User's Guide
Note
Before using this information and the product it supports, read the information in Appendix F, “Notices,” on page 157.

First Edition (September 2004)
This edition applies to version 1.1.3 of IBM Tivoli Resource Model Builder and to all subsequent releases and modifications until otherwise indicated in new editions.

© Copyright Sax Software Corporation 2004. All rights reserved.

© Copyright International Business Machines Corporation 2004. All rights reserved. US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
Contents

Tables ........................................ vii

About this guide. .......................... ix
Who should read this guide ................. ix
What this guide contains ................... ix
Publications .................................. x
  IBM Tivoli Resource Model Builder library . x
  Related publications ....................... x
  Accessing publications online .......... x
  Ordering publications ..................... xi
Accessibility ................................ xi
Contacting software support ............... xi
Conventions used in this book .......... xii
  Typeface conventions ..................... xii
  Operating system-dependent variables and paths xii

Chapter 1. Overview ..................... 1
About IBM Tivoli Resource Model Builder . 1
  Compatibility mode ....................... 2
  Common Information Model implementation . 2
  IBM Eclipse SDK ......................... 2

Chapter 2. Getting started
quick-reference guide ................. 3

Chapter 3. Installation ................. 5
Software requirements .................. 5
Hardware requirements ................ 6
Installing IBM Tivoli Resource Model Builder with
  IBM Eclipse SDK ....................... 7
Installing IBM Tivoli Resource Model Builder . 8
Migrating existing resource models .... 9
Uninstalling IBM Tivoli Resource Model Builder . 11
Running IBM Tivoli Resource Model Builder . 12

Chapter 4. Designing a resource model 13
Resource model concepts ................ 13
  Common Information Model (CIM) classes . 13
    Data collection .......................... 14
    Synchronous data collection .......... 14
    Asynchronous data collection ....... 14
    Data filtering ......................... 14
  Cycles .................................. 14
  Events and indications ................ 14
  Attributes ................................ 15
  Actions .................................. 15
  Thresholds ................................ 16
  Parameters ................................ 16
    Unrestricted list ..................... 16
    Restricted list ....................... 16
  Logging .................................. 16
  Decision tree script ................... 18
  Dependencies ............................ 18
  Preliminary steps ....................... 19
  Designing a resource model tutorial ........ 19
  Gathering data for problem analysis ...... 19
  Determining which resources to monitor .... 19
  Defining the resource model .......... 20
    Defining a cycle time .................. 20
    Defining a filter and a sort order .... 20
    Defining the problems ................. 20
  Defining thresholds and parameters .... 21
  Defining the data to log ............... 22
  Defining the monitoring algorithm ...... 22

Chapter 5. Creating a resource model 23
About resource model names ............ 23
Creating a basic resource model ....... 24
  Adding a CIM data source ............. 26
  Adding a DM Classic data source ...... 27
  Adding a Custom Script data source ... 27
  Selecting a class ....................... 28
  Selecting properties ................... 31
  Selecting monitoring sources .......... 32
  Specifying a custom script ............ 33
  Applying filtering conditions ......... 34
  Specifying event triggering conditions . 35
  Selecting the properties to log ...... 37
  Changing the internal name .......... 38
  Entering a cycle time ................... 39
  Specifying the project file name ...... 39
  Importing a Tivoli Distributed Monitoring
    (Classic Edition) collection .......... 40
Creating an empty resource model ..... 41

Chapter 6. Editing resource models . . 45
Using the Editing tool .................. 45
  Editing general settings ............... 45
  Editing CIM classes ..................... 47
  Editing DM classic probes .............. 50
  Editing events ......................... 51
  Adding actions .......................... 54
  Editing thresholds ...................... 55
  Editing parameters ..................... 56
  Editing data to log ..................... 57
  Editing dependencies ................... 59
  Editing source ......................... 60
Working with the VisitTree Wizard ...... 61
  Creating a resource model from a CIM/WMI
    class .................................. 61
  Creating a resource model from a Distributed
    Monitoring (Classic Edition) Collection data
    source ................................ 65
Creating a resource model from a custom script . 67

Chapter 7. Testing and debugging a
resource model .......................... 71
Running a resource model ............... 71
Using the JavaScript Console .......... 73
Using the RMB Event Viewer ........................................ 74
Using the RMB Indication Viewer .................................. 76
Using the RMB Logging Viewer ....................................... 77
Debugging a JavaScript resource model ............................... 78
Debugging a Visual Basic resource model ............................... 79
Testing using the M12 Java-based data provider .................. 80

Chapter 8. Building a resource model .......................... 83
Generating the Package .................................................... 83
Generating resource model settings in XML ......................... 84
Generating the Java Message Catalog ................................. 85
Generating the Message Catalog ....................................... 86
Generating the Tivoli Enterprise Console BAROC ............... 87
Generating an IBM Tivoli Monitoring configuration file ....... 87
Generating a MOF file. .................................................. 88
Generating a script file ................................................ 88
Generating HTML documents .......................................... 89

Appendix A. Command-line options .............................. 91
Tivoli command syntax .................................................. 91
Using commands with IBM Tivoli Resource Model Builder ....... 91
List of commands ....................................................... 92
addInterp ............................................................... 93
bdamepkg ............................................................ 94
bdldpkg .............................................................. 95
bldebaroc ............................................................. 96
convert ................................................................. 97
expconf ................................................................. 98
expmsgcat ........................................................... 99
expmsgcat .......................................................... 100
expxmlxml .......................................................... 101
extract ................................................................. 102
impxmlxml ........................................................... 103
pack ................................................................. 104
setcopyright ......................................................... 105
setdectree .......................................................... 106
setdep .............................................................. 107
setsedotbms ......................................................... 108
setver ............................................................... 109

Appendix B. Service object method library ..................... 111
Basic Object Methods ................................................ 111
   General settings ..................................................... 111
      Method GetModelName ............................................ 112
      Method GetCycleTime ............................................ 112
      Example .......................................................... 112
   Dynamic model .................................................... 112
      Method GetNumOfInst ............................................ 112
      Method GetNumProperty ......................................... 113
      Method GetStrProperty .......................................... 113
      Method AssociateParameterToClass ............................. 113
      Method AssociateParameterToClassProperty ................. 114
      Method CallDMNumProbe ........................................ 114
      Method CallDMStrProbe ......................................... 115
      Example .......................................................... 115
   Thresholds ........................................................ 116
      Method GetThreshold .............................................. 116
   Parameters ........................................................ 117
      Method GetNumParameterCount .................................. 117
      Method GetStrParameterCount .................................. 117
      Method GetNumParameter ........................................ 117
      Method GetStrParameter ......................................... 118
      Example .......................................................... 118
   Events ............................................................. 119
      Method SendEventEx ............................................... 119
      Example .......................................................... 119
   Logging ........................................................... 120
      Method LogInstEx .................................................. 120
   Utilities .......................................................... 121
      Method Trace ...................................................... 121
      Method GetInterp ................................................ 121
      Method Shell ...................................................... 121
      Method GetShellStdOut ......................................... 122
      Method GetShellStdErr ........................................ 122
      Method GetShellRetCode ....................................... 123
      Example .......................................................... 123
   Mapping tables ..................................................... 124
      Method CreateMap ................................................. 124
      Method SetMapNumElement ..................................... 124
      Method SetMapStrElement ...................................... 124
      Method GetMapNumValue ....................................... 125
      Method GetMapStrValue ........................................ 125
      Method RemoveMapElement ..................................... 125
      Method RemoveMapAll .......................................... 126
      Method ExistsMapElement ..................................... 126
      Method DestroyMap ............................................. 126
      Example .......................................................... 127
   Advanced Object Methods ....................................... 127
      General settings ................................................ 127
         Method SetModelName .......................................... 128
         Method SetCycleTime .......................................... 128
         Example ........................................................ 128
      Dynamic Model ............................................... 128
         Method DefineClass ............................................ 128
         Method DefineClassAsync ................................... 129
         Method RemoveClass .......................................... 130
         Method CollectData ........................................... 130
         Method CollectClassData .................................... 131
         Method DefineDMNumProbe ................................ 131
         Method DefineDMStrProbe ................................... 131
         Example ........................................................ 132
      Thresholds ....................................................... 132
         Method DefineThreshold ....................................... 132
         Example ........................................................ 132
      Parameters ....................................................... 133
         Method DefineStrParameter .................................. 133
         Method DefineNumParameter .................................. 133
         Method AddStrParameter ...................................... 133
         Method AddNumParameter ..................................... 134
         Method RemoveStrParameter ................................ 134
         Method RemoveNumParameter ................................ 134
         Method ReplaceStrParameter ................................ 135
         Method ReplaceNumParameter ................................ 135
         Example ........................................................ 136
      Events .......................................................... 136
         Method DefineEvent ............................................. 136
         Example ........................................................ 137
Logging .......................... 137
    Method DefineLogInst  ........ 137
Generic functions .................. 137
    Method EndVisit  ............. 137
    Method Dispose  .............. 138
    Method GetProfileName  ...... 138
    Method GetGlobalCounter  ... 138
Deprecated methods ................. 138
Exceptions ........................ 138

Àppendix C. Instrumentation Library
Type Interface ..................... 143
    ILT Public Operations  ... 143
        enumerateInstances  .. 143
    getProperty  .............. 143
    getMultipleProperties  .. 144
    setProperty  ............. 144
    invokeMethod  ............ 145
    invokeMethod  ............ 146
    create  .................. 147
    destroy  ................ 147
ILT Support Classes ................ 148
    M12ClassPath  ............. 148
    M12IdentityElement  .... 148
M12ObjectIdentity .................. 148
    M12PropertySet  .......... 148
    M12Exception  ............ 149
    ParameterSet  ............ 149
    ParameterSetList  ....... 149
Writing a provider for UNIX ....... 149
    Creating a MOF file for UNIX  149
ILT Sample  .................... 150
Supported CIM Types ............... 152

Àppendix D. Resource Model
Troubleshooting .................... 153

Àppendix E. Accessibility .......... 155
    Using assistive technologies  155
    Magnifying what is displayed on the screen  155
    Documentation in accessible formats  155
    Using alternative text  ......... 155

Àppendix F. Notices ................. 157
    Trademarks  ................ 159

Index ............................. 161
### Tables

1. Quick-reference information for getting started with IBM Tivoli Resource Model Builder ........................................ 3
2. Guidelines for installation and configuration ........................................ 5
3. Hardware requirements description ........................................ 6
4. Designing resource model concepts ........................................ 13
5. Creating resource model concepts ........................................ 23
6. Editing resource model tools ........................................ 45
7. VisitTree Wizard concepts ........................................ 61
8. Testing and debugging resource model concepts ........................................ 71
9. Building resource model concepts ........................................ 83
10. Command quick reference table ........................................ 92
11. Visual Basic and JavaScript example comparisons ........................................ 111
12. Visual Basic and JavaScript language comparisons ........................................ 111
13. Basic Object Method general settings examples ........................................ 112
14. Basic Object Method dynamic model examples ........................................ 115
15. Basic Object Method threshold examples ........................................ 116
16. Basic Object Method parameter examples ........................................ 118
17. Basic Object Method event examples ........................................ 119
18. Basic Object Method utility examples ........................................ 123
19. Basic Object Method mapping examples ........................................ 127
20. Advanced Object Method general settings examples ........................................ 128
21. Advanced Object Method dynamic model examples ........................................ 132
22. Advanced Object Method threshold examples ........................................ 132
23. Advanced Object Method parameter examples ........................................ 136
24. Advanced Object Method event examples ........................................ 137
25. Method exception list ........................................ 139
About this guide

The IBM Tivoli Resource Model Builder User’s Guide describes how to use the IBM® Tivoli® Resource Model Builder software to create, modify, debug, and package resource models for use with IBM Tivoli Monitoring products.

Who should read this guide

This book is for system administrators and database administrators. It explains the concepts you should know to use the IBM Tivoli Resource Model Builder product. It contains information about how IBM Tivoli Resource Model Builder is integrated into the Tivoli environment.

Readers should be familiar with the following:

- Windows® or UNIX® operating systems
- Tivoli environment
- IBM Tivoli Monitoring

What this guide contains

This guide contains the following sections:

- **Chapter 1, “Overview,”** on page 1
  Provides an overview of IBM Tivoli Resource Model Builder features.
- **Chapter 2, “Getting started quick-reference guide,”** on page 3
  Provides a quick-reference overview of IBM Tivoli Resource Model Builder procedures.
- **Chapter 3, “Installation,”** on page 5
  Provides installation steps to install IBM Tivoli Resource Model Builder.
- **Chapter 4, “Designing a resource model,”** on page 13
  Provides information on designing a resource model.
- **Chapter 5, “Creating a resource model,”** on page 23
  Provides instructions on how to create a resource model.
- **Chapter 6, “Editing resource models,”** on page 45
  Provides instructions on how to edit resource models.
- **Chapter 7, “Testing and debugging a resource model,”** on page 71
  Provides information on testing resource models.
- **Chapter 8, “Building a resource model,”** on page 83
  Provides instructions on how to build resource models.
- **Appendix A, “Command-line options,”** on page 91
  Provides command information.
- **Appendix B, “Service object method library,”** on page 111
  Provides information about service objects.
- **Appendix C, “Instrumentation Library Type Interface,”** on page 143
  Provides information about the ILT library and writing a provider in UNIX.
Publications

This section lists publications in the IBM Tivoli Resource Model Builder library and related documents. It also describes how to access Tivoli publications online and how to order Tivoli publications.

IBM Tivoli Resource Model Builder library

The following guides are available in the IBM Tivoli Resource Model Builder library:

  Describes how to install and configure IBM Tivoli Resource Model Builder.
  Provides the latest information about known product limitations and workarounds.

Related publications

To use the information in this book effectively, you must have some prerequisite knowledge, which you can find in the following books:

- *IBM Tivoli Monitoring User's Guide*
  Provides detailed information about using IBM Tivoli Monitoring.
- *Tivoli Management Framework User's Guide*
  Provides detailed information about profiles and profile management.
- *Tivoli Management Framework Planning and Installation Guide*
  Provides detailed information about server and hardware requirements.
- *Tivoli Management Framework Reference Guide*
  Provides additional information about command-line commands.
- *Tivoli Enterprise Console User's Guide*
  Provides more detailed information about using the Tivoli Enterprise Console.
- *Tivoli Business Systems Manager User's Guide*
  Provides more detailed information about using Tivoli Business Systems Manager.

The following documents also provide useful information:

The *Tivoli Software Glossary* includes definitions for many of the technical terms related to Tivoli software. The *Tivoli Software Glossary* is available, in English only, at the following Tivoli software library Web site:


Access the glossary by clicking the Glossary link on the left panel of the Tivoli software library window.

Accessing publications online

IBM posts publications for the IBM Tivoli Resource Model Builder at the following Web address:

This documentation is also available, along with other Tivoli product documentation, on the Tivoli Software Information Center Web site. Access the Tivoli Software Information Center by first going to the Tivoli software library at the following Web address:


Scroll down and click the **Product manuals** link. In the Tivoli Technical Product Documents Alphabetical Listing window, click the **IBM Tivoli Monitoring** link to access the product library at the Tivoli Information Center.

**Note:** If you print PDF documents on other than letter-sized paper, select the **Fit to page** check box in the Adobe Acrobat Print window. This option is available when you click **File → Print**. **Fit to page** ensures that the full dimensions of a letter-sized page print on the paper that you are using.

### Ordering publications

You can order many Tivoli publications online at the following Web site:


You can also order by telephone by calling one of these numbers:

- In the United States: 800-879-2755
- In Canada: 800-426-4968

In other countries, see the following Web site for a list of telephone numbers:

http://www.ibm.com/software/tivoli/order-lit/

### Accessibility

Accessibility features help users with a physical disability, such as restricted mobility or limited vision, to use software products successfully. With this product, you can use assistive technologies to hear and navigate the interface. You can also use the keyboard instead of the mouse to operate all features of the graphical user interface.

For additional information, see the Appendix E, “Accessibility,” on page 155.

### Contacting software support

If you have a problem with any Tivoli product, refer to the following IBM Software Support Web site:


If you want to contact software support, see the **IBM Software Support Guide** at the following Web site:

http://techsupport.services.ibm.com/guides/handbook.html

The guide provides information about how to contact IBM Software Support, depending on the severity of your problem, and the following information:

- Registration and eligibility
• Telephone numbers and e-mail addresses, depending on the country in which you are located
• Information you must have before contacting IBM Software Support

Conventions used in this book

This guide uses several conventions for special terms and actions, operating system-dependent commands and paths, and margin graphics.

Typeface conventions

This guide uses the following typeface conventions:

**Bold**

- Lowercase commands and mixed case commands that are otherwise difficult to distinguish from surrounding text
- Interface controls (check boxes, push buttons, radio buttons, spin buttons, fields, folders, icons, list boxes, items inside list boxes, multicolumn lists, containers, menu choices, menu names, tabs, property sheets), labels (such as Tip, and **Operating system considerations**)
- Keywords and parameters in text

*Italic*

- Words defined in text
- Emphasis of words (words as words)
- New terms in text (except in a definition list)
- Variables and values you must provide

**Monospace**

- Examples and code examples
- File names, programming keywords, and other elements that are difficult to distinguish from surrounding text
- Message text and prompts addressed to the user
- Text that the user must type
- Values for arguments or command options

Operating system-dependent variables and paths

This guide uses the UNIX convention for specifying environment variables and for directory notation.

When using the Windows command line, replace `$variable` with `%variable%` for environment variables and replace each forward slash (/) with a backslash (\) in directory paths.

**Note:** If you are using the bash shell on a Windows system, you can use the UNIX conventions.
Chapter 1. Overview

IBM Tivoli Resource Model Builder is a programming tool for creating, modifying, testing, and packaging resource models for use with IBM Tivoli Monitoring products.

IBM Tivoli Resource Model Builder enables you to specify events, parameters, and thresholds for existing and new resource models. You can identify problems in real time, notify administrators, and take autonomic, corrective action.

New features for IBM Tivoli Resource Model Builder, Version 1.1.3 include the ability to create and package a resource model for the Autonomic Management Engine (AME).

Samples of Best Practice Resource Models have been provided for you to use within the IBM Tivoli Resource Model Builder. These resource models are intended to be used as working examples for creating new resource models.

About IBM Tivoli Resource Model Builder

Resource models are used at an endpoint to collect and analyze data regarding the state and performance of different resources, such as disks, memory, CPU, or applications. After you create a resource model, you add it to your Tivoli management region.

Pre-existing resource models that were created with IBM Tivoli Monitoring Workbench, Version 5.1.1, or IBM Tivoli Resource Model Builder Version 1.1.x can be used with IBM Tivoli Resource Model Builder Version 1.1.3. However, resource models cannot be loaded into previous versions of the tool.

IBM Tivoli Monitoring provides a set of default resource models. You can use IBM Tivoli Resource Model Builder to expand the collection of available resource models according to the requirements of your system.

When you create a resource model, you specify sets of values, such as the following values:

- Thresholds you do not want a resource to exceed
- Ranges of instances to limit the data collection
- Events to generate if a monitored resource state is not satisfactory

When you create a resource model, you can also specify that the monitoring results must be sent to the Tivoli Enterprise Console server or Tivoli Business Systems Manager. Event notifications enable you to perform more complete management of all your resources.

Settings that you configure are automatically included in the decision tree script. The script contains the settings and the monitoring algorithm that governs the complete process. The monitoring algorithm is developed using either Visual Basic or JavaScript™ and specifies how to use each setting to create the process more suitable to your needs.
Compatibility mode

The compatibility mode enables you to use Tivoli Distributed Monitoring (Classic Edition) monitoring sources and custom scripts inside an IBM Tivoli Monitoring resource model.

To collect data about system resources, IBM Tivoli Monitoring relies on a Common Information Model (CIM) Object Manager implementation. This implementation is Windows Management Instrumentation (WMI) for Windows and an IBM Tivoli Monitoring endpoint engine for the other supported operating systems. With the compatibility mode, IBM Tivoli Monitoring collects data not only from CIM data sources, but also from Tivoli Distributed Monitoring (Classic Edition) monitoring sources.

Common Information Model implementation

IBM Tivoli Monitoring implements the Common Information Model (CIM) standard from the Distributed Management Task Force (DMTF). CIM is a model for describing management information in a network environment. The model applies the basic techniques of the object-oriented paradigm.

To obtain data from the monitored resources, IBM Tivoli Monitoring might use processes included in the endpoint’s operating system. On Windows systems, it uses the WMI, which is Microsoft’s implementation of CIM. On UNIX, Linux, and OS/400® operating systems, the information collection agent is incorporated in the IBM Tivoli Monitoring endpoint engine based on CIM specifications.

The service object API enables the resource model scripts to use CIM objects, monitoring collections, and custom scripts in any combination. WMI (on Windows) and the CIM Object Manager (CIMOM on other supported operating systems) are responsible for loading the providers that retrieve performance and availability data from system and application resources. At the provider layer, an Instrumentation Library Type (ILT) interface is provided, using a parameter-passing mechanism. The CIM provider is implemented by an ILT Manager for Java®.

IBM Eclipse SDK

IBM Eclipse Software Developer’s Kit (SDK) is compatible with Eclipse and can be installed and used with any open source and commercial Eclipse-based offering. Eclipse is an award-winning, open source framework for the construction of powerful software development tools and rich desktop applications. Leveraging the Eclipse plug-in framework to integrate technology on the desktop saves technology providers time and money by enabling them to focus their efforts on delivering differentiation and value for their offerings. Eclipse is a multi-language, multi-platform, multi-vendor supported environment that it is built by an open source community of developers and it is provided royalty-free by the Eclipse Foundation. Eclipse is written in the Java language, includes extensive plug-in construction toolkits and examples, and can be extended and deployed on a range of desktop operating systems including Windows, Linux, QNX and Macintosh OS X. Full details on Eclipse and the Eclipse Foundation are available at http://www.eclipse.org.
# Chapter 2. Getting started quick-reference guide

This chapter provides a quick-reference table of the procedures you can perform with IBM Tivoli Resource Model Builder. The table below includes cross-references to the location of additional information about the procedures. The procedures are listed in a suggested order, but do not have to be followed in this order.

<table>
<thead>
<tr>
<th>Goal</th>
<th>Refer to</th>
</tr>
</thead>
</table>
| **1.** Prepare to create a resource model by performing the following actions:  
  • Review resource model concepts  
  • Define what to monitor  
  • Practice with the tutorial |  
  • “Resource model concepts” on page 13  
  • “Preliminary steps” on page 19  
  • “Designing a resource model tutorial” on page 19 |
| **2.** Create a resource model using one of the following wizards:  
  • Basic Resource Model Wizard to create a complete resource model without writing any scripts or code.  
  —OR—  
  • Empty Resource Model Wizard to create an empty resource model that you can configure at a later time. |  
  • “Creating a basic resource model” on page 24  
  —OR—  
  • “Creating an empty resource model” on page 41 |
| **3.** *(Optional)* Edit your resource model using one of the following tools:  
  • Editor tool to edit or configure your resource model in no particular order.  
  —OR—  
  • VisitTree Wizard to use a wizard to guide you through resource model configuration. |  
  • Chapter 6, “Editing resource models,” on page 45  
  —OR—  
  • “Working with the VisitTree Wizard” on page 61 |
| **4.** Test and debug your resource model so you can correct any errors before building and deploying the resource model. Certain errors can prevent you from building the resource model. | Chapter 7, “Testing and debugging a resource model,” on page 71 |
| **5.** Build your resource model so you can:  
  • Install it on the server  
  • View events in the Tivoli Enterprise Console  
  • Display information in IBM Tivoli Monitoring windows  
  • Generate a MOF file  
  • Generate a JavaScript or Visual Basic Script file | Chapter 8, “Building a resource model,” on page 83 |
Chapter 3. Installation

This chapter provides the following installation requirements and procedures for IBM Tivoli Resource Model Builder:

Table 2. Guidelines for installation and configuration

<table>
<thead>
<tr>
<th>Goal</th>
<th>Refer to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Verify your environment meets the software requirements.</td>
<td>“Software requirements” on page 5</td>
</tr>
<tr>
<td>Verify your environment meets the hardware requirements.</td>
<td>“Hardware requirements” on page 6</td>
</tr>
<tr>
<td>Install IBM Tivoli Resource Model Builder, Version 1.1.3 using one of the following options:</td>
<td>Perform one of the following procedures:</td>
</tr>
<tr>
<td>• Install IBM Tivoli Resource Model Builder, Version 1.1.3 with IBM Eclipse SDK if you want to install both products into your environment. —OR—</td>
<td>• “Installing IBM Tivoli Resource Model Builder with IBM Eclipse SDK” on page 7</td>
</tr>
<tr>
<td>• Install only the IBM Tivoli Resource Model Builder, Version 1.1.3 for environments that already have IBM Eclipse SDK installed.</td>
<td>• “Installing IBM Tivoli Resource Model Builder” on page 8</td>
</tr>
<tr>
<td><em>(Optional)</em> Migrate existing resource models created with previous versions of IBM Tivoli Resource Model Builder.</td>
<td>“Migrating existing resource models” on page 9</td>
</tr>
</tbody>
</table>

Software requirements

To install the IBM Tivoli Resource Model Builder, you must have one of the following software products installed and running:

• Windows 2000 Professional
  —OR—
• Windows XP Professional

IBM Tivoli Resource Model Builder is independent of the Tivoli Management Framework environment. You can install it on a computer that is not within the Tivoli Management Framework.

IBM Tivoli Resource Model Builder is a Windows-only tool. Although it does not run on UNIX or OS/400 systems, you can create resource models for UNIX and OS/400 systems.

