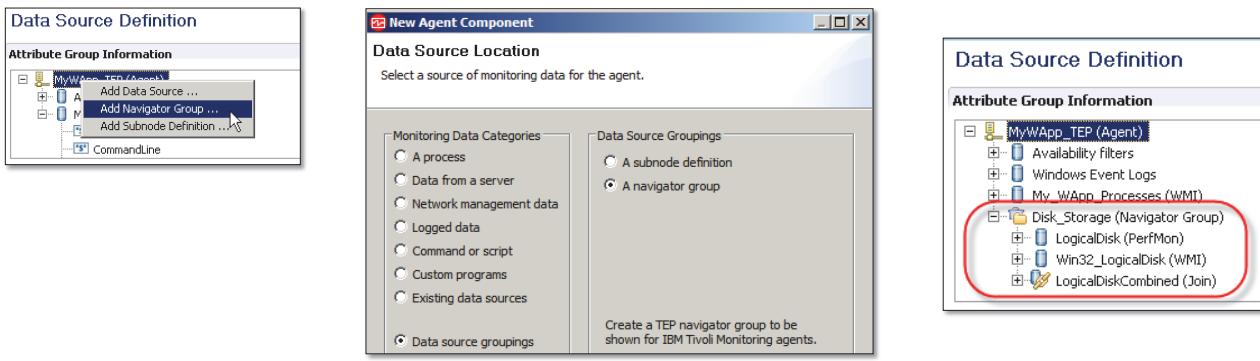
Each data source has a default Navigator item with which the monitored data is associated.

With the Navigator group feature in Agent Builder, you can associate multiple attribute groups with one Navigator item in the Tivoli Enterprise Portal. You can define the Navigator group and then the attribute groups that are associated with it during one Agent wizard session. You can also move existing attribute groups into a new Navigator item.

No data appears in the initial workspace for a Navigator group. Although each attribute group is associated with the item, you must manually create a workspace and views to show the data.

Steps for creating navigator groups

1. Select to add
2. Give name and help
3. Add first data source
4. Add other data sources



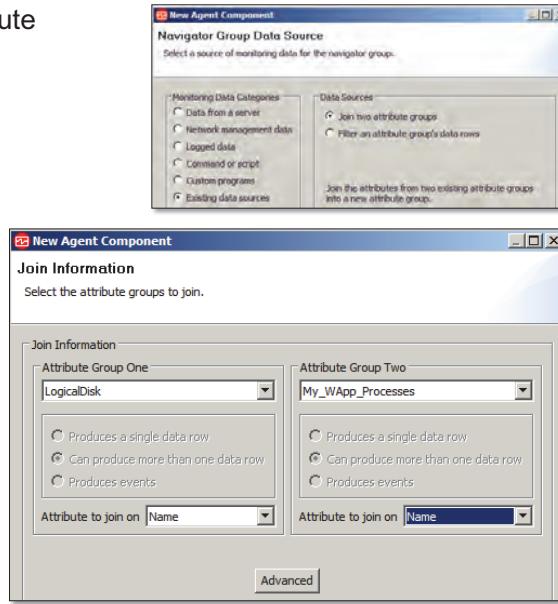
Steps for creating navigator groups

The process of creating a Navigator group is simple:

1. Indicate that you want to create a Navigator group.
2. Name the Navigator group and enter Help information.
3. Add at least one data source in the normal way.
4. Add more data sources in the normal way or by dragging and dropping the data sources in the **Data Sources** tab.

Joining attribute groups

- Create an attribute group from two or more attribute groups
- All source attributes must have unique display names
 - Rename as needed
- Process
 - Single rows: Identify attribute groups to join
 - Multiple row
 - Identify attribute groups attributes to join
 - Select attribute to join on
- In AB, no attributes shown under new, combined data source
 - Edit source attributes



Monitoring Windows resources

26

© Copyright IBM Corporation 2017

Joining attribute groups

When you monitor multiple data sources, you can join two data sources to create a third data source that contains attributes from the original two data sources. Joining is most useful when the agent collects data from two different types of data sources and their attributes are more useful when you use them together.

For example, both WMI and Perfmon contain different information about the same disk storage. By combining the two data sources, you have a more useful data source.



Note: The option for combining data sources in the Agent wizard is shown only after you create one or more data sources for the agent.

To join two data sources, you must ensure that attribute names are unique for both base data sources. If you want to join two data sources that both have attributes with the same name, rename one of the attributes.

For single row joins, identify the two attribute groups. If both base data sources are defined to return more than one row, you must identify an attribute from each of the base data sources that you join.

When you join two data sources, you create a third data source. In Agent Builder, no attributes are shown under this new data source, but the data source contains all of the attributes that the base data sources contain.

The results of a join operation vary, depending on the number of rows that each of the base data sources supports.

Join results: Single-to-single join

Results: one row with all attributes

Attribute 1	Attribute 2	Attribute 3
16	some text	35

Attribute Group 1 (single row)

Attribute 4	Attribute 5	Attribute 6	Attribute 7
5001	more data	56	35

Attribute Group 2 (single row)

Attribute1	Attribute2	Attribute3	Attribute4	Attribute5	Attribute6	Attribute7
16	some text	35	5001	more data	56	35

Resulting Attribute Group (single row)

Join results: Single-to-single join

If you define both data sources to return only a single row of data, then the resulting join data source has one row of data that contains all of the attributes from both base data sources.

Join results: Single-to-multiple-row join

Results: single row added to each row in multiple-row

Attribute 1	Attribute 2	Attribute 3
16	some text	35

Attribute Group 1 (single row)

Attribute4	Attribute5	Attribute6	Attribute7
user1	path1	56	35
user2	path2	27	54
user3	path3	44	32

Attribute Group 2 (multiple-row row)

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

Resulting Attribute Group (multiple-row)

Join results: Single-to-multiple-row join

If you join a single row data source to a multiple-row data source the resulting data source contains the same number of rows as the multiple-row data source. The data from the single-row data source is appended to each row of the multiple-row data source.

Join results: Multiple-row-to-multiple-row join

Results: matching row added to matching row

Attribute 1	Attribute 2	Attribute 3
16	some text	35
27	some text	54
39	some text	66

Attribute4	Attribute5	Attribute6	Attribute7
user1	path1	56	35
user2	path2	27	54
user3	path3	44	32

Attribute Group 2 (multiple row)

Attribute Group 1 (multiple row)

Attribute1	Attribute2	Attribute3	Attribute4	Attribute5	Attribute6	Attribute7
16	some text	35	user1	path1	56	35
27	some text	54	user2	path2	27	54

Resulting Attribute Group (multiple rows joined on Attribute3 and Attribute7)

Join results: Multiple-row-to-multiple-row join

Finally, if you define both base data sources to return more than one row, you must identify an attribute from each of the base data sources on which to complete the join. The resulting data source contains rows of data where the value for the attribute in the first data source matches the value of the attribute from the second data source.

Enabling self-describing agent

- Agent Builder agents can be self-describing (default)
 - Agent bundles application support
 - Agent automatically seeds the agent's support files to the management server: APM server, Tivoli Enterprise Monitoring Server, and Tivoli Enterprise Portal Server
 - Required for IBM Application Performance Management
 - Requires IBM Tivoli Monitoring version 6.2.3+ installed and self-description enabled in IBM Tivoli Monitoring (default)
- For integration with IBM Tivoli Monitoring, see the IBM Tivoli Monitoring Installation and Setup Guide and the IBM Tivoli Monitoring Administrator's Guide

Agent Content

The advanced information for the agent can be accessed by clicking the links below or by opening the [Outline View](#).

- [Default Operating Systems](#): lists the default operating systems selected for this agent.
- [Self-Describing Agent](#): lists the settings for bundling support files with the agent.
- [Environment Variables](#): lists the environment variables defined in this agent.

Self Describing Agent

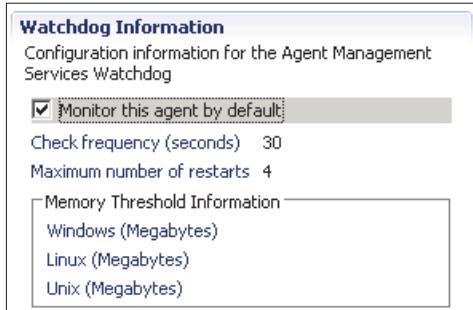
Self Describing Agent

The self describing agent capability allows you to create agent support file bundles for your agent. These support file bundles are used to automatically seed the IBM Tivoli Monitoring Servers.

Enable self-description for this agent

Enabling agent watchdog

- Agent Management Services Watchdog manages the life cycle of the Agent Builder agents like IBM Tivoli Monitoring manages the base agents
 - Automatic restart if they fail
 - Track their status in a common workspace
- Configure the Watchdog on the Agent Information tab of Agent Editor
Enabled by default
- Agent Management Services is an IBM Tivoli Monitoring 6.2.2 service available to most agents

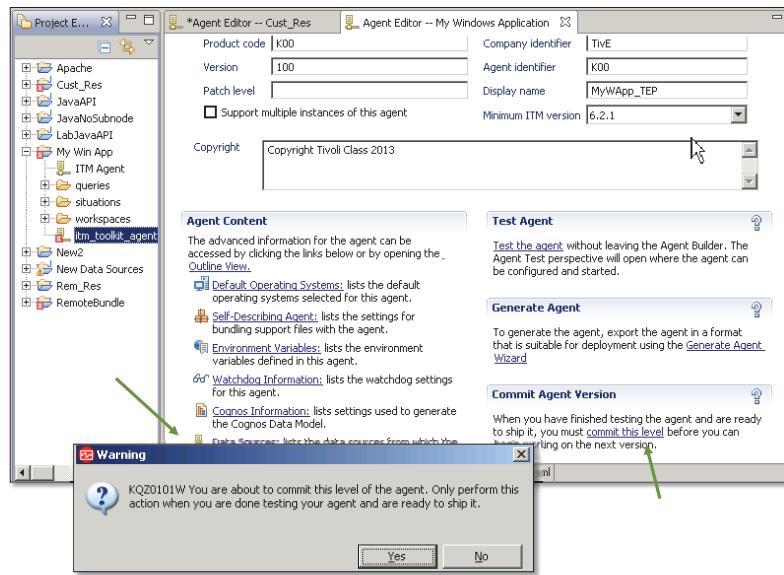


Committing an agent

- Optional step
- IBM Tivoli Monitoring requires updated agents support all previous agent versions
- Committing ensures critical components cannot be modified in future releases (see Student Guide for list)
- Performed when agent is completed and ready to go into production
- After committing, a new version number must be entered
- Limit of 1024 versions

Commit process

- Save final version
- Click the commit this level
- When you create next version
 - Enter a new version number
 - Prompted on next save (allows you to generate outputs without changing version number)
 - Patch level now available



Monitoring Windows resources

33

© Copyright IBM Corporation 2017

Commit process

Lesson 5 Testing an attribute group and full agent

IBM Training

IBM

Lesson 5: Testing an attribute group and full agent

- Types
 - Individual attribute group
 - Full agent
 - Install and test
- Individual attribute group
 - Click **Test in Wizard or Data sources** tab
 - Select local or remote target
 - Define environment variable
 - Configure agent properties
 - Start agent
 - Collect data
- In attribute group testers, you can create, delete, and modify attributes
 - **OK** saves changes; **Cancel** does not save
 - Environment variables and configuration property values are not saved

The screenshot shows the WMI Test application window. At the top, it says "WMI Test" and "103 data rows returned at Apr 14, 2017 4:49:27 PM". Below that is a toolbar with buttons for "Start Agent", "Collect Data", "Stop Agent", "Check Results", "Set Environment", and "Configuration". The main area is titled "Results" and contains a table with the following data:

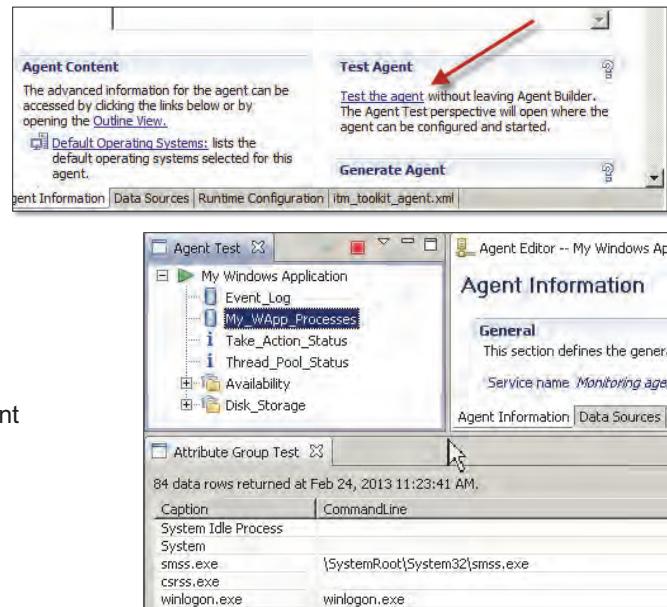
Description	ExecutablePath	ExecutionState	Handle	HandleCount	InstallDate	Kernel
System Idle Process		0	0	0	3564	
System		0	4	764	3483	
smss.exe	C:\Windows\system32\smss.exe	0	256	32	2106	
csrss.exe	C:\Windows\system32\csrss.exe	0	348	1717	3463	
csrss.exe	C:\Windows\system32\csrss.exe	0	416	410	5584	
wininit.exe	C:\Windows\system32\wininit.exe	0	424	86	5772	
services.exe	C:\Windows\system32\services.exe	0	476	325	5289	
winlogon.exe	C:\Windows\system32\winlogon.exe	0	500	110	5304	
lsass.exe	C:\Windows\system32\lsass.exe	0	528	778	4165	
lsm.exe	C:\Windows\system32\lsm.exe	0	540	204	2964	

You can use attribute group testing to test the attribute groups of the agent you create with Agent Builder, one attribute group at a time. In this lesson, you learn to test a new data source attribute group in Agent Builder.

You can use attribute group testing to test the attribute groups of the agent that you create 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 start testing from the Agent wizard when you define the attribute groups of a new agent. You can also start testing from the Agent Component wizard when you add attribute groups to an existing agent. IBM Agent Builder 6.3.4

Testing the full agent

- You can move between editing and testing the full agent
- Process
 - Click **Test the agent** on the Agent Information tab
 - Set environment variables and agent configuration as needed
 - Start the agent and test each attribute group to test its data
 - Edit agent as needed
 - Return to full agent tester, stop, and restart agent
 - Confirm modifications



Monitoring Windows resources

35

© Copyright IBM Corporation 2017

Testing the full agent

Use full agent testing to test all attribute groups of your agent together and to move easily between editing and testing the agent. You can also use full agent testing to test data sources that cannot be tested by using the attribute group test function.

You can use full agent testing to run the agent in the same way it runs in when you deploy it into a monitoring environment.

Student exercises



Exercise 1: Monitoring Windows resources

Exercise 2: Install and confirm the updated AB1 agent

Student exercises

Open your Student Exercises book and complete the exercises for this unit.

Summary

Now that you have completed this unit, you should be able to perform the following tasks:

- Add monitoring of Windows Management Instrumentation (WMI) data sources to an agent
- Add monitoring of Windows Performance Monitor (Perfmon) data sources to an agent
- Add monitoring of Windows Log Event data to an agent
- Edit data source attributes
- Create a Navigator Group that contains data from multiple sources
- Join two attribute groups into a combined attribute group
- Test a new data source attribute group and full agent in Agent Builder

Unit 5 Monitoring processes and command return codes

IBM Training



Monitoring processes and command return codes

© Copyright IBM Corporation 2017
Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

In this unit, you learn about availability monitoring of processes and command return codes. Unlike the Windows-only data sources, these data sources are run on multiple operating systems.

Scenario business objectives

- Company has a web application that provides order processing and client management
 - Application runs on multiple HTTP servers
 - Application pulls data from DB2 database
 - Windows and Linux installations
- Create a single agent that can be installed on both the Windows and Linux application servers and monitors the availability of the HTTP server processes

Scenario business objective

The training in this unit focuses on the knowledge and skills that you need to meet this scenario.

Scenario solution

Data Source Definition

Attribute Group Information

- AB2 (Agent)
 - Availability filters
 - HTTP_Win (Process)
 - HTTP_Lin (Process)
 - HTTP PID File (Command Return Code)

- AB2 agent on Windows and Linux hosts
- Availability monitoring of the HTTP server processes (Process data source)
- Availability monitoring of HTTP server by running a command that checks for the existence of an HTTP server PID file (Command Return Code data source)

Report						
Node	Timestamp	Application Component	Name	Status	Full Name	Type
lin4:01	07/11/15 20:22:18	HTTP PID File	ls /var/run/httpd2.pid	PASSED	N/A	FUNCTIONALITY TEST
lin4:01	07/11/15 20:22:18	HTTP_Lin	httpd2-prefork	UP	/usr/sbin/httpd2-prefork	PROCESS
lin4:01	07/11/15 20:22:18	HTTP_Lin	httpd2-prefork	UP	/usr/sbin/httpd2-prefork	PROCESS
lin4:01	07/11/15 20:22:18	HTTP_Lin	httpd2-prefork	UP	/usr/sbin/httpd2-prefork	PROCESS
lin4:01	07/11/15 20:22:18	HTTP_Lin	httpd2-prefork	UP	/usr/sbin/httpd2-prefork	PROCESS
lin4:01	07/11/15 20:22:18	HTTP_Lin	httpd2-prefork	UP	/usr/sbin/httpd2-prefork	PROCESS
lin4:01	07/11/15 20:22:18	HTTP_Lin	httpd2-prefork	UP	/usr/sbin/httpd2-prefork	PROCESS

Scenario solution

The scenario solution is a single, multiplatform agent that you can install on both Windows and Linux. The agent monitors related, but different items on each system.

You add availability monitoring of the HTTP server process so that when you install the agent on Windows, it monitors the Windows HTTP server process. But when you install the agent on Linux, it monitors the Linux HTTP server process.

Furthermore, the HTTP server produces a process ID (PID) file whenever it is running. Using availability monitoring of a command return code, you can create platform-specific commands that confirm the existence of this file.



Important: For the remainder of the class, deploy all agents in the IBM Tivoli Monitoring environment and not the IBM Application Performance Management environment. Nevertheless, all information and skills that you learn in the remainder of the course apply to agents that you create for both environments.

Learning objectives

After completing this unit, you should be able to perform the following tasks:

- Add monitoring of multiplatform processes to an agent
- Add monitoring of multiplatform command return code to an agent

Learning objectives

The learning objectives that this slide shows include the skills that you need to implement the scenario solution.

Unit outline

- Lesson 1: Monitoring processes
- Lesson 2: Monitoring command return codes
- **Exercise 1: Monitoring processes and command return codes**
- **Exercise 2: Install and confirm the AB2 agent**

Unit outline

As do each of the later units in this course, this unit introduces new data sources. You learn about availability monitoring of processes and command return codes. Unlike the Windows-only data sources that the previous unit introduces, these data sources are run on multiple operating systems.

Lesson 1 Monitoring processes

IBM Training

IBM

Lesson 1: Monitoring processes

Availability Monitoring Review

- Types
 - Windows Services
 - Processes
 - Command Return Codes
- All data returned to Availability node for all types
- Cannot manipulate the attributes returned
- Some attributes are not valid on some types

Availability		
Application Component	Name	Status
Apache2.4	Apache2.4	UP
DB2 - DB2COPY1 - DB2	DB2	UP
DB2DAS - DB2DAS00	DB2DAS00	UP
DB2 Governor (DB2COPY1)	DB2GOVERNOR_DB2COPY1	DOWN
DB2 License Server (DB2COPY1)	DB2LICD_DB2COPY1	DOWN

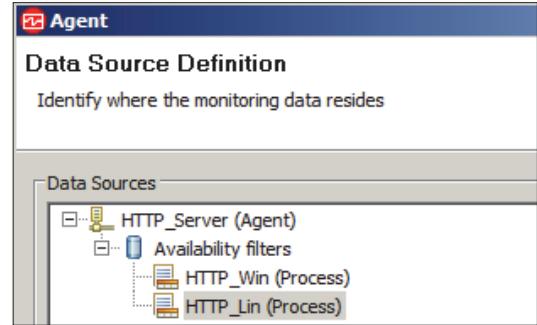
Node	Timestamp	Application Component	Name	Status	Full Name	Type
		Virtual Size	Page Faults per Sec	Working Set Size	Thread Count	PID
		Percent Processor Time	Command Line	Functionality Test Status	Functionality Test Message	

All availability filters, Windows services, process, and command return codes on an agent return their data to a single availability attribute group and Navigator item. This behavior is unlike performance monitors in which each data source has its own attribute group and Navigator item. Although you can change which filters to monitor, you cannot specify what data to return. Because some availability filter types return different data, not all columns in the data table apply to all filter types.

For example, all availability filters return application component, name, and status information, but only the command return code availability filter returns functionality test status and functionality test message attributes.

Process monitoring

- Can monitor one or more processes
 - Add each individually
- If agent is for multiple operating systems, one process name for each operating system
- Supported operating systems
 - Windows
 - Linux
 - UNIX

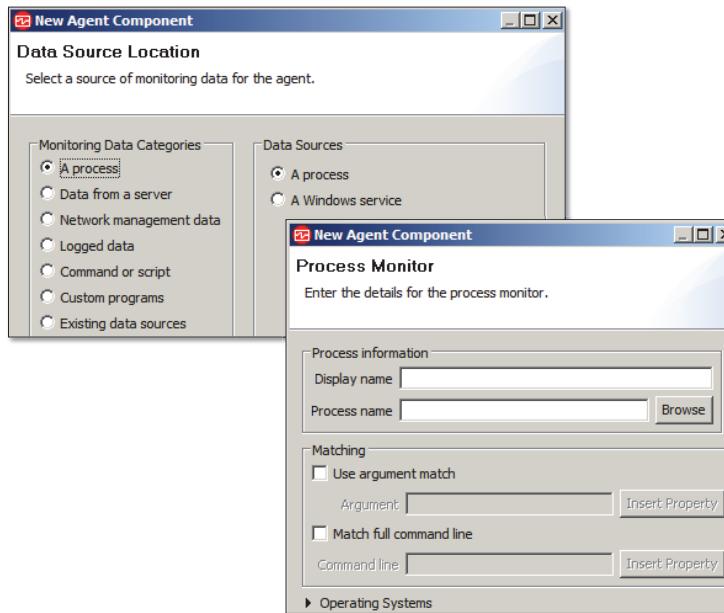


Process monitoring

You can configure an agent to monitor one or more processes. The agent can run on multiple operating systems so that you can identify a different process for each operating system. An Agent Builder agent can monitor processes on a wide range of Windows, Linux, and UNIX versions.

Adding a process

- Display Name
 - Shown in Tivoli Enterprise Portal
 - Enter manually or accept browser default
 - Browser multiple selection always uses default
- Process name
 - Monitored process
 - Manually must match exactly
 - Can browse for one or more
- Operating Systems
 - Browser automatically selects based on processes selected
- Matching
 - Refines Process Name



Monitoring processes and command return codes

8

© Copyright IBM Corporation 2017

Adding a process

You add processes to the agent one at a time. The required properties are Display name, Process name, and Operating System. Display name is the name that is visible in the management consoles and reports. Process name is the correct process name for the target operating system. Operating System is one or more operating system versions on which this process is monitored.

You can enter the Display name and Process name values either manually or automatically with the Browser function. As it is with the other browsers under *Data from a server*, if you manually enter the information, you must manually define each attribute to retrieve. When you use the browser to identify the data to gather, the browser automatically defines the attribute group and attributes needed, based on information that the browsed source provides. You can select more than one process to monitor in the browser.

If more than one process exists with the same name, you can use the matching options to limit which of the processes the agent monitors. Otherwise, the agent monitors all of them.



Note: Insert Property relates to adding runtime configuration, which this course covers.

Browsing processes

- Standard actions
 - Browse local or remote processes
 - Select one or more processes to monitor
 - Refresh
 - Search
 - Sort by column
- Remote browsing
 - IBM Tivoli Monitoring managed systems
 - OS agent must be installed and running on target
 - Identify Tivoli Enterprise Portal Server host, and then select managed system
 - Multiplatform, from all Agent Builders
 - Windows operating systems
 - Secure Shell (SSH)

Process Identifier	Name	Command Line
1552	db2dasrm.exe	"C:\Program Files\IBM\SQL..."
1584	db2mgmtsvc...	"C:\Program Files\IBM\SQL..."
5168	db2sysse.exe	C:\PROGRA~1\IBM\SQLI...
5296	db2systray.exe	"C:\PROGRA~1\IBM\SQLI...
5932	db2sec.exe	"C:\Program Files\IBM\SQL...

Monitoring processes and command return codes

9

© Copyright IBM Corporation 2017

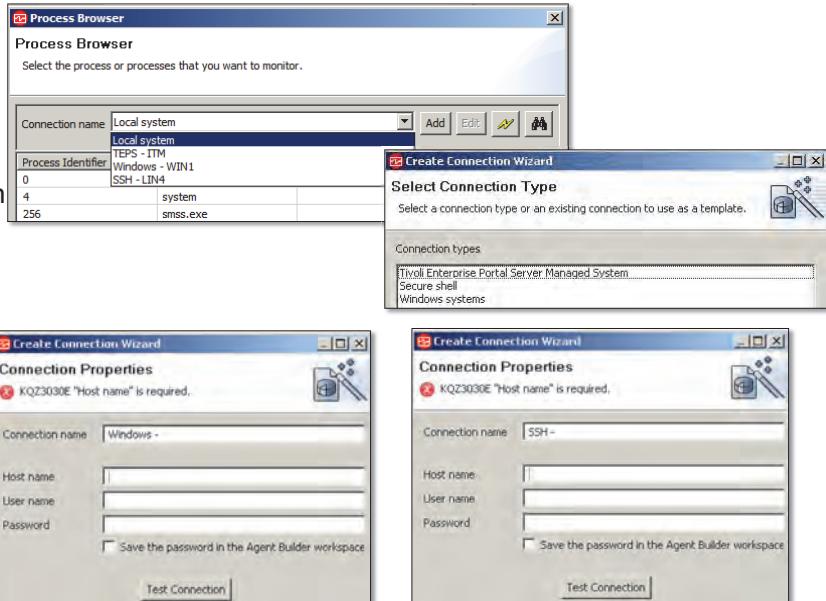
Browsing processes

With the Process Browser, you can locate and select one or more processes to monitor. You can complete many standard actions in this browser, such as the following examples:

- Local or remote browsing
- Refreshing the view
- Locating processes with the Find feature
- Selecting the specific processes that you want to monitor

Defining remote connections

- Define connections for new remote systems to browse
- Definitions shared between data sources
- Click Add to create a new definition
- Click Test Connection to confirm



Monitoring processes and command return codes

10

© Copyright IBM Corporation 2017

Defining remote connections

You can connect to and browse the processes that run on remote systems. You can connect to Windows systems, any system that accepts a secure shell (SSH) connection, and any Tivoli Monitoring managed system that runs an operating system agent.

Note: Browsing a remote host does not determine which host the agent monitors. Browsing identifies only the information to gather. Determining the host to monitor is a function of where you install the agent and how you configure the agent after you install it.

Lesson 2 Monitoring command return codes

IBM Training

IBM

Lesson 2: Monitoring command return codes

- Another availability monitor
- Functionality test
 - User-created script, executable file, query, or system command that tests the availability of an application or resource
 - Generates return codes
- Agent definition
 - Contains the functionality test, command return codes, and corresponding status and test messages
- Installed agent
 - Runs the functionality test
 - Analyzes the return codes
 - Returns availability information based on the current return code

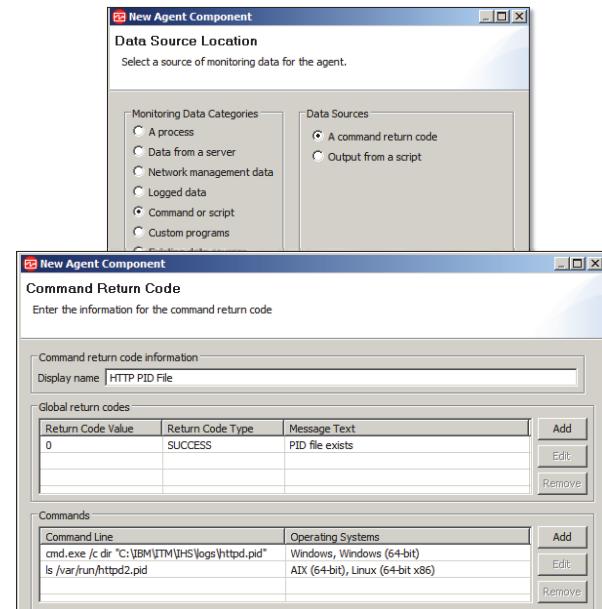
Availability		
Application Component	Name	Status
Apache2.4	Apache2.4	UP
DB2 - DB2COPY1 - DB2	DB2	UP
DB2DAS - DB2DAS00	DB2DAS00	UP
DB2 Governor (DB2COPY1)	DB2GOVERNOR_DB2COPY1	DOWN
DB2 License Server (DB2COPY1)	DB2LICD_DB2COPY1	DOWN

Monitoring command return codes is another availability data source.

You can test to see whether the monitored application or computer is available with a *functionality test*. A functionality test runs a user-created script, file, query, or system command that is an application-specific mechanism for determining whether the application is available. The functionality test must generate sufficient return codes so that the agent can analyze the return codes after running the test and return status information that it bases on the return codes. Set up a specific return code for each descriptive state and define a message for each return code that the test includes. The agent runs the specified command and determines the state of the application by analyzing the return code of the command. Define a return code for each value the command returns.

Adding a command return code data source

- Display name
 - Shown in Tivoli Enterprise Portal and Performance Management console as application name
- Command
 - One, and only one, per unique platform
- Command return codes
 - Create a return code for each unique return code that can be generated
 - Two for each command
 - Each requires a status and message
 - Create globally if shared status and message
 - Create per command if unique status and message



Monitoring processes and command return codes

12

© Copyright IBM Corporation 2017

Adding a command return code data source

When adding a command return code monitor, you must configure the display name, the functionality test commands, and the return codes.

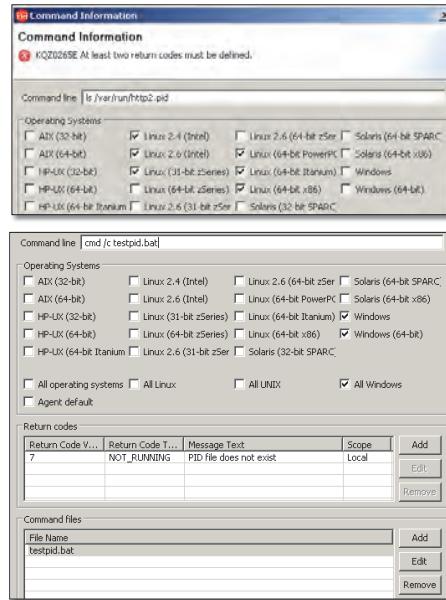
Display name is shown in the Application Component attribute in the Tivoli Enterprise Portal and Performance Management console.

The functionality test is the command that is run on a specific operating system. You can have only one command per each unique operating system. Each command must have at least two command return codes: 0 and another numeric code. Each command return code that you define in the agent must have a status and a message.

You define return codes either globally or per command. Global return codes are shared among all of the commands that you define for the functionality test. Per command return codes are unique to a specific command. Global return codes are good for any return codes that do not overlap meaning between individual commands. Per command return codes are best when specific return codes overlap between commands. For example, your agent has a Windows command and a Linux command that both return 0, indicating the functionality test is successful, and 1, indicating the functionality test fails. You can define these codes globally and share them. The same is true if the Linux command returns a 2 when the functionality test fails because the Windows and Linux return codes do not overlap. But if the Linux command returns a 1 when successful and a 2 when it fails, the Linux command needs a unique 1 return code.

Defining commands

- Command line
 - The command runs at the host
 - Quotation marks around commands with spaces
 - Test in local command prompt
 - Windows:
 - Use full command and file extension (such as path\file\command.bat)
 - Might have to run shell (cmd /c command)
 - Displayed in Tivoli Enterprise Portal: Name
- Platform
 - Required
 - Only one command per unique platform
- Return Code
 - One for each return code unique to this command
 - Overrides same global return code
- Files
 - One or more are run by the command



Monitoring processes and command return codes

13

© Copyright IBM Corporation 2017

Defining commands

Enter the command that you want to run at the host in the command-line field. Use quotation marks around commands with spaces. Always test your command locally before you add it to an agent.

For Windows commands, use the full path, command, and file extension. Depending on the command, you might have to run the command shell with **cmd /c** to generate return codes. The command is also the display name in the management console and reports.

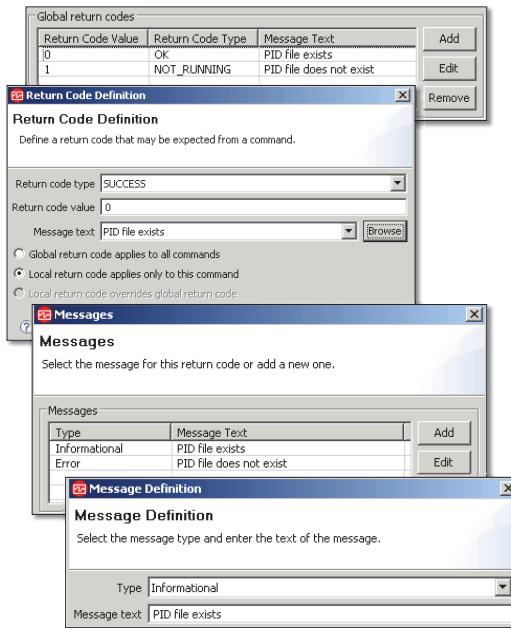
Select the operating system or group of operating systems to which this command applies. You can have only one command for each unique operating system.

Define each possible return code. Define a return code as global if it is the same for all commands or operating systems. Define the return code as local if it is unique to this command or operating system.

Add files that you want carried with the agent and used by your command. Run one of the files with the command.

Defining return codes

- Return Codes
 - Create one for each possible return code: global or per command
 - Two per command required
 - Set the type from list of seven: Success, General_Error, Already_Running, Not_Running, Prereq_Not_Running, Warning, and Dependent_Not_Running
 - Add a message from list
- Message
 - Created by user
 - Can be applied to global and command-specific return codes
 - Three types
 - Informational
 - Warning
 - Error
 - Text is displayed in functionality test message
- Socket and Java API data sources have a subfunction for monitoring errors



Defining return codes

Define a return code for each return code for all commands this functionality test generates. You must also define at least two error codes. You can define error codes globally and apply to all functionality tests, or you can define them uniquely for a specific functionality test. Each return code must have the return code value, a type, status, and a message. Return code types are predefined:

- ALREADY_RUNNING
- DEPENDENT_NOT_RUNNING
- GENERAL_ERROR
- NOT_RUNNING
- SUCCESS
- PREREQ_NOT_RUNNING
- WARNING

Messages are not predefined and you must be create them. Messages are shared across all data sources within a single agent. Messages have a type and message text. The message types are information, warning, and error.

Socket and Java API data sources have a subfunction for monitoring error codes that you configure similarly to defining return codes. Each data source has independent code and type values, but they share messages.



Note: The information that is given here for defining return codes applies to both global and per command return codes.

Student exercises



Exercise 1: Monitoring processes and command return codes

Exercise 2: Install and confirm the AB2 agent

Open your Course Exercises book and complete the exercises for this unit.

Summary

Now that you have completed this unit, you should be able to perform the following tasks:

- Add monitoring of multiplatform processes to an agent
- Add monitoring of multiplatform command return code to an agent

Unit 6 Monitoring custom data sources

IBM Training

IBM

Monitoring custom data sources

© Copyright IBM Corporation 2017
Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

This unit introduces you to custom data sources. Custom data sources require you to provide the instrumentation that is needed to gather the data that you want monitored. Custom data sources include monitoring with scripts, sockets, Java API, and parsing log files. It also introduces the runtime configuration feature, showing you how to create custom runtime configuration parameters. This unit finishes the installation topic by showing you how to generate Agent Builder output with the command-line interface (CLI).

Scenario business objectives

- Company has resources that require custom monitoring data be generated and gathered
- Create an agent that can gather custom data that is generated by a customer-provided item:
 - Command or script
 - Socket connection
 - Log file
 - Java API

Scenario business objectives

The training in this unit focuses on the knowledge and skills that you need to meet this scenario.

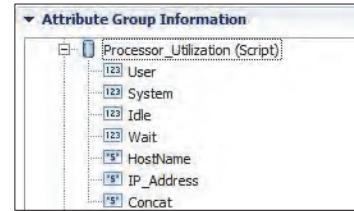
Scenario solution: Script

- Deploy a script locally and remotely that parses processor utilization information from the vmstat command
- Add attributes to capture the host name and IP address of the monitored host

```
lin4:- * vmstat 1 2
procs .....memory..... swap.....io....system....cpu....
r b swpd free buff cache si so bi bo in cs us sy id wa st
0 0 0 1261992 17132 380340 0 0 144 7 114 154 1 1 82 16 0
0 0 0 1261992 17132 380352 0 0 0 101 144 0 0 100 0 0
lin4:- #
```

```
lin4:~/AB_Files # cd /root/AB_Files/
lin4:~/AB_Files # ls
AgentGenerator.log SocketTestDL.pl SocketTestDR.pl SocketTestEL.pl SocketTest.pl
script1.sh SocketTestDL.pl~ SocketTestDR.pl~ SocketTestER.pl
lin4:~/AB_Files # cat script1.sh
vmstat 1 2 | tail -1 | awk '{print $13","$14","$15","$16}'
lin4:~/AB_Files #
```

```
lin4:~/AB_Files # ./script1.sh
8,8,67,17
```



Report

Node	Timestamp	User	System	Idle	Wait
lin4:02	05/12/15 01:09:24	0	0	100	0

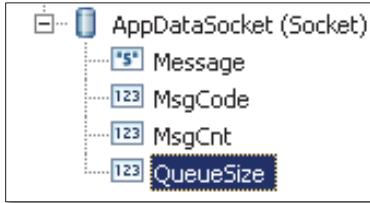
Scenario solution: Script

You create four agent solutions for this scenario. In the first, you create a script that generates an agent that deploys and runs a script. The script generates the data and the agent gathers and monitors the data. The key to this solution is to create a script or command that generates the data in the correct format.

Scenario solution: Socket

Monitor application data sent to the agent over a socket connection

- Configure the application to pass the correctly formatted data to the agent through the socket connection
- Configure the agent to listen to and gather data from that socket connection



Message	MsgCode	MsgCnt	QueueSize
A message from perl host name	1	2	123
More from perl	456	123	789

Scenario solution: Socket

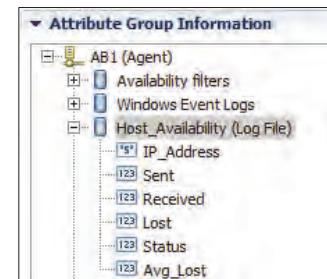
You also enable the agent to establish a socket connection with an application and monitor the application.

Scenario solution: Log file

- Monitor a log file that captures ping data on Windows
- Parse from the log file certain values that show the success of each ping.
- Create attributes to derive from the data a status indicator of Good or Failure and of the average number of lost packets

```
Pinging lin4.ibm.edu [192.168.1.107] with 32 bytes of data:
Reply from 192.168.1.107: bytes=32 time=1ms TTL=64

Ping statistics for 192.168.1.107:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 0ms, Maximum = 1ms, Average = 0ms
```



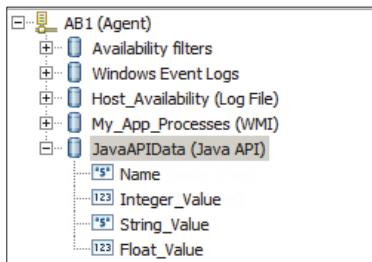
IP_Address	Sent	Received	Lost	Status	Avg_Lost
192.168.1.107	4	4	0	Good	0
192.168.1.103	4	4	0	Good	0
192.168.1.107	4	4	0	Good	0
192.168.1.103	4	4	0	Good	0

Scenario solution: Log file

In the second solution, you create an agent that monitors a log file that captures ping data. The agent must be able to parse the log file to pull the IP address, packets sent, packets received, and packets lost. Then the agent creates an attribute that states the status of the host that is based on the total number of packets received. Specifically, it reports the host status as Good if it receives four packets and Failure if it receives no packets.

Scenario solution: Java API

- Monitor application data sent to the agent by using the Java API data source
- Add custom Java code to the agent to monitor the application



Name	Integer_Value	String_Value	Float_Value
value of Name0	0	value of String_Value1	1.12
value of Name2	2	value of String_Value3	3.12
value of Name4	4	value of String_Value5	5.12

Scenario solution: Java API

Additionally, you add a Java API data source that enables the agent to monitor an application or resource with custom Java code.

Learning objectives

After completing this unit, you should be able to perform the following tasks:

- Add local and remote monitoring of script output to an agent
- Add local and remote monitoring through a socket connection
- Add monitoring of log files to an agent
- Add monitoring by a custom Java application
- Create runtime configuration properties
- Generate agent installer files from the command-line interface (CLI)

Learning objectives

The learning objectives include the skills that you need to implement the scenario solution.

Unit outline

- Lesson 1: Monitoring script output
- Lesson 2: Defining custom attributes
- Lesson 3: Creating derived attributes
- Lesson 4: Monitoring through a socket connection
- Lesson 5: Generating agent output from the command line
- **Exercises 1 - 2**
- Lesson 6: Monitoring log files
- Lesson 7: Custom runtime configuration
- Lesson 8: Java API
- **Exercises 3 - 4**

This slide gives you an outline of the lesson material in this unit.

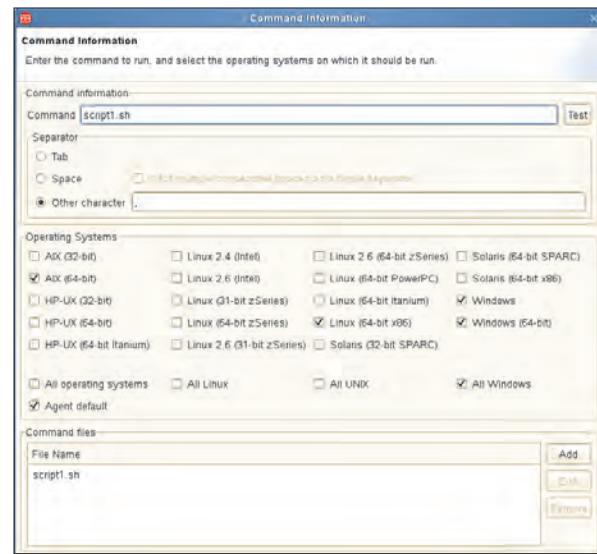
Lesson 1 Monitoring script output

IBM Training

IBM

Lesson 1: Monitoring script output

- Monitor gathers data output from a command or script run in the local command prompt
- Can distribute scripts and supporting files
- Can match different commands and scripts to relevant operating system
- Can establish SSH session and run script remotely



In this lesson, you learn to use a script to create the data you want monitored. After completing this lesson, you should be able to add local and remote monitoring of script output to an agent.

In many cases, application data is not available through a standard management interface. You can use scripts or external programs to generate data and have the agent collect and monitor that data. The agent can run system commands, scripts, and programs, or you can include the scripts and programs as part of the agent.

One agent can run different commands that match the requirements of different operating systems, but each OS-specific command must generate the same data in the same format.

Command or script output

- Each command or script defines an attribute group
- Output must be data only, no extraneous data
- Data must be delimited for each attribute and exactly match the attribute property definitions
- Multiple rows of data must be separated with a line break

```
lin4:~ # vmstat 1 2
procs .....memory..... swap... io... system... cpu...
r b swpd free buff cache si so bi bo in cs us sy id wa st
0 0 0 1261992 17132 380340 0 0 144 7 114 154 1 1 82 16 0
0 0 0 1261992 17132 380352 0 0 0 101 144 0 0 100 0 0
lin4:~ #

lin4:~/AB_Files # cd /root/AB_Files/
lin4:~/AB_Files # cat script1.sh
vmstat 1 2 | tail -1 | awk '{print $13","$14","$15","$16}'
lin4:~/AB_Files # 

lin4:~/AB_Files # ./script1.sh
8,8,67,17
```

Command or script output

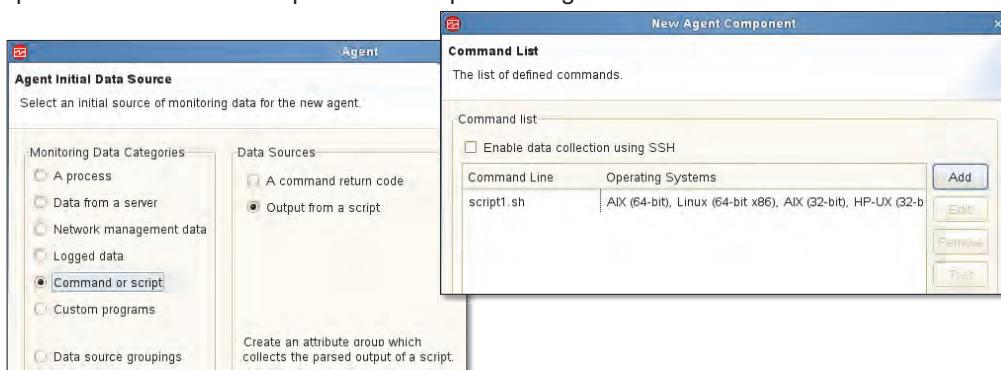
The key to this monitoring is the command or script. Each command or script defines the attribute group. The output of the command or script contains only values for each attribute of the attribute group that is delimited by a separator that you define. To return multiple rows of data, separate each row of attribute data with a line break.



Note: Command must write its output to the standard output stream, not something else like standard error.

Adding script monitoring to an agent

- Identify **Output from a script** as data source
- Select **Enable data collection using SSH** to add SSH configuration parameters and allow remote monitoring
- Add one or more commands per unique operating system
Platform-specific commands or scripts allow multiplatform agent



Adding script monitoring to an agent

After you identify that your data source is a script, you are taken to the Command List window. From this window, you can add one or more commands. Each line in this window represents a command for a specific operating system that generates the data that you want. You can have only one line for each unique operating system.

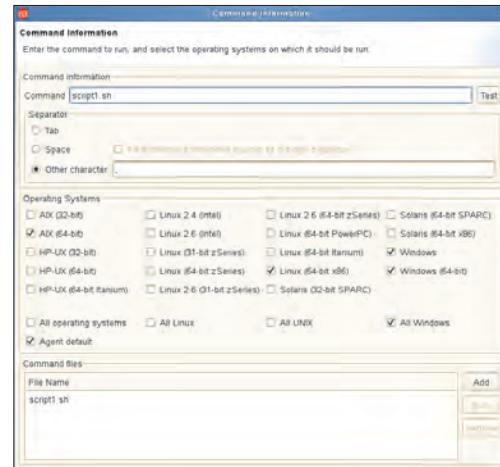
The data each of these commands generates are stored in the same attributes in the same attribute group. The data that each command generates must be similar in the number of attributes and the data types. However, the way the data is generated can differ, as in the following examples:

- You might run different commands.
- You might use one or more scripts to generate the data.
- The delimiter might be different.

Also, from this window, you can manage each command by editing, removing, and testing the commands.

Remote script command information

- **Command** is either
 - System command run on the remote system
 - Script delivered to remote system and run
- Enter separator that matches command or script output
- **Operating Systems** is where the agent is hosted
- Command files are uploaded to remote system
 - Agent does not adjust files
 - Must be runnable on remote architecture
 - End of line must match remote architecture
 - If you distribute more than one script, enter lead script as the command
- Environment variables and runtime configuration information not currently uploaded
 - Most UNIX systems require a period and slash (./) at the beginning of the command



Remote script command information

Each command has these three components:

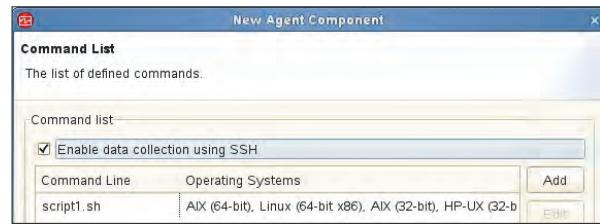
- **Command**: The command that generates the data. It might be a local command already available on the agent host, or it might match a file that the agent carries.
- **Separator**: Identifies what character separates the attributes and enables the agent to correctly parse the data that the command generates.
- **Operating systems**: Identifies the operating systems on which you can install the agent. It is suggested that you select the All operating systems check box when you use the SSH data collection features.

Environment variables and Runtime Configuration Information parameters set in the agent are not currently uploaded to the remote system. Most UNIX systems do not add the current working directory to the path, so add the prefix ./ to the command.

The agent can also carry one or more files. The main command must run one of these files. All entries that are listed in Command Files are uploaded to the remote system. The SSH data provider does not adjust the contents of the Command Files. Write the files to run on the remote computer architecture. Ensure end-of-line conventions match the remote computer operating system.

Remote script (SSH)

- Can optionally run scripts or remote commands by secure shell (SSH)
- Select **Enable data collection using SSH**
- Test window supports remote testing of scripts
- Might be hosted on any supported operating system
- Authentication: either user name with password or public key Public Key authentication tested with RSA and DSA
- Supports these authentication methods:
 - SSH v2 using RSA or DSA keys
 - AES or 3DES encryption
 - MD5 or SHA1 message authentication



Remote script (SSH)

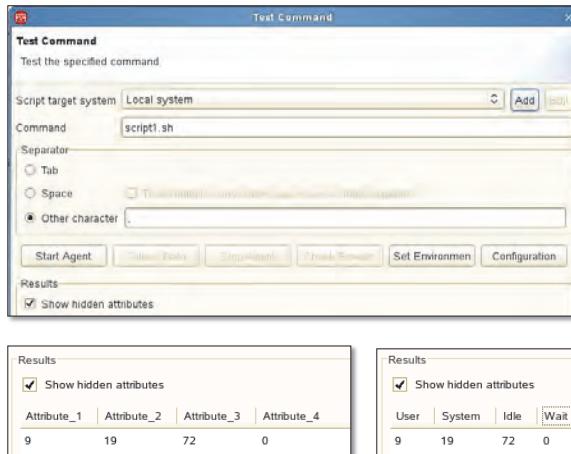
The Script data provider can run scripts or commands on a remote computer by using secure shell (SSH). To enable SSH connections, select **Enable data collection using SSH**. The Test window for the script also supports remote testing of scripts. Any supported OS can host agents with Remote Script data providers.

Complete the authentication by using either user name and password, or public key authentication. The process of generating and installing public keys to target servers is outside the scope of the agent and is done before starting the agent. Public Key authentication is tested with RSA and DSA key pairs that the OpenSSH ssh-keygen tool generates.

The SSH transport supports the SSH v2 protocol that uses RSA or DSA keys, AES, or 3DES encryption, and MD5 or SHA1 message authentication that the GSKit component of Agent Builder provides.

Testing a command

- You can test your command and its output before installing the agent
- You can establish an SSH session with a remote system to test the script
- Attempts to define attributes to match results
 - You can keep attribute definitions
 - Cancel to discard attribute definitions
- You can edit attributes with **Test Command**



Testing a command

With the Test Command window, you can test your command and confirm that it generates the data that you want before you package and install the agent. Furthermore, it attempts to generate attributes by selecting an attribute type for each data point generated by the command.

The basic goal is for your command to generate the correct output and the Test Command window to parse the data correctly, creating the correct number of attributes.

While your command is creating the correct number of attributes, you can modify the attributes. You might want to make the following basic attribute changes:

- Give each attribute a name
- Set the attribute type

If you need any environment variables for the command, click **Set Environment** to show the Environment Variables window, where you can set the variables and the values. The Environment Variables window lists all of the environment variables that the command uses while running. Initially, the list is blank.



Note: If you click **OK** from this window, you exit the Test Command window and save your attributes. Click **Cancel** to exit the Test Command window without saving the attributes that you define.

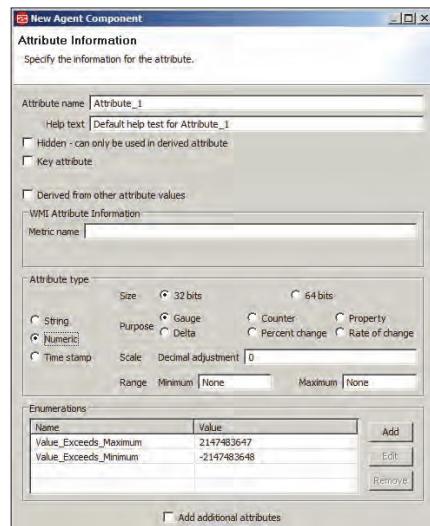
Lesson 2 Defining custom attributes

IBM Training



Lesson 2: Defining custom attributes

- Initial attributes can be created by the tester or manually
- Standard actions
 - Set attribute name
 - Define Help text
 - Choose to display in Tivoli Enterprise Portal or not
 - Set as key attribute, if required
 - Set the attribute type based on the data type



Note: Modifying attribute parameters applies to most data sources, especially for custom attributes

This lesson describes how you create custom attributes and their properties. After completing this lesson, you are able to edit existing attributes and create custom attributes.