Note that the IBM Tivoli Resource Model Builder no longer ships with certain Unix tools required to run certain Resource Models from within the IBM Tivoli Resource Model Builder. If you want to run any of the following from within the IBM Tivoli Resource Model Builder, the appropriate tool must be in the system path:

• Actions – requires Perl
Custom Script Resource Models – might require Perl or sh depending on the script
DM Classic Probes – might require many different Unix tools. View the source to determine what to add.

A complete set of Unix tools might be found using Cygwin, from http://www.cygwin.com/. Ensure that you install Perl, because it is not installed by default. After installation you will need to set your path to include the cygwin\bin directory.

For reference, the following files have been removed from the <install location>\eclipse\plugins\com.ibm.tivoli.rmb.model_1.1.3\os\win32\x86 directory:

- Awk.exe
- Basename.exe
- Bash.exe
- Cat.exe
- Cmp.exe
- Cp.exe
- Cut.exe
- Cygwin1.dll
- Cygwin19.dll
- Date.exe
- Dd.exe
- Dimname.exe
- Echo.exe
- Egrep.exe
- Env.exe
- Expr.exe
- Fgrep.exe
- Find.exe
- Gencat.exe
- Getopt.exe
- Grep.exe
- Gzip.exe
- Ls.exe
- Mkfile.exe
- Mv.exe
- Perl.exe
- Printenv.exe
- Pwd.exe
- Rm.exe
- Slashes.exe
- Sleep.exe
- Tail.exe
- Tar.exe
- Tee.exe
- Touch.exe
- Tr.exe
- Uname.exe
- Wc.exe
- Xargs.exe

---

### Hardware requirements

The following table lists the minimum hardware requirements for IBM Tivoli Resource Model Builder:

*Table 3. Hardware requirements description*

<table>
<thead>
<tr>
<th>Hardware</th>
<th>Minimum Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Disk space for installation of IBM Tivoli Resource Model Builder with IBM Eclipse SDK</td>
<td>139 MB</td>
</tr>
<tr>
<td>Disk space for installation of IBM Tivoli Resource Model Builder</td>
<td>68 MB</td>
</tr>
<tr>
<td>RAM</td>
<td>256 MB</td>
</tr>
<tr>
<td>Processor</td>
<td>600 MHz</td>
</tr>
<tr>
<td>Minimum resolution</td>
<td>1024 x 768</td>
</tr>
</tbody>
</table>
Installing IBM Tivoli Resource Model Builder with IBM Eclipse SDK

Objective
To install the following products into an environment so you can create and customize resource models:

- IBM Tivoli Resource Model Builder, Version 1.1.3
- IBM Eclipse SDK

Background information
IBM Tivoli Resource Model Builder provides two installation options for your environment. Since IBM Tivoli Resource Model Builder requires IBM Eclipse SDK, you can choose to install both products in one installation process by following this procedure.

Alternatively, you can install only the IBM Tivoli Resource Model Builder if your environment already has IBM Eclipse SDK. See “Installing IBM Tivoli Resource Model Builder” on page 8 for instructions to install IBM Tivoli Resource Model Builder only.

Required authorization
Administrator (on the operating system)

Before you begin
Complete a free registration and download the installation files from the following Web site:


Verify your environment meets the software and hardware prerequisites before beginning installation.

Note: During installation, certain installation screens might not load immediately. After clicking Next, a blank screen might appear. It may take several seconds for the next panel to appear.

When you finish
None

Procedure
1. From Windows Explorer, double-click the product folder from the directory in which it was downloaded.
2. Double-click RMB_Installer.exe to launch the installer and click Next to display the upgrade notice.
3. Select the Full Installation option.
4. Click Next to display the installation directory window and click Next to accept the default directory paths.

Additional Information: You can optionally change the installation path by typing text in the text field, or clicking Browse to browse to a directory.
5. Click **Next** to confirm the settings window and click **Next** to launch the installation wizard.

6. Complete one of the following readme options:
   - Read the Readme file for the software by doing the following:
     a. Accept the **Yes** default and click **Next** to display the Readme file.
     b. Click **Finish** to complete and close the installation.
   - OR—
   - Select **No** and click **Next** to complete and close the installation.

---

**Installing IBM Tivoli Resource Model Builder**

**Objective**
To install IBM Tivoli Resource Model Builder, Version 1.1.3 in an environment that already has IBM Eclipse SDK so you can create and customize your own resource models.

**Background information**
The following procedure lists instructions for installing the IBM Tivoli Resource Model Builder only. Follow this procedure if you already have IBM Eclipse SDK in your environment.

IBM Tivoli Resource Model Builder provides two installation options for your environment. Since IBM Tivoli Resource Model Builder requires IBM Eclipse SDK, you can choose to install both products in one installation process. If you do not have IBM Eclipse SDK in your environment, see “Installing IBM Tivoli Resource Model Builder with IBM Eclipse SDK” on page 7 for instructions on how to install both products.

**Required authorization**
None

**Before you begin**
Complete a free registration and download the installation files from the following Web site:


Verify your environment meets the software and hardware prerequisites before beginning installation.

**Note:** During installation, certain installation screens might not load immediately. After clicking **Next**, a blank screen might appear. This behavior corrects itself, and within seconds, the new window opens.

**When you finish**
None

**Procedure**
1. From Windows Explorer, select the folder to which the installation files were downloaded.
2. Double-click **RMB_Installer.exe** to launch the installer and click **Next** to display the upgrade notice.

3. Click **Next** to display the installation directory window.

4. Select the **Feature Installation** option.

5. Click **Next** to display the installation directory window and click **Next** to accept the default directory paths.

   *Additional Information:* You can optionally change the installation path by typing text in the text field, or clicking **Browse** to browse to a directory. The IBM Eclipse SDK directory contains an eclipse subdirectory that, in turn, contains an .eclipseproduct file. If the lower text field is left blank, the Resource Model Builder files are installed into the same directory as the IBM Eclipse SDK. The Resource Model Builder files can be installed into a different directory (an Eclipse extension). Specify this directory by entering a directory path in the lower box.

6. Click **Next** to confirm the settings window and click **Next** to launch the installation wizard.

7. Complete one of the following readme options:
   - Read the Readme file for the software by doing the following:
     - Accept the **Yes** default and click **Next** to display the Readme file.
     - Click **Finish** to complete and close the installation.
       —OR—
   - Select **No** and click **Next** to complete and close the installation.

---

### Migrating existing resource models

#### Objective

To migrate existing resource models so you can use them with IBM Tivoli Resource Model Builder, Version 1.1.3.

#### Background information

You can migrate existing resource models created with IBM Tivoli Monitoring Workbench, Version 5.1.1. By migrating existing resource models, you can use the same monitoring configurations from your previous monitoring environment.

#### Required authorization

None

#### Before you begin

You must delete any spaces in your existing resource model names before migration. Migrating a resource model name with spaces generates an error.

You must create a Tivoli Management Project before you can migrate existing resource models. Perform the following steps to create a Tivoli Management Project:

1. Select **File** from the toolbar to display the **File** drop-down list.
2. Select **New** from the toolbar to display the **New** pop-up list.
3. Select **Project** to display the New Project window.
4. Select **Tivoli Management Project** from the left field to display it in the right field.
5. Click on Tivoli Management Project in the right field to highlight the name.
6. Click Next to display the Project Name window.
7. Enter a name for the project in the Project name text box.
8. (Optional) Clear the Use default check box and specify a different directory location for the project.
9. Click Finish.

When you finish
After you have migrated your pre-existing resource models to IBM Tivoli Resource Model Model Builder, you can no longer use them in the previous tool, IBM Tivoli Monitoring Workbench.

Procedure
You can perform this procedure from the command-line interface or the IBM Tivoli Resource Model Builder.

Command line
Use the convert command to migrate resource models. For more information about the convert command, see “convert” on page 97.

Desktop
1. Click File from the toolbar to display the File drop-down menu.
2. Click Import to display the Import window.
3. Click Old ITM Workbench Files to highlight the file name.
4. Click Next to display the file selection window.
5. Click **Browse** to select the file name and location.

6. Click **Open** to display the selected file in the **File to import** field.

7. Click **Next** to display the new file name window.

8. Select a folder by doing one of the following:
   - Type the folder name in the **Enter or select the folder** field.
   - OR—
   - Double-click on a folder name to display the folder in the **Enter or select the folder** field.

9. Click **Finish** to import the file.

---

**Uninstalling IBM Tivoli Resource Model Builder**

**Objective**

To uninstall IBM Tivoli Resource Model Builder, Version 1.1.3 so you can remove the software from your environment.

**Background information**

None

**Required authorization**

None
Before you begin

None

When you finish

Three MOF files required by IBM Tivoli Monitoring remain in the WMI after you remove the software. These MOF files (TMW_base10.mof, TMW_Resources10.mof, and ILTManagerforJava.mof) are added to WMI during installation. They provide the WMI classes for the standard Tivoli WMI interfaces.

If you do not have IBM Tivoli Monitoring or IBM Tivoli Monitoring components on your endpoints or system, you can optionally choose to remove these files manually. If you removed the files, but need to them replaced in your environment, you can reinstall IBM Tivoli Resource Model Builder to replace these files.

Procedure

Do the following to uninstall IBM Tivoli Resource Model Builder, Version 1.1.3:

1. From the Control Panel, open the Add/Remove Programs window to display the list of installed programs.
2. Click on Resource Model Builder 1.1.3 for IBM Eclipse SDK to highlight the program.
3. Click Change/Remove to launch the wizard and click Next to display the features to uninstall window.
   Additional Information: By default, all features are selected. Clearing one option automatically clears all of the options.
4. Click Next to confirm product installation and click Next to remove the software and display the completion window.
5. Click Next to close the wizard.

Running IBM Tivoli Resource Model Builder

IBM Tivoli Resource Model Builder requires you to have Administrator authorization for the following reasons:

• Consistency of the running environment for the distributed resource model and, therefore, the ability to load, test, and run a resource model that is designed within the Resource Model Builder.
• Ability to build and design the resource model while accessing CIM/WMI, which requires Administrator authorization.
• Ability to access parts of the operating system, which require Administration authorization. The Resource Model Builder is installed on a development system and designed for developers who require this type of access.
Chapter 4. Designing a resource model

The following chapter describes how to design a resource model. Before you create a resource model, you must have a clear idea of your objectives and prerequisites. Review the information in this chapter to help you clarify the objectives and prerequisites of your environment.

The following is an overview table of designing resource model concepts:

Table 4. Designing resource model concepts

<table>
<thead>
<tr>
<th>Goal</th>
<th>Refer to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reviewing resource model concepts</td>
<td>“Resource model concepts” on page 13</td>
</tr>
<tr>
<td>Completing preliminary steps</td>
<td>“Preliminary steps” on page 19</td>
</tr>
<tr>
<td>Designing a sample resource model</td>
<td>“Designing a resource model tutorial” on page 19</td>
</tr>
</tbody>
</table>

Resource model concepts

Resource models represent the core of the IBM Tivoli Resource Model Builder. In a resource model, you can specify the resources you want to monitor at run time, the resource data to be collected, and the monitoring algorithm to gather and analyze the data.

Resource models contain the following elements:

- Common Information Model (CIM) classes
- Cycles
- Events
- Thresholds
- Parameters
- Logging
- Decision tree script
- Dependencies

Common Information Model (CIM) classes

The Common Information Model (CIM) class is considered the essential element of a resource model. The CIM classes determine:

- The resources to monitor
- The properties of each resource
- The mechanism to collect the data

CIM classes describe the status of each resource. A CIM class contains a list of properties relevant to a specific resource. The IBM Tivoli Resource Model Builder shows the list of available CIM classes contained in the CIM repository of WMI. You can select which CIM classes you would like to include in your resource model from this list. If the CIM class you need is not in this list, for example, if you are creating a resource model for a non-Windows operating system, you can easily add it. For more information, see “Selecting a class” on page 28.
The instances of a CIM class, as well as the implementation that retrieves the data, are provided by an associated data provider. On Windows, this data provider can be a WMI provider (a DLL file), or a Java provider (a JAR file).

**Data collection**

Resource models collect data in two modes: synchronous and asynchronous.

**Synchronous data collection:** Synchronous data collection is the routing of data to the resource model as soon as it is collected. By default, data is collected at each cycle, routed to the resource model, and analyzed. Data collection can span multiple cycles.

**Data sorting:** With synchronous data collection, data is sorted in either ascending (from the lowest to the highest) or descending (from the highest to the lowest) order. When data needs to be sorted, you must specify a rule by which to sort the data.

**Asynchronous data collection:** Asynchronous data collection means that data collection starts within one cycle and continues into the following cycle, when the data is made available. Between two cycles, the collected data is stored in cache. Typically, you use an asynchronous collection when monitoring asynchronous resources, like the Windows Event Log.

Asynchronous collection is available for Windows resource models only. It is not supported on the other operating systems.

**Data filtering:** You can filter the monitored data by writing a WMI Query Language (WQL) query, specifying that the resource model collect only those instances that satisfy certain conditions. WQL is a query language used by WMI. For more information see the following WMI Web site:


Filtering is available for Windows resource models only. It is not supported on the other operating systems.

**Example:** You can specify that you want to collect only process instances where CPU usage is greater than 60%. In this case, you can write a WQL WHERE clause that reads:

"where PercentProcessorTime > 60"

The value expressed in this query is fixed and cannot be changed from IBM Tivoli Monitoring windows.

**Cycles**

When a resource model runs on an endpoint, it gathers data at regular intervals, known as cycles. See “Defining a cycle time” on page 20 for more information.

**Events and indications**

An indication occurs when the state of a given resource meets defined criteria. By itself, an indication does not trigger any specific action. Indications are aggregated to become an event.
An event is a notification of change in the status of a resource. The event notifies
the system administrator about the state of a specific resource. Events can trigger
an action, and, if enabled, send a notification to the Tivoli Enterprise Console
server or the Tivoli Business Systems Manager. When you define an event, you
also specify whether these indications must be consecutive, or whether the
sequence can be interrupted by one or more monitoring cycles that do not register
any indication (holes).

When you define the number of holes, you specify an acceptable and consecutive
number of cycles in which no indication occurs. If this number is exceeded, the
aggregation rule breaks and accumulation of indications restarts at zero.

Attributes
Attributes can be string or numeric values, and can be chosen according to the
information that is collected by the resource model. You can associate relevant
attributes to an event.

A key attribute is the most representative attribute and uniquely identifies the event
to which it refers. Key attributes play a fundamental role in the event aggregation
process.

For example, an event might indicate that disk space is not sufficient. By specifying
attributes, such as available disk space, you can generate a more precise indication
of the problem.

When selecting an attribute name, choose a name that does not match an attribute
name in the base Tivoli Enterprise Console event slot (severity names can match,
however). Duplicate attribute names generate an error when the resource model is
imported to IBM Tivoli Monitoring.

Actions
You can configure one or more recovery actions to correct a monitored issue in
your environment. When configured, these actions automatically trigger when a
specific event occurs and typically restore a satisfactory system service level. The
system provides notification when the action successfully completes.

You can associate three types of actions with an event: programs and CIM
methods.

Programs
On all supported platforms, you can run a program as an action. You can
also indicate a set of arguments, such as environment variables, that are
associated with the execution of the program.

CIM methods
On Windows-based resource models, you can use a CIM class method as
an action. In CIM, each class has attributes that characterize the object and
a set of methods that are actions related to that object. Actions invoke
methods that are provided by WMI. An action can invoke one method at a
time.

When you select a class within the IBM Tivoli Resource Model Builder, the
related methods that are defined by WMI are automatically displayed. A
method can be static or non-static. A static method is defined with respect
to a class. A non-static method is defined with respect to an instance.
For example, if you are monitoring Windows services, and an event generates when a service stops, you can associate the event with a recovery action that restarts the interrupted service.

You can implement a CIM method within the Java IBM Tivoli Monitoring Engine (UNIX). The Instrumentation Library Type (ILT) needs to support the InvokeMethod ILT public operation.

**Thresholds**

A threshold is a numeric value that is specified in the customization phase and used according to the monitoring algorithm written in the script. By default, IBM Tivoli Resource Model Builder specifies a threshold value, but you can modify the value from within the IBM Tivoli Monitoring windows when defining a profile. Typically, this value represents a limit for a satisfactory resource state. If the monitored resource exceeds this limit, an indication occurs.

For example, if you are monitoring disk space and do not want it to drop under 70%, you can set 70 as the threshold. The system generates an indication each time disk space is less than 70%.

Within each threshold, you can add a description to explain how each value is used within the monitoring algorithm and the logic of the specific value.

**Parameters**

While thresholds are only numeric values, parameters can be lists of numbers or strings. Parameters represent the instances you want to monitor or a limit you do not want your resource to exceed, depending on how you define this setting in your script. Parameters display in the IBM Tivoli Monitoring windows.

Within the IBM Tivoli Resource Model Builder, you can define default values and let another user customize the settings in the IBM Tivoli Monitoring profile. When you specify the parameters, choose from the following types of lists:

- **Unrestricted List**
- **Restricted List**

**Note:** Regardless of the type of list you choose, IBM Tivoli Resource Model Builder displays the list content in the script as either a numeric or string list.

**Unrestricted list**

Unrestricted lists allow you to restrict the types of values accepted to only strings or only numbers. String lists can be used to specify parameter values such as application names. Numeric lists can be used to define event IDs or port numbers.

**Restricted list**

Restricted lists allow you to specify whether a user will be able to select one or more values from the parameter list on the IBM Tivoli Monitoring Desktop and allows you to enable or disable a description field for each parameter in the list.

**Logging**

Logging enables you to store data regarding the attributes of a resource. A local database stores the data and can be accessed using the IBM Tivoli Monitoring Web Health Console. For more information, see the *IBM Tivoli Monitoring User’s Guide*. 

---

**Note:** You can specify what and how to log within IBM Tivoli Resource Model Builder. However, by default, logging is disabled in the IBM Tivoli Monitoring profile. You must enable logging in the IBM Tivoli Monitoring profile in order to log information.

The logging function contains the following elements:

**Context**
A general problem to which the resource activity relates, such as disk space.

**Resource**
The state of the resource that you want to log, in relation to the defined context, such as a logical disk. A single context can contain more than one resource.

**Properties**
Specific attributes of the defined resource, such as free space on the logical disk. For each resource, you can specify multiple properties, which can be both numeric and string values. For the specified properties, you must define key properties that clearly identify the instance of a resource. When the property being logged is numeric, you can select one of the following property units to associate with each instance:
- Percent
- MSec
- HSc
- Second
- Minute
- Hour
- Day
- Bytes/Sec
- Megabytes/Sec
- Kilobytes/Sec
- Reads/Sec
- Queries/Sec
- Queries/Min
- Quantity
- Gigabyte
- Megabyte
- Kilobyte
- Byte
- 4 Kilo Pages/Sec
- Other

By default, any logged numeric property has the unit *Other* assigned to it. You do not need to change this to a different unit.

The database browser in the IBM Tivoli Monitoring Web Health Console shows historical (logged) data for a specific resource model on an endpoint. With logged data, you can identify specific resource problems over the past 6, 12, or 24 hours.

For more information, see the IBM Tivoli Monitoring Web Health Console documentation in the *IBM Tivoli Monitoring User’s Guide*. 
Decision tree script

IBM Tivoli Resource Model Builder automatically generates a decision tree script. The decision tree script contains the information you specified when you created and configured your resource model. If you change one of the settings in the IBM Tivoli Resource Model Builder editor panels, the corresponding data, if any, is automatically updated in the script. You cannot modify these settings directly in the script.

The decision tree script contains three basic functions and one subroutine that, by default, produce a return value equal to 0. However, you can specify different values to be returned under specific circumstances.

The decision tree script includes the following default functions:

**Main** IBM Tivoli Resource Model Builder uses a subroutine to debug the resource model. (The IBM Tivoli Monitoring monitoring engine does not call it.) This main routine is only created in VBA not the JavaScript. This subroutine performs the following actions:

- Creates the TMWService.Utilities object
- Calls the SetDefaultConfiguration function
- Calls the Init function
- Enters the monitoring loop (collect data, run VisitTree script, and sleep for a cycle time)

**Note:** Do not modify this subroutine. Modifying this subroutine produces unpredictable results.

**SetDefaultConfiguration**

Initializes the object based on the settings defined in the Events, Thresholds, Parameters, and Actions windows. If you change a setting in these windows, this function updates the corresponding data. You cannot directly change the settings in this function. SetDefaultConfiguration is called only once, when the resource model starts. If necessary, you can write additional initialization code at the end of this function.

**Init** Called when the settings defined in SetDefaultConfiguration are replaced with values specified in IBM Tivoli Monitoring profiles. The new values replace the previous settings.

**VisitTree**

Contains the monitoring algorithm and is called at each cycle. The VisitTree checks and implements the algorithm. It processes the collected data according to thresholds and parameters settings, and, if necessary, sends an event. You can define the algorithm in such a way that the execution of the data collection is entirely controlled by the algorithm itself. For example, you can run a specific collection based upon the results of a previous collection.

**Dependencies**

In some cases, you might need to transfer additional files to run a resource model on an endpoint. For example, to add a new class to the CIM repository, you need the files that define the class (MOF) and the provider (DLL for WMI Provider and JAR for Java Provider). A **MOF file** is a text file that contains definitions of classes and instances using the Managed Object Format (MOF) language.
In these cases, you can add dependencies to your resource model, and transfer the required files with the resource model.

### Preliminary steps

Before you create a resource model, you must first create an overall design, determine the objectives and functions of your resource model and identify some general specifications. Perform the following tasks to design your resource model:

1. Gather data for problem analysis to determine what you want your resource model to monitor.
2. Determine which resources you want to monitor.
3. Clarify the settings for the resource model you are going to create.
4. Define the problems you want the resource model to monitor.
5. Define a list of thresholds and parameters to use as external inputs.
6. Define the monitoring algorithm (draw a simple flow chart of the decision tree script).
7. Define what data you want to log.

After you have performed these tasks, you can proceed with creating the resource model.

### Designing a resource model tutorial

The following scenario guides you through the creation of a resource model on a Windows system.

In this example, a server is configured with one processor, and you want to determine if the CPU has enough capacity. Therefore, the objective of the resource model is to monitor CPU usage.

#### Gathering data for problem analysis

Determine the types of data needed to make a proper assessment of CPU usage. Gather information about the processor capacity and the CPU capacity used by the most frequently run processes.

#### Determining which resources to monitor

Consider what type of resource monitoring provides the required information. Windows resource models access data provided by WMI, so you must determine if WMI defines the resource you want to monitor. In our example, you need the following data about the processor and its processes:

**Processor**
- Current CPU usage
- Processor type

**Processes**
- Process name
- Process ID
- Process CPU usage

In this example, WMI provides all of the above data.
Defining the resource model

Determine in more detail the objectives of what you must monitor and what attributes are required to meet the objectives. Many of these values are default values that you can change later in the IBM Tivoli Monitoring profiles. Whenever possible, use the default values.

In this example, you can expand upon your overall objective of optimizing CPU usage by monitoring which processes are monopolizing resources. You can also examine why and when these processes require a large percentage of CPU.

Defining a cycle time

Define how often you want the resource model to cycle through the resources to collect data. Minimize the impact of the monitoring activity on system performance by selecting an appropriate cycle time. Consider two main factors: how fast a given resource changes status, and how the cycling increases overhead on the system. In this example, the resources change their status quite frequently. A cycle time of 60 seconds is a reasonable compromise between monitoring status variations and saving system performance.

Defining a filter and a sort order

Note: This feature is available on Windows resource models only.

In this example, you do not need to examine processes with low CPU consumption because they do not affect CPU performance too severely. You can apply a filter to our data collection so that you collect data only for the processes using at least 20% of CPU capacity.

You can also sort the results in ascending order of CPU usage. You can configure our resource model to monitor only the specific processes that exceed a threshold. If none of our processes exceed the 20% CPU usage threshold, no processes are monitored. This configuration optimizes system performance during monitoring.

Defining the problems

Identify the potential problems and assign a name to the events generated by each problem. In this example, you want to identify the following problems:

- Excessive CPU usage
- Excessive process consumption

Configure events to notify you of persistent indications of the problem. Define the number of occurrences that generate events. In this example, 10 consecutive occurrences of either problem generates an event.

To qualify each event in more detail, examine the attributes to associate with an event. By defining attributes, you can obtain specific information regarding the presence of a problem as well as the possible causes. For each event in this example, the following attributes provide useful information:

<table>
<thead>
<tr>
<th>Event</th>
<th>Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>High CPU usage</td>
<td>Current CPU usage</td>
</tr>
<tr>
<td></td>
<td>High use process name</td>
</tr>
<tr>
<td></td>
<td>High use process ID</td>
</tr>
<tr>
<td></td>
<td>High use process usage</td>
</tr>
<tr>
<td></td>
<td>Processor type</td>
</tr>
</tbody>
</table>
### Event Attributes

<table>
<thead>
<tr>
<th>Event</th>
<th>Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>High use process</td>
<td>Process name</td>
</tr>
<tr>
<td></td>
<td>Process ID</td>
</tr>
<tr>
<td></td>
<td>Process CPU usage</td>
</tr>
</tbody>
</table>

In this example, assume that the **Excessive CPU usage** event returned the following results:

<table>
<thead>
<tr>
<th>Attributes</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>Current CPU usage</td>
<td>90%</td>
</tr>
<tr>
<td>High use process name</td>
<td>Mail Application</td>
</tr>
<tr>
<td>High use process ID</td>
<td>443</td>
</tr>
<tr>
<td>High use process usage</td>
<td>10%</td>
</tr>
<tr>
<td>Processor type</td>
<td>Pentium 133™</td>
</tr>
</tbody>
</table>

Based upon this data, the high use process is not affecting CPU performance. You can therefore determine that the high use process is not likely to be the cause of High CPU use. Perhaps CPU performance is low because of several concurrent processes. You can also consider the type of processor you are using. Perhaps the system load is too heavy and you need a more powerful processor.