Attribute name is the name of the attribute as it is shown in data display consoles and reports. It is up to 63 characters long and composed of letters, numbers, and the underscore (_).

Help text provides Help text for the attribute that is often shown with the attributes.

Hidden indicates whether the attribute is visible in the Tivoli Enterprise Portal or Performance Management console. In cases that use the attribute in calculations with other attributes, there are reasons not to show the base value. For instance, a number that represents a byte count wraps so quickly that it is of little use.

Key attribute sets the attribute as a key in the table.

Attribute type defines the type of data that the attribute holds and determines how the monitoring system manages the data.

String can contain any UTF-8 characters. The defined length that is the total length of the buffer that is allocated to contain the string. In non-ASCII characters, this attribute might take more than 1 byte per character. Data aggregation in the warehouse shows the most current value that is collected during the period.

- Maximum size** is the maximum number or maximum character length of the value.
- Integer** is a number.

- **Size:** Indicates whether the number is a 32-bit or 64-bit number. 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 -9x10¹⁸ to 9x10¹⁸.
- Purpose
 - **Counter:** A positive integer value that contains values that generally increase over time. Data aggregation in the warehouse shows the total, high, low, and most current values.
 - **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.
 - **Gauge:** Integer values where the values returned are higher and lower 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.
 - **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.
 - **Property:** A property of the object that does not frequently change. Data aggregation in the warehouse shows the most current value that is collected during the period.
 - **Rate of change:** An integer value that represents the difference between the current value and the previous value that 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.
 - **Timestamp:** A string attribute whose format conforms to the format CYYMMDDHHMMSSmmm, where C=1 for the 21st century.

Adding and modifying attribute parameters is not limited to script and log file monitors. They can apply to attributes for most data sources, especially if you manually add attributes to those data sources.



Note: The following pages describe scale, range, and enumerations.

Scale

- The number of digits to the right of the decimal
- Floating point values can be collected and displayed as decimal values instead of as integers
- IBM Tivoli Monitoring and IBM Application Performance Management represent these values as integers
 - The agent handles scaling the value so that the correct data is displayed
- Because decimal values are simulated with 32-bit or 64-bit integers, adding a scale reduces the range of values you can represent
 - For this reason, Agent Builder makes decimal values as 64-bit by default

Used Pct	Mount	Free Pct
66	/	33.93
65	/home	34.75
0	/dev/shm	100.00
43	/data	56.50
100	/media/01242009	0.00

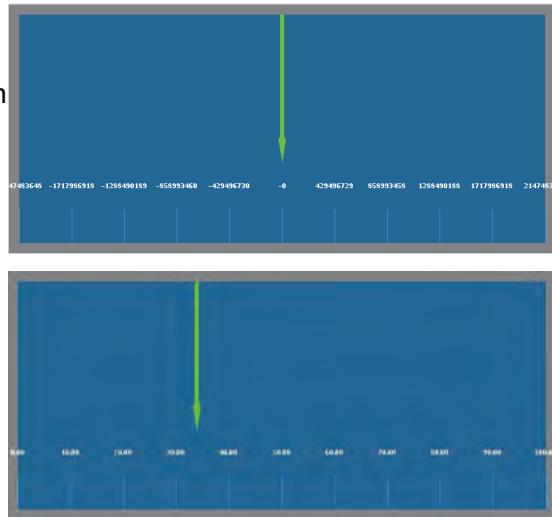
Scale

Scale determines how many decimal places are in the number. Each decimal place reduces the range 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 you specify a nonzero decimal adjustment, the number is manipulated internally as a floating point number. This process might reduce the precision of large 64-bit numbers.

Range

- Produces a more useful initial view in some graphs
 - Provide the minimum and maximum values
 - When not specified, the range shown is from Min Int to Max Int or from Min Long to Max Long
- Agent developers or users can still customize ranges in their view properties

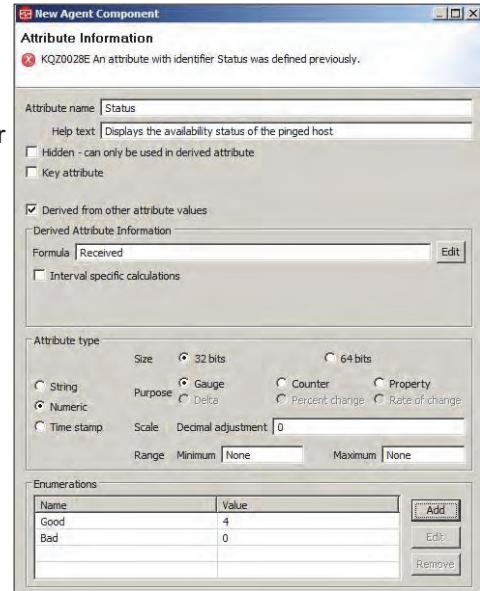


Range

Range indicates the expected range of the value. If no minimum or maximum ranges are given, default values are used. Some graphical Tivoli Monitoring workspace views use the range to produce a more useful initial view.

Enumerations

- With enumerations, you can display alternative names for values
- Numeric enumerations
 - Supported for Gauge, Counter, or Property, 32 bits or 64 bits
 - Not allowed if a scale is defined
 - Not allowed for Delta, Percent change, or Rate of change
- String enumerations
 - Make data more meaningful with string enumeration



Enumerations

The enumeration name is shown in the Tivoli Enterprise Portal or in the IBM Application Performance Management console when the corresponding value is received in the attribute from the agent. For example, a value of 0 might show as the word Normal, while a value of 1 might show as Error.

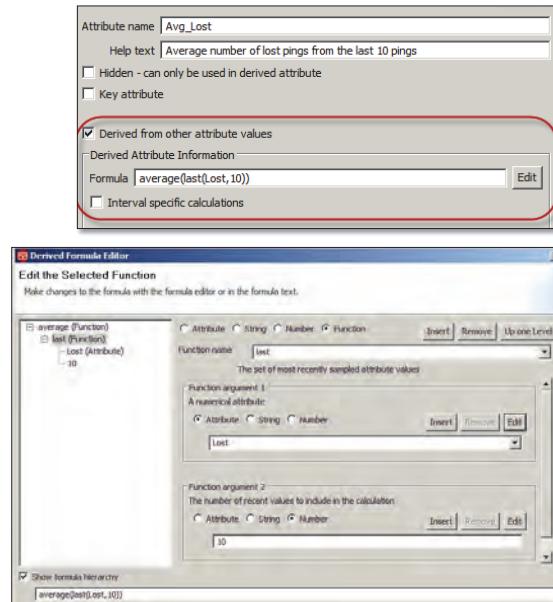
Lesson 3 Creating derived attributes

IBM Training

IBM

Lesson 3: Creating derived attributes

- You create derived attributes based on existing attributes with simple calculations
- Process
 - Click **Derived from other attribute values**
 - Enter formula or click **Edit** to open Editor
- Formula Editor features
 - Helps build the expression for a derived attribute
 - Syntax checks each change
 - Allows insertion of these items:
 - Attributes
 - Constants
 - Operators
 - Functions



Monitoring custom data sources

19

© Copyright IBM Corporation 2017

This lesson teaches you how to derive new data from existing attributes. After completing this lesson, you are able to create derived attributes.

You can create derived attributes based on existing attributes by using simple calculations. For numeric operations, all input attributes are numeric attributes.

For string concatenation, all input attributes are string attribute values. The text of two attributes combines into one and is saved with a new attribute name.

The Formula Editor is a graphical editor that aids in the creation of formulas in the Agent Builder.

Operators

Formula syntax:

Attribute1 operator Attribute2 operator Attribute3...

Examples	Operators	Examples
Numeric	<ul style="list-style-type: none">- Minus (Attribute1 minus attribute2)+ Plus (Attribute1 plus attribute2)* Multiply(Attribute1 times attribute2)/ Divide (Attribute1 divided by attribute2)% Percent (Attribute1 times 100 divided by attribute2)() Operation control	Difference: Width - Depth Sum: Width + Depth Area: Width * Depth Ratio: Width / Depth Percent: Width % Depth Height: L1*(L1+L3)
Text concatenation	+ Plus (Attribute1 plus attribute2)	Date_Time: Data + Time

This slide lists mathematical and text operators that you can use in derived attributes.

Derived attributes functions

- Derived expressions can include functions
- Some functions work on one value (absolute value, square root, and so on) and some work on lists of values (min, max, average, and so on)
- A function that expects a single value can be passed an attribute, a constant, or the result of some other function
 - `abs (Attr_A)`
 - `round (atof(Attr_B))`
- A function that expects a list can be passed a set of previous values of one attribute or any number of separate attributes
 - `Min (last(Attr_A, 10)`
 - `Max (Attr_A, Attr_B, Attr_C, 100)`

Derived attribute functions

Derived expressions can include functions.

List of derived attributes functions

- **Mathematical:** abs, average, ceiling, count, delta, floor, last, max, min, rate, round, stddev, sqrt, sum
- **Look up:** getenv, InstanceName, ipAddressToName, nameToIpAddress
- **Parsing and replacement:** matches, replaceAll, replaceFirst, tokenize
- **Conversions:** atof, atoi, itoa, NetWareTimeToTivoliTimestamp, StringToTivoliTimestamp, TivoliLogTimeToTivoliTimestamp, UTCtoLocalTime, UTCtoGMT
- **Events:** cumulativeSum, eventThreshold, isSummaryEvent, occurrenceCount, summaryInterval

List of derived attributes functions

- Mathematical
 - **abs** returns the absolute value of a value.
 - **average** returns the average value from a set.
 - **ceiling** returns the least integer that is not less than the argument.
 - **count** returns the count of values in a set.
 - **delta** returns the difference between the most recent value of an attribute and a previously collected value of that attribute.
 - **floor** returns the greatest integer that is not greater than the argument.
 - **last** returns the last n values for an attribute.
 - **max** returns the maximum value from a set.
 - **min** returns the minimum value from a set.
 - **rate** returns the rate of change per second between the most recent value of an attribute and a previously collected value of that attribute.
 - **round** rounds a value instead of the default truncation.
 - **stddev** returns the standard deviation of a set.
 - **sqrt** returns the square root of a value.
 - **sum** returns the sum of a set.

- Look up
 - **getenv** returns the value of an environment variable or configuration parameter.
 - **InstanceName** returns the instance name of the multiple-instance agent.
 - **ipAddressToName** converts an IP address to a host name.
 - **nameToIpAddress** converts a host name to an IP address.
- Parsing and replacement
 - **matches** returns a Boolean, true or false, indicating whether a regular expression matches a value.
 - **replaceAll** replaces all occurrences of substrings that match a regular expression with a replacement string.
 - **replaceFirst** replaces the first occurrence of a substring that matches a regular expression with a replacement string.
 - **tokenize** parse a string attribute on a separator and return the nth token.
- Conversions
 - **atof** converts a string into a floating point number.
 - **atoi** converts a string into an integer value.
 - **itoa** converts an integer value into a string.
 - **NetWareTimeToTivoliTimestamp** converts a Novell NetWare hexadecimal time value to an IBM Tivoli Monitoring time stamp.
 - **StringToTivoliTimestamp** converts a date and time string to an IBM Tivoli Monitoring time stamp.
 - **TivoliLogTimeToTivoliTimestamp** converts a Tivoli log file time stamp to an IBM Tivoli Monitoring time stamp.
 - **UTCtoLocalTime** converts the Coordinated Universal Time (UTC) value into the local time in a Tivoli time format. The argument is an integer time_t format. Added for UA parity.
 - **UTCtoGMT** converts the Coordinated Universal Time (UTC) into the Greenwich Mean Time (GMT) in a Tivoli time format. The argument is an integer time_t format. Added for UA parity.
- Events
 - **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.
 - **eventThreshold** Returns the threshold value that is configured for the attribute group that generates the event.
 - **isSummaryEvent** Returns 0 if it is a single event from a data source, or 1 if the event is a flow control summary event.
 - **occurrenceCount** Returns the number of matching events that are represented by a flow

control summary event, including the first event), or 1 if it is a single event from a data source.

- **summaryInterval** Returns the summary interval that is configured for the attribute group that generates the event, in seconds.

Function examples

- `Last (attrA, 10)`
 - Returns the last 10 values collected for attribute attrA
- `Average (last (attrA, 10))`
 - Returns the average of the last 10 values collected for attribute attrA
- `min (attrA, atoi(getenv(KXX_NUM_COLLECTIONS)))`
 - Returns the minimum of the values returned for attrA
 - The value is over the number of collections defined by the configuration value NUM_COLLECTIONS that is converted to a number from the string value as returned from the environment
- All of these can be combined with any other function, attributes, and mathematical operators.
 - `min (last (attrA, 10)) + min (last (attrB, 10))`
 - `min (attrA, attrB, attrC)`

Function examples

This slide shows some function examples.

Conditional expressions

- Create IF/THEN/ELSE
- Format: condition ? true_value : false_value
 - The expression in the Condition section must evaluate to true or false.
 - Operators are ==, !=, <, <=, >, >=, &&, ||, !

Conditional expressions

You can use conditional expressions in derive attribute formulas. 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.

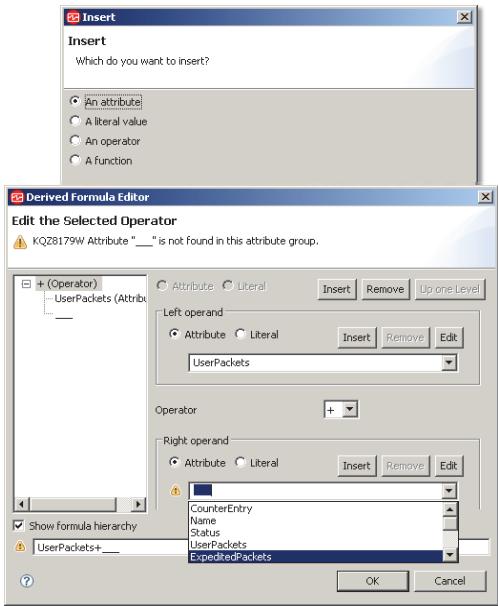
The expression in the Condition section must evaluate to true or false. Use operators (==), (!=), !(<), (<=), (>), (>=), (&&), (||), (!) to form expressions that return True or False.

You can edit simple operands, attributes, and literals, without having to change the selected component to the operand.

You can edit complex operands, which consist of other operators or functions, by clicking **Edit**. This action highlights the operand component instead of the entire conditional expression.

Formula Editor

1. Click **Insert**
2. Select the entity to add
3. Keep selecting **Insert** for appropriate operations, attributes, or functions
4. Navigate in the expression by using the view on the left
5. Use the right pane to edit appropriate operations, attributes, or functions
6. Type in the field on the bottom
 - Changes in one pane are automatically reflected in the other sections
 - Errors are displayed at the top until the formula is legal



Monitoring custom data sources

25

© Copyright IBM Corporation 2017

Formula Editor

This slide shows the basic process that you use to create a formula with the Formula Editor.

When the Formula Editor is visible, 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. You can edit the operator and its two operands 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 field.
- Formula hierarchy tree view, option: Select the **Show formula hierarchy** check box to show the formula hierarchy tree. The state of the check box is saved to use in subsequent invocations of the Formula Editor.

Lesson 4 Monitoring through a socket connection

IBM Training

IBM

Lesson 4: Monitoring through a socket connection

- Allows user-written code in any programming language that can use a socket to pass data to the agent
- The process can be long running or connect when it has data to provide
- Formatted data can be written to a socket monitored by the agent
 - Defaults to localhost connections only, but can be configured to accept remote socket connections
 - Data is passed in XML format



In this lesson, you learn how to enable an agent to receive data from an application through a socket connection.

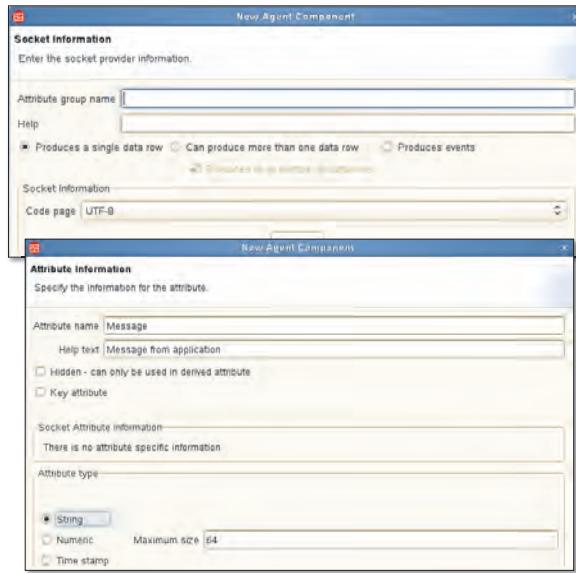
After completing this lesson, you are able to add local and remote monitoring through a socket connection.

Use the socket data source to provide data to the agent from an external application. The agent passively listens to a port. 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 runs. 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, the user controls this action.

You must configure the agent to set which port to listen to and if remote connections are allowed.

Defining the socket attribute group and attributes

- No tester or method to automatically define attributes
- Each attribute group and attribute must be manually created
- Attribute group: Important to set the number of rows gathered and code page correctly
- Attributes: Important to set the type correctly

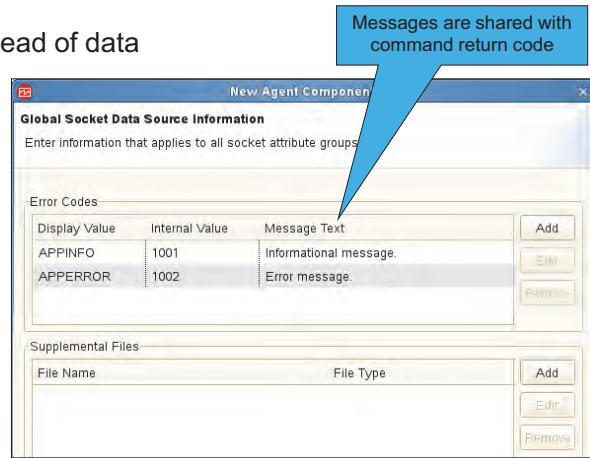


Defining the socket attribute group and attributes

Agent builder has no tester or method to automatically define attributes for the socket data source. You must manually create each attribute group and attribute.

Defining error codes

- Error codes can be sent to the agent instead of data
- Assists problem determination
- Includes an error code, a displayed value, and a logged message



Defining error codes

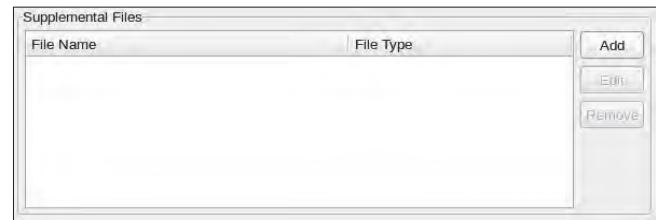
You can configure error codes to send to the agent when the monitored application cannot collect data. You must define the error code, display values, and messages. These error code definitions are shared with the command return code data source.



Note: The error code creation process is the same as described in [Unit 5, “Monitoring processes and command return codes”](#) and is not repeated here.

Adding supplemental files

- Client files can be included with the agent
- The agent does not use them
The files are available for your client to use



File Type	Description
Executable	Select this option if you want to include an executable with the agent. The agent does not use these files.
Library	Select this option if you want 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.

Adding supplemental files

You can add files to package with the agent. These files are copied to the agent system when you install the agent. The agent does not use the files, but the files are available for your client to use. The agent does not manage your client's lifecycle.

Configuring the listening port and remote connections

- Runtime configuration parameter: Listening port
 - 0 indicates any available port
- Port used written to a file that the socket client can read
 - UNIX: /tmp/kxx_[instanceName]cps.properties
 - Windows: %CANDLE_HOME%\TMAITM6\kxx_[instanceName]cps.properties
 - File contents:
CPS_PORT=<port number>
- To allow remote socket connections, set environment variable

Environment Variables	
Environment variables set at agent runtime	
Name	Value
CDP_DP_ALLOW_REMOTE	YES

Configuring the listening port and remote connections

To allow remote socket connections, set the CDP_DP_ALLOW_REMOTE environment variable.

The port the agent listens on is the only configuration parameter for a remote script data source that uses SSH. A value of 0 for the port indicates that the agent picks an available port.

The port that is used is written to a file that the socket client can read.

- UNIX: /tmp/kxx_[instanceName]cps.properties
- Windows: <install_dir>\TMAITM6\kxx_[instanceName]cps.properties
- File contents: CPS_PORT=<port number>

Sending data

- Client connects to defined port or reads agent file for port

- Client sends properly formatted data
 - Data must match attribute group definition
 - Order matters
 - Skips derived attributes

- Sample for the Socket_Demo attribute group

```
<socketData><attrGroup name="Socket_Demo"><in>
<a v="Wake County, North Carolina"/><a v="1"/>
<a v="140"/></in></attrGroup></socketData>\n
```

- Attributes are string value followed by two numerical values
- Data types match Socket_Demo attribute group
- Attribute order matches attribute group
- Use multiple instance tags <in> for each row to send multiple rows

Sending data

The client application connects to the defined port or reads the agent file and connects to the port it specifies. The client sends properly formatted data to the agent. The data that is sent must match the attribute group definition and order. Derived attributes in the attribute group are skipped.

The slide shows a sample of properly formatted data for a socket data source. The attributes are a string value followed by two numerics. The data types and the order of the attributes match the Socket_Demo attribute group. Multiple rows can be sent by including multiple instances of the tag <in> that contain attribute data for one row.

More data samples

- Sending an error code instead of data

```
<socketData><attrGroup name="Socket_Demo"><error rc="1000"/></attrGroup></socketData>\n
```

- Sending data to a subnode named instanceName

```
<socketData subnode="instanceName"><attrGroup name="Socket_Demo"><in><a v="Wake County, North Carolina"/><a v="2"/><a v="140"/></in></attrGroup></socketData>\n
```

- Sending data to attribute group that produces events is no different

- Must include only one row of data in each message

- No data collection request

- Updated only when socket client sends
 - Data continues to be available from agent until updated by client

More data samples

Sending data for an attribute group that produces events is no different than sending data for attribute groups that contain sampled data. There is no data collection request for sampled attribute groups. The attribute group data is updated only when the socket client sends an update. That data is available from the agent until the socket client updates it.

If you send data to an attribute group that produces events, you must include only one row of data in each message.

Notes on coding data

- Include backslash *n* (\n) to indicate the end of a message
 - Agent reads text between the newline characters
 - No \n characters in attribute values
- Code pages
 - Agent Builder uses UTF8
 - Specify UTF8 in agent, if UTF8 used by client
 - Specify local code page, if UTF8 not used by client and agent converts

Notes on coding data

Always include the newline code \n to indicate the end of a message. The agent reads text between the newline characters to parse a single message. Make sure \n characters do not occur in your attribute values.

Agent Builder uses the UTF8 code page. Specify UTF8 in the attribute group definition if you can provide data in UTF8. Specify the local code page if you want the agent to convert the data to UTF8 for you from the local code page.

Formatting attributes

Replace this special character	With this text
\n	

&	&
<	<
>	>
“	"
‘	'

- No special formatting for numerical values, such currency symbols or thousands separators
- Time stamps must follow the IBM Tivoli Monitoring stamp format:
CYYMMDDHHMMSSmmm
Sample: 1110505181030000 is the time: May 5, 2011 6:10:30 PM

Formatting attributes

String attribute values must replace special characters. Use the replacements that are shown in the slide if any special characters are present in your attribute values.

Numeric values must not have special formatting such as currency symbols or thousands separators.

Time stamps must follow the IBM Tivoli Monitoring stamp format.

Take Actions

- Socket client can receive Take Action requests from agent

- Registration requires a prefix

```
<taskPrefix value="K04"/>\n
```

- Take action request

```
<taskRequest id="1">\n<task command="K04_restart_device" user="sysadmin"/> </taskRequest>\n
```

- Take action response is return code

```
<taskResponse id="1" rc="0"/>\n
```

- 0 is success

- All others are an error

Take Actions

A long running socket client can register to receive Take Action requests from the agent. Sample Take Action code is shown in the slide and includes a registration prefix from the agent, the Take Action from the agent, and a response from the client. Use the agent's product code or a similar unique identifier for the registration prefix. The Take Action response is a return code. A return code of 0 indicates success, anything else indicates an error.

Lesson 5 Generating agent output from the command line

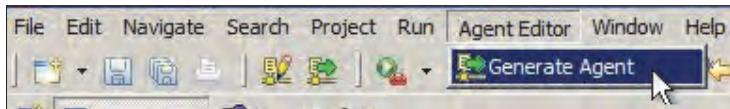
IBM Training

IBM

Lesson 5: Generating agent output from the command line

Agent output and installation review

- Types
 - Install locally: Install agent and ITM application support directly on local computer
 - Output agent installer: Create script-based installers for installing agent or Tivoli Monitoring application support Archived in single zip and tar files
- Can be done from within Agent Builder graphical user interface or command-line interface
- In Agent Builder, performed with Agent Generator wizard



- A standard agent must be installed on the target
- This lesson focuses on using the Agent Builder CLI command to perform both types of outputs

Note: This lesson completes the installation topic

This lesson teaches you how to run the Agent Builder output wizard from a command line without the Agent Builder graphical user interface. After completing this lesson, you will be able to generate agent installer files from the CLI.

This slide provides a review of the various installation processes for an Agent Builder agent. This lesson focuses on using the Agent Builder CLI command to complete all three types of outputs.

Note: This lesson completes the installation instructions that this course teaches.

Agenttoolkit command

- Use **agenttoolkit** command to generate agent output from command line
 - In the Agent Builder installation directory
 - Windows: <install_dir>\Agentbuilder
 - UNIX/Linux: <install_dir>/Agentbuilder
- All standard outputs created
 - Quick (local) installation
 - Compressed installation files

Agenttoolkit command

The IBM Agent Builder contains a command-line interface (CLI) that you can use to generate the agent without starting the Eclipse graphical user interface (GUI). This slide shows the location and file name for the Windows, UNIX, and Linux versions of the command.