After specifying each event with attributes, determine which attribute is the key attribute, if any. In this example, the **High CPU usage** event has no key attribute. For the **High use process** event, the key attributes are the process name and process ID.

### Defining thresholds and parameters

Specify the thresholds and parameters to govern the data collection. For more information about thresholds and parameters, see "Thresholds" on page 16 and "Parameters" on page 16.

In this example, you are using thresholds and not parameters. The following thresholds might be sufficient for this example:

<table>
<thead>
<tr>
<th>Thresholds</th>
<th>Default Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximum CPU usage</td>
<td>90%</td>
</tr>
<tr>
<td>High use process</td>
<td>25%</td>
</tr>
<tr>
<td>Processes considered</td>
<td>10</td>
</tr>
</tbody>
</table>

**Maximum CPU usage** sets an indication to occur each time the CPU usage exceeds 90%. **High use process** sets an indication to occur each time a single process uses more than 25% CPU. **Processes considered** limits the number of monitored processes to the 10 high use processes in the data collection.

You can describe the logic behind these values in the description of each threshold and give each value a function when you specify it in the decision tree script.
Defining the data to log

Define the data to log for this resource model. For this example, logging the attributes provides relevant and important data. Over time, the log provides overall information about the CPU usage.

Defining the monitoring algorithm

Determine the monitoring algorithm by drawing a flow chart that shows how the algorithm works. The flow chart reflects what you write in your script. Describe how all the components interact with each other and what the final results reveal.
Chapter 5. Creating a resource model

IBM Tivoli Resource Model Builder provides two wizards for creating new resource models. The Basic Resource Model Wizard provides a series of configuration windows with default values already filled in. You can use this wizard to create simple resource models starting from a selected CIM class taken from the WMI repository, a monitoring collection, or a custom script. The required Visual Basic or JavaScript code generates automatically.

The Empty Resource Model Wizard provides simple configuration windows so you can create an empty resource model. You can later define the resource model elements without following a specific sequence.

Creating a resource model in the IBM Tivoli Resource Model Builder produces a complete package that you can install on the Tivoli management region server through the command line.

The following is an overview table of the topics covering creating a new resource model:

<table>
<thead>
<tr>
<th>Goal</th>
<th>Refer to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reviewing resource model naming conventions</td>
<td>“About resource model names” on page 23</td>
</tr>
<tr>
<td>Creating a basic resource model so you can complete a fully configured resource model.</td>
<td>“Creating a basic resource model” on page 24</td>
</tr>
<tr>
<td>Creating an empty resource model so you can configure the resource model at any time and in no particular order.</td>
<td>“Creating an empty resource model” on page 41</td>
</tr>
</tbody>
</table>

About resource model names

Notes:

1. The decision tree script in IBM Tivoli Resource Model Builder does not recognize double-byte characters.
2. If you write text in a non-English language, then export the resource model and install it on the server, translated message catalogs need to be installed separately. For more information, see “Generating the Message Catalog” on page 86.

Resource models and their components have the following names and descriptions that are used in different contexts; for example, names used in scripts are different from those displayed in the IBM Tivoli Monitoring windows.

Internal Name

The names of your resource models, thresholds, parameters, and events as named in the decision tree script. These names are used inside a monitoring algorithm written in Visual Basic or JavaScript. The names must be alphanumeric, start with an alphabetic letter, and contain no blanks.
Descriptive Name

The names of your resource models, thresholds, parameters, events and actions as they display in the IBM Tivoli Monitoring windows and the message catalog. Use more description in descriptive names than you do for internal names. Descriptive names are not subject to naming conventions.

Description

A description of your resource models, thresholds, parameters, events and actions. Here you can specify what the specific element does and the logic on which it is based. For thresholds and parameters, you can explain the values specified in the decision tree script.

Alias

A name for the selected class of a dynamic model. This name is used in the decision tree script. Use a meaningful but simple alias. If you do not type an alias, the class name called in the decision tree script is the Selected Class name that includes the whole path.

Category Internal Name

A name for the resource model. This name displays in internal databases and defines groups that include similar resource models.

An error occurs if you try to import a resource model with a previously used internal name. Create a new name for the resource model and use the same name for the Category Internal Name and Category Descriptive Name resource models.

Category Descriptive Name

A descriptive name for the resource model. This name displays in the IBM Tivoli Monitoring windows when the operator opens a profile containing that resource model. It also displays in the message catalog. This name defines groups that include similar resource models.

An error occurs if you try to import a resource model with a previously-used internal name. Create a new name for the resource model and use the same name for the Category Internal Name and Category Descriptive Name resource models.

Creating a basic resource model

Objective

To use the Basic Resource Model Wizard to create a complete resource model from either a CIM class taken from the WMI repository, a monitoring collection, or a custom script.

Background information

If you create a resource model using the basic resource model creation wizard, you do not have to write any scripts or code. A resource model created using the wizard is just like any other resource model. After creating it, you can modify and customize it by editing the decision tree script, adding parameters, or writing filtering queries.

The Basic Resource Model Wizard enables you to create a resource model starting from the following contexts:

- A CIM class from the WMI repository
**Additional Information:** For non-Windows operating systems, you can import CIM classes into WMI. These classes do not need a WMI data provider. They exist in WMI so that you can work with them in IBM Tivoli Resource Model Builder.

- A monitoring collection
- A custom script

**Before you begin**

You must create a Tivoli Management Project before you can create a resource model with the wizard. Perform the following steps to create a Tivoli Management Project:

1. Select **File** from the toolbar to display the **File** drop-down list.
2. Select **New** from the toolbar to display the **New** pop-up list.
3. Select **Project** to display the New Project window.
4. Select **Tivoli Management Project** from the left field to display it in the right field.
5. Click on **Tivoli Management Project** in the right field to highlight the name.
6. Click **Next** to display the Project Name window.
7. Type a name for the project in the **Project name** text box.
8. (Optional) Clear the **Use default** check box and specify a different directory location for the project.
9. Click **Finish**.

**When you finish**

Check the Problems view to verify that you do not have any errors in your resource model. If the Problems view displays errors, correct the errors before running the resource model. The Problems view displays in the bottom pane of the IBM Tivoli Resource Model Builder.

**Procedure**

1. Click **File** from the toolbar to display the **File** drop-down menu.
2. Click **New** to display the **New** pop-up menu.
3. Click **Basic Resource Model Wizard** to launch the wizard and display the scripting language and operating system window.
4. Select one of the following scripting languages:
   - **VBA Script Resource Models (Windows only):** Limits the operating system type to w32-ix86 (Windows) only.
   - **JavaScript Resource Model (all platforms):** Enables the resource model to run on all operating systems.
5. (For **JavaScript only**) Select any of the following operating systems on which the resource model will run:

   - **w32-ix86**
     The Windows operating system on Intel™ ix-86 hardware.
   - **solaris2**
     The Sun Solaris 2 operating system.
   - **aix4–r1**
     The IBM AIX® operating system.
hpux10
The HP UNIX operating system.

linux–ix86
The Linux operating system on Intel ix-86 hardware.

linux–s390
The Linux operating system on the zSeries® platform.

os400
The IBM OS/400 operating system.

os2–ix86
The OS/2® operating system.

linux-ppc
The Linux operating system on iSeries™ or pSeries® platforms.

6. Click **Next** to open the Data Source Selection Page window.

7. Select one of the following wizards:
   - **CIM/WMI Data Source**
     See "Adding a CIM data source" on page 26
   - **Collection Data Source**
     See "Adding a DM Classic data source" on page 27
   - **Script Data Source**
     See "Adding a Custom Script data source" on page 27

**Additional Information:** You can add more than one data source to a resource model.

**Adding a CIM data source**

To create a resource model with the CIM data source wizard, perform the following steps:

1. Select a class containing the specific properties of the resource to monitor.
   See "Selecting a class" on page 28

2. Create a list of properties to monitor that are relevant to the class you selected.
   See "Selecting properties" on page 31

3. *(Optional)* Apply filtering criteria to data collection.

   **Note:** Available only for Windows resource models.
   See "Applying filtering conditions" on page 34

4. *(Optional)* Specify when you want an event to be generated and whether you want it to be sent to the Tivoli Enterprise Console server or the Tivoli Business Systems Manager console.
   See "Specifying event triggering conditions" on page 35

5. *(Optional)* Select the properties to log.
   See "Selecting the properties to log" on page 37

6. *(Optional)* Add another (CIM, monitoring collection, or custom script) data source.

7. *(Optional)* Change the generated internal names in the summary page.
   See "Changing the internal name" on page 38

8. Specify the time interval to elapse between two successive monitoring cycles.
   See "Entering a cycle time" on page 39

9. Specify the name of the new resource model and the project file in which to store it.
Adding a DM Classic data source

To import a monitoring collection created with Tivoli Distributed Monitoring (Classic Edition) using the Distributed Monitoring (Classic Edition) Collection data source wizard, perform the following steps:

1. Select the required file format to import, and browse to the file location.
   See “Importing a Tivoli Distributed Monitoring (Classic Edition) collection” on page 40.

2. Select the monitoring sources you want to import in your resource model.
   See “Selecting monitoring sources” on page 32.

3. Optional: Specify when you want an event to generate and whether you want to send it to the Tivoli Enterprise Console server or Tivoli Business Systems Manager.
   See “Specifying event triggering conditions” on page 35.

4. Optional: Select the properties to log.
   See “Selecting the properties to log” on page 37.

5. (Optional) Add another (CIM, monitoring collection, or custom script) data source.

6. (Optional) Change the generated internal names in the summary page.
   See “Changing the internal name” on page 38.

7. Specify the time interval to elapse between two successive monitoring cycles.
   See “Entering a cycle time” on page 39.

8. Specify the name of the new resource model and the project file in which to store it.
   See “Specifying the project file name” on page 39.

When you have completed the procedure, the IBM Tivoli Resource Model Builder produces a resource model that contains both the definition of the imported monitoring source and the code necessary to implement the logic of the analysis.

Adding a Custom Script data source

IBM Tivoli Resource Model Builder enables you to use resource models to launch shell commands or scripts and retrieve the output.

Include the commands or the scripts you want to launch in the resource model. These commands and scripts are embedded with the resource model’s other components.

To include a custom script in a resource model, perform the following general steps:

1. Specify the command you want to launch when you run this resource model.
   See “Specifying a custom script” on page 33.

2. Optional: Specify when you want an event to generate and whether you want to send it to the Tivoli Enterprise Console server or Tivoli Business Systems Manager.
   See “Specifying event triggering conditions” on page 35.

3. Optional: Select the properties to log.
   See “Selecting the properties to log” on page 37.
4. *(Optional)* Add another (CIM, monitoring collection, or custom script) data source.

5. *(Optional)* Change the generated internal names in the summary page.
   See “Changing the internal name” on page 38

6. Specify the time interval to elapse between two successive monitoring cycles.
   See “Entering a cycle time” on page 39

7. Specify the name of the new resource model and the project file in which to store it.
   See “Specifying the project file name” on page 39

### Selecting a class

**Objective**

To choose the class containing the specific properties of the resource you want to monitor.

**Background information**

The class list shows the available classes. When you select a class, all properties corresponding to the selected class are listed in the Class Properties table.

The CIM repository of WMI stores the available CIM classes. WMI providers provide class instances. If the resource you want to monitor is not described by any class in the CIM repository, you can add a new class.

For example, if you are creating a resource model to use on a UNIX-based operating system, you need to add the specific CIM class to the CIM repository of WMI. You do not need a WMI data provider for this class. It exists in the CIM repository solely to be available when you work with the IBM Tivoli Resource Model Builder.

To add a new class to the CIM repository, you must write a Managed Object Format (MOF) text file that defines the class and the provider. For Windows systems, you can use either a native WMI provider or a Java provider. For non-Windows systems, you must use a Java provider.

To run the resource model on an endpoint, you must transfer the MOF file and the provider (DLL file for a WMI provider, or a jar file for a Java provider) to the endpoint. Do this by adding a dependency to the resource model.

Note that there is no explicit relationship between the CIM classes in WMI and the MOF files that have been added as dependencies. That is, when a MOF file is compiled into WMI, you must still manually add the file as a dependency. Likewise, if a MOF file has been added as a dependency, you must still compile the file into WMI to be used as a data source in the resource model.

If you run a collection test at this point, the resource model collects data about all the properties of the selected class. However, if you are using a CIM class that does not have either a WMI provider or Java provider for Windows available through WMI, you cannot perform a collection test.

**Before you begin**

Specify the scripting language in which the resource model is to be written and the operating systems on which it will run.
When you finish
Create a list of properties to monitor that are relevant to the class you selected. See "Selecting properties" on page 31.

Procedure
1. Select a class to display the associated class properties in the table on the left.

Additional Information: You can also perform the following operations in this window:

- Change the namespace by doing the following:
  a. Enter the new name in the Namespace text box.
  b. Click the lightning icon (shown below) to connect to the namespace.

  Additional Information: If you are connected to a namespace, you are not prompted to enter your user name and password, if the current ones are valid. If you are prompted with the Logon window, do the following:
  1) Enter your user name and password.
  2) Click Logon to connect to the namespace.

- Search for specific classes from the class list by doing the following:
  a. Click the binocular icon (shown below) to open the Class Search window.
b. Enter a search phrase in the **Search phrase** text box.

c. *(Optional)* Select any search options to filter your search.

d. Click **Search** to display the results in the bottom of the window.

e. Select a class to highlight the class.

f. Click **OK** to include the selected class in the class list.

*Add a class by doing the following:*

a. Click the plus-sign icon (shown below) to open the Add Class window.

```
+  
```

b. Enter text in the **Superclass** and **New class name** text fields.

c. Click **OK** to add the class to the class list.

*Delete a class by doing the following:*

**Note:** You cannot delete system classes. System classes begin with an underscore.

a. Select the class.

b. Click the minus–sign icon.

```
-  
```

c. Click **OK** to verify the class deletion.

2. *(Optional)* Click **Collection Test** to retrieve all instances of a selected class from CIM/WMI.

**Additional Information:** As you progress through the pages, you can use the collection test button to show the inclusion of selected properties, while removing unwanted properties.

3. *(Optional)* Click **Show Description** to display information about the properties of the selected class that was included by the CIM class provider.
4. *(Optional)* Click **MOF Compiler** to launch the MOF compiler wizard. Use this wizard to import MOF files into the WMI repository.

5. Click **Next** to proceed with the wizard.

**Selecting properties**

**Objective**

To create a list of properties to monitor that are relevant to the class you selected.

**Background information**

*Properties* are specific attributes of the defined resource. For example, a property of the resource logical disk is free space. For each resource, you can specify multiple properties, which can be either numeric or string values.

**Before you begin**

Choose the class containing the specific properties of the resource to monitor. See “Selecting a class” on page 28.

**When you finish**

Optionally set filtering conditions. See “Applying filtering conditions” on page 34.

— OR —

Specify the event triggering conditions. See “Specifying event triggering conditions” on page 35.

**Procedure**

Do the following to select properties to monitor:

1. Select the specific properties to monitor from the **Available Properties** table.

2. Move the property to the **Selected Properties** table by doing one of the following:
   - Double-click the name of the property.
   - OR —
   - Click the right arrow.
Additional Information: You can select multiple properties from the list and move them together using the arrow buttons.

3. (Optional) Click Collection Test to collect instances for all the properties of the resource.
   Additional Information: If you click Back and run a collection test from the previous window, the resource model collects data from all properties.
4. Click Next to proceed with the wizard.

Selecting monitoring sources

Objective
To specify the monitoring sources to import into the resource model.

Background information
This list displays all sources that belong to the selected monitoring collection. You can import all sources or a subset of them. To view the details about a specific monitoring source, select it in the Available Sources table and click Details.

Before you begin

When you finish
None

Procedure
Do the following to select monitoring sources to import:
1. Select the monitoring sources you want to import in your resource model from the Available Sources list.
   Additional Information: If you select a source that has arguments, the Monitoring Source Argument Configuration window opens.
2. Click Next to proceed with the wizard.
Specifying a custom script

Objective
To import a shell command or custom script as a data source.

Background information
IBM Tivoli Resource Model Builder enables you to use resource models to launch shell commands or scripts and retrieve the output.

Before you begin
Select the scripting language and operating systems.

When you finish
None

Procedure
1. Specify the command you want to launch when you run this resource model by doing one of the following:
   • Type the command in the Shell Command text box. Be sure to include any parameters after the command or script.
   — OR —
   • Click Browse to locate the file that contains your script. After the script is located, the name of the script displays in the Shell Command text box. Be sure to include any parameters after the script.

   Note: If running a Perl script, ensure that the Perl interpreter be clearly listed. For example, when using a script titled logger.pl with parameter test.txt, enter perl logger.pl test.txt in the Shell command text box.

   Additional Information: You can import a custom script (like a bash or perl script) into a resource model to monitor its return values. IBM Tivoli Resource Model Builder imports that file and adds it to the dependencies of the resource model. The file containing the script automatically installs on the endpoint with
the resource model.

2. In the **Script Type** field, select one of the following to indicate the type of output you expect from the command:
   - **String**
     — OR —
   - **Numeric**

3. In the **Monitoring Engine** field, specify the engine version.
   - **ITM** - code generated uses Svc.Shell
     — OR —
   - **AME** - code generated uses Svc.ShellCmd

4. Click **Next** to proceed with the wizard.

**Applying filtering conditions**

**Objective**
To apply filtering conditions to a resource model.

**Note:** This option is available for Windows resource models only.

**Background information**
Applying filtering conditions to a resource model is optional. In the CIM Classes editor page, you can manually enter a "where" clause. From the Set Filtering Conditions window, you can obtain the same result without writing any code. You can build the conditions that make up your query by clicking buttons and selecting items from lists. Buttons are enabled or disabled, complying with the logic of the expression.

The **Selected Properties** table lists the properties you have selected for the specified class. You can use each of them to create a filter and define the instances you want to collect.
The filtering query that you build displays in WQL language in the lower pane of the window.

Before you begin
Create a list of properties to monitor that are relevant to the class you have selected. See “Selecting the properties to log” on page 37.

When you finish
Specify when you want an event to be generated and whether you want it to be sent to the Tivoli Enterprise Console server or to Tivoli Business Systems Manager. See “Specifying event triggering conditions.”

Procedure
Build a query by doing the following:
1. Double-click a property to open the Filter Condition Definition window.
   Additional Information: The possible conditions vary depending on whether the property is a string or a numeric value. As an option, you can click "NOT" to collect instances that do not meet a certain criteria, or "(" to form complex or grouped queries.
2. Select a filtering condition symbol from the drop-down list.
3. Enter a corresponding value in the text box.
4. Click OK to return to the Filter Condition Definition window.
5. Click Delete to delete the last condition displayed in the Condition table.
6. (Optional) Click Collection Test to see the result of the filtered data collection.
7. Click Next to proceed with the resource model wizard.

Specifying event triggering conditions

Objective
To specify when you want an event to generate and whether you want it sent to the Tivoli Enterprise Console server or Tivoli Business Systems Manager.

Background information
Only events can notify that there is a problem in the resource state, trigger an action and, if enabled, send a notification to the Tivoli Enterprise Console server or to the Tivoli Business Systems Manager.

Before you begin
Select one or more properties from the monitored properties and specify the conditions that must be satisfied to generate an event. For more information about events, see “Events and indications” on page 14

When you finish
After creating a trigger condition, the data source displays the triggers below it. You can remove a trigger by right-clicking on it. Double-clicking opens the trigger for editing. Right-clicking on the selected data source allows that data source to be deleted along with all triggers created with it.

You have the option to select properties to log, as described in “Selecting the properties to log” on page 37. You can also add another CIM, monitoring collection, or custom script data source. Then, specify the resource model cycle time as described in “Entering a cycle time” on page 39.
Procedure

1. Double-click a property to open the Event Triggering Condition window.
2. Select one of the following logical operators from the drop-down list:

   **Numeric property options:**

   - > Records the value of the selected property, compares it to the value you specify in the text box, and triggers an event if the current value is greater than the specified value.
   - < Records the value of the selected property, compares it to the value you specify in the text box, and triggers an event if the current value is less than the specified value.
   - = Records the value of the selected property, compares it to the value you specify in the text box, and triggers an event if the current value is equal to the specified value.
   - <> Records the value of the selected property, compares it to the value you specify in the text box, and triggers if the current value is not equal to the specified value.
   - >= Records the value of the selected property, compares it to the value you specify in the text box and triggers an event if the current value is larger than or equal to the specified value.
   - <= Records the value of the selected property, compares it to the value you specify in the text box and triggers an event if the current value is smaller than or equal to the specified value.

   **Increases at least by**
   Checks the current value of the selected property against the previous value and triggers an event if the current value is greater than the previous one by a value equal to or greater than the value you specify in the text box.

   **% increases at least by**
   Checks the current value of the selected property as a percentage of the previous value and triggers an event if current value is greater than the previous one by a value equal to or greater than the value you specify in the text box.

   **Decreases at least by**
   Checks the current value of the selected property against the previous value and triggers an event if the current value is smaller than the previous one by a value equal to or greater than the value you specify in the text box.

   **% decreases at least by**
   Checks the current value of the selected property as a percentage of the previous value and triggers an event if current value is smaller than the previous one by a value equal to or greater than the value you specify in the text box.

   **Changes by**
   Compares the previous and current values of the selected property and triggers an event if between these two values there is a difference equal to the value you specify in the text box.

   **Changes at least by**
   Compares the previous and current values of the selected property
and triggers an event if between these two values there is a difference equal to or greater than the value you specify in the text box.

**Is out of the range**

Comparing the current value of the selected property against the defined range limits, and triggers an event if the current value falls outside the specified range.

**String property options:**

**Is equal to**

Records the value of the selected property, compares it to the value you specify in the text box, and triggers an event if the current value is equal to the specified value.

**Is not equal to**

Records the value of the selected property, compares it to the value you specify in the text box, and triggers an event if the current value is different from the specified value.

**Contains**

Records the value of the selected property, compares it to the value you specify in the text box, and triggers an event if the current value contains the specified value.

3. Enter a string or numeric value in the text box next to the drop-down list. 
   Additional Information: Specifying the logical operator and value builds the logical expression.

4. Enter the number of occurrences of the specified condition that must be detected before the resource model generates an event.

   **Note:** The number of occurrences must be greater than zero.

5. Enter the number of holes allowed before the resource model generates an event.

   **Note:** The number of holes must be greater than, or equal to, zero.

6. Select the **Event severity** from the drop-down list.

7. (Optional) Select the first check box if you want an event to generate when the conditions that had previously caused an event are resolved.

8. (Optional) Select the respective check boxes if you want to send this event to the Tivoli Enterprise Console server or Tivoli Business Systems Manager.

9. Click **OK** to return to Specify the Event Triggering Conditions window.

10. (Optional) Repeat this procedure for each event you want to generate.

11. Click **Next** to proceed with the wizard.

**Selecting the properties to log**

**Objective**

To select properties to log so you can store specific data about a resource.

**Background information**

As an optional feature, setting and enabling logging provides monitoring results of one or more properties. A local database stores the data which you can access with IBM Tivoli Monitoring Web Health Console. For more information, see the *IBM Tivoli Monitoring User’s Guide*. By default, logging is disabled in IBM Tivoli Monitoring.
When configured, this logged data can also be sent to the Tivoli Enterprise™ Data Warehouse.

**Before you begin**
Specify event triggering conditions as described in "Specifying event triggering conditions" on page 35.

**When you finish**
You can add another CIM, monitoring collection, or custom script data source. You can also change a generated internal name, and then specify cycle time.

**Procedure**
Do the following to select a property to log:

1. Move an available property to the **Logged Properties** table by doing one of the following:
   - Double-click the property.
   - OR—
   - Click the right arrow.

2. Click **Finish** to continue with the wizard.

**Changing the internal name**

**Objective**
To change any internal names of the resource model so you can customize resource model internal names.

**Background information**
The internal name window contains all of the various internal names created by the wizard. In addition to the resource model name, you can change the parameter, threshold, and other resource model component names. You can customize any of the names to a meaningful naming convention for your environment.
Before you begin
None

When you finish
None

Procedure
Do the following to change the internal name:
1. Type a new name for the resource model in the Model Internal Name text box.
2. Click Next to save the updated name and continue with the wizard.

Entering a cycle time

Objective
To specify the cycle time for this resource model so you can run the resource model at appropriate time intervals for your environment.

Background information
The cycle time specifies the time between two successive monitoring cycles. Evaluate the resource model and the amount of resources it requires to run before assigning the value. Having a resource model cycle too frequently can monopolize the resources of the machines on which it is running.

This value is expressed in seconds.

Before you begin
None

When you finish
None

Procedure
Do the following to set the cycle time:
1. Enter the cycle time (a positive number in seconds) in the text box.
2. Click Next to proceed with the wizard.

Specifying the project file name

Objective
To specify the name of the resource model you are creating and the project file in which it will be stored.

Background information
None

Before you begin
None

When you finish
None

Procedure
1. Enter the name of the project file or select one from the tree view.
2. Specify the name of the resource model you are creating.
Note: Supported file extensions include only the following:

- \texttt{jrm} for JavaScript resource models
- \texttt{vrm} for VBA Script resource models

3. Click \texttt{Finish} to save the resource model and complete the creation process.

\textbf{Importing a Tivoli Distributed Monitoring (Classic Edition) collection}

\textbf{Objective}
To select the required file format to import and browse to the file location.

\textbf{Background information}
Use this wizard to import a monitoring collection created with Tivoli Distributed Monitoring (Classic Edition).

\textbf{Before you begin}
Select the scripting language and operating systems.

\textbf{When you finish}
Select monitoring sources as described in “Selecting monitoring sources” on page \pageref{selecting-monitoring-sources}.

\textbf{Procedure}
Do the following to select the required file format to import and browse to the file location:

1. Select one of the following file types:
   - CSL
     - OR —
   - Dumped Monitoring Collection

2. Click \texttt{Browse} to select the location of the file.
**Creating an empty resource model**

**Objective**
To create an empty resource model so you can later customize the resource model without following a specific sequence.

**Background information**
None

**Before you begin**
You must create a Tivoli Management Project before you can create a resource model with the wizard. Perform the following steps to create a Tivoli Management Project:
1. Select **File** from the toolbar to display the **File** drop-down list.
2. Select **New** from the toolbar to display the **New** pop-up list.
3. Select **Project** to display the New Project window.
4. Select **Tivoli Management Project** from the left field to display it in the right field.
5. Click on **Tivoli Management Project** in the right field to highlight the name.
6. Click **Next** to display the Project Name window.
7. Enter a name for the project in the **Project name** text box.
8. (Optional) Clear the **Use default** check box and specify a different directory location for the project.
9. Click **Finish**.

**When you finish**
You can define the resource model elements by opening the resource model tree structure and double-clicking on the element on the left side of the perspective, or by expanding the view and using the element tabs on the bottom of the view.

Check the Problems view to verify that you do not have any errors in your resource model. If the Problems view displays errors, correct the errors before running the resource model. The Problems view displays in the bottom pane of the IBM Tivoli Resource Model Builder.

**Procedure**
1. Select **File** from the toolbar to display the **File** drop-down menu.
2. Click **New** to display the **New** pop-up menu.
3. Select **Empty Resource Model Wizard** to launch the empty resource model wizard and display the General Settings window.
4. Type a name for your resource model in the **Internal Name** text box.

   *Additional Information:* This name must be alphanumeric, start with a letter, and contain no blanks. The name can contain underscore characters, although...
not as a first character. For more information about naming conventions, see “About resource model names” on page 23.

5. Type a descriptive name for your resource model in the Descriptive Name text box.

6. Type a description of your resource model in the Description text box. Additional Information: This text describes the resource model, such as the decision-tree logic and threshold settings.

7. Type a name for the category in which this resource model belongs in the Category Internal Name text box. Additional Information: Internal databases use this name to define groups that include similar resource models and for grouping resource models in the profile Category display.

   This name must be alphanumeric, start with an alphabetic letter and contain no blanks. This name can contain underscores. All resource models with the same Category Internal name display under the same category option.

8. Type a descriptive name for the category in which this resource model belongs in the Category Descriptive Name text box. Additional Information: This name displays in the IBM Tivoli Monitoring windows and the message catalog.

9. Type the number you want to indicate the major version of your resource model in the Major Version text box.

10. Type the number you want to indicate the minor version of your resource model in the Minor Version text box.

11. Type a value for the time interval (in seconds) between two successive data collections in the Cycle Time text box. Additional Information: This value is the default value that you can modify in the IBM Tivoli Monitoring windows.

12. Select one of the following scripting languages:
   - **VBA Script Resource Models (Windows only)** Limits the operating system type to w32-ix86 (Windows) only.
     —OR—
   - **JavaScript Resource Model (all platforms)** Enables the resource model to run on all operating systems.

13. Select any of the following operating systems on which the resource model will run:

    **w32-ix86**
    The Windows operating system on Intel ix-86 hardware.

    **solaris2**
    The Sun Solaris 2 operating system.

    **aix4–r1**
    The IBM AIX operating system.

    **hpux10**
    The HP UNIX operating system.

    **linux–ix86**
    The Linux operating system on Intel ix-86 hardware.

    **linux–s390**
    The Linux operating system on the zSeries platform.

    **os400**
    The IBM OS/400 operating system.
os2-ix86
The OS/2 operating system.

linux-ppc
The Linux operating system on iSeries or pSeries platforms.

14. Click Next to proceed with the wizard.

15. Specify the project file name by doing the following:
   a. Enter the name of the project file or select one from the tree view.
   b. Specify the name of the resource model being created.
   c. Click Finish to save the resource model and complete the creation process.
Chapter 6. Editing resource models

This chapter describes the procedures to edit resource models. You can edit your resource model by clicking on an edit page in the IBM Tivoli Resource Model Builder editor tool or by using the VisitTree Wizard.

The following table lists the tools you can use to edit resource models:

<table>
<thead>
<tr>
<th>Goal</th>
<th>Refer to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Editing the resource model with the editing tool.</td>
<td>“Editing general settings” on page 45</td>
</tr>
<tr>
<td>Editing the resource model with the VisitTree Wizard to provide screens to guide the editing process.</td>
<td>“Working with the VisitTree Wizard” on page 61</td>
</tr>
</tbody>
</table>

Using the Editing tool

You can edit your resource model by clicking the editing pages in the Tivoli Management Perspective window. Choose to edit one of the following topics:

- Editing general settings
- Editing CIM classes
- Editing DM classic probes
- Editing events
- Adding Actions
- Editing thresholds
- Editing parameters
- Editing data to log
- Editing dependencies
- Editing source

Editing general settings

Objective
To define the general settings for a resource model.

Background information
None

Before you begin
None

When you finish
Check the Problems view to verify that you do not have any errors in your resource model. If the Problems view displays errors, correct the errors before running the resource model. The Problems view displays in the bottom pane of the IBM Tivoli Resource Model Builder.
Procedure
1. Open the resource model.
2. Click the General Settings page.
3. Enter or change information on this page as necessary.

Additional Information:

Internal name
The name of the resource model as used by the VisitTree code and from the command line. The internal name must be unique across all IBM Tivoli Monitoring based products, so prefix it with a descriptive or categorical name such as "DB2_Dataspace," rather than the generic "DatabaseSpace." This name must be alphanumeric, start with a letter, and contain no blanks.

Descriptive name
The name that displays in the IBM Tivoli Monitoring profile list of resource models and in the IBM Tivoli Monitoring Web Health Console.

Category internal name
A name for the resource model. This name displays in internal databases and defines groups that include similar resource models.

This name must be alphanumeric, start with an alphabetic letter, and contain no blanks. All resource models that have the same Category Internal name display under the same category option.

Category descriptive name
A descriptive name for the resource model. This name displays in the IBM Tivoli Monitoring windows when the operator opens a profile containing that resource model. It also displays in the message catalog. This name defines groups that include similar resource models.

Major version
The major and minor version numbers assigned to a resource model prevent installation of an older version than the one currently installed. Initial resource models must have a Major version of 1 and Minor version of 0, such as version 1.0.

Minor version
The major and minor version numbers assigned to a resource model prevent installation of an older version than the one currently installed. Initial resource models must have a Major version of 1 and Minor version of 0, such as version 1.0.

Cycle time
The time interval (in seconds) between two successive data collections. You can modify the displayed default value in the IBM Tivoli Monitoring windows.

Disable TBSM Events
Check this box if you do not want events sent to Tivoli Business Systems Manager. Clear the check box to send events to Tivoli Business Systems Manager.

Description
Describes the resource model. You can document the decision tree logic as well as explain threshold settings.
Copyright Statement
Modify this field if you would like to add information to include in the files generated from this resource model, such as the Baroc file and message catalogs.

Scripting Language
Specifies the scripting language used in the resource model.

VBA Script Resource Models (Windows only)
Limits the operating system type to w32-ix86 (Windows) only.

JavaScript Resource Model (all platforms)
Enables the resource model to run on all operating systems.

Supported Platforms
The operating systems on which the resource model runs.

w32–ix86
The Windows operating system on Intel ix-86 hardware.
solaris2
A Sun Solaris 2 operating system.
aix4–r1
An IBM AIX operating system.
hpux10
An HP Unix operating system.
linux–ix86
The Linux operating system on Intel ix-86 hardware.
linux–s390
The Linux operating system on the zSeries platform.
linux–ppc
The Linux operating system on iSeries or pSeries platforms.
os400
An IBM OS/400 operating system.
os2–ix86
An OS/2 operating system.

4. Save the changes and close the resource model or go to another editing page.

Editing CIM classes

Objective
To edit or add the resources you want to monitor and collect the data.

Background information
The class list shows the available classes. When you select a class, all properties corresponding to the selected class are listed in the Class Properties table.

The CIM repository of WMI stores the available CIM classes. WMI providers provide class instances. If the resource you want to monitor is not described by any class in the CIM repository, you can add a new class.

For example, if you are creating a resource model to use on a UNIX-based operating system, you need to add this CIM class to the CIM repository of WMI. You do not need a WMI data provider for this class. It exists in the CIM repository to be available only when you work with the IBM Tivoli Resource Model Builder.
To add a new class to the CIM repository, you must write a Managed Object Format (MOF) text file that defines the class and the provider. For Windows systems, you can use either a native WMI provider or a Java provider. For non-Windows systems, you must use a Java provider.

To run the resource model on an endpoint, you must transfer the MOF file and the provider (DLL file for a WMI provider, or a jar file for a Java provider) to the endpoint. Do this by adding a dependency to the resource model.

If you run a collection test at this point, the resource model collects data about all the properties of the selected class. However, if you are using a CIM class that does not have either a WMI provider or Java provider for Windows available through WMI, you cannot perform a collection test.

**Before you begin**

None

**When you finish**

Check the Problems view to verify that you do not have any errors in your resource model. If the Problems view displays errors, correct the errors before running the resource model. The Problems view displays in the bottom pane of the IBM Tivoli Resource Model Builder.

**Procedure**

1. Open the resource model.
2. Click the CIM Classes page.
3. Work with an existing class, add a new class, or launch the WMI MOF Compiler Wizard.

*Additional Information:* The following editing options are available:

**Add CIM data source**

Opens the Add new CIM data source window, in which you can do the following:

- Search for specific classes from the class list by doing the following:
  a. Click the binocular icon to open the WMI Class Search window.
  b. Enter a search phrase and search options.
  c. Click **Search** to display the search results on the bottom of the window.
  d. Select a class and click **OK**.

- Add a class by doing the following:
  a. Click the plus-sign icon to open the Add Class window.
  b. Enter a name for the **Superclass** and the **New class name**.
  c. Click **OK** to add the new class to the class list.

- Delete a class by doing the following:
  a. Select a class to delete to highlight the class.
  b. Click the minus-sign icon to delete the class.
  c. Click **OK** to confirm the deletion.

- Perform a collection test by doing the following:

  *For Windows resource models only:* Click **Collection Test** to see an example of the collected instances of the class defined. If you are performing an asynchronous collection, data collection starts when you click **Collection Test**. A pop-up window opens enabling you to
click OK to stop the data collection. In this way, you are simulating a cycle time interval. The Collected Instances window opens and shows the results of data collection.

- Click Show Description to display information about the properties of the selected class included with the CIM class provider.
- Click MOF Compiler to launch the WMI MOF Compiler Wizard and import MOF files into the WMI repository.

MOF Wizard
Launches the WMI MOF Compiler Wizard so you can import MOF files into the WMI repository. After compiling a MOF file from this page, you must reconnect to your CIM repository for it to open in the CIM/WMI navigator.

Selected
The class name and full path.

Use Alias
The name used in the decision tree script. If you do not assign an alias, the decision tree calls the class name and path displayed in the Selected field. Use a simple, but meaningful alias.

Class properties
The properties for the class. Select the properties you want to list from the Available table and move them to the Selected table by clicking the right-arrow icon. Move properties from the Selected table to the Available table by clicking the right-arrow icon. You cannot remove key properties from the Selected table.

Collection Information
The sort option for the collected data. The sort options are:

- No Sort
  Implies no sequence order. Type the number of cycles you want to elapse between two successive data collections in the Every text box.
  - Every = 0, runs the collection one time
  - Every = 1, runs the collection at every cycle
  - Every = \( n \), runs the collection at every \( "n" \) cycles
  - Every = -1, runs the collection based on the monitoring algorithm of the VisitTree section of the Decision Tree Script. This value is set automatically when you select the Collect on demand check box.

- Sort Ascending
  Sorts from the lowest value to the highest. Type the key element for the sort in the Sorting Field text box. For Windows resource models, type the number of instances in your data collection in the Number of Instances text box. Type the number of cycles you want to elapse between two successive data collections In the Every text box.
  - Every = 0, runs the collection one time
  - Every = 1, runs the collection at every cycle
  - Every = \( n \), runs the collection at every \( "n" \) cycles
  - Every = -1, runs the collection based on the monitoring algorithm of the VisitTree section of the Decision Tree Script. This value is set automatically when you select the Collect on demand check box.

- Sort Descending
Sorts from the highest value to the lowest on the basis of the sorting key element you choose. Type the key element for the sort in the **Sorting Field** text box.

For Windows resource models only, type the number of instances that you want in your data collection in the **Number of Instances** text box. Type the number of cycles you want to elapse between two successive data collections in the **Every** text box.

- **Every** = 0, runs the collection one time
- **Every** = 1, runs the collection at every cycle
- **Every** = n, runs the collection at every "n" cycles
- **Every** = -1, runs the collection based on the monitoring algorithm of the **VisitTree** section of the Decision Tree Script. This value is set automatically when you select the **Collect on demand** check box.

- **Asynchronous**

  **Note:** Asynchronous collection can be selected only for those classes whose providers support \_InstanceOperationEvent. For more information, see the WMI Web site.

  **For Windows resource models only:** Starts data collection at a cycle and carries it until the following cycle, when the collection is made available. Specify the following information that you want to obtain:
  - Instance creation
  - Instance modification
  - Instance deletion

  **WHERE Clause**

  **For Windows resource models only:** A WQL where statement (query) to define filtering criteria for data collection, if required.

  **Note:** When entering a clause manually, the clause must begin with "where."

4. Save the changes and close the resource model or go to another editing page.

**Editing DM classic probes**

**Objective**
To import a monitoring source (Distributed Monitoring Classic Probe).

**Background information**
When you define a resource model, you can select CIM classes for the resources, import a monitoring source (Distributed Monitoring Classic Probe), or do both. The steps below show how to import a classic DM probe into the resource model.

**Before you begin**
None

**When you finish**
Check the Problems view to verify that you do not have any errors in your resource model. If the Problems view displays errors, correct the errors before running the resource model. The Problems view displays in the bottom pane of the IBM Tivoli Resource Model Builder.
Procedure
1. Open the resource model.
2. Click the Classic Probes page.
3. Click Add DM Classic Probe to open an import selection window.
4. Do one of the following to import files:
   a. Click the Collection Scripting Language File (.csl) radio button.
   b. Click OK to browse to the file location.
   c. Select the file to import and click Open to display the Preprocessing Settings window.
   d. Specify the preprocessor path in the Preprocessor text box.
   e. Specify any preprocessor options in the Preprocessing Options field.
   —OR—
   a. Click the Dumped Monitoring Collection radio button.
   b. Click OK to browse the file location and select the file to import.
5. Click OK to import the file and return to the DM Classic Probes page.
6. (Optional) Select a file and click Remove to remove the selected classic probe from the resource model.
7. Save the changes and close the resource model, or go to another editing page.

Editing events

Objective
To define an indication for the resource model and the aggregation settings that determine when a sequence of indications produce an event.

Background information
When you define an event, you must specify the attributes that qualify it. In particular, you must select the most significant attribute for the aggregation process.

While defining an event, you can also indicate if you want the system to notify the Tivoli Enterprise Console server and Tivoli Business Systems Manager that an event was generated. The IBM Tivoli Monitoring operator can change these selections when the resource model is included in an IBM Tivoli Monitoring profile. You can also define the degree of severity and the message that specifies the notification of the event.

Before you begin
None

When you finish
Check the Problems view to verify that you do not have any errors in your resource model. If the Problems view displays errors, correct the errors before running the resource model. The Problems view displays in the bottom pane of the IBM Tivoli Resource Model Builder.

Procedure
1. Open the resource model.
2. Select the Events page.
3. Edit an existing event or click Add Event to add a new event to the resource model.

Additional Information: Customize the following event options:

Internal Name
The internal name of an event must be unique within the entire set of resource models installed on a specific IBM Tivoli Monitoring server. If two or more resource models specify events with identical names, then you cannot install those resource models on the same IBM Tivoli Monitoring server. Follow naming conventions and best practices as described in Chapter 4, "Designing a resource model," on page 13.

Descriptive name
The descriptive name of this event that is informative and recognizable for logging and reporting information.

Message
The message the event sends to the Tivoli Enterprise Console server or Tivoli Business Systems Manager. When typing the message, specify event attributes that are variables between "@" symbols. For example: "This process @ProcessName@, @ProcessID@ is consuming too much CPU."

Send to
- TBSM To send the event notification to the Tivoli Business Systems Manager server as a default.
- TEC To send the event notification to the Tivoli Enterprise Console server as a default. When you select Send to TEC the Default Event Hierarchy check box displays.

Default Event Hierarchy
Clear this box to create event hierarchy in Tivoli Enterprise Console by specifying a parent for this event in the Event Parent text box. By default, this box is selected and the event parent is set to TMW_EVENT.

Event Parent
The parent of this event in the Tivoli Enterprise Console hierarchy. This enables you to organize events into hierarchies that are understood by the Tivoli Enterprise Console. When the resource model is saved, the parent is saved to the workspace XML file. By default, the event parent is set to TMW_EVENT.

Severity
The degree of severity for notification to the Tivoli Enterprise Console server or Tivoli Business Systems Manager. Make a selection from the drop-down list.

Description
The descriptive text that is displayed on the IBM Tivoli Monitoring windows and that provides information for the operator running the resource model.

Properties
Add or remove properties from the event. Properties detail an event by allowing you to obtain as much information as you need from the event. For example, an event might indicate that your disk subsystem has problems. By associating properties such as bytes per second, disk name, or available disk space, you can identify the problem easier.
Do the following to add an property:

a. Click **Add** or right-click inside the table.

b. Enter the name of the property in the **Name** column.

**Note:** If you are going to enable logging for this resource model, use the same names for properties here and in the logging definitions when specifying the same metric.

c. Select the **Type** column and select the type from the drop-down list.

d. Select the **Is Key** column to make this the key property. When selected, a check mark displays in this column. Key properties are the most meaningful and play a fundamental role in the event aggregation process.

e. Select the **Unit** column and select the applicable property unit.

Use the up and down arrow buttons to specify the order of the properties.

Click **Remove** to remove the selected property from the event.

**Note:** The same property units that can be assigned to logged numeric properties, which are **Other** by default.

**Aggregation Settings**

The conditions under which this indication becomes an aggregated event. Only aggregated events trigger actions, notify you of a problem, and send notification to the Tivoli Enterprise Console and Tivoli Business Systems Manager.

**Clearing Event**

Select **Clearing Event** if you want the system to send a clearing event when the circumstances that generated the event are resolved.

The Tivoli Enterprise Console server and Tivoli Business Systems Manager use the clearing event to close the corresponding error event.

For example, if the resource being monitored is a service, the event is sent when the service is not available. The event might have an action associated with it to restart the service (or the restart might be done manually). When the engine detects that the service is available again, a clearing event can be sent to close the original event.

**Number of occurrences**

The number of indications that must occur before an event generates. The number of occurrences must be greater than zero. This is the default value for this resource model. You can change this value at runtime in the IBM Tivoli Monitoring application.

**Number of holes**

The maximum number of monitoring cycles allowed with no indications for an event to be generated. The number of holes must be greater than or equal to zero. This is the default value for this resource model. The user can change this value at runtime in the IBM Tivoli Monitoring application.
4. *(Optional)* Click **Action** to associate a recovery action with the event, as described in "Adding actions" on page 54.

5. Save the changes and close the resource model, or go to another editing page.

### Adding actions

#### Objective
To associate recovery actions with an event.

#### Background information
It is possible to associate one or more recovery actions with a specific event. These actions automatically trigger when the event occurs and can restore a satisfactory system service level.

Actions are associated with either the execution of a CIM method or the execution of a program. For example, if you are monitoring Windows NT® services and an event generates when a service stops, you can associate the event with a recovery action that restarts the interrupted service.

#### Before you begin
Add an event or select an event to modify from the Event page. See, "Editing events” on page 51.

#### When you finish
Check the Problems view to verify that you do not have any errors in your resource model. If the Problems view displays errors, correct the errors before running the resource model. The Problems view displays in the bottom pane of the IBM Tivoli Resource Model Builder.

#### Procedure
1. Click **Action** from the **Event** page to open the **Event Actions** for window.
2. Do one of the following to add actions using either the CIM or program method:
   - **Add a CIM Method action by doing the following:**
     a. Click **Add CIM Method** to open the CIM Action Browser window.
        
        *Additional Information:* The CIM Action Browser window lists available classes, and for each class, its instance keys, methods and parameters.
        
        The list in the upper part of the **Methods** group shows all the methods of the selected class, and specifies whether the methods are static or non-static.
     b. Select the class that includes the required action by doing any of the following:
        - Select a class from the currently displayed list.
        - Change the **Namespace** by doing the following:
          1) Enter the new name in the Namespace text box.
          2) Click the lightning icon to open the Logon window.
          3) Enter your User name and Password, if necessary.
             
             *Additional Information:* If you are connected to a namespace, you are not prompted to enter your user name and password if the current ones are valid.
          4) Click **Logon** to connect to the namespace.
        - Search for a specific class from the class list by doing the following:
1) Click the binoculars icon to open the WMI Class Search window.
2) Enter a search phrase and search option.
3) Click Search to display the search results on the bottom of the window.
4) Select a class and click OK.
   - Add a class by doing the following:
     1) Click the plus–sign icon to open the Add Class window.
     2) Enter a name for the Superclass and the New class name.
     3) Click OK to add the new class to the class list.
   c. **Optional:** Select a key attribute from the Instance keys group box.
      Additional Information: The Instance Key group box displays only if you select non-static methods. Key attributes for the selected class display. Click on an attribute in the Event attribute column to display a drop-box. You can select "Not set" or any key attribute.
   d. Select a method from the Methods group box.
   e. Select a parameter from the Parameters for group box.
   f. Type a descriptive name for the class in the Descriptive name text box.
   g. Click OK to add the class and return to the Event Actions for window.
• **Add a Program action by doing the following:**
  a. Click Add Program from the Actions window to open the Program Action Definition window.
  b. Type an internal name in the Internal name text box.
  c. Type the Shell command (the command line which launches a process) in the Shell command text box, or browse for a predefined command.
     Note: For Windows and UNIX endpoints, always indicate a slash ("/") in the Shell command path (for example, c:/action.bat).
  d. Enter a descriptive name and a description, in the corresponding text boxes.
  e. Click OK to complete the definition of this action and return to the Event Actions for window.
3. Click OK to save the changes and return to the Events editing page.
4. Close the resource model, or go to another editing page.

**Editing thresholds**

**Objective**
To edit or define thresholds for a resource model.

**Background information**
For each resource model, you can define one or more thresholds. Usually, the numeric value you enter in this window represents a limit above or below which you do not want your resource to perform. The default values that you define here can be modified later when this resource model is included in a profile on the IBM Tivoli Monitoring server.

**Before you begin**
None
When you finish
Check the Problems view to verify that you do not have any errors in your resource model. If the Problems view displays errors, correct the errors before running the resource model. The Problems view displays in the bottom pane of the IBM Tivoli Resource Model Builder.

Procedure
1. Open the resource model.
2. Select the Thresholds page.
3. Select the threshold to edit or click Add Threshold to add a new threshold.
   
   Additional Information: Complete the following threshold options:

   Internal name
   The internal name used by the VisitTree function to determine if a property has exceeded a threshold value. This name must be alphanumeric, start with an alphabetic character, and contain no blank spaces.

   Descriptive name
   The name that is used in IBM Tivoli Monitoring windows.

   Default Value
   The numeric value associated with this threshold by default. This value can be changed by the operator from the IBM Tivoli Monitoring windows.

   Description
   The description of this threshold as it displays on the IBM Tivoli Monitoring windows. This description should include the descriptive name of the event with which this threshold is associated, the names of the attributes being monitored, and a relational comparison of the threshold to the attribute.

   For example, when the counter, PrcUsrTime, exceeds this threshold, the event, Ev_w2k_Thread_PrcUsrTime_too_high, generates.

   4. (Optional) Click Remove to remove a threshold.
   5. Save the changes, or go to another page to continue making changes.

Editing parameters

Objective
To edit or define one or more lists of parameters.

Background information
For each resource model, you can define one or more lists of parameters. To define parameters, you must select the most appropriate type of list and then enter the required information. The lists of numbers or strings you specify for each parameter can represent the instances you want to monitor or a limit you do not want your resource to exceed. The default values that you define here can be modified later when this resource model is included in a profile on the IBM Tivoli Monitoring server.

Before you begin
None
When you finish
Check the Problems view to verify that you do not have any errors in your resource model. If the Problems view displays errors, correct the errors before running the resource model. The Problems view displays in the bottom pane of the IBM Tivoli Resource Model Builder.

Procedure
1. Open the resource model.
2. Select the Parameters page.
3. Select the parameter to edit or click Add Parameter to add a new parameter.
   Additional Information: Complete the following parameter options:
   
   Internal Name
   The internal name used by the VisitTree function. This name must be alphanumeric, start with an alphabetic character, and contain no blank spaces.

   Descriptive Name
   The name that is used in the IBM Tivoli Monitoring displays.

   Description
   The descriptive text that is displayed on the IBM Tivoli Monitoring windows and provides information for the operator running the resource model.

   List type and values
   When you specify the parameters, you can choose one among the following kinds of lists:
   
   Unrestricted List
   Unrestricted lists offer the option to restrict the types of values accepted to only strings or only numbers. String lists can be used to specify parameter values such as application names. Numeric lists can be useful to define event IDs or port numbers.