You can use this command to generate the three standard Agent Builder outputs.

Command syntax

- Quick (local) Installation CLI
 - Options

```
agenttoolkit.{bat /sh} project_dir -generatelocal itm_install_dir
```
 - Example

```
agenttoolkit.{bat /sh} C:\AB_Class\MyAgent -generatelocal C:\IBM\ITM
```
- Compressed Installation Files CLI
 - Options

```
agenttoolkit.{bat /sh} project_dir -generatezip output_dir
```
 - Example

```
agenttoolkit.{bat /sh} C:\AB_Class\MyAgent -generatezip C:\Output
```

Quick, local, Installation CLI

As with the GUI version of this command, this option initiates the following actions:

- Loads and validates the **itm_toolkit_agent.xml** file
- Generates the files for installing the agent
- Installs agent and application support into the local IBM Tivoli Monitoring environment
- Removes installation files

This slide shows the options that you use and provides an example. This table defines each variable that this option uses.

Variable	Definition
project_dir	Specifies the name of the directory that contains the itm_toolkit_agent.xml file
itm_install_dir	Specifies the location where IBM Tivoli Monitoring is installed, for example C:\IBM\ITM

Compressed Installation Files CLI

As with the GUI version of this command, this option initiates the following actions:

- Loads and validates the **itm_toolkit_agent.xml** file
- Generates the compressed installation files so that the agent can install on another computer

The slide shows the options that you use and provides an example. This table defines each variable that this option uses.

Variable	Definition
project_dir	Specifies the name of the directory that contains the itm_toolkit_agent.xml file
output_dir	Specifies the name of a directory where the compressed file is written

Student exercises



Exercise 1: Create an agent to monitor local and remote systems with scripts and socket connections
Exercise 2: Install and confirm the updated AB2 agent

Student exercises

Open your Course Exercises book and complete the exercises for this unit.

Lesson 6 Monitoring log files

IBM Training

IBM

Lesson 6: Monitoring log files

- Can contain important information
- Agent definition is more complex
 - Different file names and locations on different hosts
 - Must correctly parse the file
 - Keep certain information
What is a record (set of data values)
 - What is a field of data
 - Ignore unwanted information
 - Manual work needed for attributes
 - How to process the log file
- XML log file parsing

ping.txt - Notepad

Pinging lin4.ibm.edu [192.168.1.107] with 32 bytes of data:
Reply from 192.168.1.107: bytes=32 time=1ms TTL=64
Reply from 192.168.1.107: bytes=32 time<1ms TTL=64
Reply from 192.168.1.107: bytes=32 time<1ms TTL=64
Reply from 192.168.1.107: bytes=32 time<1ms TTL=64

Ping statistics for 192.168.1.107:
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
Minimum = 0ms, Maximum = 1ms, Average = 0ms

Pinging win1.ibm.edu [192.168.1.103] with 32 bytes of data:
Reply from 192.168.1.103: bytes=32 time<1ms TTL=128
Reply from 192.168.1.103: bytes=32 time<1ms TTL=128
Reply from 192.168.1.103: bytes=32 time<1ms TTL=128
Reply from 192.168.1.103: bytes=32 time<1ms TTL=128

Ping statistics for 192.168.1.103:
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
Minimum = 0ms, Maximum = 0ms, Average = 0ms

Pinging lin4.ibm.edu [192.168.1.107] with 32 bytes of data:

Report

Node	Timestamp	IP Address	Sent	Received	Lost	Status
ITM:03	07/12/15 18:24:31	192.168.1.107	4	4	0	Good
ITM:03	07/12/15 18:24:28	192.168.1.103	4	4	0	Good
ITM:03	07/12/15 18:24:25	192.168.1.107	4	4	0	Good
ITM:03	07/12/15 18:24:22	192.168.1.103	4	4	0	Good

In this lesson, you learn how to monitor data that is stored in log files. After completing this lesson, you should be able to add monitoring of log files to an agent.

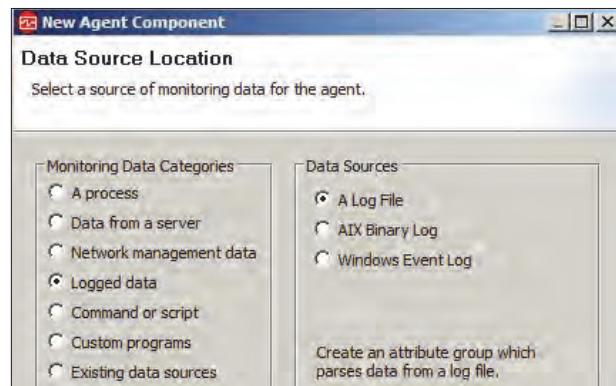
Log files can contain important data that you might want to monitor. Like monitoring script output, monitoring log files is different from most of the other types of monitoring done by your Agent Builder agent. The data from most data sources is well-defined, making it easier for you to create an agent that can gather the correct data. As you saw with monitoring script output, this situation is not true for monitoring log files.

The log file location and file name can be different on the different hosts you want to monitor. Log files can contain unwanted information; so the file must be correctly parsed to remove just the information you do not want and return only the data that you do want.

The attributes need manual adjustment to correctly match the data that is gathered and to select the correct data. You must construct the agent to correctly process the log file each time that it gathers data.

Process overview

- Identify log file as data source
- Specify the log file to monitor (attribute group)
Can be platform specific
- Define record identification (rows to return)
- Define how to process the log file
- Define field identification
Global or attribute
- Test log file parsing
- Define attributes



Note: All steps are not required for simple log files: delimited, data only, single line per record. Steps can be iterative, testing, and redefining.

Process overview

This slide gives an overview of the process of creating an agent that monitors a log file. It also shows the Data Source Location window where you identify a log file as the data source. This lesson describes each of these steps in detail.

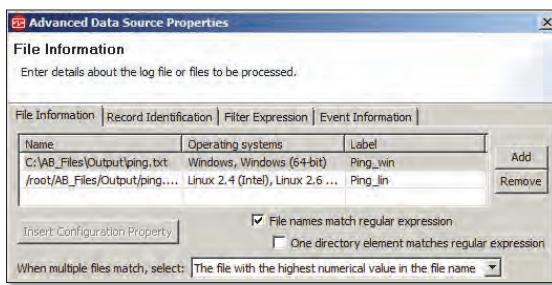
As you see with script monitoring, Agent Builder has a feature that you use to test your agent's ability to parse the log file. It can also interactively define the attributes to hold the data so that you can confirm that your agent gathers and retains the correct data before you deploy the agent.



Note: Not all steps are necessary for simple log files, such as ones that contain only the data to retrieve or the data is consistently delimited and a record is contained on a single line. The order of the steps can vary and is often iterative, reconfiguring, and retesting until you get the results you want.

Specifying the log file to monitor

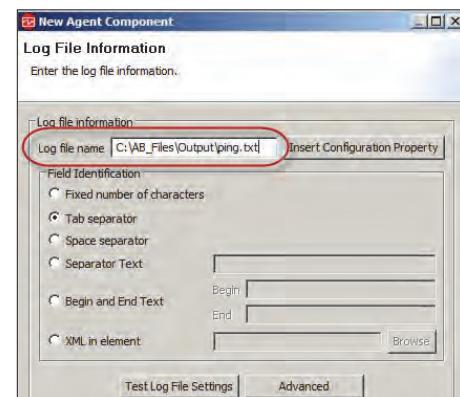
- If single log file, enter the path and file name
- If different per operating system
 - Click **Advanced > File Information**
 - Enter path and file name, operating system, and unique label
- For variable
 - Insert Configuration Property
 - Dynamic naming
 - Regular expression



Monitoring custom data sources

42

© Copyright IBM Corporation 2017



Specifying the log file to monitor

After you indicate that you are creating an agent that monitors a log file, the first thing you must identify is the log file to monitor.

If the log file on each host to monitor is the same name and in the same location, enter the full path and file name in the **Log file name** field.

If the file name or path to the file differs for different operating systems, you can identify the files names and paths in the **Advanced Log File Attribute Group Information** window. Each log file entry requires a unique label.

Part or all of the log file name can come from a runtime configuration property. Click **Insert Configuration Property** and select a configuration property to include part or all of the name.

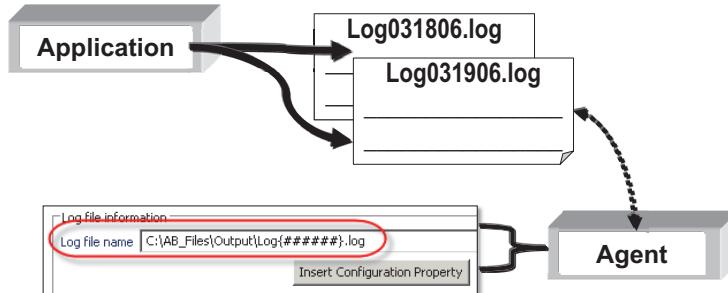


Note: Creating and managing custom runtime configuration properties is described in [Unit 6, Lesson 7](#) on page 6-65.

You can also use dynamic naming for file names that can change.

Dynamic file name support

- Application file names can change based on date, time, size, or certain application criteria.
- You can support this action by specifying a pattern in the **Log file name** field.



Dynamic file name support

For applications that create output files with dynamic file names, you can configure the **Log file name** field to match patterns. This feature accepts that files that are monitored by the agent do not always have a fixed and predictable file name. Many products and applications use a date or time stamp that they insert into the file names.

Dynamic file name examples

{#####}.abc	Matches numeric file names of eight characters and file extension .abc. Example: 11111118.abc or 11111119.abc
{#####??}.abc	Matches numeric file names of eight characters and ignores the last two positions in the file name. Example: 11111100.abc, 11111201.abc, 11111302.abc
IN{#####}.log	Matches file names starting with IN followed by six numeric characters and file extension .log. Example: IN111111.log, IN111112.log, IN111114.log
{YYYYMMDD}.log	Defines candidate files with names such as 20060930.log or 20061015.log.
MY{YYDDD}.log	Requests files with names such as MY05202.log, MY06010.log, or MY06350.log.

Dynamic file name examples

This slide shows dynamic file name examples.

You represent a dynamic number within the file with a hash (#) sign. The wildcard question mark (?) specifies to ignore the character. The asterisk (*) specifies to ignore the rest of the string. Use the letters Y (year), M (month), and D (day) to specify numbers that represent a date.

The file with the highest matching number is considered the current source.

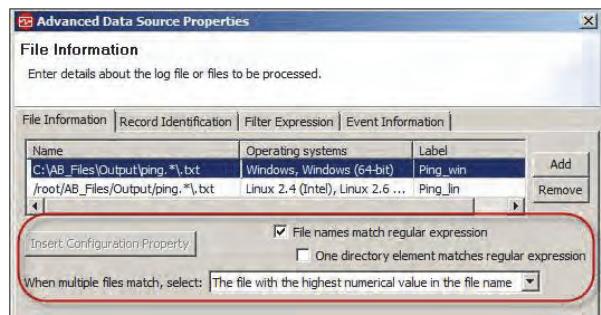
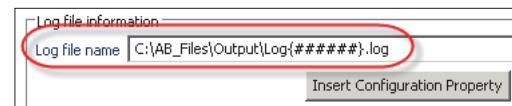
For more information, see the *IBM Agent Builder User's Guide*.

Regular expressions for file names

- Can set up monitoring of multiple log files and variable names
- Process
 - Enter regular expression
 - Select **File names match regular expression** in Advanced

Tip: Define regular expression in agent, not runtime configuration

- Windows: Use [...] instead of a backslash (\)
- Log File RegEx Statistics attribute group



Note: Use regular expressions to specify file name patterns.

Regular expressions for files names

File name matching supports regular expressions, which allows monitoring of multiple log files or log files with variable names.

To use regular expressions, enter the regular expression into the **Log file name** field. In Advanced, select **File names match regular expressions**.

The agent developer defines the regular expression and does not require an agent user to see it.

Do not use the backslash (\) on Windows hosts. The agent might make a mistake about what directory to look in for the matching files. Use ellipses in brackets [...] instead.

The Log File RegEx Statistics attribute group contains information that shows the statistics of the log file regular expression processing.

Regular expressions

- Regular expression: pattern that describes a string
- Syntax examples
 - Alternatives:
`gray|grey` gray or grey
`[aeiou]` any vowel
`[1-5]` numbers 1-5, `[a-m]` letters a-m
 - Grouping:
`gr(a|e)y` gray or grey
 - Quantification: ?, *, + after an element r
`color?r` matches both color and colour
`ab*c` matches ac, abc, abbc, abbbc, and so on
`ab+c` matches abc, abbc, abbbc, and so on, but not ac

Note: For more information on regular expression usage, see Agent Builder User's guide

Regular expressions

A *regular expression*, or *pattern*, is an expression that describes a set of strings. Here are a few examples of regular expression:

- **Alternation:** A vertical bar separates alternatives. For example:

`gray|grey` can match gray or grey.

- **Grouping:** Use parentheses to define the scope and precedence of the operators, among other uses. For example, equivalent patterns that both describe the set of *gray* and *grey* are as follows:

`gray|grey` and `gr(a|e)y`

- **Quantification:** A quantifier after a token, such as a character or group, specifies how often that preceding element is allowed to occur. The most common quantifiers are the question mark (?), the asterisk (*), and the plus sign (+).

? The question mark indicates that there is zero or one of the preceding element. For example, `color?r` matches both *color* and *colour*.

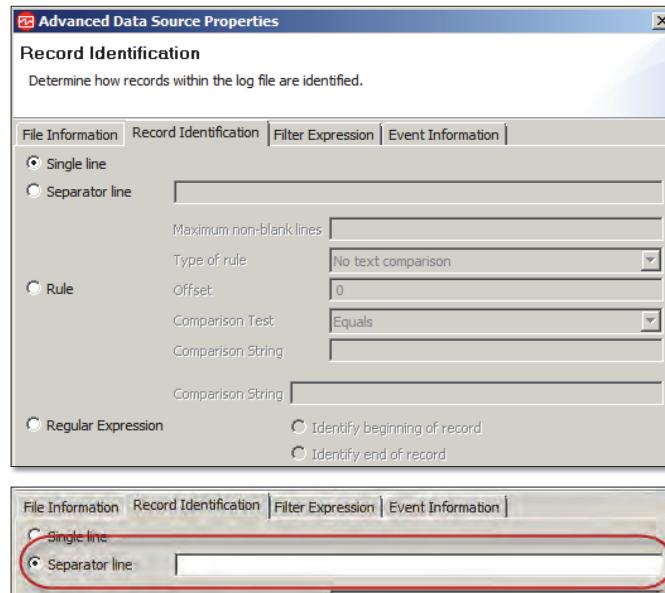
* The asterisk indicates that there are zero or more of the preceding element. For example, `ab*c` matches *ac*, *abc*, *abbc*, *abbbc*, and so on.

+ The plus sign indicates that there is one or more of the preceding element. For example, `ab+c` matches *abc*, *abbc*, *abbcc*, and so on, but not *ac*.

For more information about the syntax of individual filters, see the *IBM Agent Builder User's Guide*. Also, There are many examples and tutorials available on the internet regarding regular expressions.

Defining record identification

- Defines what is a record (row of data)
- Single line
 - Default option
 - Defined by line return
 - Every line is a record
- Separator line
 - Text found at the beginning of a line that indicates the end of the record.



Defining record identification

The first level of parsing you can define is a record. A record is one or more lines in the log file and must contain a full set, single row, of the data you want to gather with the agent.

You can define a record in four ways:

- Single line
- Ending pattern
- Rule
- Regular expression

Single line is the default option. With this setting, the agent treats each line of the log file as a record.

If you want to change the default configuration, select **Advanced** from the Log File Information window and click the **Record Identification** tab.

Use *Separator line* to identify a sequence of characters that indicate the end of a record. This option defines the record as the previous 100 lines or until the previous record. It then filters out any lines in the log file that do not match the pattern.

Defining record identification (continued)

- Rule
 - Maximum number of lines that make up a record
 - Type of rule
 - No text comparison
 - Identify beginning of record
 - Identify ending of record
 - Comparison offset, condition, and text
 - Nonmatching rows default to single line
- Regular Expression
 - Pattern that is used to indicate the beginning or end of a record

Maximum non-blank lines: 1
Type of rule: Identify beginning of record
Offset: 0
Comparison Test: Equals
Comparison String:
Rule

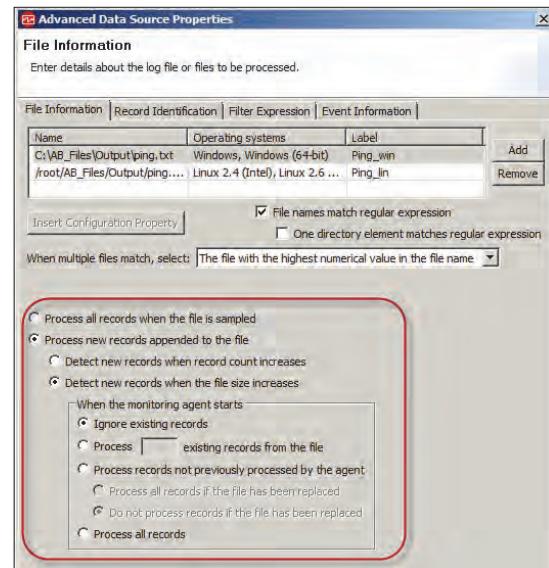
Comparison String:
Regular Expression Identify beginning of record
Identify end of record

Defining record identification rule

Select the *Rule* option if you want the agent to treat multiple lines in the log files as a single record. With Rule, you can identify a maximum number of lines that make up a record and a sequence of characters that indicate the beginning or end of a record. With Rule, the character sequence can occur at a specific offset within a line. You can also specify the beginning or end of a record where a particular character sequence does not occur (the != selection).

Defining how to process the log file

- Controls when and which records
- All records when sampled
- Only when new records added
 - Detect new records by
 - Record count
 - File size increase
 - When agent starts
 - Ignore existing records
 - Process x records
 - Process unprocessed
 - Process file if replaced
 - Do not process file if replaced
 - Process all records



Defining how to process the log file

You can control how the log file is processed in several ways. The default method is shown in the screen capture. The agent processes new records that are appended to the file based on an increase in the file size. Furthermore, when the agent starts, it ignores existing records.

Process all records when the file is sampled enables processing of all records in the entire file every time a situation runs against the data source or a sample is taken. This process reports the same records every time unless they are removed from the file. This selection does not produce event data when new records are written to the file. To process correctly, a file must have at least two records.

Process new records appended to the file enables processing of new records that are appended to the file while the agent is running. It produces an event record for every record added to the file. If the file is replaced, the first record changes in any way, it processes the entire file, and produces an event for each record in the file.

If you choose to process new records that are appended to the file, you can also select how to detect new records. **Detect new records when record count increases** enables detecting new records when the number of records in the file increases, whether the size of the file changes. This selection is useful when an entire log file is preallocated before any records are written to the file. You can select this option for files that are not preallocated, but it is less efficient than monitoring the size of the file.

If you choose to detect new records based on increases in the size of the file, you can also select 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.

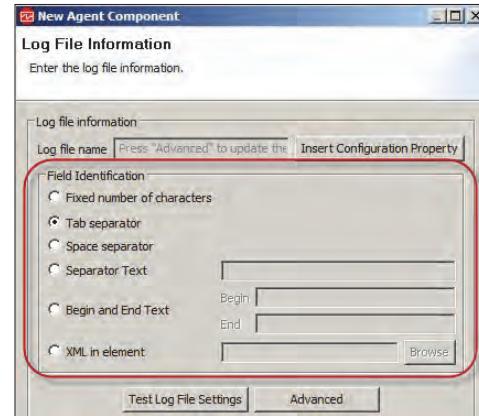
Process ____ existing records from the file enables production of events for a fixed number of records from the end of the file at the time the agent starts.

Process records not previously processed by the agent enables the monitoring agent to maintain restart data so that the agent knows which records it 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 ran. This option involves a little extra processor and disk usage each time a record is added to the file.

If you choose to process records that the agent has not previously processed, you can select what to do when the agent starts and the existing file is replaced. **Process all records if the file has been replaced** enables the production of events for all records in the file if the current information about the monitored file and the information that is stored in the restart data do not match. Examples of mismatches include the file name or file creation time changes, the file size decreases, and the file last modification time is earlier than it was. **Do not process records if the file has been replaced** disables processing of any existing records in the file if the current monitored file does not match the monitored file that is stored in the restart.

Defining global field identification

- Field identification defines how to identify and separate attributes from one another
 - Global and per attribute
- Global field identification defines the general identifier if no attribute identifier is defined
- Generally, it is the identifier that works for most attributes
- If one or more fields, need a unique identifier, use per attribute field identifiers



Defining global field identification

Field identification enables your agent to identify which data in the log file is a specific attribute to retrieve. You can set these parameters globally or for each attribute.

Global field identification defines the default identifier if no identifier is specified per attribute. Generally, it is the identifier that works for the most attributes. Field identifiers that are set for individual attributes override global field identifiers.

The ways in which a data field can be identified are defined here:

- **Fixed number of characters** indicates that a fixed number of characters separates the attributes.
- **Tab separator** indicates that tabs separate the attributes.
- **Separator Text type** indicates that a common text delimiter, such as a comma, semicolon, or blank space separates the attributes. Type the separator in the field.
- **Begin and End Text** indicates that specific text at the beginning and end of the data separates the fields of data, or attributes. Enter both **Begin** and **End** text.

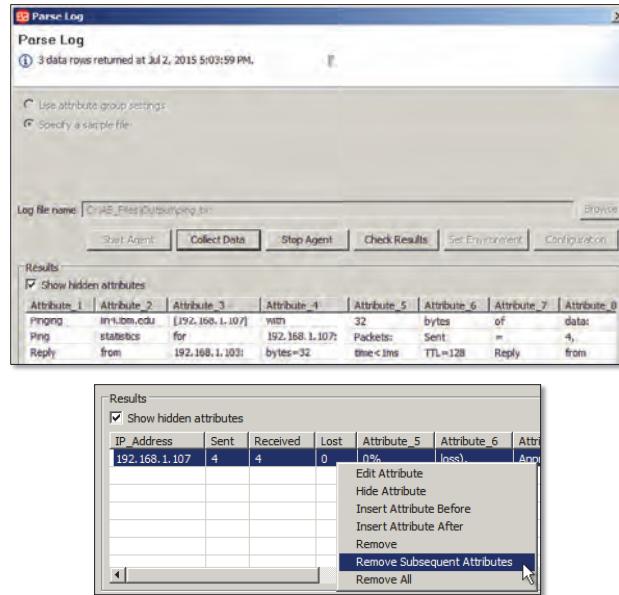


Note: Unless you click **Advanced** and enter the information in that window, the information that you provide here in the Log File Information windows makes the following assumptions:

- It monitors only one log file at a time.
- Each line of the log file contains all the fields necessary to fill the attributes to define.
- Each attribute is separated from the others in the same way.

Testing log file settings

- Use to test your log file settings and its output before installing the agent
- Browse to locate a sample log file
 - Can have different location and file name from the actual target file
- Define attributes to match results
 - Can keep attribute definitions
 - Cancel to not keep attribute definitions
- You can edit, add, remove, and hide attributes
 - Reparse to see new results



Testing log file settings

After you select the options for the log source, you can test your log file settings and their parsing ability.

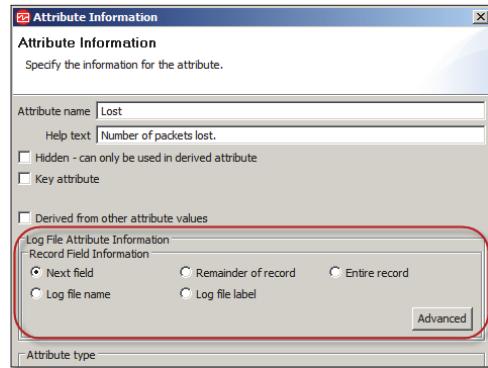
1. Click **Test Log File Settings** to open the Parse Log window.
2. Click **Browse** to select a sample log file, or type the name of the file in the text box.
3. Click **Start Agent** and **Collect Data** to parse the log file based on the current settings. The Agent Builder parses the specified log file and creates a default set of attributes.

You can modify attributes by either clicking the column heading or by right-clicking in the column and selecting **Edit Attribute**. Some attribute settings affect the log file parsing by setting more specific field identification. You can also insert, remove, and hide attributes by right-clicking the appropriate column and selecting the action from the menu. A hidden attribute is not shown in the management console. However, you can use the value in a derived attribute.

When you change an attribute, the data becomes unusable. The results are cleared from the table, and you must parse the file again to see how the changes affect the presentation of the data. When you are finished editing, click **OK** to exit and save your attribute changes. Click **Cancel** to exit without saving.

Defining basic attribute field identification

- Defining per attribute field identifiers requires the attributes be defined
- You can manually define them
- The Test Command feature automatically defines attributes based on how it parses the log file



Defining basic attribute field identification

Attribute field identification is done only after the attributes are defined. The Test area in Agent Builder automatically defines attributes. You can also create them manually.



Note: Definitions for standard attribute fields and attribute types are taught in the first lesson in this unit and are not described here.

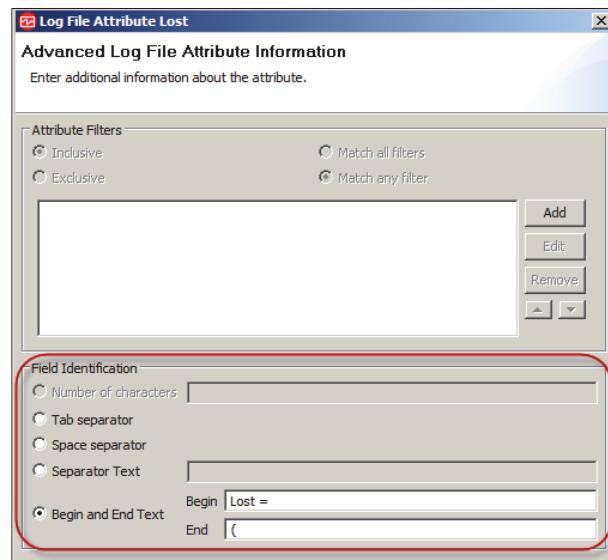
In addition to the fields that are applicable to all of the data sources, the Attribute Information window for the Log File data source has the following fields in the **Record Field Information** area that determine what data goes in the attribute:

- 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 box. This option is the default for all attributes, except for the last field, if automatically generated in the Test area.
- Remainder of record** shows the rest of the record after it parses previous attributes. This attribute is the last one where the attributes are automatically defined in the Test area, except for possibly the log file name or log file label.
- Entire record** shows the entire record. This situation is the only attribute, except for attributes that are not parsed from the log file, such as the log file name, log file label, or a derived attribute.
- 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 pane.