   Restricted List
   Restricted lists allow you to specify whether a user will be able to select one or more values from the parameter list in IBM Tivoli Monitoring Desktop, and enable or disable a description field for each parameter in the list. Depending on the setting, one or more of the parameters can be selected by default.

   Note: Regardless of the type of list you choose, IBM Tivoli Resource Model Builder displays the list content in the script as either a numeric or string list.

4. (Optional) Click Remove to remove the selected parameter.
5. Save the changes, or go to another page to continue making changes.

Editing data to log

Objective
To define what data to log for this resource model.
**Background information**

Use the logging function to store data about the attributes of the resource you are monitoring. To implement logging in a resource model, first define what data to log, then call the `DefineLogInst` method in the decision tree script.

In the resource models you create, logging is disabled by default, but you can enable it from the IBM Tivoli Monitoring windows.

Logged data is stored in a database that can be accessed through the IBM Tivoli Monitoring Web Health Console.

**Before you begin**

None

**When you finish**

Check the Problems view to verify that you do not have any errors in your resource model. If the Problems view displays errors, correct the errors before running the resource model. The Problems view displays in the bottom pane of the IBM Tivoli Resource Model Builder.

**Procedure**

1. Open the resource model.
2. Select the **Logging** page.
3. Select the resource to edit or click **Add Logging** to add a new resource.

   **Note:** The logging resource internal name is the concatenation of the Context and Resource. The internal name of a logging resource must be unique within the entire set of resource models installed on a specific IBM Tivoli Monitoring server.

   **Additional Information:** Complete the following logging options:

   **Context**
   
   The name of a general problem to which the resource logging relates. This name displays in the IBM Tivoli Monitoring Web Health Console and is also the first parameter passed in the `Svc.LogInst` or `Svc.LogInstEx` methods.

   **Resource**
   
   The name displayed in the IBM Tivoli Monitoring Web Health Console historical data and also the second parameter passed in the `Svc.LogInst` or `Svc.LogInstEx` methods.

   **Properties**
   
   The PropertyDefinition is used to include the specific properties to log. These are attributes of the resource you have specified. The attributes listed under Properties include:

   **Name** Displays the name of the property.
   **Type** Displays the property type.
   **Is key** Specifies if the property is a key property.
   **Unit** Property Units are shown based on the unit (type in the schema) only. The Category is inferred from the type as follows:

<table>
<thead>
<tr>
<th>Category</th>
<th>Unit (Type in schema)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRC (percent)</td>
<td>Percent (PRC)</td>
</tr>
</tbody>
</table>
### Category

<table>
<thead>
<tr>
<th>Category (time)</th>
<th>Unit (Type in schema)</th>
</tr>
</thead>
<tbody>
<tr>
<td>TM (time)</td>
<td>MSec (Milliseconds)</td>
</tr>
<tr>
<td></td>
<td>HSec (Hundredths of seconds)</td>
</tr>
<tr>
<td></td>
<td>Second (Sec)</td>
</tr>
<tr>
<td></td>
<td>Minute (Min)</td>
</tr>
<tr>
<td></td>
<td>Hour (Hr)</td>
</tr>
<tr>
<td></td>
<td>Day (Day)</td>
</tr>
</tbody>
</table>

| RT (rate)           | Bytes/Sec (Bps)                        |
|                     | Megabytes/Sec (MBps)                   |
|                     | Kilobytes/Sec (KBps)                   |
|                     | Reads/Sec (Rps)                        |
|                     | Queries/Sec (Qps)                      |
|                     | Queries/Min (Qpm)                      |

| QTY (quantity)      | Quantity (QTY)                         |
|                     | Gigabyte (GB)                          |
|                     | Megabyte (MB)                          |
|                     | Kilobyte (KB)                          |
|                     | Byte (B)                               |

To add properties:

a. Click **Add** to display the Add New Logging Property window.

b. Enter the name of the property in the **Name** text box.

**Note**: Use the same names for properties here as in the logging definitions when specifying the same metric.

c. Click either **String** or **Numeric** in the **Type** field.

d. Click either **True** or **False** in the **Key** field.

True makes this the key attribute. Key attributes are the most meaningful and play a fundamental role in the event aggregation process.

e. Click **OK** to save your changes and return to the **Logging** page.

4. Type the name of a general problem that the resource logging relates to in the **Context** text box.

5. Type the name of the resource whose state you want to log in relation to the specified context in the **Resource** text box.

6. Save the changes, or go to another page to continue making changes.

## Editing dependencies

### Objective
To transfer additional files with the resource model.

### Background information
To run a resource model on an endpoint, you might need to transfer additional files. For example, to add a new class to the CIM repository, you need the files that define the class (MOF) and the provider (DLL for WMI provider or JAR for Java provider). A MOF file is a text file that contains definitions of classes and instances using the Managed Object Format (MOF) language.

In these cases you can add dependencies to your resource model and transfer the required files with the model.
Before you begin
None

When you finish
Check the Problems view to verify that you do not have any errors in your resource model. If the Problems view displays errors, correct the errors before running the resource model. The Problems view displays in the bottom pane of the IBM Tivoli Resource Model Builder.

Procedure
1. Open the resource model.
2. Select the Dependencies page.
3. Select the file to modify and click Modify selected, or click Add to add a new file.
4. Browse to the file you want to transfer.
5. Select one of the following operating system options to support this file:
   • all — All operating systems.
   —OR—
   • desired platform.
6. Click OK to copy the dependency into the Dependencies page.
7. Save the changes, or go to another page to continue making changes.

Note: Extract Selected allows you to externally save the selected dependency file for future use.

Editing source

Objective
To edit the source code of a resource model.

Background information
You might want to modify the source when you are debugging the resource model and the logging instances need to be examined in the Logging view. For example, you can add the line “Svc.EnableLogging(1)” to the Default Config function.

The decision tree script implements the settings defined during the creation of the resource model. In the decision tree script, you have to write a monitoring algorithm that controls the whole process. The monitoring algorithm is written in Visual Basic or JavaScript and displays on the right of the main window when you open a resource model.

Note: The IBM Tivoli Resource Model Builder implements the Sax™ interpreter, which differs slightly from other Visual Basic interpreters, such as Microsoft® Visual Studio. For more information, refer to the following website:

http://www.saxsoft.com/Basic/Details

In the script, you can edit the predefined functions. You can add additional code, but you cannot modify the settings specified in the previous windows. If you try to modify them from the script, the new settings are not saved. However, you can add additional steps to the functions, as long as you are careful writing the new lines of code.
When you write the monitoring script, you must specify how your resource model works. When you specify the event attributes, you must list them in the same order as they display in the Events window.

Before you begin
None

When you finish
Test the resource model source by clicking on the right-arrow in the Source page menu. The results display in the RMB Event Viewer.

Check the Problems view to verify that you do not have any errors in your resource model. If the Problems view displays errors, correct the errors before running the resource model. The Problems view displays in the bottom pane of the IBM Tivoli Resource Model Builder.

Procedure
1. Open the resource model.
2. Select the Source page.
3. Edit the source code.

Note: Changes made in the decision tree script are not reflected on the other IBM Tivoli Resource Model Builder pages for this resource model.
4. Save the changes, or go to another page to continue making changes.

Working with the VisitTree Wizard

The VisitTree Wizard provides a step-by-step configuration tool to create a resource model. You can choose to create a resource model from one of three data sources: CIM/WMI, collection, or custom script. You must provide the IBM Tivoli Resource Model Builder with the specific data source in order for that data source button to be selected in the wizard.

The following table lists the concepts for working with the VisitTree Wizard:

<table>
<thead>
<tr>
<th>Goal</th>
<th>Refer to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Creating a resource model from a CIM/WMI class data source.</td>
<td>&quot;Creating a resource model from a CIM/WMI class&quot; on page 61</td>
</tr>
<tr>
<td>Creating a resource model from a collection data source.</td>
<td>&quot;Creating a resource model from a Distributed Monitoring (Classic Edition) Collection data source&quot; on page 65</td>
</tr>
<tr>
<td>Creating a resource model from a custom script.</td>
<td>&quot;Creating a resource model from a custom script&quot; on page 67</td>
</tr>
</tbody>
</table>

Creating a resource model from a CIM/WMI class

Objective
To use the VisitTree Wizard to create a resource model from a CIM/WMI class.
Background information
None

Before you begin
None

When you finish
None

Procedure
1. Click the VisitTree Wizard icon to display the Data Source Selection Page window.
2. Click CIM/WMI Data Source to display the CIM/WMI Data Sources window.
3. Click a CIM class in the Defined Classes field to highlight the CIM class.
4. (Optional) Click Collection Test to retrieve all instances of a selected class from CIM/WMI.

Additional Information: As you progress through the wizard, you can use the collection test to show the inclusion of selected properties while removing unwanted properties.
5. Click **Next** to display the Specify the Event Triggering Conditions window.

6. **(Optional)** Specify event trigger conditions by doing the following:
   a. In the **Selected Properties** field, click a monitored property to highlight the property.
   b. Click the left-arrow to display the Event Triggering Condition window.
   c. Customize the following event trigger options to suit your environment:

   **<Selected Class Property>**
   Select one of the following logical operators to associate with the property:
   - **Is equal to**
     Select **Is equal to** and enter text in the text box to search for events that equal the specified text.
   - **Is not equal to**
     Select **Is not equal to** and enter text in the text box to search for events that do not equal the specified text.
   - **Contains**
     Select **Contains** and enter text in the text box to search for events that contain the specified text.

   **<Selected Class Property>**
   Select a logical operators to associate with the property and then enter the qualifying text in the text box. Operator choices include: >, <, =, <=. 
   - **Increases at least by**, % **increases at least by**, **Decreases at least by**, % **decreases at least by**, **Changes at least by**, **Changes by**, and **Is out of the range**.

   **Number of occurrences of this condition before the event is generated**
   Specify the number of occurrences of the specified condition that must be detected before the resource model generates an event.
Number of allowed monitoring cycles during which this condition does not occur
Enter the number of holes allowed before the resource model generates an event.

Event severity
Select the degree of severity to apply to this event. This severity is optionally passed to the Tivoli Enterprise Console and Tivoli Business Systems Manager. Choices include: CRITICAL, FATAL, HARMLESS, MINOR, and WARNING.

Send an event when a triggering condition that previously existed is cleared
Select this text box if you want an event to be generated when the conditions that previously caused an event have resolved.

Send this event to TEC
Select this option if you want the Tivoli Enterprise Console server to be notified when this event generates.

Send this event to TBSM
Select this option if you want Tivoli Business Systems Manager to be notified when this event generates.

d. Click OK to save your edits and return to the Specify Event Triggering Conditions window.

Additional Information: The Event Triggers field now displays the following event trigger options:

Name      The name of the Class property.
Condition The logical operator associated with the class property.
Value     The specified value associated with the condition.
Severity  The specified severity value for the event.

7. Click Next to display the Select the Properties to Log window.
8. **(Optional)** Do the following to log monitoring results for one or more monitored properties:
   a. In the **Available Properties** field, select the property for which you want to monitor results to highlight the property.
   b. Click the right-arrow to move the property to the **Logged Properties** field.

9. Click **Finish** to save your changes and return to the Data Source Selection Page window.

**Creating a resource model from a Distributed Monitoring (Classic Edition) Collection data source**

**Objective**
To use the VisitTree Wizard to create a resource model from a dumped monitoring collection.

**Background information**
None

**Before you begin**
You must import a dumped collection in order to use the wizard to create a resource model. For instructions, see "Importing a Tivoli Distributed Monitoring (Classic Edition) collection" on page 40.

**When you finish**
None

**Procedure**
1. Click the VisitTree Wizard icon to display the Data Source Selection Page window.
2. Click **Collection Data Source** to launch the Collection Data Source Wizard and display the Select Sentry Monitors window.
3. Click on any Sentry Monitor in the **Available Sentry Monitors** table to highlight the monitor.

4. *(Optional)* Click **Details** to display a detailed view of the selected monitor in the Monitoring Source Details window.

5. Click on the right-arrow to move the monitor to the **Selected Sentry Monitors** table.

6. Click **Next** to display the **Specify the Event Triggering Conditions** window.

7. Click on a property in the **Selected Properties** table to highlight the property.

8. Click the right-arrow to display the **Event Triggering Condition** window.

9. Specify the event triggering conditions as described in "Specifying event triggering conditions" on page 35.

10. Click **OK** to save your changes and return to the **Specify the Event Triggering Conditions** where the new changes display in the **Event Triggers** table.

11. Click **Next** to display the Select the Properties to Log window.

12. *(Optional)* Select the properties to log as described in "Selecting the properties to log" on page 37.

13. Click **Finish** to save your resource model and return to the Data Source Collection Page window.

14. Click **Finish** to display the Summary Page window.

15. *(Optional)* Change any internal name as described in "Changing the internal name" on page 38.
16. Click Next to display the Finalizing Creation window.

17. Perform one of the following steps:
   - Replace existing VisitTree code with the proposed code on the right by using the right-arrow.
     
     **Note:** Using the right-arrow button replaces all of the current code with the proposed code.

   —OR—

   - Highlight sections of the proposed code, and cut and paste the sections into the current code.

18. Click Finish to save your changes and close the wizard.

Creating a resource model from a custom script

Do the following to use the VisitTree Wizard to create a resource model from a custom script:

1. Click the VisitTree Wizard icon to display the Data Source Selection Page window.
2. Click Script Data Source to display the Specify the Dependency window.
3. Specify the shell command in the **Shell Command** text box.
4. Select one of the following script types:
   - **String**
   —OR—
   - **Numeric**
5. In the Monitoring Engine field, specify the engine version.
   - ITM - code generated will use Svc.Shell
     — OR —
   - AME - code generated will use Svc.ShellCmd
6. Click Next to display the Specify the Event Triggering Conditions window.

7. Select a class from the Selected Properties table to highlight the class.
8. Click the right-arrow icon to display the Event Triggering Condition window.
9. Configure these options as described in “Specifying event triggering conditions” on page 35
10. Click OK to return to Specify the Event Triggering Conditions window.
11. Click Next to display the Select the Properties to Log window.
12. Select properties to log as described in “Selecting the properties to log” on page 37
13. Click Finish to complete the edits and return to the Data Source Collection Page.
14. Click Finish to display the Summary Page window.
15. Click Next to display the Finalizing Creation window.

16. Perform one of the following steps:
   • Replace existing VisitTree code with the proposed code on the right by using the right-arrow.

     **Note:** Using the right-arrow button replaces all of the current code with the proposed code.

   —OR—

   • Highlight sections of the proposed code, and cut and paste the sections into the current code.

17. Click Finish to save your changes and close the wizard.
Chapter 7. Testing and debugging a resource model

This chapter describes the procedures to test and debug a new resource model.

The following is an overview table of testing and debugging a new resource model:

<table>
<thead>
<tr>
<th>Goal</th>
<th>Refer to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Running a resource model so you can test it on a local environment and return results.</td>
<td>“Running a resource model” on page 71</td>
</tr>
<tr>
<td>Reviewing the results of a locally run resource model using one of the following tools:</td>
<td>“Using the JavaScript Console” on page 73; “Using the RMB Event Viewer” on page 74; “Using the RMB Indication Viewer” on page 76; “Using the RMB Logging Viewer” on page 77</td>
</tr>
<tr>
<td>• JavaScript Console</td>
<td></td>
</tr>
<tr>
<td>• RMB Event Viewer</td>
<td></td>
</tr>
<tr>
<td>• RMB Indication Viewer</td>
<td></td>
</tr>
<tr>
<td>• RMB Logging Viewer</td>
<td></td>
</tr>
<tr>
<td>Debugging one of the following resource model types:</td>
<td>“Debugging a JavaScript resource model” on page 78; “Debugging a Visual Basic resource model” on page 79</td>
</tr>
<tr>
<td>• JavaScript resource model</td>
<td></td>
</tr>
<tr>
<td>—OR—</td>
<td></td>
</tr>
<tr>
<td>• Visual Basic resource model</td>
<td></td>
</tr>
<tr>
<td>Testing a resource model that has an M12 Java-based data provider.</td>
<td>“Testing using the M12 Java-based data provider” on page 80</td>
</tr>
</tbody>
</table>

Running a resource model

Objective

To run a resource model so you can test it in a local environment.

Note: You must use a data provider that runs on Windows to perform this procedure.

Background information

After creating your resource model, you can test it. To test a resource model, run the script to make sure it does not generate any errors.

Before you begin

Check the Problems view to verify that you do not have any errors in your resource model. If the Problems view displays errors, correct the errors before you run the resource model. The Problems view displays in the bottom pane of the IBM Tivoli Resource Model Builder:
When you finish

If an error displays, correct the error. For debugging a JavaScript resource model, see “Debugging a JavaScript resource model” on page 78. For debugging a Visual Basic resource model, see “Debugging a Visual Basic resource model” on page 79.

When you have fixed all the errors, run the resource model again. To make sure that the resource model actually detects the problems that you have specified, simulate those problems on your workstation when you run the resource model.

Procedure

1. Open the resource model.
2. Select the Source page to display the Source window.

3. Enable logging in the SetDefaultConfiguration method by doing the following:
   a. In the Decision Tree Script, scroll down to the text that reads ”Place your additional initializing code below.”
   b. Type the following text into the script:
      
      ```javascript
      Svc.EnableLogging(1)
      ```

4. Click the Run icon (right-arrow icon, shown below) from the menu bar to start testing.

**Additional Information:** Depending on the debug tool on your system, you might receive a prompt to choose a debugging tool. If you receive this prompt, select a debugging tool to begin the testing process.

5. Review the resource model data using any of the following views:
   - JavaScript Console
   - RMB Event Viewer
• RMB Indication Viewer
• RMB Logging Viewer

Additional Information: Use these views to see what information is sent to the IBM Tivoli Monitoring engine. For example, you can determine if calls to TMWServices.LogInst are being made at the appropriate times, and if the information being logged is correct. If the information is not correct, it can lead to problems distributing data into the Tivoli Enterprise Data Warehouse.

Using the JavaScript Console

Objective
To use the JavaScript Console so you can view the line number of a syntax error in a JavaScript resource model.

Note: This console is applicable to JavaScript resource models only.

Background information
This panel displays information if there is an error in the script. The line number of the error displays to assist in identifying the error.

The view is configurable to start or stop showing debug information. You can use the clear icon (shown below) to clear the contents of the view.

Before you begin
None

When you finish
Run the resource model again to verify the correction of any errors.
Procedure

1. Launch the JavaScript Console by running the resource model as described in “Running a resource model” on page 71.

2. Check the syntax errors displayed in the console.
   Additional Information: If no errors display in the JavaScript Console, the resource model is free of syntax errors, and you can exit this procedure.

3. Locate the line of code in which the error occurred.
   Additional Information: The line number in which the error occurred displays in the console.

4. Correct the syntax errors in the source script.

Using the RMB Event Viewer

Objective
To use the RMB Event Viewer so you can review the events that occurred for the resource model.

Background information
The views are configurable to start or stop showing debug information. In any of the views, you can use the clear icon (shown below) to clear the contents of the view.

Before you begin
Run the resource model as described in “Running a resource model” on page 71.
When you finish

Run the resource model again to verify the correction of any errors.

Procedure

1. Click the RMB Event Viewer tab to display the RMB Event Viewer in the bottom pane of the window.

You can view the following information with the RMB Event Viewer:

Event name
   The name of the event.

Event key
   The key attribute selected for the indication aggregation process.

Def occs
   The number of occurrences that you defined in the Events page for the indication aggregation process.

Cur occs
   The current number of occurrences.

Def holes
   The number of holes that you defined in the Events page.

Curr holes
   The current number of holes.

Event sent
   TRUE means an event was generated and sent. FALSE means an event was not generated and was not sent during the last cycle.
Using the RMB Indication Viewer

Objective
To use the RMB Indication Viewer so you can review the indications that occurred for the resource model.

Background information
The RMB Indication Viewer provides a view in which you can see the contents of calls to Svc.SendEvent and Svc.SendEventEx.

The view is configurable to start or stop showing debug information. You can use the clear icon (shown below) to clear the contents of the view.

Before you begin
Run the resource model as described in “Running a resource model” on page 71.

When you finish
Run the resource model again to verify the correction of any errors.

Procedure
1. Click the RMB Indication Viewer tab to display the RMB Indication Viewer in the bottom pane of the window.

2. You can review the following information:

   Event name
   Displays the event name generated by the resource model.
Numeric params
- Numeric parameters correlate with an event’s numeric attributes. This field shows each attribute’s value when the indication was sent.

String params
- String parameters correlate with an event’s string attributes. This field shows each attribute’s value when the indication was sent.

Progressive number
- Displays the cycle in which the indication occurred.

Using the RMB Logging Viewer

Objective
- To use the RMB Logging Viewer so you can review the events that occurred for the resource model.

Background information
- The RMB Logging Viewer provides a view in which you can see the contents of calls to Svc.LogInst and Svc.LogInstEx in real time.

To view the logging information you must add the following line at the end of the setDefaultConfiguration() function:

```java
Svc.EnableLogging(1);
```

**Note:** This is only applicable when testing your resource model in the Resource Model Builder and must be removed before packaging and deploying the resource model in an IBM Tivoli Monitoring environment.

The view is configurable to start or stop showing debug information. You can use the clear icon (shown below) to clear the contents of the view.

Before you begin
- Run the resource model as described in “Running a resource model” on page 71

When you finish
- Run the resource model again to verify the correction of any errors.
Procedure

1. Click the **RMB Logging Viewer** tab to display the **RMB Logging Viewer** in the bottom pane of the window.

2. You can review the following information:
   - **Resource**: Displays the logging resource.
   - **Context**: Displays the logging context.
   - **Keys**: Displays the key logging properties and their values.
   - **Metrics**: Displays the numeric properties and their values for the logging object.
   - **Categories**: Displays the string properties and their values for the logging object.

**Debugging a JavaScript resource model**

**Objective**

To debug a JavaScript resource model so you can correct any errors in a resource model.

**Background information**

You can debug a JavaScript resource model with debugging tools, such as Microsoft Visual Studio™ or Script Debugger for Windows™. If you do not have a debugging program in your environment, you can obtain a free program by searching for a script debugger, or by downloading one from the following Web site:
You can also debug a JavaScript resource model without a debugging program. Use the trace and logging files to search for resource model errors.

Before you begin
None

When you finish
When you have fixed all the errors, run the resource model again. To make sure that the resource model actually detects the problems that you specified, simulate those problems on your workstation when you run the resource model. See “Running a resource model” on page 71

Procedure
1. Run the resource model as described in “Running a resource model” on page 71

   Additional Information: If your system has a debugging program installed, the debugger window automatically pops up after the resource model runs. If you get a prompt asking you if you want to debug an error, click Yes to launch the debugger.

2. Determine the cause of the error by doing one of the following:
   • Use the IBM Tivoli Resource Model Builder tools by doing the following:
     a. Check the JavaScript Console for a line number of where the error occurred.

        Additional Information: Errors displayed in the JavaScript Console are a result of syntax errors in the code. If you resolve these errors, and the resource model runs without errors, you do not need to examine the trace or log files. If no error displays in this view, then the error is not syntax-related, and you should go to the next step.
     b. Examine the tracing (trace file) or logging (log file) to determine where a logic error is located.

   —OR—

   • Use a debugging program to resolve script errors by doing the following:
     a. From this debugger window, step into the code to set break points.
     b. Run the resource model again to pinpoint and correct any code errors.
     c. If there continues to be a problem in the resource model, check the log and check the last item that was traced to determine the origin of the error.

Debugging a Visual Basic resource model

Objective
To debug a Visual Basic resource model so you can correct any errors in the resource model.

Background information
In the decision tree script, lines containing an error are highlighted in the Visual Basic editor.
Before you begin
None

When you finish
When you have fixed all the errors, run the resource model again. To make sure that the resource model actually detects the problems that you have specified, simulate those problems on your workstation when you run the resource model.

Procedure
1. Run the resource model as described in “Running a resource model” on page 71.
2. Determine error origins by stepping into the code and setting break points in the Visual Basic editor.
3. Run the resource model again to pinpoint and correct any code errors.
4. If there continues to be a problem in the resource model, check the log for the last item that was traced to determine the cause of the error.

Testing using the M12 Java-based data provider

Objective
To test a resource model that retrieves its data from a Java provider so you can identify and correct problems.

Background information
After creating a resource model that retrieves its data from a Java data provider, you can test it in the IBM Tivoli Resource Model Builder. Note that testing only works if the Java data provider can run on Windows and has a MOF file that is compatible with WMI.

Before you begin
Make sure the MOF file for your Java provider is compiled into the WMI repository. If the MOF file is not correctly registered in WMI, a warning appears in the Problems view.

When you finish
None

Procedure
1. Copy the Java provider jars and any other supporting files to the following directory:

   `<RMBInstallDir>\eclipse\plugins\com.ibm.tivoli.rmb.model_1.1.3\Lcfnew\Itm\Pacs\<Category>`

   where:

   `<RMBInstallDir>`
   The directory into which you installed the IBM Tivoli Resource Model Builder.

   `<Category>`
   The internal category name of the resource model you are testing.
2. Add the following line of code at the end of the function "SetDefaultConfiguration":

```
Svc.SetCategory("<Category>");
```

where:

```
<Category>
```

The internal category name of the resource model you are testing.

3. *(Optional)* If your resource model (or the data provider that your resource model uses) relies on context information, you can insert calls to the following Service Object method in the Init section of the decision tree to set this information.

```
void setPropertyContext(String key, String value)
```

**Note:** Only use this method during testing in the IBM Tivoli Resource Model Builder.

4. Run the resource model by clicking the Run icon (right-arrow icon, shown below) from the menu bar to start testing.

5. *(Optional)* Verify that indications and events are being sent as configured through the RMB Indication Viewer and RMB Event Viewer views.
Chapter 8. Building a resource model

This chapter describes the procedures to build a complete resource model package or an individual component of the package.

The following table provides an overview of the procedures for building a resource model:

<table>
<thead>
<tr>
<th>Goal</th>
<th>Refer to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Building a resource model into a complete package so it can be installed on the IBM Tivoli Monitoring server.</td>
<td>“Generating the Package” on page 83</td>
</tr>
<tr>
<td>Building a Java message file to display information in IBM Tivoli Monitoring windows.</td>
<td>“Generating the Java Message Catalog” on page 85</td>
</tr>
<tr>
<td>Building a message file to display information in IBM Tivoli Monitoring windows.</td>
<td>“Generating the Message Catalog” on page 86</td>
</tr>
<tr>
<td>Building and exporting a TEC BAROC file to view resource model events on the Tivoli Enterprise Console.</td>
<td>“Generating the Tivoli Enterprise Console BAROC” on page 87</td>
</tr>
<tr>
<td>Generating a configuration file for the IBM Tivoli Monitoring engine.</td>
<td>“Generating an IBM Tivoli Monitoring configuration file” on page 87</td>
</tr>
<tr>
<td>Generating a MOF File for the resource model.</td>
<td>“Generating a MOF file” on page 88</td>
</tr>
<tr>
<td>Generating a JavaScript or Visual Basic Script file for this resource model.</td>
<td>“Generating a script file” on page 88</td>
</tr>
<tr>
<td>Generating an HTML document to view the contents and configurations of a resource model.</td>
<td>“Generating HTML documents” on page 89</td>
</tr>
</tbody>
</table>

Generating the Package

Objective

To build your resource model into a complete package so it can be installed on the Tivoli management region.

Background information

After testing your resource model, you must generate a tar (tape archive format) file and then install it on the Tivoli management region. When you run the Generating the package procedure, the package generated includes the script, dependencies, and settings necessary to install the resource model.

Before you begin

Test the resource model, as described in “Running a resource model” on page 71.

When you finish

You must generate the package of your resource model so that you can deploy the resource model to the Tivoli management region. Completing all the steps in this
procedure installs the generated file on the Tivoli management region. After the file is installed on the Tivoli management region, your resource model is ready to function.

**Procedure**

You can perform this procedure from the command-line interface or the IBM Tivoli Resource Model Builder.

**Command line**

Use the `bldpkg` command to package a resource model. For more information about the `bldpkg` command, see “`bldpkg` on page 95”.

Use the `bdamepkg` command to package a resource model for the Autonomic Management Engine. For more information about the `bdamepkg` command, see “`bdamepkg` on page 94”.

**Desktop**

1. Open the resource model.
2. Click RMB from the top menu to display the RMB drop-down menu.
3. Select Generate Package and choose one of the following options:
   - ITM (tar): Creates a .tar file for use on a distributed monitoring system.
   - AME (zip): Creates a .zip file for use with Autonomic Management Engine
4. Select the file to export in the Export Package window.
5. Enter the destination directory or click Browse to open the Save As window.
6. Install the resource model. Your installation method varies based on the type of package generated, and the version of IBM Tivoli Monitoring you are using. Enter the following command from the command line:
   
   `wdmrm -add <outputPackage> [.tar]`

   where `<outputPackage>` is the name of the file.

---

**Generating resource model settings in XML**

**Objective**

To generate an XML file of your resource model settings so you can use the XML file for resource model review and implementation.

**Background information**

None

**Before you begin**

Test the resource model, as described in “Running a resource model” on page 71.

**When you finish**

You must generate the package of your resource model so that you can deploy the resource model to the Tivoli management region. Completing all the steps in this procedure installs the generated file on the Tivoli management region. After the file is installed on the Tivoli management region, your resource model is ready to function.
Procedure
1. Open the resource model for which you want to generate a settings XML file.
2. Click RMB from the top menu to display the RMB drop-down menu.
3. Select Generate RM Settings XML.
4. Select the file destination directory and click Save to generate the file.

Generating the Java Message Catalog

Objective
To build a Java message file containing all the information that is displayed in IBM Tivoli Monitoring windows.

Background information
You can build a Java message file containing all the information that is displayed in IBM Tivoli Monitoring windows. This is useful if you want to translate your text information into other languages. The IBM Tivoli Resource Model Builder is designed to generate resource model packages that contain all the strings for the English locale, therefore DBCS characters are not supported. All localized strings should be handled separately from the IBM Tivoli Resource Model Builder.

Export the English version of the message catalog and Java message catalog, then translate them into the desired language, compile them using Gencat tool (included in the Application Development Environment), and finally install them manually in the message bundle on the Tivoli management environment server.

NLS Note
Before compiling DBCS message catalogs (such as Japanese) with Gencat, you must convert the original files into UTF8 files issuing the following Framework command:
```text
wiconv
```

Before you begin
Test the resource model, as described in “Running a resource model” on page 71.

When you finish
None

Procedure
You can perform this procedure from the command-line interface or the IBM Tivoli Resource Model Builder.

Command line
Use the `expjmsgcat` command to package a Java message file. For more information about the `expjmsgcat` command, see “`expjmsgcat`” on page 99.

Desktop
1. Open your resource model.
2. Click RMB from the top menu to display the RMB drop-down menu.
3. Click **Generate Java Message Catalog** to open the Build Java Message Catalog window.
4. Save the file with a .java extension.

---

### Generating the Message Catalog

#### Objective
To build a message file containing all the information that is displayed in IBM Tivoli Monitoring windows.

#### Background information
You can build a message file containing all the information that is displayed in IBM Tivoli Monitoring windows. This is useful if you want to translate your text information into other languages. The IBM Tivoli Resource Model Builder is designed to generate resource model packages that contain all the strings for the English Locale, therefore DBCS characters are not supported. All localized strings should be handled separately from the IBM Tivoli Resource Model Builder.

Export the English version of the message catalog and Java message catalog, then translate them into the desired language, compile them using the Gencat tool (included in the Application Development Environment), and finally install them manually in the message bundle on the Tivoli management environment server.

---

**NLS Note**

Before compiling Japanese message catalogs (such as Japanese) with Gencat, you must convert the original files into UTF8 files issuing the following Framework command:

```
wiconv
```

---

#### Before you begin
Test the resource model, as described in "Running a resource model" on page 71.

#### When you finish
None

#### Procedure
You can perform this procedure from the command-line interface or the IBM Tivoli Resource Model Builder.

**Command line**
Use the `expmsgcat` command to package a message file. For more information about the `expmsgcat` command, see "expmsgcat" on page 100

**Desktop**
1. Open your resource model.
2. Click **RMB** from the top menu to display the **RMB** drop-down menu.
3. Click **Generate Message Catalog** to open the Build Message Catalog window.
4. Save the file with a .msg extension.
Generating the Tivoli Enterprise Console BAROC

Objective
To build and export a Tivoli Enterprise Console BAROC file so you can see your resource model events on the Tivoli Enterprise Console.

Background information
To see your resource model events on the Tivoli Enterprise Console, you have to build and export a Tivoli Enterprise Console BAROC file. This file contains all the event definitions specified in the BAROC language. This file can be installed on a Tivoli Enterprise Console rule base and allows the Tivoli Enterprise Console to display the events of your resource model.

Before you begin
Test the resource model, as described in “Running a resource model” on page 71.

When you finish
None

Procedure
You can perform this procedure from the command-line interface or the IBM Tivoli Resource Model Builder.

Command line
Use the bldtecbaro c command to generate a Tivoli Enterprise Console BAROC file. For more information about the bldtecbaro c command, see “bldtecbaro c” on page 96.

Desktop
1. Open your resource model.
2. Click RMB from the top menu to display the RMB drop-down menu.
3. Select Generate TEC BAROC to open the Build TEC BAROC window.
4. Select the file to save.
5. Save the file with a .baroc extension.
6. Click Finish.

Generating an IBM Tivoli Monitoring configuration file

Objective
To generate a resource model configuration file for IBM Tivoli Monitoring engine use and interpretation.

Background information
IBM Tivoli Monitoring uses a resource model configuration file to run the resource model. The configuration file describes the contents of the resource model for the IBM Tivoli Monitoring engine.

Before you begin
Test the resource model, as described in “Running a resource model” on page 71.
When you finish
None

Procedure

Command line
Use the `expconf` command to generate an IBM Tivoli Monitoring configuration file. For more information about the `expconf` command, see ["expconf" on page 98](#).

Desktop
1. Open your resource model.
2. Click RMB from the top menu to display the RMB drop-down menu.
3. Select Generate ITM Configuration File to open the Generate ITM Config File window.
4. Select the location in which to save the file with a .conf extension.
5. Click Save to save the file.

Generating a MOF file

Objective
To generate a Managed Object Format (MOF) file for this resource model.

Background information
A MOF file contains the resource model indications and data sources. It is used on Windows engines to set up CIM classes that the Windows engine needs to run.

Before you begin
Test the resource model, as described in ["Running a resource model" on page 71](#).

When you finish
None

Procedure
1. Open your resource model.
2. Click RMB from the top menu to display the RMB drop-down menu.
3. Select Generate MOF to open the Build MOF File window.
4. Select the file to save with a .mof extension.
5. Click Finish to save the file.

Generating a script file

Objective
To generate a JavaScript or Visual Basic Script file for this resource model so you can review the script for debugging purposes.

Background information
None
Before you begin
Test the resource model, as described in “Running a resource model” on page 71.

When you finish
None

Procedure
1. Open your resource model.
2. Click RMB from the top menu to display the RMB drop-down menu.
3. Select Generate Script to open the Build JavaScript File or Build Visual Basic Script File window.
4. Select the file to save with a .js or .vba extension.
5. Click Finish to save the file.

Generating HTML documents

Objective
To build a HTML document that contains all the information related to your resource model and included in the descriptive sections of the resource model components.

Background information
A generated HTML document enables you to view the contents and configurations of your resource model.

Before you begin
Test the resource model, as described in “Running a resource model” on page 71.

When you finish
None

Procedure
1. Open your resource model.
2. Click RMB from the top menu to display the RMB drop-down menu.
3. Select Generate HTML Docs to open the HTML Creation window.
4. Select the file to save with a .html extension.
5. Click Finish to save the file.
Appendix A. Command-line options

This chapter lists and describes the IBM Tivoli Resource Model Builder commands. Table 10 on page 92 summarizes these command-line options. After the table, a section for each command describes how to run the command by covering the following information:

Purpose
Lists the purpose of the command.

Format
Specifies the syntax that you enter on the command line. The syntax contains the command name, and a list of the parameters for the command. A definition of each parameter follows the command name.

Examples
The example for the command contains a brief description of the example and an example of the syntax.

Usage
P Provides an explanation of the command and its purpose.

Comments
Provides additional commands or text that you can refer to for more information.

Tivoli command syntax

The following special characters define Tivoli command syntax:

[ ] Identifies elements that are optional. Those not enclosed in brackets are required.

... Indicates that you can specify multiple values for the previous element. Separate multiple values by a space, unless otherwise directed by a command’s information.

If the ellipsis for an element follows a closing bracket, use the syntax within the brackets to specify multiple values. For example, to specify two administrators for the option [–a admin]..., use –a admin1 –a admin2.

If the ellipses for an element is within the brackets, use the syntax of the last element to specify multiple values. For example, to specify two hosts for the option [–h host ...], use –h host host2.

| Indicates mutually exclusive information. You can use the element on either the left or right of the vertical bar.

‘{ }’ Delimits a set of mutually exclusive elements when one of them is required. If the elements are optional, they are enclosed in brackets ([ ]).

In addition to the special characters, the typeface conventions described in the Preface are used.

Using commands with IBM Tivoli Resource Model Builder

When running commands for use with IBM Tivoli Resource Model Builder, first set your PATH environment variable to include the location of the wrmbcli file:

< RMB_INSTALL_DIR >\eclipse\plugins\com.ibm.tivoli.rmb.model_1.1.3
where `<RMB_INSTALL_DIR>` is the name of the installation directory (by default, this name is IBM Tivoli Resource Model Builder).

Before using the commands in this section, bring up a command prompt and change to the current directory so you can access the files easily. For example, cd to the following directory:

```
Program Files\IBM Tivoli Resource Model Builder\eclipse\<workspace>\[<current project directory>]
```

## List of commands

The following table lists the name and purpose statement for each command option for the `wrmcli` command:

<table>
<thead>
<tr>
<th>Command</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>addInterp</td>
<td>Adds platforms (interps) to the resource model.</td>
</tr>
<tr>
<td>bldamepkg</td>
<td>Builds the output .zip package file for the resource model.</td>
</tr>
<tr>
<td>bldpkg</td>
<td>Builds the output .tar package file for the resource model.</td>
</tr>
<tr>
<td>bldteccbaroc</td>
<td>Builds the BAROC Tivoli Enterprise Console® event file for the resource model.</td>
</tr>
<tr>
<td>convert</td>
<td>Migrates resource models from IBM Tivoli Monitoring Workbench, Version 5.1.1 to IBM Tivoli Resource Model Builder.</td>
</tr>
<tr>
<td>expconf</td>
<td>Generates the IBM Tivoli Monitoring configuration file for the resource model.</td>
</tr>
<tr>
<td>expjmsgcat</td>
<td>Exports the Java message catalog from the resource model.</td>
</tr>
<tr>
<td>expmsgcat</td>
<td>Exports the message catalog from the resource model.</td>
</tr>
<tr>
<td>exptmsxml</td>
<td>Exports the messages from the specified resource model workspace into the TMS XML format.</td>
</tr>
<tr>
<td>extract</td>
<td>Extracts the resource model XML file and dependency files from the specified workspace file and places them in the current directory or the directory specified.</td>
</tr>
<tr>
<td>imptmsxml</td>
<td>Imports the TMS XML formatted messages into the specified resource model workspace.</td>
</tr>
<tr>
<td>pack</td>
<td>Packs the Model.xml file from resmodDirectory and any dependencies specified in the depDirectory into the specified workspace.</td>
</tr>
<tr>
<td>setcopyright</td>
<td>Imports a copyright statement into files that the resource model generates. The file containing the copyright statement must be an ASCII text file.</td>
</tr>
<tr>
<td>setdectree</td>
<td>Sets the decision tree implementation of the resource model workspace or the output workspace to the decision tree contained in the specified decision tree file.</td>
</tr>
<tr>
<td>setdep</td>
<td>Adds the dependency file for the specified interp to the specified resource model workspace or output workspace.</td>
</tr>
<tr>
<td>setsendtotbsm</td>
<td>Denotes whether events in the specified resource model are forwarded to Tivoli Business Systems Manager.</td>
</tr>
<tr>
<td>setver</td>
<td>Sets the major and minor versions of the resource model workspace or output workspace.</td>
</tr>
</tbody>
</table>
addInterp

Purpose

Adds interp files for the resource model.

Format

\texttt{wrmbcli <resmodworkspace> -addInterp \textless w32-ix86, hpux10, os400, solaris2, linux-ix86, os2-ix86, aix4-rl, linux-s390, linux-ppc\textgreater}

where:

\textless resmodworkspace\textgreater is the name of the resource model workspace.

\textless addInterp \textless w32-ix86, hpux10, os400, solaris2, linux-ix86, os2-ix86, aix4-rl, linux-s390, linux-ppc\textgreater are the names of the platform.

Examples

In the following example, a linux-ppc and hpux10 are added to the Process resource model.

\texttt{wrmbcli Process.jrm -addInterp linux-ppc,hpux10}

Usage

The interps specified will be added as supported platforms for the workspace. This only applies to Javascript workspace files.

Comments

None
bldamepkg

Purpose
Builds the zip output package file for the resource model.

Format
wrmbcli <resmodworkspace> -bldamepkg <outputPackage>

where:

<resmodworkspace> is the name of the resource model workspace.

<outputPackage> is the name of the output package (.zip file) for the resource model.

Examples
In the following example, a package is being built for a Win32_PnPDevices workspace and Win32_PnPDevices is the name of the output package:
wrmbcli PnPDevices.jrm -bldamepkg Win32_PnPDevices.zip

Usage
The .zip file is created in the current directory. After you create and test a resource model, you must package the resource model into a .zip file. This file is the package used by the Autonomic Management Engine (AME) to install the resource model.

When you run bldamepkg, you generate all of the following into one .zip file:
• IBM Tivoli Monitoring configuration file
• Message catalog
• AME-specific package

Comments
For more information on packaging a resource model, see “Generating the Package” on page 83.
bldpkg

**Purpose**

Builds the .tar output package file for the resource model.

**Format**

```
wrmbcli <resmodworkspace> -bldpkg <outputPackage>
```

where:

- `<resmodworkspace>` is the name of the resource model workspace.
- `<outputPackage>` is the name of the output package (.tar file) for the resource model.

**Examples**

In the following example, a package is being built for a Win32_Process workspace and Win32_Process is the name of the output package:

```
wrmbcli Win32_Process.jrm -bldpkg Win32_Process.tar
```

**Usage**

The .tar file is created in the current directory. After you create and test a resource model, you must package the resource model into a .tar file. This file is the package used by the IBM Tivoli Monitoring Tivoli management region server to install the resource model.

When you run **bldpkg**, you generate all of the following into one .tar file:

- IBM Tivoli Monitoring configuration file
- Message catalog
- Platform-specific packages that contain the IBM Tivoli Monitoring configuration file, message catalog, defined dependencies, and script file

After building your .tar file, install the resource model on your IBM Tivoli Monitoring 5.1.x Tivoli Management Region server by typing the following command:

```
wmrm -add <outputPackage>.tar
```

where `<outputPackage>` is the name of the .tar file.

**Comments**

For more information on packaging a resource model, see “Generating the Package” on page 83.
bldtecbaroc

Purpose
Builds the Tivoli Enterprise Console BAROC event file for the resource model.

Format
wrmbcli <resmodworkspace> -bldtecbaroc <tecbaroc>

where:

<resmodworkspace> is the name of the resource model workspace.

<tecbaroc> is the name of the Tivoli Enterprise Console BAROC event file for the resource model.

Examples
wrmbcli OracleUser.vrm -bldtecbaroc Oracle.baroc

Usage
The bldtecbaroc command creates the baroc file in the current directory. You can add this file to a rule base on the Tivoli Enterprise Console server so that a resource model can send events to this server.

Comments
For more information on generating a Tivoli Enterprise Console BAROC file, see "Generating the Tivoli Enterprise Console BAROC" on page 87.
convert

Purpose
Migrates resource models from IBM Tivoli Monitoring Workbench, Version 5.1.1 to IBM Tivoli Resource Model Builder.

Format
wrmbcli <OldResourceModel> -convert <NewResourceModel>

where:

<OldResourceModel> is a .dmws or .dmjsws file.

<NewResourceModel> is a .vrm or .jrm file.

Examples
wrmbcli TMW_EventLog.dmws -convert TMW_EventLog.vrm

Usage
The file specified in the <NewResourceModel> must be the correct type for the <OldResourceModel> specified. For example, conform to the following conventions:

• If the old resource model is .dmws, then the new resource model must be .vrm.

—OR—

• If the old resource model is .dmjsws, then the new resource model must be .jrm.

Comments
None
expconf

**Purpose**
Generates the IBM Tivoli Monitoring configuration file for the resource model.

**Format**
```
wrmbcli <resmodworkspace> -expconf <confFile>
```
where:
- `<resmodworkspace>` is the name of the resource model workspace.
- `<confFile>` is the name of the IBM Tivoli Monitoring configuration file to be generated.

**Examples**
In the following example, the IBM Tivoli Monitoring configuration file WizRM_time_sh.conf is being generated for the WizRM_time_sh.jrm workspace:
```
wrmbcli WizRM_time_sh.jrm -expconf WizRM_time_sh.conf
```

**Usage**
The specified file is created in the current directory. IBM Tivoli Monitoring uses a resource model configuration file to run the resource model. The configuration file describes the contents of the resource model for the IBM Tivoli Monitoring engine.

**Comments**
For more information on packaging a resource model, see “Generating the Package” on page 83.
expjmsgcat

Purpose
Exports the java message catalog from the resource model.

Format
wrmbcli <resmodworkspace> -expjmsgcat <javamessagecatalog>

where:
<resmodworkspace> is the name of the resource model workspace.
<javamessagecatalog> is the name of the java message catalog.

Examples
wrmbcli DB2Agents.jrm -expjmsgcat DB2Agents.java

Usage
The expjmsgcat command creates the java message catalog file in the current directory.

Comments
For more information on generating a java message catalog file, see “Generating the Java Message Catalog” on page 85.
expmsgcat

**Purpose**
Exports the message catalog from the resource model.

**Format**
```
wrmbcli <resmodworkspace> -expmsgcat <messagecatalog>
```
where:
- `<resmodworkspace>` is the name of the resource model workspace.
- `<messagecatalog>` is the name of the message catalog.

**Examples**
```
wrmbcli CIM_Product.vrm -expmsgcat CIM_Product.msg
```

**Usage**
The `expmsgcat` command creates the message catalog file in the current directory.

**Comments**
For more information on generating a message catalog file, see “Generating the Message Catalog” on page 86.
exptmsxml

Purpose
Exports the messages from the specified resource model workspace into the Tivoli Message Standard (TMS) XML format.

Format
wrmbcli <resmodworkspace> -exptmsxml <TMSXMLfile>

where:

<resmodworkspace> is the name of the resource model workspace.
<TMSXMLfile> is the name of the file into which the messages will be exported.

Examples
wrmbcli CIM_Error.jrm -exptmsxml XMLmsg.xml

Usage
Exports the messages from the specified resource model workspace into the specified file in TMS XML format.

Comments
None
extract

Purpose
Extracts the resource model XML file and dependency files from the specified workspace file and places them in the current directory or the directory specified.

Format
wrmbcli <resmodworkspace> -extract

where:

<resmodworkspace> is the name of the resource model workspace.

<extract directory> is the name of the directory to which you want the files extracted.

Examples
wrmbcli NT_Memory.jrm -extract

Usage
The extract command can have two results, depending on the number of arguments. The resource model XML and dependency directories and files are extracted to the current directory or the extract directory if one is provided.

Comments
None
imptmsxml

Purpose
Imports the Tivoli Message Standard (TMS) XML formatted messages into the specified resource model workspace.

Format
wrmbcli <resmodworkspace> -imptmsxml <TMSXMLfile>

where:

<resmodworkspace> is the name of the resource model workspace.

<TMSXMLfile> is the name of the formatted file of messages to be imported into a resource model workspace.

Examples
wrmbcli Win32_Thread.vrm -imptmsxml XMLmsg.xml

Usage
Imports the specified TMS XML formatted file of messages into the specified resource model workspace.

Comments
None
pack

Purpose

Packs the Model.xml file from resmodDirectory and any dependencies specified in the depDirectory into the specified workspace.

Format

wdmwbcli <resmodworkspace> -pack <resmodXMLdir> [depDirectory]

where:

<resmodworkspace> is the name of the resource model workspace.
<resmodXMLdir> is the name of the resource model directory.
<depDirectory> is the name of the dependency directory.

Examples

wdmwbcli NT_Dump.vrm -pack XMLdir dependencies

Usage

The -pack command looks for a Model.xml file in the resmodXMLdir directory and any dependencies stored in the appropriate directory structure in depDirectory.

Comments

None
setcopyright

Purpose
Imports a copyright statement into files that the resource model generates. The file containing the copyright statement must be an ASCII text file.

Format
wrmbcli <resmodworkspace> -setcopyright <copyrightfile>

where:

<resmodworkspace> is the name of the resource model workspace.
<copyrightfile> is the name of the copyright file to add.

Examples
In the following example, a file is being specified that defines a copyright for the Win32_Process workspace.
wrmbcli Win32_Process.jrm -setCopyright copyright.txt

Usage
The copyright statement in the file will be added to the workspace. It will be appended to the beginning of all created files except the script file and the HTML docs. This file must be a plain text file.

Comments
None
setdectree

Purpose
Sets the decision tree implementation of the resource model workspace or the output workspace to the decision tree contained in the specified decision tree file.

Format
wrmbcli <resmodworkspace> -setdectree <dectreefile> [<outputworkspace>]

where:

<resmodworkspace> is the name of the resource model workspace.
<dectreefile> is the file that contains the decision tree.
<outputworkspace> is the name of the output workspace.

Examples
wrmbcli CIM_Class.vrm -setdectree DecisionTree.txt

Usage
This command produces two results depending on the number of arguments. If the output workspace is not provided, the specified workspace is modified to use the decision tree provided in the plain text file. If the output workspace is specified, the first workspace is left intact and the output is a copy of it that uses the decision tree from the plain text file.

Comments
None
setdep

Purpose

Adds one or more dependency files for the specified interp to the specified resource model workspace or output workspace.

Format

```
wrmbcli <resmodworkspace> -setdep <interp> <depfile1> [<depfile2>]
[<outputworkspace>]
```

where:

- `<resmodworkspace>` is the name of the resource model workspace.
- `<interp>` is the name of the interp.
- `<depfile1>`, `<depfile2>` is the name of the dependency file or files to be added to the workspace.
- `<outputworkspace>` is the name of the output workspace.

Examples

```
wrmbcli Test_bat.jrm -setdep w32-ix86 test.bat Test_bat_w32-ix86.jrm
```

Usage

This command produces two results depending on the number of arguments. If the output workspace is not provided, the given workspace is modified to include this dependency. If the output workspace is given, the first workspace is left intact and the output is a copy of it that includes the dependencies.

Comments

None
setsendtotbsm

Purpose
Denotes whether events in the specified resource model are forwarded to Tivoli Business Systems Manager.