Defining advanced attribute field identification

Advanced

- Field Identification:
- Defines the delimiters for this specific attribute
- Overrides global delimiter



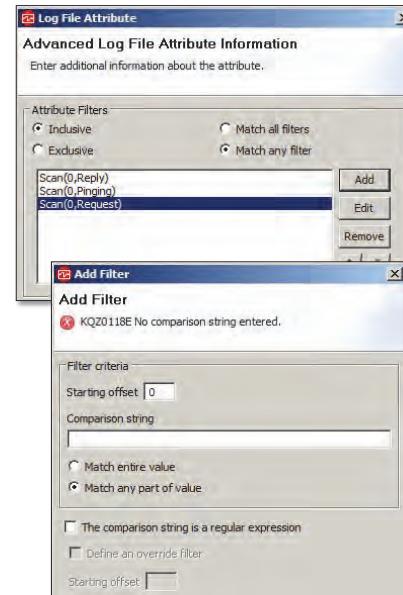
Defining advanced attribute field identification

Field Identification indicates how to override the global, attribute group, field delimiters for this one attribute only. The **Field Identification** parameters are the same as they are for global field identifiers and are previously defined.

Use attribute filters to select or reject records within a log file, not to identify what data goes in an attribute. They are described in ["Defining record identification"](#) on page 6-51.

Filtering records with attribute filters

- Used to include or exclude records based on matching attribute values
- Attribute filters
 - Identify field values to match
 - Inclusive setting keeps records that match and excludes those that do not match
 - Exclusive setting excludes records that match and keeps records that do not match
 - If more than one filter set, choose if all or only one filter must succeed



Filtering records with attribute filters

Sometimes the global record identification that you define does not capture all the records that you want or does not exclude records that you do *not* want. Often you can use attribute filters to include records you want or to exclude records you do not want.

- Inclusive* indicates that if the attribute filter succeeds, keep the record.
- Exclusive* indicates that if the attribute filter succeeds, do not keep the record.

If more than one filter is added to any specific attribute, you can select whether all filters must succeed or any one filter must succeed for the attribute filter set as a whole to succeed. If the filter set does not succeed, the log file record is not retrieved.



Note: You must define attributes before you can use them to filter records.

Attribute filtering with regular expressions

- Regular expressions allowed in attribute filters
 - Primarily for log files, but available to all data sources
 - Uses more processor cycles
- Process
 - Enter regular expression
 - Select regular expression
 - If needed, enter replacement value

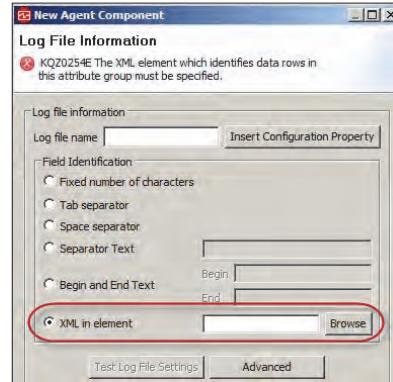


Attribute filtering with regular expressions

Attribute filters accept regular expressions. An attribute filter that uses a regular expression uses more processor cycles than an attribute filter that does not.

Monitoring XML files

- Eliminates the need to create begin/end tags for each element.
- Eliminates line restrictions
- Browser reads document elements to allow the file to be parsed
- Runtime also handles XML elements or whole documents so that users can parse documents (data) or a file of elements (usually representing events).
- Both copy and tail modes are supported, just like other log file parsing
 - Note: Many XML files are not really legal XML documents. They are just a series of elements (Java XML parser declares this practice illegal).



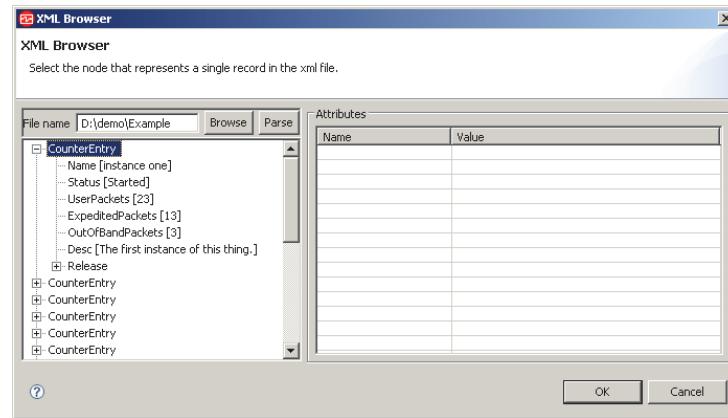
Monitoring XML files

The Log File data collector can monitor XML files and provides parsing instrumentation. This function eliminates the need to create begin and end tags for each element and line restrictions.

A browser reads document elements to parse the file.

XML Browser

- Click Browse on XML in element line.
- Select a file name to parse
- Choose the XML element that defines the record
- Attributes come from the XML elements within the chosen element.



Monitoring custom data sources

57

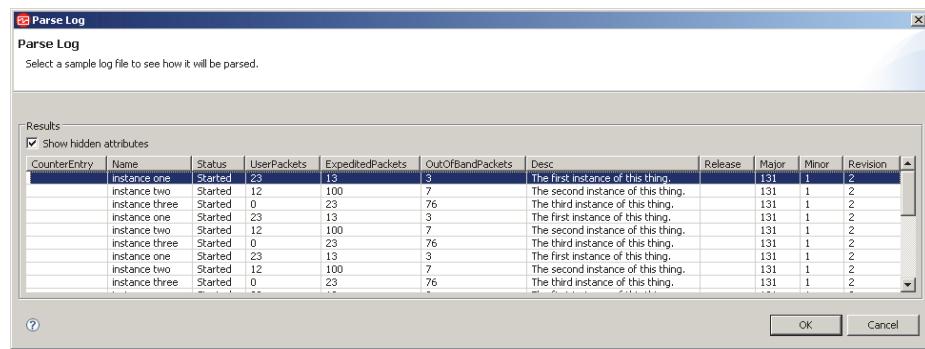
© Copyright IBM Corporation 2017

XML Browser

The XML browser identifies all possible attributes of the record by looking at the child tags and their attributes.

Testing XML log file parsing

- Select **Test Log File Settings** to view the results and populate attributes from the elements contained in the selected XML element.
- Edit, delete, or hide attributes as appropriate



Testing XML log file parsing

Use the Test facility to confirm that the XML parsing works as you expect.

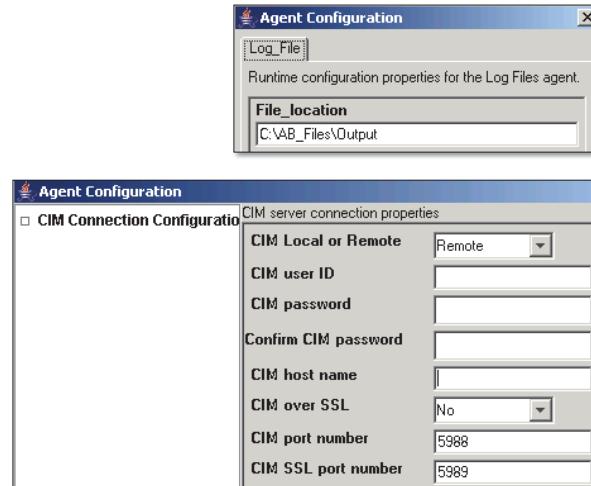
Lesson 7 Custom runtime configuration

IBM Training

IBM

Lesson 7: Custom runtime configuration

- Runtime configuration
 - Agent configuration that can be set at agent installation or configuration
 - Might require values, might provide defaults
- Some data sources allow custom runtime
 - Described in this unit
- Some data sources provide required runtime configuration
 - Described in next unit



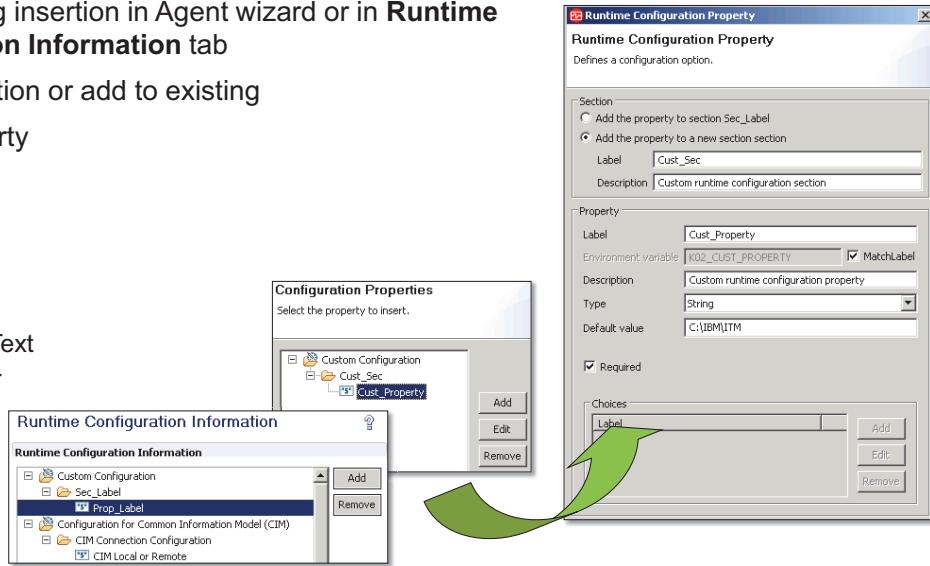
This lesson teaches you how to add custom runtime configuration properties. After completing this lesson, you can create custom runtime configuration properties.

Runtime configuration is agent configuration that you can set at agent installation or configuration. With it, you customize the agent for the specific host. You might set certain values, some of which might be required or have default values that you can modify.

With some data sources, you can create custom runtime configurations. Some data sources provide the necessary runtime configuration that is needed to configure the agent. This unit describes custom runtime configuration. The next unit describes data source provided runtime configuration.

Creating custom runtime configuration

- Create during insertion in Agent wizard or in **Runtime Configuration Information** tab
 - Create a section or add to existing
 - Define property
- Types:
- String
 - Password
 - Numeric
 - Choice
 - Read Only Text
 - File Browser



Creating custom runtime configuration

You can create the properties at the point of insertion by accessing the Configuration Properties window with the **Insert Property** or the **Insert Configuration Property** button. You can also create them in the **Runtime Configuration Information** tab in the Agent Editor window after initially defining your agent with the Agent wizard.

From these windows, you can add, remove, and edit the agent's runtime configuration properties.

Place each property in a section. In the **Section** area, select to add the property to the current section or to create a section. If new, complete the following fields:

- **Label:** Text that describes the properties in this section.
- **Description:** Description of the properties in this section.

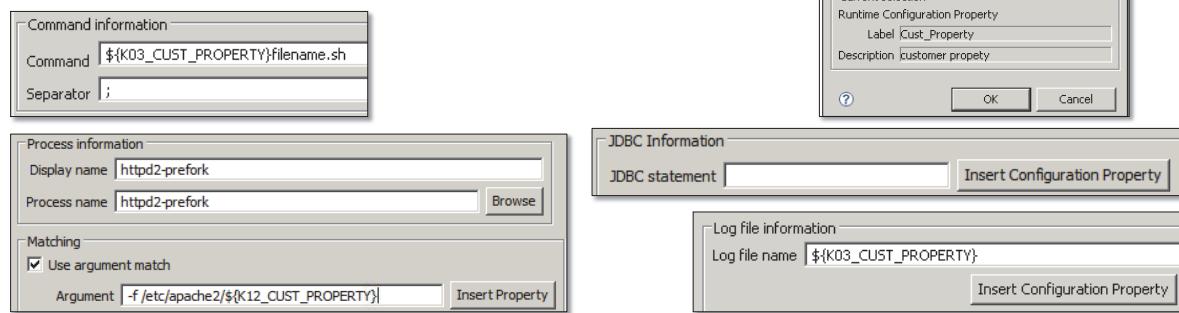
In the **Property** area, complete the following fields:

- **Label:** Text that the agent configuration pane shows that identifies the information you enter.
- **Environment variable:** The environment variable is shown 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. You can clear the **Match Label** check box if you want to change the environment variable independently from the label.
- **Description:** Description of the property that you are defining.
- **Type:** Type of information that is collected, one of the following options:
 - **String:** For any alphabetic information that you need to collect, 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, asterisks obscure the data that is typed into the text box. In addition, you must 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 entered values are shown in the agent configuration pane as a group of radio buttons that allow only one selection from the group.
- **Read Only Text:** Shows text when configuring the agent, but no information is collected.
- **Separator:** Shows a horizontal separator, but no information is collected.
- **File Browser:** Collects a string, which is a file name or directory. Click **Browse** to browse the file system for the file or directory.
- **Default value:** A default value to use unless changed during agent configuration.
- **Required:** Verify that the user enters a value when the agent is configured, or clears the field if entering a value is optional.

Inserting runtime configuration variables

- You can use custom configuration properties when performing these tasks:
 - 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
- Click **Insert** or enter property value



Inserting runtime configuration variables

You can use custom runtime configuration properties in any of the instances that this slide lists. For each type, you can manually enter the variable name, as shown in the slide. For process matching and log file names, you can select the property directly from the Configuration Properties window by clicking either **Insert Property** or **Insert Configuration Property**.

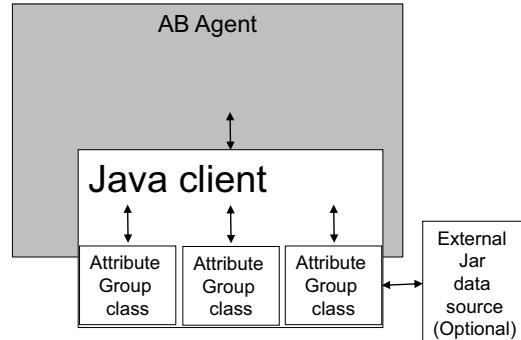
Lesson 8 Java API

IBM Training

IBM

Lesson 8: Java API

- With Agent Builder, you can develop a custom Java application to collect your monitoring data, called the Java client
- Working sample code is generated for you
 - Replace or edit sample code to generate the monitored data
- Collects data or sends events
- Agent starts and stops the Java client for you



This lesson teaches you how to integrate an Agent Builder agent with a Java application that gathers custom data to monitor. After completing this lesson, you are able to add monitoring by a custom Java application.

Use the Java API data source and the Java programming language to collect data that you cannot collect by using other Agent Builder data sources. The agent starts the Java application and sends a shutdown request when it is time to shut down. The Java application must exit only when it is requested to do so.

Process overview

1. Define the attribute group
2. Create one attribute
3. Define the main Java class and jar name
4. Define error codes
5. Add supplemental files
6. Define and finalize extra attributes
7. Create the attribute group java class
8. Create custom Java code to gather data, send and receive messages, perform Take Actions

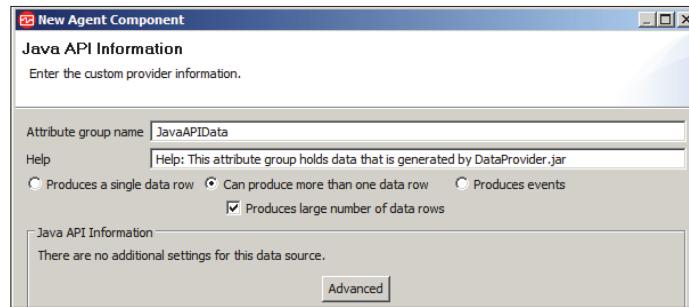


Process overview

This slide gives an overview of the process for adding a Java API data source.

Add an attribute group

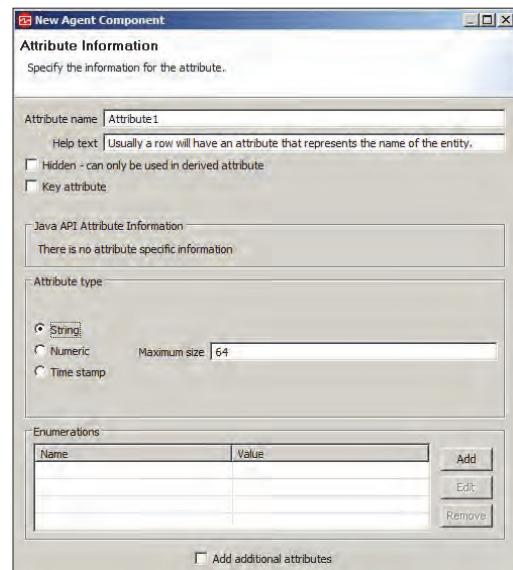
- Each attribute group must be manually created
- Attribute group
 - Sampled (single data row or more than one)
 - Event
- Attribute group class contains sample code to generate sample data for each attribute



Add an attribute group

Add an attribute to the attribute group

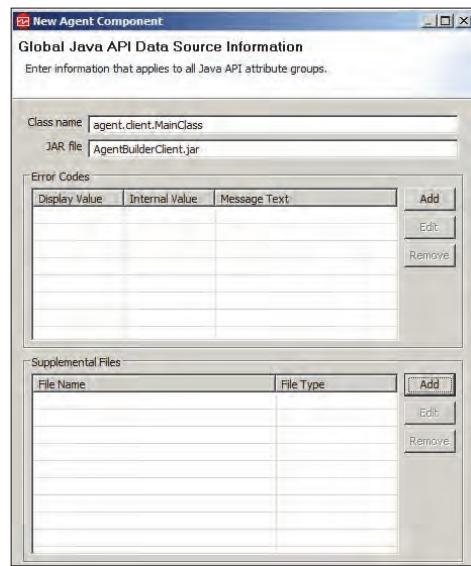
- You must manually create each attribute
- You can create attributes in the wizard or after (no browser)



Add an attribute to the attribute group

Define the client process

- Define the class name that contains the main method
- Define the JAR file that contains your classes
- Define error codes for Performance Object Status table
- Define extra files needed by your Java client



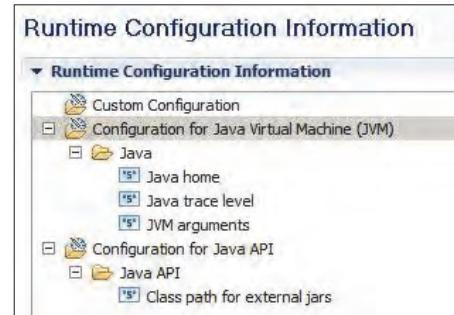
Define the client process

The class name is a fully qualified class name whose main method Java calls on when starting. 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. You package and install the JAR file with the agent.

Define the runtime configuration

- Multiple data sources use Java
- Common configuration settings Java Home
 - JVM arguments
 - Extra JAR files required
 - Trace Level
- Custom Java API configuration settings
 - Values accessed through special method



Define the configuration

Multiple data sources share the Java virtual machine the agent uses. Common configuration settings control the Java virtual machine.

- Java home
- JVM arguments
- Extra .jar files required

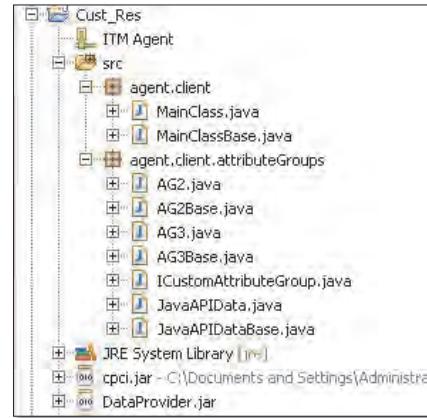
Any .jar files that you add to the agent's supplemental files are automatically included in the class path.

- Trace Level: The trace level label in the configuration corresponds to the following Java Logging trace levels:
 - Error: SEVERE
 - Warning: WARNING
 - Informational: INFO
 - Minimum Debug: Fine
 - Medium Debug: Finer
 - Maximum Debug: Finest

The Java API has custom configuration settings that the client needs to function. Access these values through the AgentConnection getConfigurationProperty("<config_prop_name>") method.

Java API generated client

- Agent Builder creates a working Java client
- Eclipse JDT was added to Agent Builder to support Java code development
- JDT Java nature added to Agent Builder agent projects that contain a Java API client
- Source files written to src folder, class files compiled to bin folder (standard JDT practices)
- New client created when the agent is first saved or when the first Java API attribute group is added and the agent is saved
- Client is customized for the attribute groups and attributes that are defined

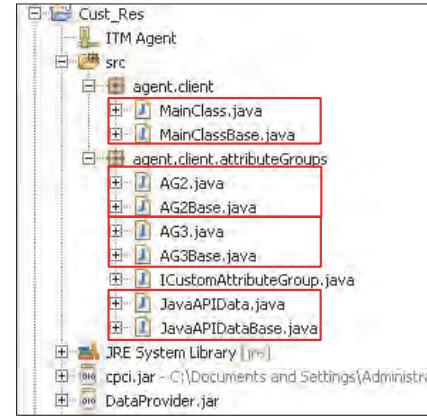


Java API-generated client

When you create an agent with one or more Java API data sources, the Agent Builder generates Java application source code. The agent project generates the code 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 post 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, that is replaced with data that is obtained from the resources you want to monitor.

Java API-generated client structure

- Classes come in pairs
 - Main pair for the client
 - Pair for each attribute group
- A single interface, which should not be modified, is also available
- One of each pair is the *Base*
 - Should not be modified by user
 - Overwritten every time agent is saved
- You create or customize the other of each pair
 - Accesses important resources, generates data, takes actions
 - Inherits from corresponding base
 - Sample written by Agent Builder only if does not exist
 - You can regenerate original by deleting file and saving agent



Java API-generated client structure

Java classes come in pairs: the main pair for the client and a pair of classes for each Java API attribute group. Also, a single interface class is included which you do not modify.

In each pair of files, there is a *helper* or *base* file with a class name that ends in **Base**. Do not modify this file. It is overwritten every time the agent is saved, so any user changes are overwritten and lost.

The other class is intended for the agent developer to modify to access resources important to the agent and complete monitoring actions such as gathering data and taking actions. This Java class inherits from the corresponding base file. It is written by Agent Builder only if the file does not exist. You can regenerate an original at any time by deleting the existing copy and saving the agent.

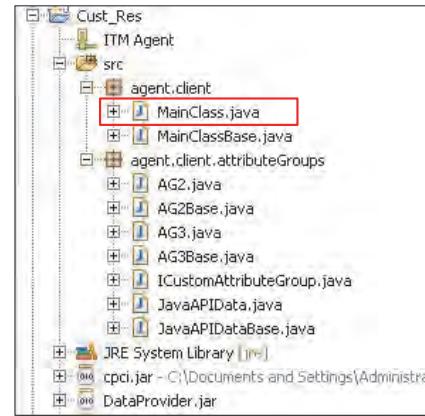
Java API generated main class

- Inherits from helper class

```
public class MainClass extends MainClassBase {
```
- Contains Java client main method that creates the main object and enters request handling loop

```
public static void main(String[] arguments) {
    MainClass client = new MainClass(arguments);

    client.acceptRequests();
}
```
- Constructor for initializing access to client resources
- Methods
 - stopDataCollection
 - takeAction
 - handleActionException



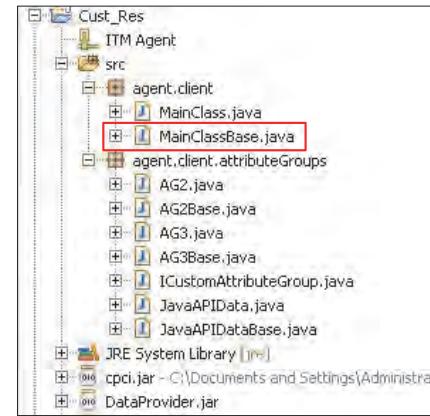
Java API-generated main class

Methods that are run in the main class:

- **stopDataCollection** method to gracefully end access to client resources
- **takeAction** method handles action requests
 - Must add code to complete the action
 - Send return code `getAgentConnection().sendActionReturnCode(0, request.getID());`
 - Should not catch Exception, but handle exceptions in `handleActionException` method
- **handleActionException** method handles exceptions that are thrown from `takeAction` method
 - Any API exception, which is derived from `CpciException`, handled by base class, and not passed to this class

Java API generated main base class

- Contains details of dealing with Java API
- Constructor
 - Initializes a Java logger to be used in all classes
 - Creates agentConnection object
 - Registers all attribute groups with the agent
 - Registers product code as take action prefix
- Methods
 - registerActionPrefix
 - isMultiThreaded
 - acceptRequests



Java API-generated main base class

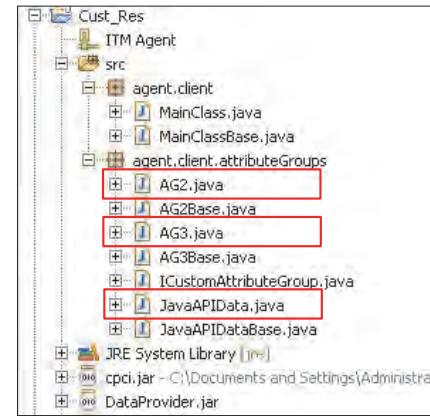
Methods that are run in the main base class override the **registerActionPrefix** method in the main class to register different prefixes or no prefix.

The **isMultiThreaded** method default is false, which means that the agent sends one request at a time and waits for response before sending the next request. Override it to return **true**, which means requests are sent to the agent as they come in.

acceptRequests is a long-running loop to wait for requests and call appropriate methods to handle.

Java API attribute group classes

- Appends attributeGroups to main package name
- Fills in Attributes object
 - Defined in base class
 - Contains typed fields to match defined attributes
- Methods
 - collectData
 - setAttributeValues
 - sendData method
 - handleException
 - stopDataCollection
- Generated code different for event attribute groups and attribute groups in subnode



Java API attribute group classes

Methods that run in attribute group classes are as follows:

- collectData method returns data for rows in the sample set
- Collects each row in a buffer, setAttributeValues method, and then sends all rows to the agent, sendData method
- handleException method to handle exception during data collection, except CpicException, handled by base class
- stopDataCollection method for attribute group-specific cleanup

Generated code is different for event attribute groups and attribute groups in a subnode.