Format
wrmbcli <resmodworkspace> -setsendtotbsm {true | false}

where:

<resmodworkspace> is the name of the resource model workspace.

true send the events to Tivoli Business Systems Manager.

false does not send the events to Tivoli Business Systems Manager.

Examples
wrmbcli W2K_memory.vrm -setsendtotbsm false

Usage
The setsendtotbsm command gives Tivoli Business Systems Manager access to the specified workspace when true, and disables the access when false.

Comments
None
setver

Purpose
Sets the major and minor versions of the resource model workspace or output workspace.

Format
wrmbcli <resmodworkspace> -setver <major> <minor> [ <outputworkspace> ]

where:
<resmodworkspace> is the name of the resource model workspace.
<major> is the major version of the resource model workspace or output workspace.
<minor> is the minor version of the resource model workspace or output workspace.
<outputworkspace> specifies the output workspace.

Examples
wrmbcli CIM_Data.jrm -setver 2 5 CIM_Data_mod.jrm

Usage
This command produces two results depending on number of arguments. If the output workspace is not provided, the given workspace is updated to use the given version numbers. If the output workspace is given, the first workspace is left intact and the output is a copy of it that uses the given version numbers.

Comments
None
Appendix B. Service object method library

This appendix describes the Class Object library, including the syntax used to express all main functions needed to create an object in a resource model. The appendix also lists guidelines for producing scripts and possible errors that can occur when you run a resource model.

The IBM Tivoli Resource Model Builder enables you to program your own functions and create customized scripts. Customized resource models are less predictable, so it is essential to test your scripts thoroughly before distributing them.

Note: The following methods are written in Visual Basic. You can write the same methods in JavaScript by modifying the syntax and complying with the code rules. The following table lists some examples of the same methods expressed in both languages:

<table>
<thead>
<tr>
<th>Visual Basic</th>
<th>JavaScript</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetNumOfInst (ClassName As String) As Long</td>
<td>int GetNumOfInst (String className)</td>
</tr>
<tr>
<td>GetNumProperty (ClassName As String, idxAs Long, PropName As String) As Double</td>
<td>double GetNumProperty (String className, int index, String propertyName)</td>
</tr>
<tr>
<td>DefineClass (Source As String, AliasName As String, RealClassName As String, WhereClause As String, NumProps As String, StrProps As String, SortType As String, SortField As String, Top as long, Every as long)</td>
<td>DefineClass (String source, String aliasName, String realName, String whereClause, String numProperties, String strProperties, String sortType, String sortField, int Top, int every)</td>
</tr>
</tbody>
</table>

The following table outlines the main differences between the two languages:

<table>
<thead>
<tr>
<th>Visual Basic</th>
<th>JavaScript</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parameter name As Parameter Type.</td>
<td>Parameter Type Parameter name.</td>
</tr>
<tr>
<td>The type of value returned by the function is expressed at the end of the function. Example: GetNumProperty(...) As Double</td>
<td>The type of value returned by the function is expressed at the beginning of the function. Example: double GetNumProperty(...)</td>
</tr>
<tr>
<td>Not applicable</td>
<td>Instructions end with a semicolon.</td>
</tr>
</tbody>
</table>

Basic Object Methods

This section describes basic methods for the TMWService Object. You can refer to it for the syntax required to call the specific methods.

General settings

The following methods help you to express functions for resource model configuration.
**Method GetModelName**

Syntax

```csharp
Object.GetModelName() As String
```

Description

Returns the name of the resource model.

Error code

S_OK

**Method GetCycleTime**

Syntax

```csharp
Object.GetCycleTime() As Double
```

Description

Returns the cycle time value.

Error code

S_OK

**Example**

*Table 13. Basic Object Method general settings examples*

<table>
<thead>
<tr>
<th>Visual Basic example:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dim modelName As String</td>
</tr>
<tr>
<td>modelName = Svc.GetModelName()</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>JavaScript example:</th>
</tr>
</thead>
<tbody>
<tr>
<td>var modelName;</td>
</tr>
<tr>
<td>modelName = Svc.GetModelName();</td>
</tr>
</tbody>
</table>

**Dynamic model**

The following methods help you to express functions for dynamic model configuration.

**Method GetNumOfInst**

Syntax

```csharp
Object.GetNumOfInst(ClassName As String) As Long
```

Parameters

- **ClassName**
  
  The name of the Dynamic Model class.

Description

Returns the number of the collected instances of the Dynamic Model class named `ClassName` in the current monitoring cycle.

Remarks

The `ClassName` must be previously defined through in the Dynamic model window.

Error codes

- S_OK
- TMWSERVICE_E_CLASS_NOT_FOUND
Method GetNumProperty

Syntax

Object.GetNumProperty(ClassName As String, idx As Long, PropName As String) As Double

Parameters

ClassName
   The name of the Dynamic Model class.

idx    The instance index.

PropName
   The name of a NUMERIC property.

Description
   Returns the value of the property named PropName of the instance number idx of the Dynamic Model class named ClassName.

Remarks
   The idx counter goes from 0 to NumOfInst -1. The ClassName must be previously defined in the Dynamic model window.

Error codes
   S_OK
   TMWSERVICE_E_PROPERTY_NOT_FOUND

Method GetStrProperty

Syntax

Object.GetStrProperty(ClassName As String, idx As Long, PropName As String) As String

Parameters

ClassName
   The name of the Dynamic Model class.

idx    The instance index.

PropName
   The name of a STRING property.

Description
   Returns the value of the property named PropName of the instance number idx of the Dynamic Model class named ClassName.

Remarks
   idx counter goes from 0 to NumOfInst -1. The ClassName must be previously defined in the Dynamic Model window.

Error codes
   S_OK
   TMWSERVICE_E_PROPERTY_NOT_FOUND

Method AssociateParameterToClass

Syntax

Object.AssociateParameterToClass(ParameterName As String, ClassName As String)

Parameters

Appendix B. Service object method library  113
**ParameterName**

The name of the parameter to use as the argument for the provider operations on the specified class.

**ClassName**

The name of the Dynamic Model class whose provider operations are to be executed using the specified parameter as an argument.

**Description**

Defines an existing parameter as the argument for the provider operations on the specified class.

**Remarks**

This method is for UNIX operating systems only. The **ClassName** must be previously defined in the Dynamic model window.

**Method AssociateParameterToClassProperty**

**Syntax**

```csharp
Object.AssociateParameterToClassProperty(ParameterName As String, ClassName As String, Property As String)
```

**ParameterName**

The name of the parameter to use as the argument for the provider operations on the specified class property.

**ClassName**

The name of the Dynamic Model class to which the property belongs.

**PropertyName**

The name of the property whose provider operations are to be executed using the specified parameter as an argument.

**Description**

Defines an existing parameter as the argument for the provider operations on the specified class property.

**Remarks**

This method is for UNIX operating systems only. The **ClassName** must be previously defined in the Dynamic model window.

**Method CallDMNumProbe**

**Syntax**

```csharp
Object.CallDMNumProbe(ProbeKey As String, Arguments As String) As Double
```

**Parameters**

- **Probekey**
  
  A key that uniquely identifies the monitoring source and is always in the form *CollectionName.MonitorName*.

- **Arguments**
  
  A string that contains the monitor arguments separated by blanks.

**Returns**

The numeric value resulting from the monitor call.

**Description**

This method runs the monitoring source identified by **ProbeKey** and returns its output.
Remarks
The monitoring source activation must return within 60 seconds, otherwise an error generates.

Error codes
- TMWSERVICE_E_PROBE_WRONG_ARGS_NUM
- TMWSERVICE_E_PROBE_NOT_LOAD
- TMWSERVICE_E_PROBE_NOT_FOUND
- TMWSERVICE_E_NO_INTERP_SUPPORT
- TMWSERVICE_E_NO_DATA
- TMWSERVICE_E_IMPLIED_ERROR
- TMWSERVICE_E_ERRORVALUE_SCRIPT_ERROR
- S_OK

Method CallDMStrProbe

Syntax
Object.CallDMStrProbe(ProbeKey As String, Arguments As String) As String

Parameters
- Probekey
  A key that uniquely identifies the monitoring source and is always in the form CollectionName.MonitorName.

- Arguments
  A string that contains the monitor arguments separated by blanks.

Returns
The string value resulting from the monitor call.

Description
This method runs the monitoring source identified by ProbeKey and returns its output.

Remarks
The monitoring source activation must return within 60 seconds, otherwise an error generates.

Error codes
- TMWSERVICE_E_PROBE_WRONG_ARGS_NUM
- TMWSERVICE_E_PROBE_NOT_LOAD
- TMWSERVICE_E_PROBE_NOT_FOUND
- TMWSERVICE_E_NO_INTERP_SUPPORT
- TMWSERVICE_E_NO_DATA
- TMWSERVICE_E_IMPLIED_ERROR
- TMWSERVICE_E_ERRORVALUE_SCRIPT_ERROR
- S_OK

Example
Table 14. Basic Object Method dynamic model examples

Visual Basic example:
Table 14. Basic Object Method dynamic model examples (continued)

| Dim id, numOfInstances As Integer  
| Dim process As String  
| numOfInstances = Svc.GetNumOfInst("TMW_Process")  
| For idx = 0 To numOfInstances - 1  
| 'retrieve the numeric property, ID, for the given instance of  
| TMW_Process id = Svc.GetNumProperty("TMW_Process", idx, "ID")  
|  
| 'retrieve the string property, Process, for the given instance of  
| Next  

JavaScript example:

```javascript
var id, numOfInstances;
var process;
numOfInstances = Svc.GetNumOfInst("TMW_Process");
for(idx = 0; idx < numOfInstances; idx++){
  //retrieve the numeric property, ID, for the given instance of
  TMW_Process id = Svc.GetNumProperty("TMW_Process", idx, "ID");

  //retrieve the string property, Process, for the given instance of
}
```

Thresholds

The following methods help you to express functions for threshold configuration.

**Method GetThreshold**

**Syntax**

```plaintext
Object.GetThreshold(ThName As String) As Double
```

**Parameters**

- **ThName**
  
  The name of the threshold.

**Description**

Returns the value of the threshold named `ThName`.

**Error codes**

- S_OK
- TMWSERVICE_E_THRESHOLD_NAME_NOT_DEFINED

**Example**

Table 15. Basic Object Method threshold examples

| Visual Basic example:  
| '<<THRESHOLDS_INFO>>  
| Svc.DefineThreshold "NewThreshold1", 3.0  
| '<<\THRESHOLDS_INFO>>  
| ....  
| Dim value As Double  
| value = Svc.GetThreshold("NewThreshold1")  

JavaScript example:
Table 15. Basic Object Method threshold examples (continued)

```csharp
//<<THRESHOLDS_INFO>>
Svc.DefineThreshold("NewThreshold1", 3.0);
//<<\THRESHOLDS_INFO>>
....
var value;
value = Svc.GetThreshold("NewThreshold1");
```

## Parameters

The following methods help you to express functions for parameters configuration.

### Method GetNumParameterCount

**Syntax**

```csharp
Object.GetNumParameterCount(ParamName As String) As Long
```

**Parameters**

<table>
<thead>
<tr>
<th>ParamName</th>
</tr>
</thead>
<tbody>
<tr>
<td>The name of a NUMERIC parameter.</td>
</tr>
</tbody>
</table>

**Description**

Returns the number of values contained by the parameter named `ParamName`.

**Error codes**

- S_OK
- TMWSERVICE_E_PARAMETER_NOT_DEFINED

### Method GetStrParameterCount

This method retrieves the number of items of string, Boolean, and choice list types. For the string lists, the returned value is the number of elements. For the Boolean lists, the returned value is the number of elements set to true. For the choice lists, the returned value is 1.

**Syntax**

```csharp
Object.GetStrParameterCount(ParamName As String) As Long
```

**Parameters**

<table>
<thead>
<tr>
<th>ParamName</th>
</tr>
</thead>
<tbody>
<tr>
<td>The name of a STRING parameter.</td>
</tr>
</tbody>
</table>

**Description**

Returns the number of values contained by the parameter named `ParamName`.

**Error codes**

- S_OK
- TMWSERVICE_E_PARAMETER_NOT_DEFINED

### Method GetNumParameter

**Syntax**

```csharp
Object.GetNumParameter(ParamName As String, idx As Long) As Double
```

**Parameters**

<table>
<thead>
<tr>
<th>ParamName</th>
</tr>
</thead>
<tbody>
<tr>
<td>The name of a NUMERIC parameter.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>idx</th>
</tr>
</thead>
<tbody>
<tr>
<td>The index of the parameter value.</td>
</tr>
</tbody>
</table>
Remarks

idx counter goes from 0 to NumOfInst-1.

Description

Returns the NUMERIC value contained at the index idx of the parameter named ParamName.

Error codes

S_OK
TMWSERVICE_E_PARAMETER_NOT_DEFINED
TMWSERVICE_E_PARAMETER_BAD_INDEX

Method GetStrParameter

This method retrieves the values of string, Boolean, and choice list types. For the string lists, the returned values are the element value. For the Boolean lists, the returned values are the values of the elements set to true. For the choice lists, the returned value is the selected value.

Syntax

Object.GetStrParameter(ParamName As String, idx As Long) As String

Parameters

  ParamName
      The name of a STRING parameter.

  idx
      The index of the parameter value.

Description

Returns the STRING value contained at the index idx of the parameter named ParamName.

Remarks

idx counter goes from 0 to NumOfInst-1.

Error codes

S_OK
TMWSERVICE_E_PARAMETER_NOT_DEFINED
TMWSERVICE_E_PARAMETER_BAD_INDEX

Example

Table 16. Basic Object Method parameter examples

<table>
<thead>
<tr>
<th>Visual Basic example:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dim numVal As Double</td>
</tr>
<tr>
<td>Dim strVal As String</td>
</tr>
<tr>
<td>Dim paramCount As Long</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>paramCount = Svc.GetNumParameterCount(&quot;NumParam&quot;)</td>
</tr>
<tr>
<td>For i = 0 To paramCount - 1</td>
</tr>
<tr>
<td>numVal = Svc.GetNumParameter(&quot;NumParam&quot;, i))</td>
</tr>
<tr>
<td>Next</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>paramCount = Svc.GetStrParameterCount(&quot;StrParam&quot;)</td>
</tr>
<tr>
<td>For i = 0 To paramCount - 1</td>
</tr>
<tr>
<td>strVal = Svc.GetStrParameter(&quot;StrParam&quot;, i))</td>
</tr>
<tr>
<td>Next</td>
</tr>
</tbody>
</table>

| JavaScript example: |
Events

The following methods express functions for events configuration.

**Method SendEventEx**

**Syntax**

Object.SendEventEx(EventName As String, mapHndl As Integer)

**Parameters**

- **EventName**
  The name of the event to send.

- **mapHndl**
  The handle of a mapping table returned by a call to CreateMap that contains all the keys required to set the event attributes.

**Description**

This method sends the event named EventName and specifies its attributes.

**Remarks**

The event named EventName must be previously defined through a DefineEvent in the SetDefaultConfiguration Subroutine. The mapping table associated with the handle mapHndl must contain, as keys, all the attributes defined for the given event.

**Error codes**

- TMWSERVICE_E_MAP_KEY_NOT_FOUND
- TMWSERVICE_E_ANALYZER_EVENT_NOT_GOT
- TMWSERVICE_E邵AWN_EVENT_FAILED
- TMWSERVICE_E_EVENT_NOT_DECLARED
- TMWSERVICE_E_EVENT_PROP_NOT_FOUND
- TMWSERVICE_E_EVENT_INDICATION_FAILED
- S_OK

**Example**

**Table 17. Basic Object Method event examples**

Visual Basic example:
Table 17. Basic Object Method event examples (continued)

```plaintext
'ventry definition section
'<<EVENTS_INFO>>
Svc.DefineEvent "MyEvent", "NumAttr", "StrAttr"
'<<EVENTS_INFO>>

.....

Dim hPropTable As Integer
hPropTable = Svc.CreateMap()
Svc.SetMapNumElement(hPropTable, "NumAttr", 1)
Svc.SetMapStrElement(hPropTable, "StrAttr", "stringVal")
Svc.SendEventEx("MyEvent, hPropTable)
```

**JavaScript example:**

```javascript
// Event definition section
//<<EVENTS_INFO>>
Svc.DefineEvent("MyEvent", "NumAttr", "StrAttr");
//<<EVENTS_INFO>>

.....

var hPropTable;
hPropTable = Svc.CreateMap();
Svc.SetMapNumElement(hPropTable, "NumAttr", 1);
Svc.SetMapStrElement(hPropTable, "StrAttr", "stringVal");
Svc.SendEventEx("MyEvent, hPropTable");
```

**Logging**

The following methods help you to express functions for data logging.

**Method LogInstEx**

**Syntax**

```plaintext
Object.LogInstEx(contextName As String,
Resource As String, mapHndl As Integer)
```

**Parameters**

- **context**
  The logging context.

- **pResource**
  The resource to which the attributes refer.

- **mapHndl**
  The handle of a mapping table returned by a call to CreateMap that contains all the keys required to set the logging attributes.

**Description**

This method logs the attributes of the resource named Resource in the given context.

**Remarks**

The context and the resource must be previously defined through a
Define LogInst in the SetDefaultConfiguration Subroutine. The mapping
table associated with the handle mapHndl must contain as keys all the
attributes defined for the given logging context.

Error codes
- TMWSERVICE_E_MAP_KEY_NOT_FOUND
- TMWSERVICE_E_ANALYZER_EVENT_NOT_GOT
- TMWSERVICE_E SPAWN EVENT FAILED
- TMWSERVICE_E_EVENT NOT DECLARED
- TMWSERVICE_E_EVENT PROP NOT FOUND
- TMWSERVICE_E_EVENT INDICATION FAILED
- S_OK

Utilities
The following methods help you to express functions for tracing activity.

Method Trace
Syntax
```
Object.Trace(LogLevel As Integer, Message As String)
```

Parameters
- LogLevel
  - The log level of the message, 0...3. (3 is the most verbose
    trace level.)
- Message
  - The message to trace.

Description
 Writes the message parameter to the trace log if the LogLevel is greater
 than the IBM Tivoli Monitoring trace level.

Remarks
 Call this method for debugging. To change the trace level from the IBM
 Tivoli Monitoring server, type the following command: wdmtrceng. For
 more information, see the IBM Tivoli Monitoring User’s Guide.

Error code
- S_OK

Method GetInterp
Syntax
```
Object.GetInterp() As String
```

Description
 Returns the interp of the workstation on which the resource model runs.

Error codes
- w32-ix86 on Windows NT and Windows 2000 operating systems; aix4-r1,
  hpux10, linux-ix86, linux-s390, solaris2, linux-ppc on UNIX operating systems.

Method Shell
Syntax
```
Object.Shell(shell As String) As String
```

Parameters
- shell  - The command line to launch the new process.
Returns
The first line of the standard output of the launched process that does not start with the # character.

Description
The Shell method runs a new program on the monitored host and retrieves its output.

Remarks
It is possible to launch customized scripts that are available on the endpoint by issuing, for example,
perl myperl.pl

The launched process must return within 60 seconds, otherwise an error generates.

Error codes
TMWSERVICE_E_NO_INTERP_SUPPORT
TMWSERVICE_E_NO_DATA
TMWSERVICE_E_IMPLIED_ERROR
TMWSERVICE_E_ERRORVALUE_SCRIPT_ERROR
S_OK

Method GetShellStdOut

Syntax
Object.GetShellStdOut() As String

Returns
The standard output of the process spawned by the last call to the "Shell" method.

Description
The GetShellStdOut method is used to get the full standard output of a spawned process. If the standard output is empty, this method returns an empty string.

Remarks
The Shell method returns only the first line of the standard output that does not start with the # character, while this method returns the full standard output.

Error codes
S_OK

Method GetShellStdErr

Syntax
Object.GetShellStdErr() As String

Returns
The standard error of the process spawned by the last call to the "Shell" method.

Description
The GetShellStdOut method gets the full standard error of a spawned process. If the standard error is empty, this method returns an empty string.

Error codes
S_OK
Method GetShellRetCode

Syntax

Object.GetShellRetCode() As Integer

Returns

The exit code of the process spawned by the last call to the “Shell” method.

Description

The GetShellRetCode method gets the exit code of a process. If the process succeeds, the exit code is usually zero.

Error codes

S_OK

Example

Table 18. Basic Object Method utility examples

<table>
<thead>
<tr>
<th>Visual Basic example:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dim interp As String</td>
</tr>
<tr>
<td>interp = Svc.GetInterp()</td>
</tr>
<tr>
<td>Svc.Trace(3, &quot;interp: &quot; + interp)</td>
</tr>
<tr>
<td>Dim out As String</td>
</tr>
<tr>
<td>out = Svc.Shell(&quot;perl myScript.pl&quot;)</td>
</tr>
<tr>
<td>Svc.Trace(3, &quot;first line of output: &quot; + out)</td>
</tr>
<tr>
<td>out = Svc.GetShellStdOut()</td>
</tr>
<tr>
<td>Svc.Trace(3, &quot;std output: &quot; + out)</td>
</tr>
<tr>
<td>out = Svc.GetShellStdErr()</td>
</tr>
<tr>
<td>Svc.Trace(3, &quot;std err: &quot; + out)</td>
</tr>
<tr>
<td>Dim retCode As Integer</td>
</tr>
<tr>
<td>retCode = Svc.GetShellRetCode()</td>
</tr>
<tr>
<td>Svc.Trace(3, &quot;return code: &quot; + retCode)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>JavaScript example:</th>
</tr>
</thead>
<tbody>
<tr>
<td>var interp;</td>
</tr>
<tr>
<td>interp = Svc.GetInterp();</td>
</tr>
<tr>
<td>Svc.Trace(3, &quot;interp: &quot; + interp);</td>
</tr>
<tr>
<td>var out;</td>
</tr>
<tr>
<td>out = Svc.Shell(&quot;perl myScript.pl&quot;);</td>
</tr>
<tr>
<td>Svc.Trace(3, &quot;first line of output: &quot; + out);</td>
</tr>
<tr>
<td>out = Svc.GetShellStdOut();</td>
</tr>
<tr>
<td>Svc.Trace(3, &quot;std output: &quot; + out);</td>
</tr>
<tr>
<td>out = Svc.GetShellStdErr();</td>
</tr>
<tr>
<td>Svc.Trace(3, &quot;std err: &quot; + out);</td>
</tr>
<tr>
<td>var retCode;</td>
</tr>
<tr>
<td>retCode = Svc.GetShellRetCode();</td>
</tr>
<tr>
<td>Svc.Trace(3, &quot;return code: &quot; + retCode);</td>
</tr>
</tbody>
</table>
Mapping tables

**Method CreateMap**

**Syntax**

```
Object.CreateMap() As Integer
```

**Returns**

The handle of a new mapping table.

**Description**

This method creates a new mapping table and returns a handle to it.

**Remarks**

To avoid memory leaks, call the DestroyMap method when the mapping table is no longer required.

**Error code**

S_OK

**Method SetMapNumElement**

**Syntax**

```
Object.SetMapNumElement(hndl As Integer, key As String, val As Double)
```

**Parameters**

- **hndl** The handle of a mapping table returned by a call to CreateMap.
- **key** The key to store.
- **val** The numeric value to store.

**Description**

This method inserts the key-val pair in the mapping table associated with the given handle.

**Error codes**

- TMWSERVICE_E_MAP_HANDLE_NOT_FOUND
- S_OK

**Method SetMapStrElement**

**Syntax**

```
Object.SetMapStrElement(hndl As Integer, key As String, val As String)
```

**Parameters**

- **hndl** The handle of a mapping table returned by a call to CreateMap.
- **key** The key to store.
- **val** The string value to store.

**Description**

This method inserts the key-val pair in the mapping table associated with the given handle.

**Error codes**

- TMWSERVICE_E_MAP_HANDLE_NOT_FOUND
- S_OK
Method GetMapNumValue

Syntax

Object.GetMapNumValue (hndl As Integer, key As String) As Double

Parameters

hndl The handle of a mapping table returned by a call to CreateMap.
key The key to lookup.

Returns

The numeric value associated with the given key.

Description

This method retrieves the value associated with the key in the mapping table linked with the given handle.

Error codes

TMWSERVICE_E_MAP_HANDLE_NOT_FOUND
TMWSERVICE_E_MAP_KEY_NOT_FOUND
S_OK

Method GetMapStrValue

Syntax

Object.GetMapStrValue (hndl As Integer, key As String) As String

Parameters

hndl The handle of a mapping table returned by a call to CreateMap.
key The key to lookup.

Returns

The string value associated with the given key.

Description

This method retrieves the value associated with the key in the mapping table linked with the given handle.

Error codes

TMWSERVICE_E_MAP_HANDLE_NOT_FOUND
TMWSERVICE_E_MAP_KEY_NOT_FOUND
S_OK

Method RemoveMapElement

Syntax

Object.RemoveMapElement (hndl As Integer, key As String)

Parameters

hndl The handle of a mapping table returned by a call to CreateMap.
key The key to remove from the mapping table.

Description

This method removes the value associated with the key in the mapping table linked to the given handle.

Error codes

TMWSERVICE_E_MAP_HANDLE_NOT_FOUND
S_OK
Method RemoveMapAll

Syntax
Object.RemoveMapAll(hndl As Integer)

Parameters
- **hndl**: The handle of a mapping table returned by a call to CreateMap.

Description
This method removes all the elements contained in the mapping table associated with the handle hndl, but it does not destroy the mapping table.

Remarks
This method does not free all the resources used by a mapping table. If you want to make them available, call the DestroyMap method.