Sample edit to collectData section

For example, change:

```
public void collectData(Request request) throws Exception {
    final String METHOD = "collectData";
    logger.entering(CLASS, METHOD, request);
    AgentConnection agentConnection = mainClient.getAgentConnection();
    Attributes row;
    // This example sends 3 data rows with arbitrary values.
    // Replace the rows with data from the monitored resource.
    for (int rowNum = 1; rowNum <= 3; ++rowNum) {
        row = new Attributes(
            "arg" + rowNum,
            "arg" + rowNum,
            10 + rowNum,
            Calendar.getInstance()
        );
    }
}
```

To:

```
row = new Attributes(
    DataProvider.getStringValue ("Name"),
    DataProvider.getIntValue("Integer_Value"),
    DataProvider.getStringValue ("String_Value"),
    DataProvider.getFloatValue("Float_Value")
);
```

Sample edit to collectData section

Edit the attribute group Java source to call the appropriate methods to gather the data. Normally this process involves updating the collectData() method.

Java API generated code changes

- Code rework necessary if you make the following changes:
 - Main class name
 - Add, remove, or rename a Java API attribute group
 - Add, remove, rename, or reorder attribute in Java API group
- Updated helper base classes are rewritten when agent is saved
- Modifiable classes are not changed and might no longer match the structure in their base classes
- Results in many compiler errors that must be fixed to match new agent structure

Java API tips and tricks

- Define all attributes within an attribute group before customizing the attribute group class
- One Java API client per agent: One main class and one or more attribute group classes
- Error codes defined by the agent are coded in the main base class so that you can reference them symbolically (NO_ERROR and GENERAL_ERROR are also there)
- If you need external .jar files to compile code, you must add them to the build path manually

Java API tips and tricks

To add the external .jar files so that the code compiles in Eclipse, you must add the .jar files to the build path manually.

1. Select the project.
2. Right-click the project, and select **Properties > Java Build Path > Libraries** tab.
3. Select one of the following options:
 - Add the .jar file from the script folder. If you add the .jar file as a supplemental file, it is placed in the script folder.
 - Add external .jar files to search the user's file system.

Java API client lifecycle

- The Java client process is started by the agent
- Single Java client to support all Java API attribute groups in the agent
- The client is told to shut down using a shutdown request
 - The client should terminate when this occurs
 - The client should also terminate if it fails to connect to and register its attribute groups with the agent
- After successfully registering attribute groups with the agent, the agent restarts the Java client if it detects it has failed

Student exercises



Exercise 3: Create an agent to monitor a log file and a custom Java data source

Exercise 4: Install and confirm the updated AB1 agent

Student exercises

Open your Course Exercises book and complete the exercises for this unit.

Summary

Now that you have completed this unit, you should be able to perform the following tasks:

- Add local and remote monitoring of script output to an agent
- Add local and remote monitoring through a socket connection
- Add monitoring of log files to an agent
- Add monitoring by a custom Java application
- Create custom attributes and runtime configuration properties
- Generate agent installer files from the CLI

Unit 7 Monitoring remote and optional resources

IBM Training



Monitoring remote and optional resources

© Copyright IBM Corporation 2017
Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

This unit teaches you how to use subnodes in your agent. You can use subnodes to enable a single installed agent to monitor multiple remote resources. You can also use them to create an agent with optional data sources that you can activate or not activate at run time. This unit also introduces several new data sources, including Simple Network Management Protocol (SNMP), HTTP server URLs, ICMP (Ping), Java Management Extension (JMX), Common Information Model (CIM), and Java Database Connectivity (JDBC). Each of these data sources requires runtime configuration, which you must refine for your agent.

Scenario 1 business objectives

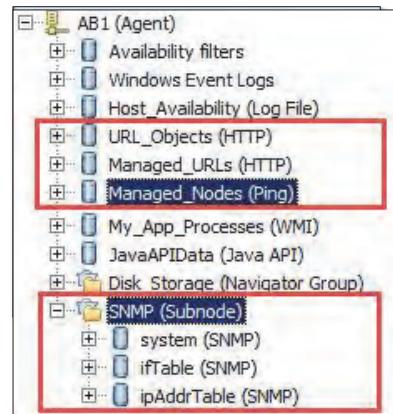
- Company has several optional resources that need to be monitored remotely
 - SNMP-managed devices
 - Device network availability
 - Web server URLs
- Create an agent that when installed on the agent host remotely monitors multiple instances of each resource
- Create an agent where optional data sources can be enabled when configuring the agent

Scenario 1 business objectives

The training in the first half of this unit focuses on the knowledge and skills that you need to meet this scenario.

Scenario 1 solution

- Single agent dedicated to monitoring resources on remote servers
- Subnode pulling identical SNMP data from a configurable number of SNMP hosts
- ICMP Ping data source testing the network availability of a large volume of network devices
- HTTP URL data source monitoring the availability of many URLs and embedded objects



Scenario 1 solution

In this solution, you create an agent that monitors multiple hosts at the same time. It pings a list of hosts to confirm their availability. It tests multiple URLs from multiple HTTP servers to confirm their availability. It uses subnodes to gather the same network data from multiple servers.

Scenario 2 business objectives

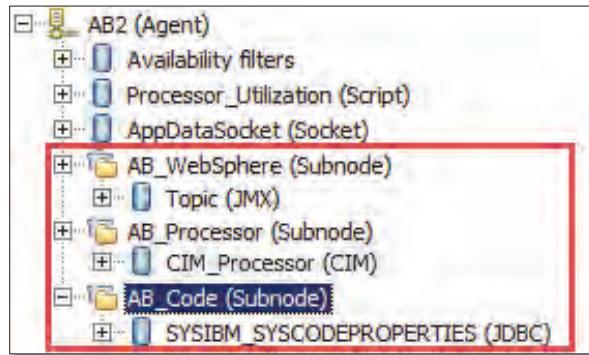
- Company has several resources that need to be monitored
 - Processes running on application servers
 - Applications running on Java EE application servers
WebSphere Community Edition, running on Windows and Linux
 - Data in a database running on application servers
- Not all application servers provide all of these services
- Create an agent that when installed and configured monitors the resources available on the specific server

Scenario 2 business objectives

The training in the second half of this unit focuses on the knowledge and skills that you need to meet this scenario.

Scenario 2 solution

- Single agent using three subnodes, one for each optional set of data points
- Subnode data sources
 - CIM data source pulling memory information from a host
 - JMX data source pulling information on Java EE applications running on an application server
 - JDBC data source pulling user information from a database



Scenario 2 solution

For this solution, you place each optional set of monitored data sources into its own subnode. With this action, you decide whether to create the subnodes when you configure the agent.

The three subnodes monitor the following data:

- CIM data source pulls memory information from a host
- JMX data source pulls information on Java Platform, Enterprise Edition applications on an application server
- JDBC data source pulls user information from a database

Learning objectives

After completing this unit, you should be able to perform the following tasks:

- Monitor remote data sources
- Add optional data sources to an agent
- Manage runtime configuration parameters
- Add monitoring of Simple Network Management Protocol (SNMP) to an agent
- Add monitoring of HTTP server URLs to an agent
- Add monitoring of network availability to an agent
- Add monitoring of Java Management Extension (JMX) data sources to an agent
- Add monitoring of Common Information Model (CIM) data sources to an agent
- Add monitor of a database with the JDBC data source to an agent

Learning objectives

The learning objectives include the skills that you need to implement the scenario solution.

Unit outline

- Lesson 1: Data source-provided runtime configuration
- Lesson 2: Subnodes
- Lesson 3: Monitoring remote systems
- Lesson 4: Monitoring Simple Network Management Protocol
- Lesson 5: Monitoring network availability with ping
- Lesson 6: Monitoring HTTP URLs and objects
- **Exercise 1: Remotely monitoring many resources**
- **Exercise 2: Install and confirm the updated AB1 agent**
- Lesson 7: Monitoring with the Common Information Model
- Lesson 8: Monitoring application servers with Java Management Extension
- Lesson 9: Monitoring databases with JDBC
- **Exercise 3: Allowing data sources to be optional**
- **Exercise 4: Install and confirm the updated AB2 agent**

This slide gives you an outline of the lesson material in this unit.

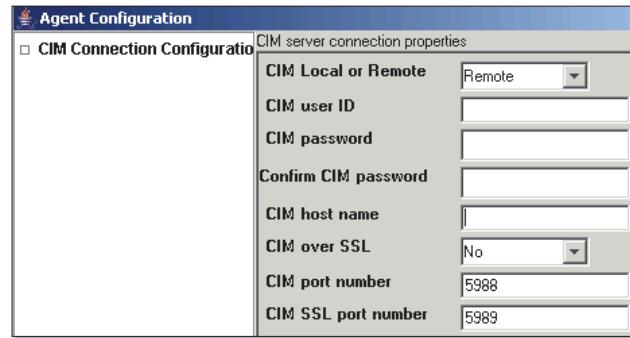
Lesson 1 Data source-provided runtime configuration

IBM Training

IBM

Lesson 1: Data source-provided runtime configuration

- These data sources require runtime configuration
 - CIM, SNMP, JMX, script SSH, socket, Java API, HTTP, ICMP, Ping, SOAP
 - Monitoring of remote data sources
 - Subnodes and multiple-instance data sources
- You can perform the following tasks:
 - Remove unnecessary parameters
 - Require parameters
 - Set default values



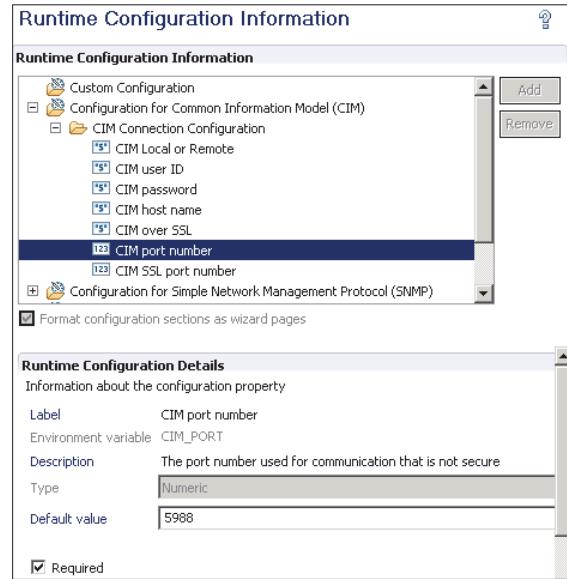
Data sources that require runtime configuration automatically identify all of the possible properties that you can set at run time. These data sources provide defaults for many properties. You can modify this list to identify exactly which parameters your agent offers and what default values you want the agent to have.

This lesson introduces data source-specific runtime configuration. After completing this lesson, you should be able to manage runtime configuration parameters.

Editing runtime configuration

Actions you can take in Agent Builder

- Create custom properties
- Turn on or off runtime configuration
- Format as wizard pages or not
- Add or remove properties
- Add, remove, or change default property values
- Make property required



Editing runtime configuration

This slide lists all the actions that you can take when you establish runtime configuration parameters in your agent.

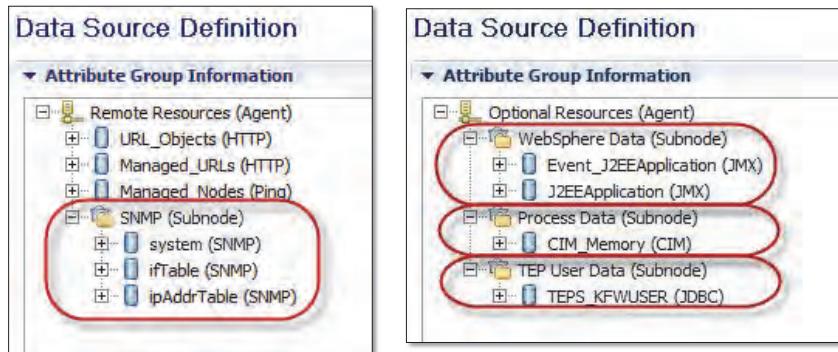
Lesson 2 Subnodes

IBM Training

IBM

Lesson 2: Subnodes

- Template of target data sources with the agent
- Applied at agent runtime to one or more targets
- Does not have to be applied, making monitoring of the included data sources optional
- Each getting its own runtime configuration properties

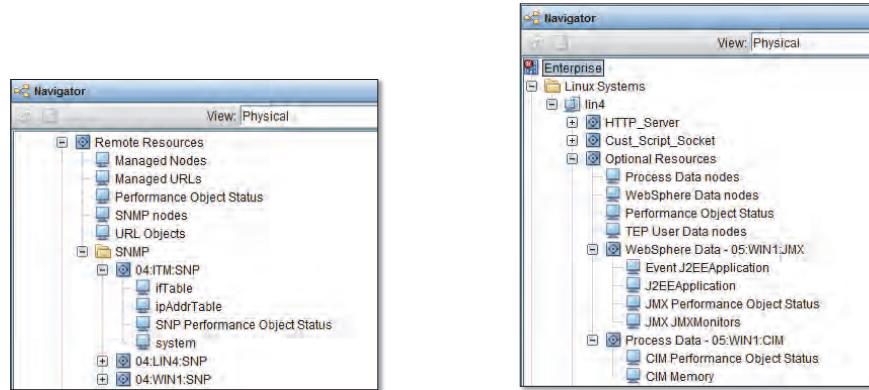


This lesson introduces you to subnodes. You use subnodes in later lessons to monitor multiple remote hosts from one agent. You also use subnodes to create agents with optional data sources that are enabled at run time.

After completing this lesson, you should be able to define a subnode and how it can be used to add optional data sources to an agent to monitor local and remote data sources,

Usage styles

- Monitor same data from multiple entities
 - Entities are typically remote hosts
 - Using multiple subnodes can monitor different data sources
- Monitor different data
 - Subnodes can represent optional data sources

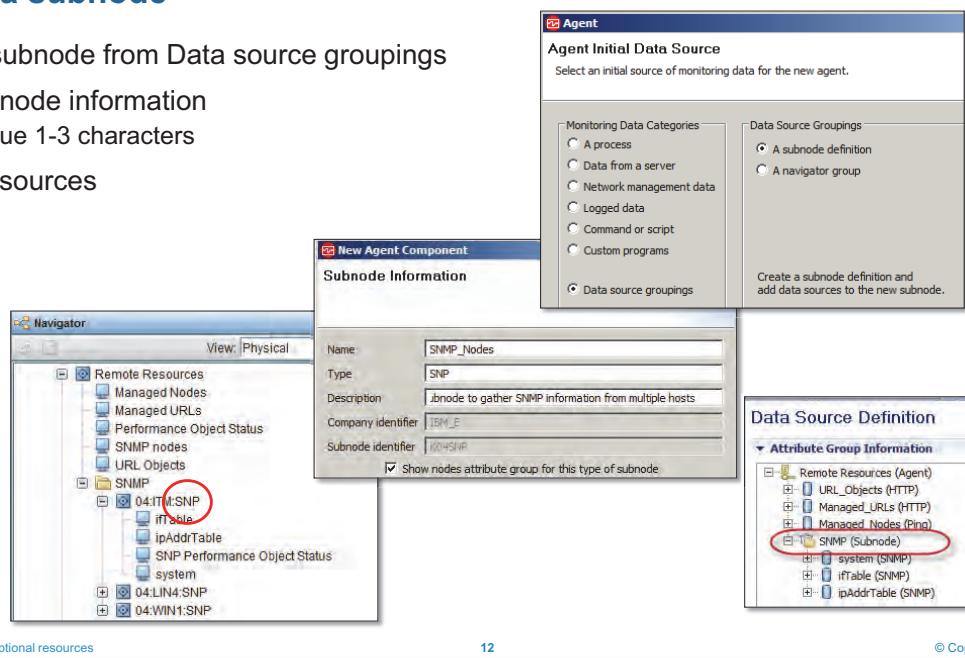


Usage styles

A single subnode can have several data sources that can be gathered from many remote hosts. Adding more subnodes enables the same agent to gather different data than the first subnode from different remote hosts.

Creating a subnode

- Select A subnode from Data source groupings
- Enter subnode information
Type: Unique 1-3 characters
- Add data sources



Monitoring remote and optional resources

12

© Copyright IBM Corporation 2017

Creating a subnode

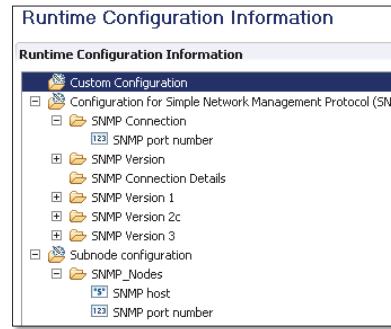
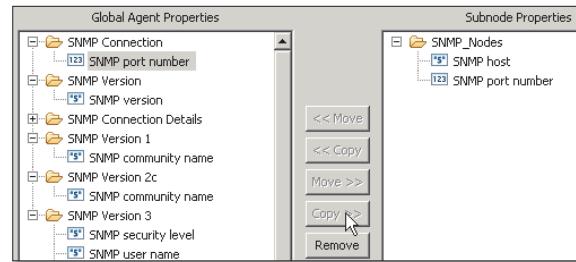
Similar to the Navigator group, you can define a subnode before or after you define the data sources that are part of it.

A subnode requires three properties: name, type, and description. The **Name** is the name of the subnode you are creating. **Type** is a unique set of 1 to 3 characters (with numbers, letters, or both) that identifies the type of the subnode you are creating. **Description** is the description for the subnode you are creating.

After the subnode is defined, you can add data source by defining new ones, moving existing data sources into the subnode, or by copying existing data sources into the subnode.

Setting subnode runtime configuration

- When the Windows data sources are added to subnode, Global Options automatically is selected
- Determine whether properties are global or subnode
 - Global are entered once for each agent and apply to all subnodes
 - Subnode properties are set for each subnode
 - Subnode properties can be both global and subnode (Copy) or only subnode (Move)



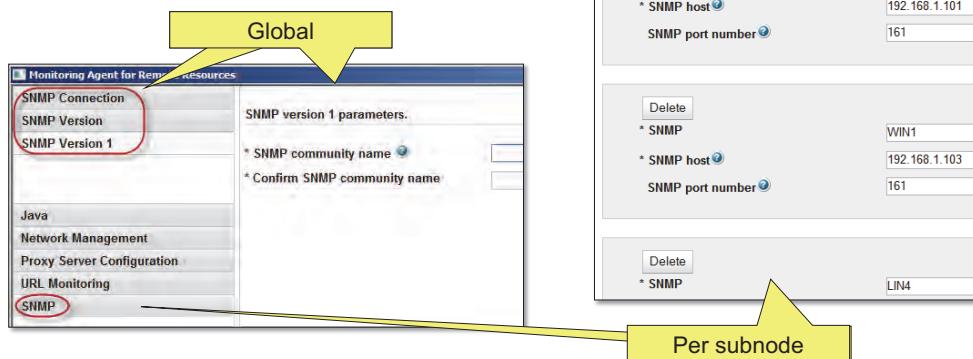
Setting subnode runtime configuration

If you add the Windows data sources (WMI, Perfmon, and Windows Event Logs), Global Options are automatically enabled, adding their configuration parameters to runtime configuration.

Besides the standard actions, you can configure in the agent with subnodes that you need to determine whether the properties are set globally, for each subnode, or both. Global properties are set one time during runtime configuration and apply to all subnodes. Subnode properties are set for each subnode. Properties that are both global and subnode, are set one time for all subnodes, but then made available again for each subnode.

Configuring an agent with subnodes

1. Set the global (shared) properties
2. Create subnode for each target
3. Set TEP display name and unique properties for each target



Configuring an agent with subnodes

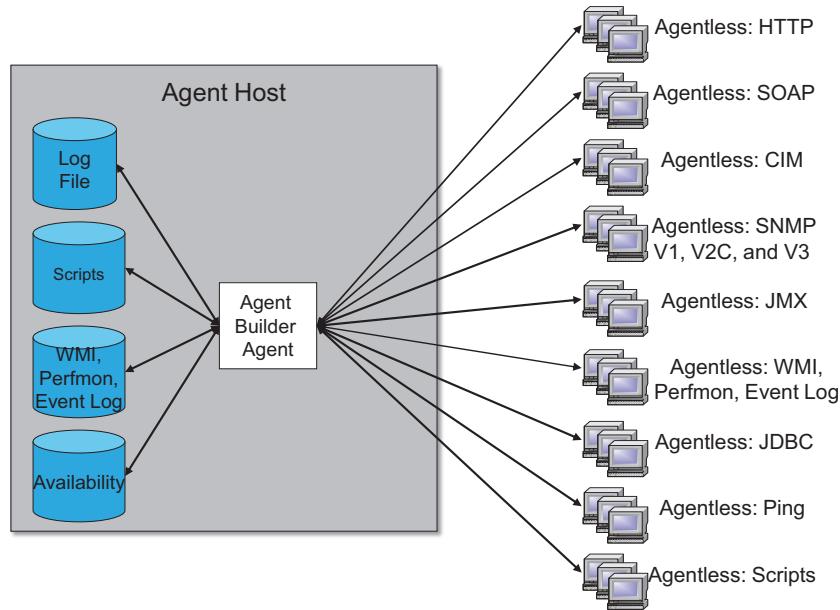
This slide shows the process for configuring an agent with subnodes. A configuration pane is available for each group of global properties. At the subnodes pane, you click **New** to define a subnode. A subpanel for that subnode opens. You set the node name, host name, and any properties unique to this subnode. You repeat this process for each subnode until you define them all.

Lesson 3 Monitoring remote systems

IBM Training

IBM

Lesson 3: Monitoring remote systems



Monitoring remote and optional resources

15

© Copyright IBM Corporation 2017

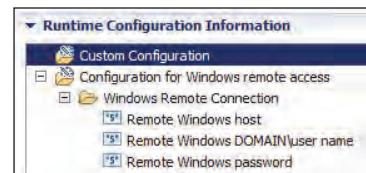
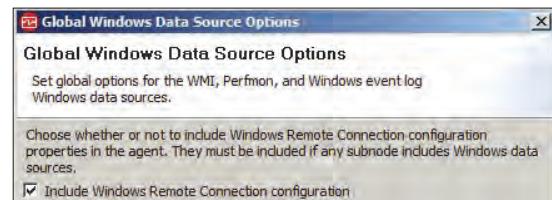
This lesson introduces monitoring of remote data sources. After completing this lesson, you should be able to describe how to remotely monitor data sources.

The diagram shows a single agent that is configured to monitor multiple local and remote data. By their placement, either local or remote, the diagram also identifies which data sources can be monitored remotely and that cannot.

Notice that WMI, Perfmon, and Windows Event Log data sources are in both local and remote categories. By default, the Windows data sources are configured only for local monitoring, but you can enable them to allow remote monitoring. The other remote data sources can be configured for either local or remote monitoring by default.

Remote monitoring and agent runtime configuration

- Runtime configuration properties must allow changing of monitoring target from the local host where the agent is installed to the remote host
- Data sources that can be monitored remotely
 - Available by default: CIM, JMX, SNMP, JDBC, HTTP, Ping, SOAP
 - Global Runtime makes available: WMI, Perfmon, Windows event log
 - Script: SSH connection must be enabled



Remote monitoring and agent runtime configuration

You can use the runtime configuration parameters to change the monitoring target from local to remote. Because of the nature of the CIM, JMX, and SNMP data sources, their runtime configuration always includes the target host. By default, the target host is set to **localhost**, but you can change it in Agent Builder or at run time to a remote host.

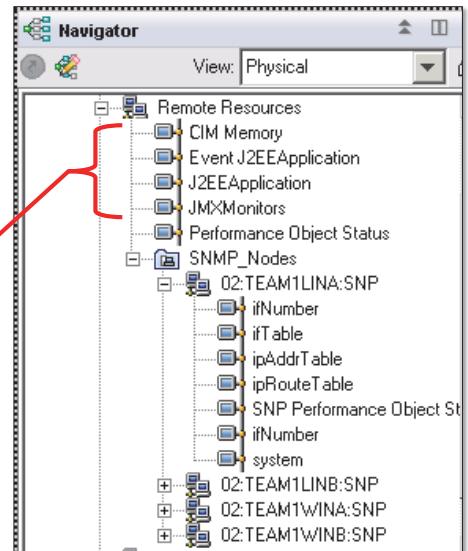
By default, the Windows data sources WMI, Perfmon, and Windows event log do not have runtime configuration. Therefore, the default is local monitoring. But you can force the agent to include the runtime configuration parameters with the **Global Options** feature. **Global Options** forces the host name, user ID, and password parameters into runtime configuration.

Unless a subnode is used, the configuration parameters for a single agent can target only one host for each data source. For example, if you configure your agent to retrieve multiple WMI and Perfmon values, they can all be retrieved from only one Windows host. Furthermore, if you include remote SNMP monitoring, it can target a different host than the Windows data sources, but it can retrieve only data from one SNMP host.

Enabling remote monitoring without subnodes

- Each data source is configured at agent to target a remote resource
 - Can target same or different remote resources per data source
 - Agent cannot monitor more than what it monitors locally
- Necessary runtime configuration properties are available on agent
- The agent host is the IBM Tivoli Monitoring managed system, not the remote hosts

Agent installed on local host, but monitoring remote systems



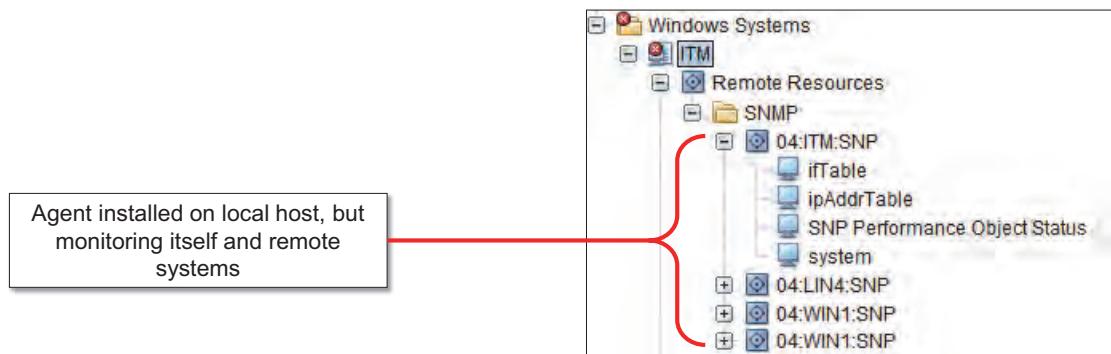
Enabling remote monitoring without subnodes

One way of enabling remote monitoring is to configure the agent data sources to target a different host than the local host. You build the agent like a standard agent and ensure that runtime configuration is available for all data sources. Then, at run time, you configure each data source to target a single remote host. Each different data source can target the same or different hosts.

In the Tivoli Enterprise Portal and the Performance Management console, the agent host is the managed system and not the remotely monitored host. This setup can greatly affect your ability to manage the monitoring of the remote host. The display in the management console looks exactly like a normal agent. Without custom display names, it might be hard to determine from what host the data is being gathered.

Enabling remote monitoring with subnodes

- Create a subnode containing data sources that exist on multiple remote systems
- At agent runtime, create a subnode for each remote system
- Use subnodes to target up to 1000 remote hosts from a single agent
- Each subnode is an IBM Tivoli Monitoring managed system



Enabling remote monitoring with subnodes

Another way of enabling remote monitoring is through a subnode. A **subnode** is a template within an agent of one or more data sources. That template can be reused at run time to target different hosts, allowing the agent to retrieve data from more than one remote host. One subnode might even be configured to monitor the same data sources on the local host.

With subnodes, each subnode is a managed system. You can manage each independently from the other. In the Tivoli Enterprise Portal, each subnode gets a unique subnode Navigator item with Navigator items for each data source, just like a standard agent. In the Performance Management console, each subnode can get its own summary dashboard and attribute details.

Lesson 4 Monitoring Simple Network Management Protocol

IBM Training



Lesson 4: Monitoring using the Simple Network Management Protocol

- Protocol used to exchange information used in network management
- Client and server model
 - Managed host runs an agent (server)
 - Agent maintains the Management Information Base (MIB)
MIB: Hierarchical database that defines variables needed by the SNMP protocol to monitor and control network components
- To monitor a component, gather the appropriate MIB data
- To control a component, modify the appropriate MIB data
- Supported versions: SNMP V1, V2C, and V3
- Can monitor SNMP events (trap)

This lesson teaches you how to monitor resources that use Simple Network Management Protocol (SNMP). After completing this lesson, you should be able to add monitoring of Simple Network Management Protocol (SNMP) to an agent.

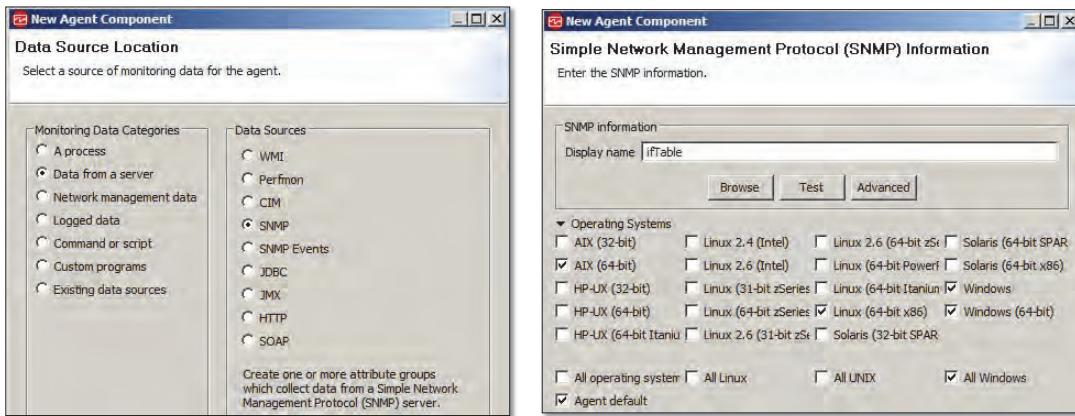
Simple Network Management Protocol (SNMP) is a protocol that is used by network hosts to exchange information used in the management of networks. SNMP network management is based on the client-and-server model. Each managed host runs a process that is called an agent. The agent is a server process that maintains the Management Information Base (MIB) database for the host.

An MIB is a hierarchical database that defines variables that the SNMP protocol needs to monitor and control network components. You monitor the component by contacting the agent and gathering the appropriate MIB variables. You control the component by contacting the agent and modifying the appropriate MIB variables.

SNMP is an open system that many users and equipment manufacturers use. Agent Builder agents can monitor SNMP V1, V2C, and V3.

Adding SNMP monitoring to an agent

- Identify SNMP as data source
- Specify SNMP object name or ID (OID)
Manually enter or use browser
- Identify agent operating systems

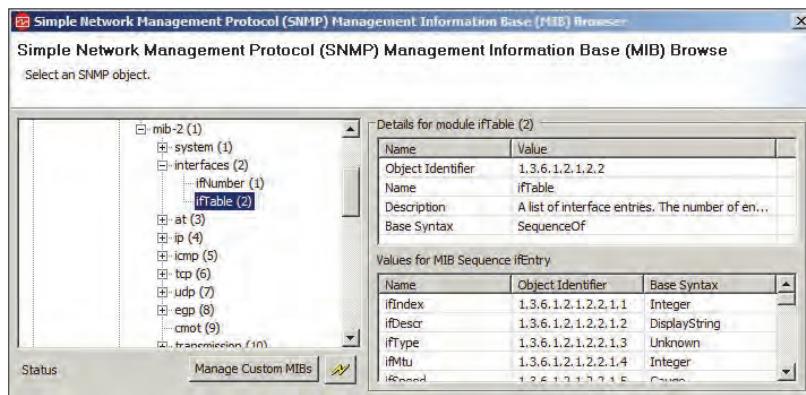


Adding SNMP monitoring to an agent

The required properties are the display name and operating system. The display name must be the correct object name or object identifier (OID). You can manually enter the display name or browse for it. The operating system is the acceptable operating system on which this data source attempts to collect data.

Browsing SNMP

- Browser displays Agent Builder MIB
 - Single, combined MIB of all known MIBs
 - Includes MIBs shipped with AB and any MIBs that you import
- Standard actions
 - Import MIB
 - Refresh
 - Select an object



Browsing SNMP

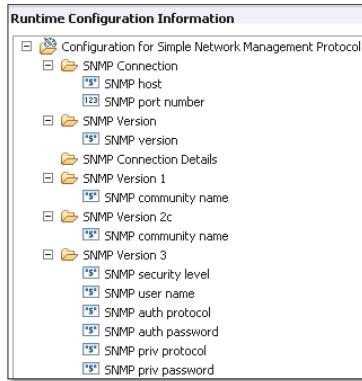
The SNMP browser displays the current Agent Builder MIB, which is a combined MIB of all MIBs known to this instance of Agent Builder. Agent Builder contains a large set of common MIBS. This superset MIB would include these MIBs, plus any that you import.



Note: Notice that there is no remote monitoring of MIBs in the standard actions.

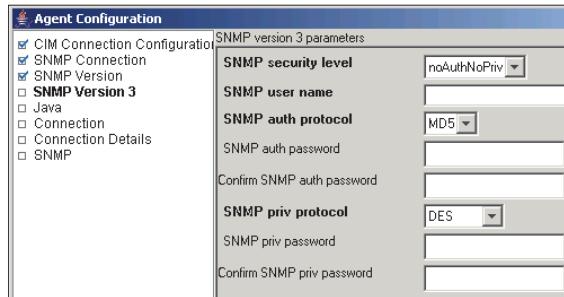
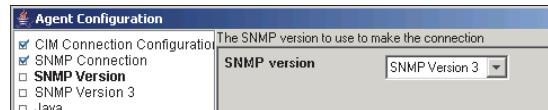
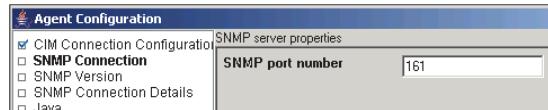
When you select an object, its values are shown on the right. Generally, the object that is selected is the attribute group and the values that are the attributes that are collected. But sometimes the object you select includes subjects that have their own values. In such cases, the agent collects multiple attribute groups with their own attributes.

SNMP runtime configuration



For each instance of an agent, all SNMP data sources must use the same version

Tip: SNMP community names are case-sensitive



SNMP runtime configuration

The SNMP data source requires runtime configuration so that you can specify which SNMP device the agent monitors. The SNMP configuration properties are added to the agent automatically. You can view, add, and change the configuration properties with the Agent Editor. You can set the defaults for most values except passwords.

Each configuration property is defined here:

Parameter name	SNMP versions that require this parameter	Parameter description
SNMP host	All	The host name or IP address of the SNMP system
SNMP port number	All	The port number of the SNMP server
SNMP version	All	The SNMP version to use to make the connection: V1, V2C, or V3
Community name	V1, V2C	The SNMP server community name
SNMP user name	V3	The user name for connecting to the SNMP agent
Auth password	V3	The authorization pass phrase for connecting to the SNMP agent

Parameter name	SNMP versions that require this parameter	Parameter description
Priv password	V3	The privacy pass phrase for connecting to the SNMP agent
Security level	V3	The security level that is used to connect to the SNMP agent: noAuthNoPriv, authNoPriv, or authPriv
Authorization protocol	V3	The authorization protocol that is used to connect to the SNMP agent: MD5 or SHA
Privacy protocol	V3	The privacy protocol that is used to connect to the SNMP agent: DES or CBC DES

For each instance of an agent, all SNMP data sources must be using the same version. If you need to monitor systems with different SNMP versions, you must make your agent multiple-instance. Each instance of the agent would monitor the systems for each SNMP version.

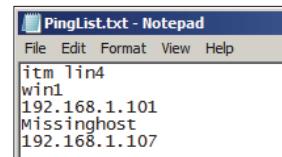
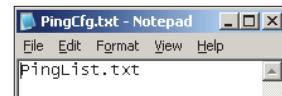
Lesson 5 Monitoring network availability with ping

IBM Training

IBM

Lesson 5: Monitoring network availability with ping

- Monitors network availability of network devices using ICMP ping
- Configuration file
 - Lists device list files
 - If not supplied, KUMSLIST file loaded from the Agent BIN
- Device list file
 - Lists host names and IP addresses of devices to monitor



Report					
Node	Timestamp	Address	Device Entry	Current Response Time	Name
ITM:04	07/24/15 16:48:37	192.168.1.101	192.168.1.101	1	itm.ibm.edu
ITM:04	07/24/15 16:48:37	192.168.1.107	192.168.1.107	1	lin4.ibm.edu
ITM:04	07/24/15 16:48:37	UNKNOWN ADDRESS	Missinghost	0	UNKNOWN HOSTNAME
ITM:04	07/24/15 16:48:37	192.168.1.103	win1	1	win1.ibm.edu
ITM:04	07/24/15 16:48:37	192.168.1.101	itm	1	itm.ibm.edu
ITM:04	07/24/15 16:48:37	192.168.1.107	lin4	1	lin4.ibm.edu

This lesson teaches you how to monitor the network availability of multiple hosts from a single agent. After completing this lesson, you should be able to add monitoring of network availability to an agent.

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 systems, add the Ping data collector to your agent.

The agent must be provided with the list of systems to be monitored by using ping configuration files. The agent does not manage or update the files that are used. The user must make all required updates to the files. Both a ping configuration file and a device list file must be created.

The ping configuration file is a text file that must be specified as a runtime configuration parameter to the agent. The contents of each line in this file must be the location of a device list file. The device list file lists host names and IP addresses of devices to monitor.

Each entry in the ping configuration file causes the agent to start a separate thread for monitoring the devices in the provided device list file. This thread loops through all the devices in the device list file, sending each a ping request. These threads run at intervals of 60 seconds or at intervals of 30 seconds plus the time it takes to ping the list, whichever is longer. If configuration does not contain device list files, the file name KUMSLISTtries to be loaded from the Agent BIN directory.

Ping behaviors and reporting

- Performance Object Status error codes:
 - Ping File Does Not Exist
 - No Ping Device Files
 - Ping Device List File Missing
- Duplicates
 - Duplicate devices in a file are removed
 - Duplicates devices across files not detected
 - Duplicate IP addresses not removed unless device entry also duplicated
- Unresolvable addresses
 - Address and Name set to 0.0.0.0, status set to Unknown, not pinged
- Invalid addresses
 - Marked with status of Invalid and not pinged
- Asynchronous
 - All pings sent asynchronously and response times can be slower than from command line

Ping behaviors and reporting

Here is a list of ping monitoring data:

- Performance Object Status error codes:
 - Ping File Does Not Exist: No ping configuration file might be found.
 - No Ping Device Files: No ping device files were found. This error code deactivates the attribute group.
 - Ping Device List File Missing: A Device listing file was missing, but others were found so the attribute group remains active.
- Duplicates:
 - Duplicate device entries in a single device entry file are removed.
 - Duplicates across device entry files are not detected.
 - Duplicate IP addresses are not removed unless the device entries are also duplicates.
- Unresolvable addresses:

A device entry that cannot be resolved has the Address and Name set to 0.0.0.0. The status is set to Unknown, and no attempt to ping is made.
- Invalid addresses:

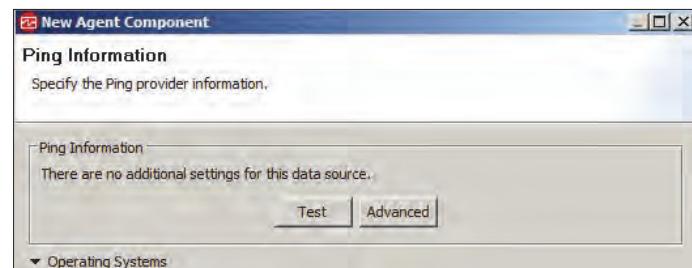
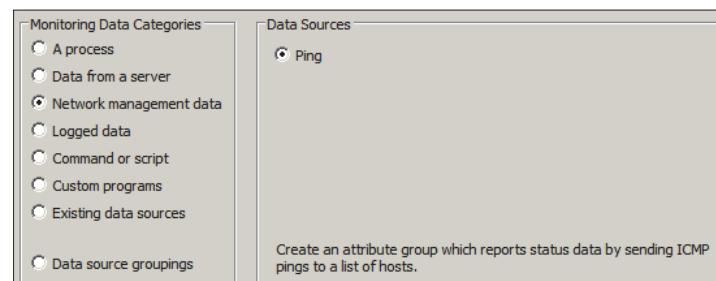
An IP address that is invalid is marked with a status of Invalid and is not pinged. Invalid address start or end with 0 or 255.

- Asynchronous:

All pings are sent asynchronously to the systems in a device list file. For this reason, response times can be slower than if you ran a single ping request from the command line.

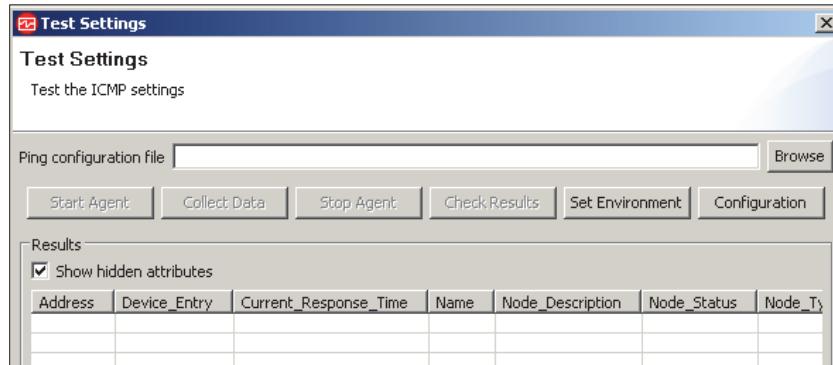
Adding a Ping data source

- Add Ping data source
- Managed Nodes attribute group created automatically
- Configuration file location and name is a runtime configuration property



Testing Ping data source

- Identify location of configuration file
- Set environment variable, runtime configuration, and Java information as needed



Testing Ping data source

You can test the Ping attribute group that you created within Agent Builder. You can start the testing procedure in the following ways:

- When you are creating the agent, click **Test** on the Ping Information page.
- After you create the agent, select an attribute group on the Agent Editor Data Source Definition page and click **Test**.

Ping attribute group and runtime configuration

- Attribute Group
 - Managed Nodes attribute group automatically create There is no need for a browser because all ping attribute groups look the same
 - The data types of the attributes cannot be modified, but derived attributes can be added
- Runtime Configuration

Runtime Configuration Information

Runtime Configuration Information

- Custom Configuration
- Configuration for ICMP Hotlist Network Management data provider (ICMP)
 - Network Management
 - Ping configuration file

Network management properties

Ping configuration file 

Data Source Definition

Attribute Group Information

- Ping Sample (Agent)
 - Managed_Nodes (ICMP)
 - Address
 - Device_Entry
 - Current_Response_Time
 - Name
 - Node_Description
 - Node_Status
 - Node_Type
 - Status_Timestamp

Lesson 6 Monitoring HTTP URLs and objects

IBM Training

IBM

Lesson 6: Monitoring HTTP URLs and objects

- Monitors availability of HTTP URLs and embedded objects
- HTTP data source reads file for URLs to monitor
- Java requirement
 - HTTP is implemented as a Custom data provider
 - Requires Java 1.5 or newer
 - Takes advantage of URL features built into Java
- Replacement for the Universal Agent HTTP data provider
 - UA KUMPURLS file format can be used
 - Enhancements
 - Extra columns or derived attributes can be added
 - Page object detection is much improved
 - The file containing the list of URLs to monitor is a config parameter
 - Can monitor URLs that use the file or FTP protocols
 - Differences
 - Interval metrics do not apply
 - Caching concept is not implemented
 - INTERVAL and CACHE keywords in KUMPURLS file are ignored
 - Adding URLs to monitor using a situation is not implemented

Report							
Node	Timestamp	URL	Response Time	Page Size	Page Objects	Total Object Size	
ITM:04	07/24/15 16:58:38	http://win1/	47	1500	8	92572	
ITM:04	07/24/15 16:58:38	http://win1/AnyBank.htm	47	857	2	92949	
ITM:04	07/24/15 16:58:38	http://win1/bad.html	32	1245	0	NOT COLLECTED	
ITM:04	07/24/15 16:58:38	http://win1/AnySiteMap.html	32	848	2	112672	
ITM:04	07/24/15 16:58:38	http://lin4	32	1500	8	92572	

This lesson teaches you how to monitor the availability of HTTP URLs. After completing this lesson, you should be able to add monitoring of HTTP server URLs to an agent.

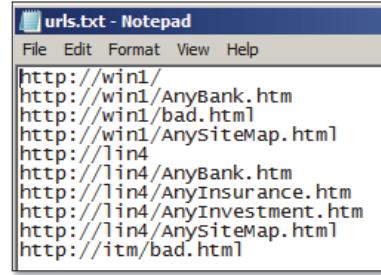
You can monitor the availability and response time of selected URLs by using an HTTP data source.

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.

URLs file

- HTTP data source reads file for URLs to monitor
- Modify file:
 - Manually
 - Take Actions
 - Http URL Add
 - Http URL Remove
- http:// is assumed unless specified otherwise
- User Data and Alias parameters are optional



The screenshot shows a Windows Notepad window titled "urls.txt - Notepad". The menu bar includes File, Edit, Format, View, and Help. The main content area contains the following list of URLs:

```
http://win1/
http://win1/AnyBank.htm
http://win1/bad.html
http://win1/AnySiteMap.htm
http://lin4
http://lin4/AnyBank.htm
http://lin4/AnyInsurance.htm
http://lin4/AnyInvestment.htm
http://lin4/AnySiteMap.htm
http://itm/bad.html
```

URLs file

The HTTP data source reads the HTTP URLs file for URLs to monitor. The file that contains the URLs to monitor can be modified either manually or through the built-in Take Actions. Manually edit the HTTP URLs file. If the agent is running, updates are implemented the next time data collection occurs. The Http URL Add Take Action adds a URL to the monitoring file. The Http URL Remove Take Action removes a URL from the monitoring file.

If the protocol is left off the URL, **http://** is assumed and the management console displays the URL with the **http://** prefix added.

The User Data and the Alias parameters that are shown in the Http URL Add Take Action are optional. They are user-specified fields that can provide additional information in the attribute group. If User Data is not specified, the default value is INITCNFG (like the UA). If Alias is not specified, the default value is an empty string (like the UA).

Adding an HTTP data source

- Add HTTP data source
- Two attribute groups created automatically
- URLs file location and name is a runtime configuration property



Adding an HTTP data source

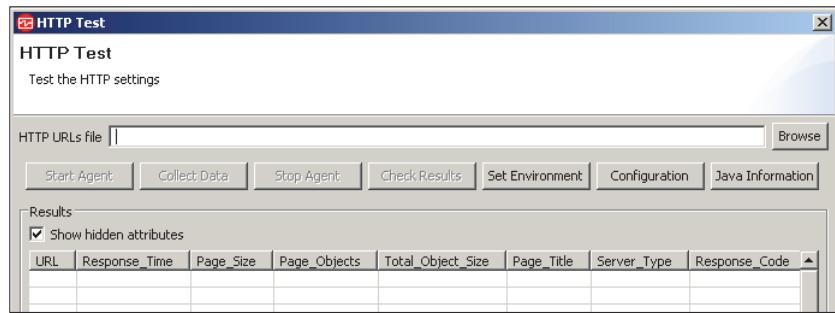
The HTTP data source creates two attribute groups:

- **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, table might have entries for the .gif and .jpg files that might be used in the website that is listed in the Managed URL report.

You can add, modify, or delete attributes.

Testing the HTTP data source

- Select to test URL or embedded object monitoring
- Identify location of URL file
- Set environment variable, runtime configuration, and Java information as necessary



Testing the HTTP data source

You can test the HTTP attribute group that you created within Agent Builder. Start the testing procedure in the following ways:

- When you are creating the agent, click **Test** on the HTTP Information page.
- After you create the agent, select an attribute group on the Agent Editor Data Source Definition page and click **Test**.

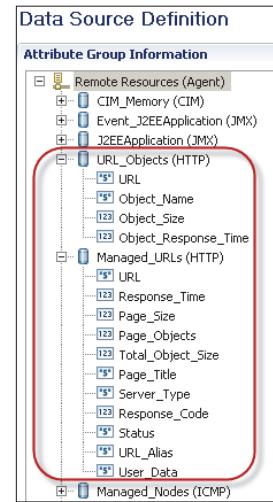
HTTP attribute groups and runtime configuration

- Attribute groups
 - Two attribute groups are automatically created
 - URLs
 - Embedded objects
 - The data types of the attributes cannot be modified, but derived attributes can be added
- Runtime configuration

Runtime Configuration Information

Runtime Configuration Information

- Custom Configuration
- Configuration for Hypertext Transfer Protocol Proxy
 - Proxy Server Configuration
 - HTTP Data Provider Proxy Information
 - Proxy Hostname
 - Proxy Port
 - Proxy User Name
 - Proxy Password
 - Configuration for Hypertext Transfer Protocol (HTTP)
 - URL Monitoring
 - HTTP URLs File
 - Page Object Collection



HTTP attribute groups and runtime 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.

Configuring the agent for HTTP monitoring

- Enter HTTP URLs file
- Indicate whether monitoring embedded objects
- Configure proxy server, if needed
 - Runtime configuration provided
 - Used for HTTP, HTTPS, and FTP requests
 - To limit proxy to subset of protocols, create multiple instances
 - One that uses the proxy
 - One that does not
 - Use different HTTP URLs files for each instance

The image shows two configuration dialog boxes. The top box is titled 'Configuration for monitoring a list of URLs' and contains fields for 'HTTP URLs File' (with a browse button) and 'Page Object Collection' (set to 'Yes'). The bottom box is titled 'Configuration for a proxy server used by HTTP providers' and contains fields for 'Proxy Hostname', 'Proxy Port' (set to 80), 'Proxy User Name', 'Proxy Password', and 'Confirm Proxy Password'.

Configuring the agent for HTTP monitoring

The main runtime configuration property is to enter HTTP URLs file path and file name. You can also indicate whether you want the agent to monitor embedded objects.

If needed, you can configure the proxy server. Runtime configuration is provided to specify the proxy server. The values are optional. If you do not have a proxy server, make sure that you do not specify a value for the proxy server host name.

If you specify a proxy server, it is used for HTTP, HTTPS, and FTP requests. If you want a proxy only for HTTP but not HTTPS or FTP, create the agent as a multiple instance agent and create multiple instances. One instance monitors the URLs that require the proxy, and one instance monitors the URLs that do not require the proxy.

Student exercises



Exercise 1: Remotely monitoring many resources

Exercise 2: Install and confirm the updated AB1 agent

Student exercises

Open your Course Exercises book and complete the exercises for this unit.

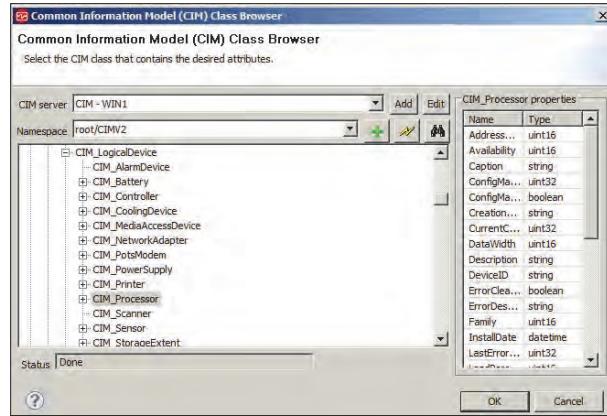
Lesson 7 Monitoring the Common Information Model

IBM Training

IBM

Lesson 7: Monitoring the Common Information Model

- Monitors any object that supports Open CIM Object Managers (CIMOM) / Pegasus: Solaris, IBM Director, z/OS Servers
- Uses CIM over HTTP or HTTPS to connect to remote systems to gather data
- Runtime configuration
 - Required
 - Includes any information needed to access a local or remote CIMOM
- All returned metrics are available for the standardized data manipulations



This lesson teaches you how to monitor resources by using the Common Information Model (CIM). After completing this lesson, you should be able to add monitoring of Common Information Model (CIM) data sources to an agent.

The Common Information Model (CIM) is an industry standard for describing data about applications and devices. With it, administrators and software management programs can control applications and devices on different operating systems in the same way, ensuring interoperability across a network.

Open CIMOM (CIM Object Manager) is a term that is used to define the implementation of the CIM_Server defined in the CIM Operations over HTTP specification. Typically, a CIMOM includes the functions of the CIM Server, repositories, providers, and the tools to manage the CIM server.

Pegasus and IBM Director are examples of Open CIMOMs. Both AIX and z/OS servers provide Open CIMOMs for monitoring and managing their systems.

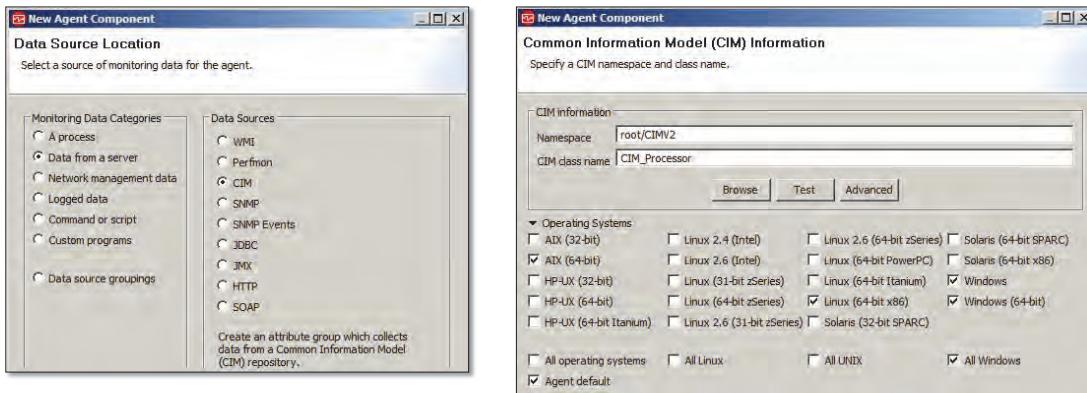


Note: IBM Director is an implementation of Pegasus.

The Agent Builder CIM data source enables your agent to gather data from Open CIMOMs. This data source requires runtime configuration, with which you can configure any user IDs, passwords, and other information that is needed to access a local or remote CIMOM.

Adding CIM monitoring to an agent

- Identify CIM as data source
- Specify namespace and class, all properties initially imported
Manually enter or use browser
- Identify operating systems



Adding CIM monitoring to an agent

Configuring a CIM data source is similar to configuring a WMI data source. WMI data sources are covered in [Unit 4, “Monitoring Windows resources”](#). You must identify the target namespace and class you want to gather. You can manually enter the data or use a browser. You must specify the operating system on which the agent is installed.



Note: If your agent is monitoring a remote CIMOM, operating system does not refer to the CIMOM host operating system.

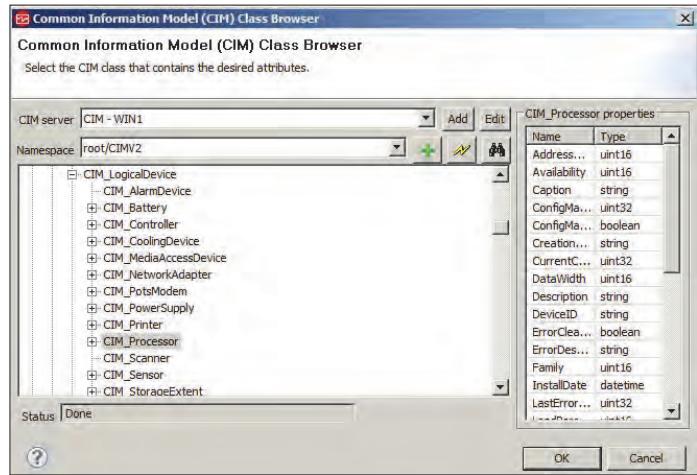
Browsing CIM

- Standard actions
 - Local or remote browsing
 - Refresh
 - Find
 - Select namespace
 - Select one class; properties are shown on right

- Remote browsing

- Add host name
- Login required
- Lists namespaces on remote system

Note: Cannot browse Solaris, but can monitor it



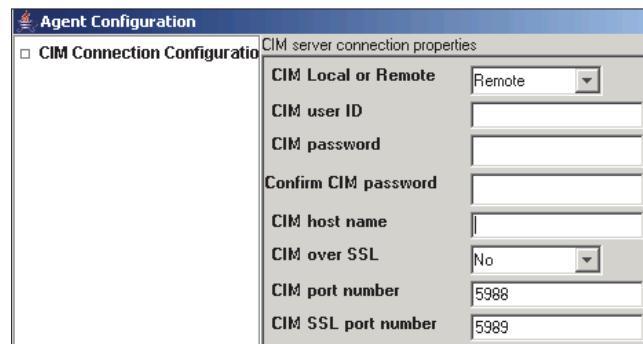
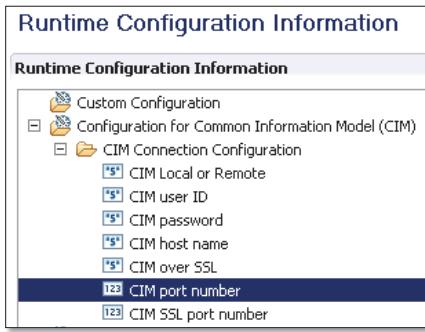
Browsing CIM

The CIM browser functions similarly to the WMI browser, with the exception that with CIM, you can browse Linux and UNIX in addition to Windows CIMOMs.



Note: You cannot browse a Solaris CIMOM, but you can monitor it by manually entering the correct namespace and CIM class name.

CIM runtime configuration



CIM runtime configuration

If you define a CIM data source in your agent, CIM configuration properties are added to the agent automatically and they are required. You can view, add, and change the configuration properties with the Agent Editor. You can set the defaults for most values except passwords.

Each configuration property is defined here:

- **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 Secure Socket Layer (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 usually different.

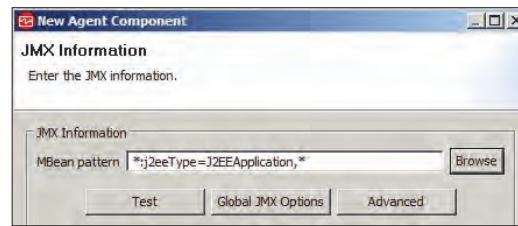
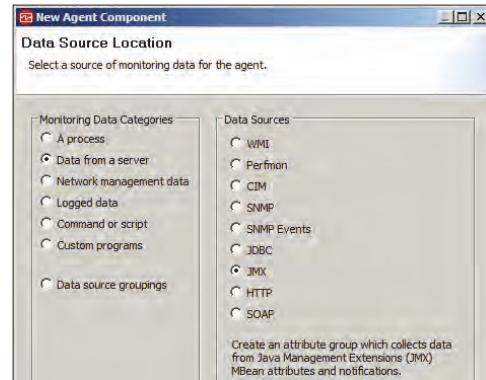
Lesson 8 Monitoring Java Management Extension

IBM Training

IBM

Lesson 8: Monitoring application servers with Java Management Extension

- Collects data from JMX MBeans (managed beans)
 - MBean is a Java object that represents a manageable resource, such as an application, a service, a component, or a device
- Supported Java EE application servers
 - WebSphere Application Server 6 and 7
 - JBOSS 4 and earlier
 - WebLogic 9, 10
 - JSR-160-Compliant Servers
- Enter the object name or object name pattern for a type of MBean
- Browser and agent requires JMX connection information



Monitoring remote and optional resources

39

© Copyright IBM Corporation 2017

This lesson teaches you how to monitor Java EE applications that use the Java Management Extensions (JMX). After completing this lesson, you should be able to add monitoring of Java Management Extension (JMX) data sources to an agent.

Java Management Extension (JMX) technology provides a simple, standard way of monitoring and managing Java Platform, Standard Edition (J2SE) resources such as applications, devices, and services. You can also use JMX technology to monitor and manage the Java Virtual Machine1 (JVM).

The Agent Builder JMX monitor collects JMX MBean information. An MBean (managed bean) is a Java object that represents a manageable resource, such as an application, a service, a component, or a device. The slide lists the specific Java Platform, Enterprise Edition (J2EE) application servers that can be monitored.

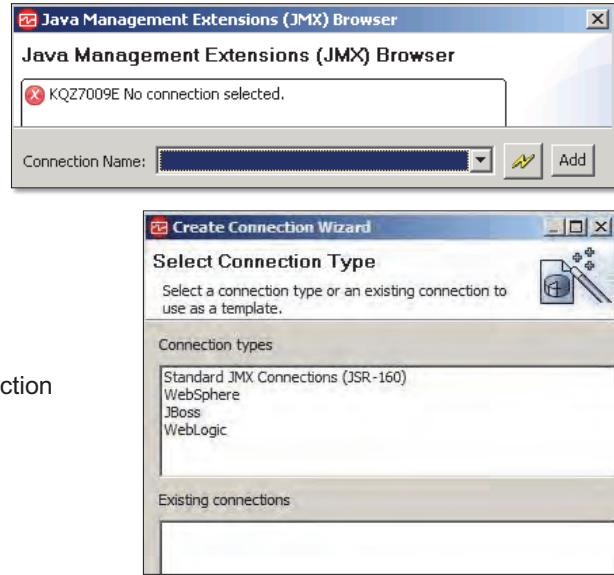
The only required parameter is the **MBean pattern**, which can identify one or more MBeans to collect. You can manually enter the MBean pattern or use a browser to build that pattern.



Hint: Although the MBean pattern is the only required parameter, if you do not use the browser and do not select a specific MBean, Agent Builder creates an attribute group with no attributes.

Browsing MBeans: Connecting to application server

- If never connected
 - Browser is empty
 - Must establish connection to JMX app
 - Connections are saved for reuse
- If previously connected
 - Browser opens to last connection
 - Can change or establish new
- To establish new, select the type of connection
 - Connection properties different for each type of connection



Browsing MBeans: Connecting to application server

To browse, you must first establish a connection to a JMX data source. If you never connected, the browser is empty, and you need to define a connection. This connection information is saved for later use.

If you previously connected, the browser opens to the previous connection. You can change to another connection or establish a new one.

To establish a new connection, you must select the type and enter the connection properties appropriate to that application server. The connection properties for each type of application server are different.

Browsing MBeans: Connecting to WebSphere Community Edition

- Log in required
 - JMX user and password
- JMX service URL
 - Edit as needed to connect to JMX service
 - For remote browsing, replace localhost with host name or IP address
- Java class path information
 - Paths to any local Java needed to connect to JMX server
- Set as agent configuration defaults
- Test Connection



Monitoring remote and optional resources

41

© Copyright IBM Corporation 2017

Browsing MBeans: Connecting to WebSphere Community Edition

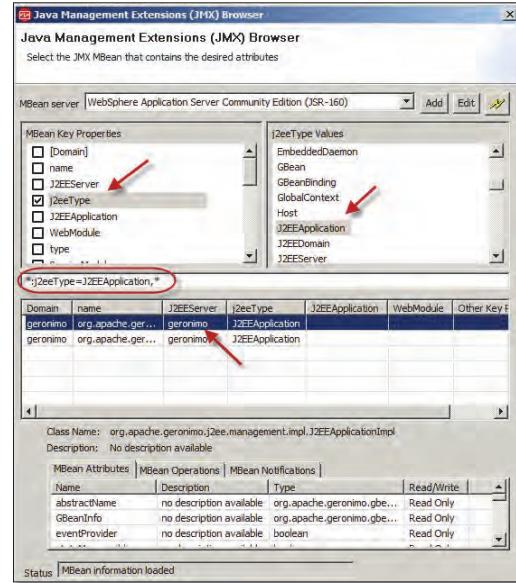
This slide documents the parameters for connecting to a WebSphere Community Edition (CE) application server. The **Connection Name** is the name under which these parameters are saved. The default **JMX user ID** is **system** with password **manager**.

The service URL is the default URL for a local connection. To connect to and browse a remote host, replace **localhost** with the remote server host name or IP address. The Java class path information is the local paths that the agent needs to establish the connection.

You can specify these parameters as the runtime configuration defaults. Use **Test Connection** to ensure that you can connect to the server.

Browsing JMX MBeans

- Standard actions
 - Local or remote browsing
 - Refresh
 - Sort by column
- Objective
 - Create an MBean pattern that gathers the data you want
 - Pattern can return one or many MBeans
- Process
 - Select MBean key properties
 - Select a value (key attribute)
 - List shows all MBean that match pattern
 - Select an MBean to view attributes, operations, and notifications



Browsing JMX MBeans

You use the two boxes in the top of the **Java Management Extensions (JMX) Browser** to construct an MBean pattern. The JMX data provider collects all MBeans that match the pattern from the JMX server that you connect to.

The initial pattern, `*.*`, matches everything. This pattern usually provides too much data, and all MBeans do not return the same types of data. You want to select **MBean Key properties** and the **Selected Key Property Values** that define an MBean pattern that selects the MBeans that you want to monitor. The pattern that you are generating is displayed below the selection boxes. Below the pattern is a list of the Mbeans that match that pattern.

If the list contains no MBeans, your pattern is too strict. If the list has too many matching MBeans, you probably want to refine your pattern. Most often, matching the **j2eeType** or **type** MBean Key Properties filters the MBeans to the ones you want.

If you select one of the MBeans that matches your pattern, the bottom section of the dialog box displays MBean attributes, operations, and notifications that are associated with that MBean.

While this MBean is selected, select **Finish**. The browser creates a JMX data source that collects MBeans with the pattern that is specified. It defines attributes that are based on the attributes available in the MBean that was selected. If no MBean was selected, no attributes are defined. If the selected MBean has associated notifications, a JMX event data source is also created automatically to subscribe to those notifications.

JMX monitors and operations

- JMX monitors and operations
 - JMX Add String Metric Watcher
 - JMX Add Gauge Metric Watcher
 - JMX Add Counter Metric Watcher
 - JMX Delete Metric Watcher
- JMX Invoke
 - Exposed as a Take Action
 - Command definition and sample

```
JMX_INVOKE [MBeanName] [Operation] [arg1] [arg2] [arg3]
```

JMX monitors and operations

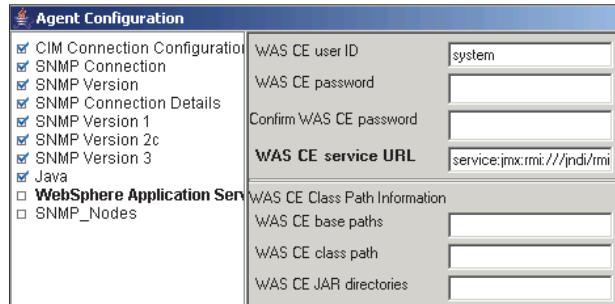
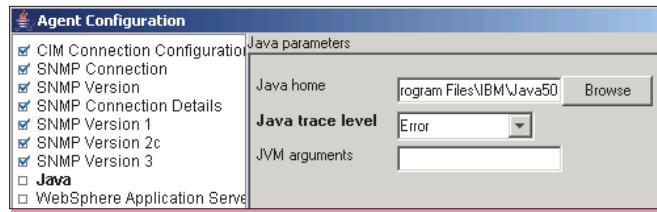
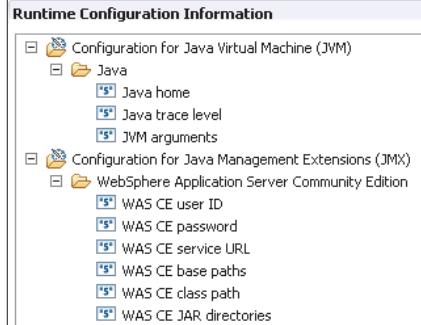
The JMX_INVOKE operation is available as a Take Action. It runs an operation on all Mbeans matching the pattern that you specify. If the operation returns data, that data is logged.

Here is the command sample and definition:

```
JMX_INVOKE [MBeanName] [Operation] [arg1] [arg2] [arg3] [arg4]
```

- The arg parameters can be empty.
- Any simple data types can be used for the arguments.
- The operation is created based on data from the MbeanInfo metadata.

JMX runtime configuration for WebSphere Community Edition

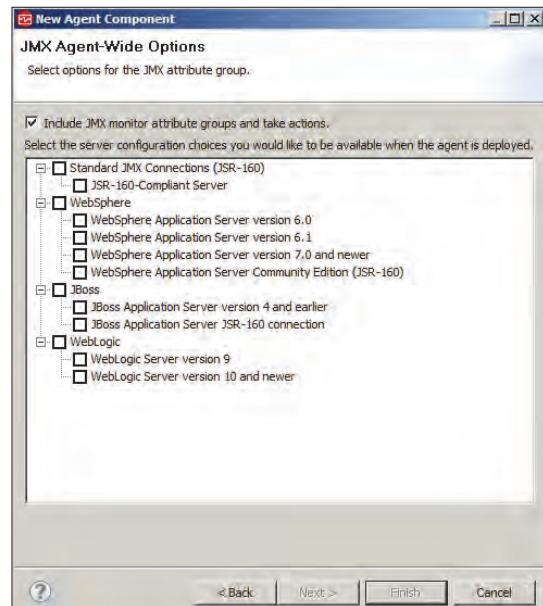


JMX runtime configuration for WebSphere Community Edition

The JMX data source requires runtime configuration so that you can configure the specific connection information for each agent. This slide shows the WebSphere Community Edition (CE) runtime configuration parameters. The runtime configuration parameters are different for each type of Java Platform, Enterprise Edition application server.

Adding other JMX types to JMX runtime configuration

- JMX automatically creates runtime configuration for the JMX type that you target
- Some JMX MBean patterns are shared across JMX types
- You can add runtime configuration for other JMX types, allowing the agent to be configured for the specific JMX type



Monitoring remote and optional resources

45

© Copyright IBM Corporation 2017

Adding other JMX types to JMX runtime configuration

Agent Builder automatically creates runtime configuration for the JMX type that you browsed.

You can add the runtime configuration parameters for one or more Java Platform, Enterprise Edition application servers so that at run time you can target different types of application servers. This procedure works only when the MBean pattern that you defined works for each type of application server that you target.

JMX reference materials

- JMX ObjectName description
<http://java.sun.com/j2se/1.5.0/docs/api/javax/management/ObjectName.html>
- Java Monitoring and Management site
<http://java.sun.com/j2se/1.5.0/docs/guide/management/>
- JDBC Overview
<http://java.sun.com/products/jdbc/overview.html>

Lesson 9 Monitoring database content with JDBC

IBM Training

IBM

Lesson 9: Monitoring database content with JDBC

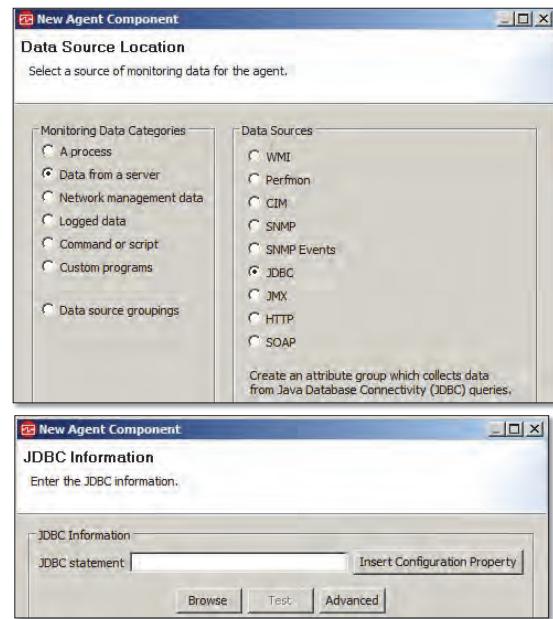
- Databases supported
 - DB2
 - Oracle
 - MS SQL Server
- Run SQL statements or stored procedures
- Can connect to remote databases
- Works in subnodes
- Process overview
 - Browse database to build data sources
 - Test JDBC statements in the Builder
 - Store connections to reuse them later
 - Attributes are created from database metadata

In this lesson, you learn to monitor databases through a JDBC connection. After completing this lesson, you should be able to add monitoring of a database with the JDBC data source to an agent.

Use the JDBC data source to collect data from databases that allow JDBC connections. Each JDBC data source that you define runs an SQL query to collect data from the database.

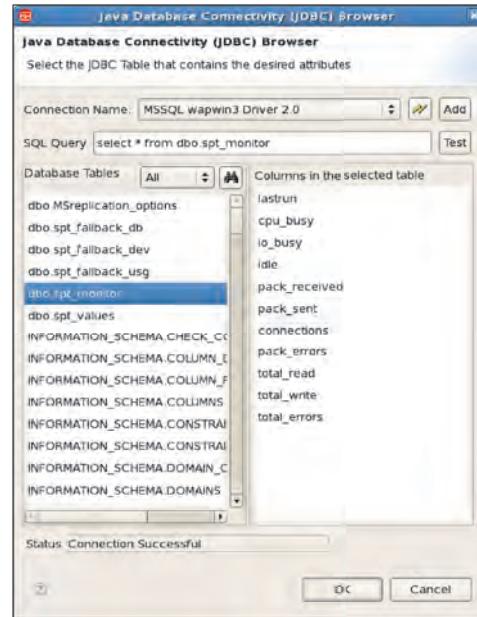
Adding a JDBC data source

- The data source requires a statement to run against the database
- Operating systems are where the agent can run, not where the database can run
- Statement can contain configuration properties



Browsing JDBC data source

- When a connection is created, browser lists database tables
 - Selecting a table lists columns
 - Selections update the query automatically
- Query can be tested
- You can select multiple tables to create multiple data sources at the same time
 - This action disables the Test and Column selection
- Search (binoculars) for a table
- Filter (choice option beside search) tables to view User tables, System Tables, Views, or All



Monitoring remote and optional resources

49

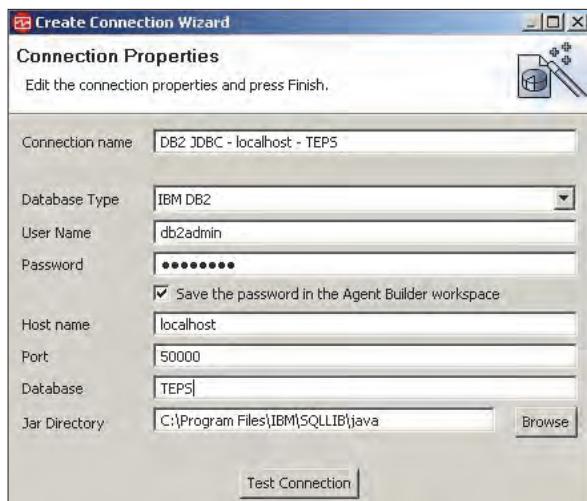
© Copyright IBM Corporation 2017

Browsing JDBC data source

Use the JDBC Browser to connect to a database and view its tables so that you can build an SQL query that collects the data that you need. When you select a table and columns, a query is generated for you. You can modify and test the query that is generated to make sure the data that is returned is what you need.

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.

JDBC connection properties for remote browsing



JDBC connection properties for remote browsing

Here are a few connection property definitions:

- **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 that 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 that are used to connect to the database. Type the path name, or click **Browse** to locate the directory.

Testing the JDBC data source

- Run queries to view data returned from the database
- Attributes are created and can be edited by selecting the column headers
- Test SQL WHERE clauses and complex queries to refine your data collection

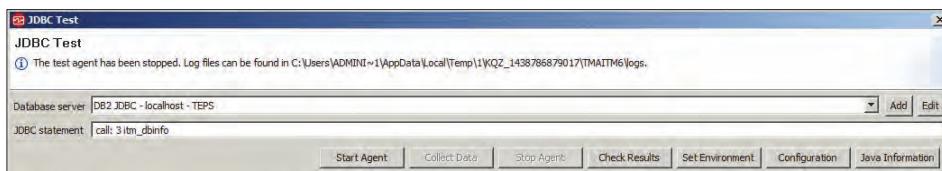
Tip: Do joins using SQL queries instead of doing joins in Agent Builder

The screenshot shows the JDBC Test application window. At the top, it says "JDBC Test" and "5 data rows returned at Jul 5, 2015 10:01:26 AM." Below this is a toolbar with buttons for "Start Agent," "Collect Data," "Stop Agent," "Check Results," "Set Environment," "Configuration," and "Java Information." The main area is titled "Results" and contains a table with the following data:

TBSpace	Definer	Create_Time	Tbspaceid	Tbspacetype	Datatype	Extentsize	Prefetchsize	Overhead	Transferrate	Remarks	Ngn
SYSCATSPACE	SYSIBM	Apr 16, 2015 11:08:17 AM	0	D	A	4	-1	7.5	0.06	No value	IBM
TEMPSPACE1	SYSIBM	Apr 16, 2015 11:08:17 AM	1	S	T	32	-1	7.5	0.06	No value	IBM
USERSPACE1	SYSIBM	Apr 16, 2015 11:08:37 AM	2	D	L	32	-1	7.5	0.06	No value	IBM
SYSTOOLSPACE	DB2ADMIN	Apr 16, 2015 12:48:23 PM	3	D	L	4	-1	7.5	0.06	No value	IBM
SYSTOOLTMPSPACE	DB2ADMIN	Apr 17, 2015 12:29:47 PM	4	S	L	4	-1	7.5	0.06	No value	IBM

Running stored procedures

- Collects data from all result sets by default
- Stored Procedure format:
 - call[:index] procedureName [args] [...]
- Samples:
 - call sp_helpdb
 - call: 2 sp_helpdb master



Running stored procedures

Here are two samples of stored procedures:

- call sp_helpdb

Runs the procedure **call sp_helpdb**, which requires no arguments. Data from all returned result sets is included in the data the data provider returns.

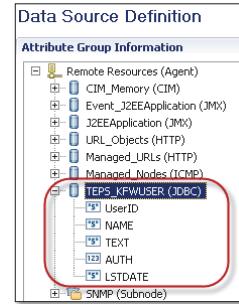
- call:2 sp_helpdb master

Runs the procedure **sp_helpdb** with the master argument. This argument is a string input argument. Only data from the second result set that the stored procedure returns is included in the data that the data provider returns.

When the index is not specified, data from all returned results sets is collected. You must ensure that the data that is returned in these cases is compatible with the attributes that 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.

JDBC attribute groups and runtime configuration

- Attribute Group
Attribute group created based on SQL query and tester
- Runtime configuration



JDBC attribute groups and runtime 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:** This parameter is 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 file. The default is to write only error data to the log file.

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

- **Class path:** Explicitly named .jar files the agent searches
- **JAR directories:** List of directories that are searched for .jar files

Student exercises



Exercise 3: Allowing data sources to be optional
Exercise 4: Install and confirm the updated AB2 agent

Open your Course Exercises book and complete the exercises for this unit.

Summary

Now that you have completed this unit, you should be able to perform the following tasks:

- Monitor remote data sources
- Add optional data sources to an agent
- Manage runtime configuration parameters
- Add monitoring of Simple Network Management Protocol (SNMP) to an agent
- Add monitoring of HTTP server URLs to an agent
- Add monitoring of network availability to an agent
- Add monitoring of Java Management Extension (JMX) data sources to an agent
- Add monitoring of Common Information Model (CIM) data sources to an agent
- Add monitor of a database with the JDBC data source to an agent



IBM Training



© Copyright IBM Corporation 2017. All Rights Reserved.