Error codes
- TMWSERVICE_E_MAP_HANDLE_NOT_FOUND
- S_OK

Method ExistsMapElement

Syntax
Object.ExistsMapElement(hndl As Integer, key As String)
As Boolean

Parameters
- **hndl**: The handle of a mapping table returned by a call to CreateMap.
- **key**: The key to check whether an element exists.

Returns
TRUE if the key exists in the given mapping table, or FALSE if it does not exist.

Description
This method checks whether the given key is contained in the mapping table associated with the given handle.

Error codes
- TMWSERVICE_E_MAP_HANDLE_NOT_FOUND
- S_OK

Method DestroyMap

Syntax
Object.DestroyMap(hndl As Integer)

Parameters
- **hndl**: The handle of a mapping table returned by a call to CreateMap.

Remarks
Call the DestroyMap to free the resources used by a mapping table.

Description
This method destroys the mapping table associated with the handle hndl.

Error codes
- TMWSERVICE_E_MAP_HANDLE_NOT_FOUND
- S_OK
Example

Table 19. Basic Object Method mapping examples

<table>
<thead>
<tr>
<th>Visual Basic example:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dim hPropTable As Integer</td>
</tr>
<tr>
<td>hPropTable = Svc.CreateMap()</td>
</tr>
<tr>
<td>Svc.SetMapNumElement(hPropTable, &quot;NumAttr&quot;, 1)</td>
</tr>
<tr>
<td>Svc.SetMapStrElement(hPropTable, &quot;StrAttr&quot;, &quot;stringVal&quot;)</td>
</tr>
<tr>
<td>Dim numAttr As Double</td>
</tr>
<tr>
<td>numAttr = Svc.GetMapNumValue(hPropTable, &quot;NumAttr&quot;)</td>
</tr>
<tr>
<td>Dim strAttr As String</td>
</tr>
<tr>
<td>strAttr = Svc.GetMapStrValue(hPropTable, &quot;StrAttr&quot;)</td>
</tr>
<tr>
<td>Svc.RemoveMapElement(hPropTable, &quot;NumAttr&quot;)</td>
</tr>
<tr>
<td>If Svc.ExistsMapElement(hPropTable, &quot;NumAttr&quot;) Then</td>
</tr>
<tr>
<td>'Error</td>
</tr>
<tr>
<td>End If</td>
</tr>
<tr>
<td>Svc.RemoveMapAll(hPropTable)</td>
</tr>
<tr>
<td>Svc.DestroyMap(hPropTable)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>JavaScript example:</th>
</tr>
</thead>
<tbody>
<tr>
<td>var hPropTable;</td>
</tr>
<tr>
<td>hPropTable = Svc.CreateMap();</td>
</tr>
<tr>
<td>Svc.SetMapNumElement(hPropTable, &quot;NumAttr&quot;, 1);</td>
</tr>
<tr>
<td>Svc.SetMapStrElement(hPropTable, &quot;StrAttr&quot;, &quot;stringVal&quot;);</td>
</tr>
<tr>
<td>var numAttr;</td>
</tr>
<tr>
<td>numAttr = Svc.GetMapNumValue(hPropTable, &quot;NumAttr&quot;);</td>
</tr>
<tr>
<td>var strAttr;</td>
</tr>
<tr>
<td>strAttr = Svc.GetMapStrValue(hPropTable, &quot;StrAttr&quot;);</td>
</tr>
<tr>
<td>Svc.RemoveMapElement(hPropTable, &quot;NumAttr&quot;);</td>
</tr>
</tbody>
</table>
| if( Svc.ExistsMapElement(hPropTable, "NumAttr") ){
| //Error 
| } |
| Svc.RemoveMapAll(hPropTable); |
| Svc.DestroyMap(hPropTable); |

Advanced Object Methods

This section contains a description of the advanced methods of the TMWSERVICE Object. Never call these methods directly, because the calls to these methods are always generated by IBM Tivoli Resource Model Builder.

General settings

The following methods help you to express functions for resource model configuration.
**Method SetModelName**

**Syntax**

```
Object.SetModelName(ModelName As String)
```

**Parameters**

- **ModelName**
  
  The name of the resource model.

**Description**

Sets the resource model name.

**Remarks**

Do not call this method outside the SetDefaultConfiguration(...) function.

**Error code**

S_OK

**Method SetCycleTime**

**Syntax**

```
Object.SetCycleTime(CycleTime As Double)
```

**Parameters**

- **CycleTime**
  
  The cycle time of the resource model.

**Description**

Sets the resource model cycle time.

**Remarks**

Do not call this method outside the SetDefaultConfiguration(...) function.

**Error code**

S_OK

**Example**

*Table 20. Advanced Object Method general settings examples*

<table>
<thead>
<tr>
<th>Visual Basic example:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Svc.SetModelName(&quot;ModelName&quot;)</td>
<td></td>
</tr>
<tr>
<td>Svc.SetCycleTime(30)</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>JavaScript example:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Svc.SetModelName(&quot;ModelName&quot;);</td>
<td></td>
</tr>
<tr>
<td>Svc.SetCycleTime(30);</td>
<td></td>
</tr>
</tbody>
</table>

**Dynamic Model**

The following methods help you to express functions for dynamic model configuration.

**Method DefineClass**

**Syntax**

```
Object.DefineClass(Source As String, ClassName As String,_,
RealClassName As String, WhereClause As String,_,
NumProps As String, StrProps As String, SortType As String,_,
SortField As String, Top as long, Every as long)
```

**Parameters**
Source
The data source name (It can be only "CIM").

ClassName
The name of the Dynamic Model class being defined.

RealClassName
The full path name of the CIM class to be associated with the Dynamic Model class being defined.

WhereClause
A where clause used to filter the instances.

NumProps
Comma-separated values with the names of class NUMERIC attributes.

StrProps
Comma-separated values with the names of class STRING attributes.

SortType
Used to sort the collected instances. It can be set to Ascending, Descending, or None.

SortField
A NUMERIC field used for sorting the collected instances.

Top
Used only when the instances are sorted, to specify the maximum number of instances to collect. Set to 0 to get all the instances.

Every
Specifies the number of cycle times that have to elapse between two successive data collections. The default is 1.

Description
Adds a new class to the resource model dynamic model.

Remarks
The ClassName must be previously defined in the Dynamic Model window.

Error codes
S_OK
TMWSERVICE_E_CLASS_NAME_NOT_VALID
TMWSERVICE_E_SOURCE_NOT_SUPPORTED

Method DefineClassAsync
Syntax
Object.DefineClassAsync(ClassName As String,_
RealClassName As String,_
EvType As String, NumProps As String, StrProps As String)

Parameters

ClassName
The name of the Dynamic Model class being defined.

RealClassName
The full path name of the CIM class to be associated with the Dynamic Model class being defined.
EvType
- Used to specify the kind of asynchronous collection. It can be set to InstanceCreation, InstanceModification, or InstanceDeletion.

NumProps
- Comma-separated values with the names of class NUMERIC attributes.

StrProps
- Comma-separated values with the names of class STRING attributes.

Description
- Adds a new class to the resource model dynamic model.

Error codes
- S_OK
- TMWSERVICE_E_CLASS_NAME_NOT_VALID

Method RemoveClass

Syntax
- Object.RemoveClass(ClassName As String)

Parameters
- ClassName
  - The name of the Dynamic Model class.

Description
- Removes the class named ClassName from the resource model dynamic model.

Error codes
- S_OK
- TMWSERVICE_E_PROPERTY_NOT_FOUND

Method CollectData

Syntax
- Object.CollectData()

Description
- Collects data for all the classes belonging to the resource model dynamic model.

Remarks
- This method is called from IBM Tivoli Monitoring engine before the VisitTree function is called. By default, it is called every cycle time, unless otherwise specified in the Every option of the Dynamic Model window. Do not call this method. If you invoke this method, a new collection of the data defined in the resource model starts. So if, for example, you are in a loop, you risk changing data during the analysis.

Error codes
- S_OK
- TMWSERVICE_E_COLLECTOR_UNABLE_TO_USE_PROV
- TMWSERVICE_E_COLLECTOR_CRITICAL_ERROR
- TMWSERVICE_E_COLLECTOR_PROV_NOT_READY
Method CollectClassData
Syntax
    Object.CollectClassData(ClassName As String)
Parameters
    ClassName
        The name of the Dynamic Model class.
Description
    Collects data only for the specified Dynamic Model class.
Remarks
    Do not call this method if you are analyzing the defined class. Invoking
    this method, a new collection of the data defined in the resource model
    starts. So if, for example, you are in a loop, you risk changing data during
    the analysis. The ClassName must be previously defined through in the
    Dynamic model window.
Error codes
    S_OK
    TMWSERVICE_E_COLLECTOR_UNABLE_TO_USE_PROV
    TMWSERVICE_E_COLLECTOR_CRITICAL_ERROR
    TMWSERVICE_E_COLLECTOR_PROV_NOT_READY

Method DefineDMNumProbe
Syntax
    Object.DefineDMNumProbe(key As String, numOfArgs As Long)
Parameters
    key
        A key that uniquely identifies the monitoring source,
        always in the form CollectionName.MonitorName.
    numOfArgs
        The number of arguments required by the monitor.
Description
    This method adds to the dynamic model a Tivoli Distributed Monitoring
    (Classic Edition) monitoring source that returns a numeric value.
Error code
    S_OK

Method DefineDMStrProbe
Syntax
    Object.DefineDMStrProbe(key As String, numOfArgs As Long)
Parameters
    key
        A key that uniquely identifies the monitoring source,
        always in the form CollectionName.MonitorName.
    numOfArgs
        The number of arguments required by the monitor.
Description
    This method adds to the dynamic model a Tivoli Distributed Monitoring
    (Classic Edition) monitoring source that returns a string value.
Error code
    S_OK
**Example**

*Table 21. Advanced Object Method dynamic model examples*

### Visual Basic example:

```
    "WHERE Name = "calc.exe"", "", "Handle,Name", "None", "", 0, 1
    "InstanceDeletion", "", "Handle,Name"
```

```
Svc.DefineDMStrProbe ("w2k_Process.NtServices", 1)
Svc.DefineDMNumProbe ("w2k_Process.HandleCnt", 1)
```

### JavaScript example:

```
    "WHERE Name = "calc.exe"", "", "Handle,Name", "None", "", 0, 1);
    "InstanceDeletion", "", "Handle,Name");
```

```
Svc.DefineDMStrProbe ("w2k_Process.NtServices", 1);
Svc.DefineDMNumProbe ("w2k_Process.HandleCnt", 1);
```

**Thresholds**

The following methods help you to express functions for threshold configuration.

### Method DefineThreshold

**Syntax**

```
Object.DefineThreshold(ThName As String, value As Double)
```

**Parameters**

- **ThName**  
  The name of a new threshold.

- **value**  
  The threshold default value.

**Description**

Defines a new threshold.

**Remarks**

This method is called inside the

```
<<THRESHOLDS_INFO>>...<<\THRESHOLDS_INFO>>
```

tags after you have defined a threshold in the Thresholds window.

**Error codes**

- S_OK
- TMWSERVICE_E_THRESHOLD_NAME_NOT_VALID

**Example**

*Table 22. Advanced Object Method threshold examples*

### Visual Basic example:

```
Svc.DefineThreshold "NewThreshold1", 1.0
```

### JavaScript example:

```
Svc.DefineThreshold("NewThreshold1", 1.0);
```
Parameters

The following methods help you to express functions for parameters configuration.

**Method DefineStrParameter**

**Syntax**

```
Object.DefineStrParameter(ParamName As String, values As String)
```

**Parameters**

- **ParamName**
  The name of a new STRING parameter.
- **values**
  Comma-separated values of the parameter default values.

**Description**

Defines a new STRING parameter.

**Remarks**

This method is called inside the `<<PARAMETERS_INFO>>...<<\PARAMETERS_INFO>>` tags after you have defined a parameter in the Parameters window.

**Error codes**

- S_OK
- TMWSERVICE_E_PARAMETER_NAME_NOT_VALID

**Method DefineNumParameter**

**Syntax**

```
Object.DefineNumParameter(ParamName As String, values As String)
```

**Parameters**

- **ParamName**
  The name of a new NUMERIC parameter
- **values**
  Comma-separated values of the parameter default values.

**Description**

Defines a new NUMERIC parameter.

**Remarks**

This method is called inside the `<<PARAMETERS_INFO>>...<<\PARAMETERS_INFO>>` tags after you have defined a parameter in the Parameters window.

**Error codes**

- S_OK
- TMWSERVICE_E_PARAMETER_NAME_NOT_VALID

**Method AddStrParameter**

**Syntax**

```
Object.AddStrParameter(ParamName As String, Value As String)
```

**Parameters**

- **ParamName**
  The name of a STRING parameter.
- **Value**
  A STRING value.

**Description**

Adds the given Value at the end of the parameter named `ParamName`. 
Remarks
This method is called after you have defined a new parameter in the Parameters window.

Error codes
S_OK
TMWSERVICE_E_PARAMETER_NOT_DEFINED
TMWSERVICE_E_PARAMETER_BAD_INDEX

Method AddNumParameter
Syntax
Object.AddNumParameter(ParamName As String, Value As Double)

Parameters

  ParamName
  The name of a NUMERIC parameter.

  Value
  A NUMERIC value.

Description
Adds the given Value at the end of the parameter named ParamName.

Remarks
This method is called after you have defined a new parameter in the Parameters window.

Error codes
S_OK
TMWSERVICE_E_PARAMETER_NOT_DEFINED
TMWSERVICE_E_PARAMETER_BAD_INDEX

Method RemoveStrParameter
Syntax
Object.RemoveStrParameter(ParamName As String, idx As Long)

Parameters

  ParamName
  The name of a STRING parameter.

  idx
  The index of a parameter value.

Description
Removes the value contained at the given index and shifts down all the values after the removed one.

Remarks
idx counter goes from 0 to NumOfInst -1.

Error codes
S_OK
TMWSERVICE_E_PARAMETER_NOT_DEFINED
TMWSERVICE_E_PARAMETER_BAD_INDEX

Method RemoveNumParameter
Syntax
Object.RemoveNumParameter(ParamName As String, idx As Long)

Parameters

  ParamName
  The name of a NUMERIC parameter.
idx  The index of a parameter value.

Description
Removes the value contained at the given index and shifts down all the values after the removed one.

Remarks
idx counter goes from 0 to NumOfInst -1.

Error codes
S_OK
TMWSERVICE_E_PARAMETER_NOT_DEFINED
TMWSERVICE_E_PARAMETER_BAD_INDEX

Method ReplaceStrParameter

Syntax
Object.ReplaceStrParameter(ParamName As String,_
idx As Long, Value As String)

Parameters

ParamName
The name of a STRING parameter.

idx  The index of a parameter value.

Value  A STRING value.

Description
Replaces the value contained at the given index with the new value.

Remarks
idx counter goes from 0 to NumOfInst -1.

Error codes
S_OK
TMWSERVICE_E_PARAMETER_NOT_DEFINED
TMWSERVICE_E_PARAMETER_BAD_INDEX

Method ReplaceNumParameter

Syntax
Object.ReplaceNumParameter(ParamName As String,_
idx As Long, Value As Double)

Parameters

ParamName
The name of a NUMERIC parameter.

idx  The index of a parameter value.

Value  A NUMERIC value.

Description
Replaces the value contained at the given index with the new value.

Remarks
idx counter goes from 0 to NumOfInst -1.

Error codes
S_OK
TMWSERVICE_E_PARAMETER_NOT_DEFINED
TMWSERVICE_E_PARAMETER_BAD_INDEX
Example

Table 23. Advanced Object Method parameter examples

<table>
<thead>
<tr>
<th>Visual Basic example:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Svc.DefineStrParameter(&quot;strParam&quot;, &quot;item1,item2&quot;) 'values are item1 and item2</td>
</tr>
<tr>
<td>Svc.AddStrParameter(&quot;strParam&quot;, &quot;item3&quot;) 'values are item1, item2, item3</td>
</tr>
<tr>
<td>Svc.ReplaceStrParameter(&quot;strParam&quot;, 2, &quot;toBeDeleted&quot;) 'values are item1, item2, toBeDeleted</td>
</tr>
<tr>
<td>Svc.RemoveStrParameter(&quot;strParam&quot;, 2) 'values are item1 and item2</td>
</tr>
<tr>
<td>Svc.DefineNumParameter(&quot;numParam&quot;, &quot;1,2&quot;) 'values are 1,2</td>
</tr>
<tr>
<td>Svc.AddNumParameter(&quot;numParam&quot;, &quot;numParam&quot;, 3) 'values are now 1,2,3</td>
</tr>
<tr>
<td>Svc.ReplaceNumParameter(&quot;numParam&quot;, 2, 2) 'values are now 1,2,2</td>
</tr>
<tr>
<td>Svc.RemoveNumParameter(&quot;numParam&quot;, 1) 'values are now 1,2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>JavaScript example:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Svc.DefineStrParameter(&quot;strParam&quot;, &quot;item1,item2&quot;); //values are item1 and item2</td>
</tr>
<tr>
<td>Svc.AddStrParameter(&quot;strParam&quot;, &quot;item3&quot;); //values are item1, item2, item3</td>
</tr>
<tr>
<td>Svc.ReplaceStrParameter(&quot;strParam&quot;, 2, &quot;toBeDeleted&quot;); //values are item1, item2, toBeDeleted</td>
</tr>
<tr>
<td>Svc.RemoveStrParameter(&quot;strParam&quot;, 2); //values are item1 and item2</td>
</tr>
<tr>
<td>Svc.DefineNumParameter(&quot;numParam&quot;, &quot;1,2&quot;); //values are 1,2</td>
</tr>
<tr>
<td>Svc.AddNumParameter(&quot;numParam&quot;, &quot;numParam&quot;, 3); //values are now 1,2,3</td>
</tr>
<tr>
<td>Svc.ReplaceNumParameter(&quot;numParam&quot;, 2, 2); //values are now 1,2,2</td>
</tr>
<tr>
<td>Svc.RemoveNumParameter(&quot;numParam&quot;, 1); //values are now 1,2</td>
</tr>
</tbody>
</table>

Events

The following methods help you to express functions for events configuration.

Method DefineEvent

Syntax

```
Object.DefineEvent(EventName As String, _
NumAttrs As String, StrAttrs As String)
```

Parameters

**EventName**

The name of a new event.

**NumAttrs**

Comma-separated values with the names of the NUMERIC attributes of the event.

**StrAttrs**

Comma-separated values with the names of the STRING attributes of the event.

Description

Defines a new Event.

Remarks

This method is called inside the

`<<EVENTS_INFO>>...<<\EVENTS_INFO>>` tags after you have defined an event in the Events window.

Error codes

S_OK

TMWSERVICE_E_EVENT_REDEFINITION
Example

Table 24. Advanced Object Method event examples

<table>
<thead>
<tr>
<th>Visual Basic example:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Svc.DefineEvent &quot;MyEvent&quot;, &quot;numAttr1,numAttr2&quot;, &quot;strAttr1,strAttr2&quot;</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>JavaScript example:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Svc.DefineEvent(&quot;MyEvent&quot;, &quot;numAttr1,numAttr2&quot;, &quot;strAttr1,strAttr2&quot;);</td>
</tr>
</tbody>
</table>

Logging

The following methods help you to express functions for data logging.

**Method DefineLogInst**

**Syntax**

```
Object.DefineLogInst(Context As String,_
Resource As String,_
Keys As String, NumAttrs As String, StrAttrs As String)
```

**Parameters**

- **Context**
  
  The name of a new logging context.

- **Resource**
  
  The resource name.

- **Keys**
  
  The resource key properties.

- **NumAttrs**
  
  Comma-separated values with the names of resource NUMERIC attributes.

- **StrAttrs**
  
  Comma-separated values with the names of resource STRING attributes.

**Description**

Defines a logging context.

**Remarks**

This method is called inside the

`<<LOGGING_INFO>>...<<\LOGGING_INFO>>` tags after you have defined a context in the Logging window.

**Error codes**

- **S_OK**
- **TMWSERVICE_E_INST_EVENT_REDEFINITION**

Generic functions

The following methods help you to express functions for generic activities.

**Method EndVisit**

**Syntax**

```
Object.EndVisit()
```

**Description**

Is called at the end of each cycle, to perform clean up operations.

**Error code**

- **S_OK**
**Method Dispose**

**Syntax**

```csharp
Object.Dispose()
```

**Description**

Is called at the end of the monitoring loop, to perform clean up operations.

**Error code**

S_OK

**Method GetProfileName**

**Syntax**

```csharp
Object.GetProfileName() As String
```

**Returns**

The profile name.

**Description**

Returns the name of the profile to which the resource model belongs.

**Error code**

S_OK

**Method GetGlobalCounter**

**Syntax**

```csharp
Object.GetGlobalCounter() As String
```

**Returns**

The value of the current monitoring loop.

**Description**

Returns the value of the current monitoring loop. When a resource model starts its monitoring this value is set to 0, and it is increased by one after each successful call to the VisitTree function.

**Error code**

S_OK

**Deprecated methods**

Some methods used with Tivoli Distributed Monitoring (Advanced Edition), Version 4.1 are replaced by new methods.

The `SendEvent` method is deprecated. To send an event, use the `SendEventEx` method.

The `LogInst` method is deprecated. To log the attributes of a resource, use the `LogInstEx` method, instead.

**Exceptions**

This section contains a table of the error codes for the methods in the service object method library. You can also get the error codes from the engine’s log (c:\default.log). You can also check the error code by catching an exception that is thrown from the method when an error occurs, and checking the error code in the exception. For example,

**VBA:**

```vba
On Error GoTo CATCH_ERROR
TMWService.CollectData
```

---

GoTo NORMAL_FLOW

CATCH_ERROR:
MsgBox ("Error occurred in CollectData." & "The hex error code is: " & Hex(Err.Number))
TMWService.Trace(0, "Error occurred in CollectData." & "The hex error code is: " & Hex(Err.Number))
TMWService.Dispose()
Exit Sub

JavaScript:

try {
    Svc.CollectData();
} catch(err) {
    var nmb;
    if (err.number > 0)
        nmb = err.number.toString(16);
    else
        nmb = (err.number + 0x100000000).toString(16);

    WScript.Echo("An error occurred in Svc.CollectData, code: "+nmb);
    Svc.Trace0, "Resource Model Builder: An error occurred during Svc.CollectData, code: "+nmb);
    Svc.Dispose();
    return;
}

This table contains a list of exceptions you might encounter, with the corresponding message.

<table>
<thead>
<tr>
<th>Exceptions</th>
<th>Description</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x80041001</td>
<td>TMWSERVICE_E_FAILED</td>
<td>Generic Error.</td>
</tr>
<tr>
<td>0x80041002</td>
<td>TMWSERVICE_E_UNABLE_TO_CONNECT_WMI</td>
<td>It is not possible to get access to WMI Service.</td>
</tr>
<tr>
<td>0x80041003</td>
<td>TMWSERVICE_E_ANALYZER_EVENT_NOT_FOUND</td>
<td>The TMW_AnalyzerEvent cannot be found. The MOF file compilation might have failed.</td>
</tr>
<tr>
<td>0x80041004</td>
<td>TMWSERVICE_E_CANNOT_GET_SINK</td>
<td>It is not possible to get the Event Sink.</td>
</tr>
<tr>
<td>0x80041005</td>
<td>TMWSERVICE_E_CANNOT_RESTART_COLLECTOR</td>
<td>The data collector cannot be restarted.</td>
</tr>
<tr>
<td>0x80041006</td>
<td>TMWSERVICE_E_ANALYZER_EVENT_NOT_GOT</td>
<td>The definition of the TMW_AnalyzerEvent was not found in the CIM Repository.</td>
</tr>
<tr>
<td>0x80041007</td>
<td>TMWSERVICE_E_SPAWN_EVENT_FAILED</td>
<td>It is not possible to spawn a new instance of the event.</td>
</tr>
<tr>
<td>0x80041008</td>
<td>TMWSERVICE_E_EVENT_NOT_DECLARED</td>
<td>The event or the logging context has not been defined.</td>
</tr>
<tr>
<td>0x80041009</td>
<td>TMWSERVICE_E_EVENT_INDICATION_FAILED</td>
<td>WMI errors occurred while the TMW_AnalyzerEvent, or the TMW_InstEvent was being sent.</td>
</tr>
<tr>
<td>Exceptions</td>
<td>Description</td>
<td>Remarks</td>
</tr>
<tr>
<td>---------------</td>
<td>--------------------------------------------------</td>
<td>----------------------------------------------</td>
</tr>
<tr>
<td>0x8004100a</td>
<td>TMWSERVICE_E_EVENT_WRONG_NUM_OF_ARGS</td>
<td>Wrong number of arguments.</td>
</tr>
<tr>
<td>0x8004100b</td>
<td>TMWSERVICE_E_CIM_CLASS_NOT_FOUND</td>
<td>The class has not been defined.</td>
</tr>
<tr>
<td>0x8004100c</td>
<td>TMWSERVICE_E_THRESHOLD_NAME_NOT_VALID</td>
<td>The name for the threshold is not allowed.</td>
</tr>
<tr>
<td>0x8004100d</td>
<td>TMWSERVICE_E_THRESHOLD_NAME_NOT_DEFINED</td>
<td>The threshold has not been defined.</td>
</tr>
<tr>
<td>0x8004100e</td>
<td>TMWSERVICE_E_EVENT_REDEFINITION</td>
<td>The event was already defined.</td>
</tr>
<tr>
<td>0x8004100f</td>
<td>TMWSERVICE_E_PARAMETER_NOT_DEFINED</td>
<td>The parameter has not been defined.</td>
</tr>
<tr>
<td>0x80041010</td>
<td>TMWSERVICE_E_PARAMETER_NAME_NOT_VALID</td>
<td>The name for the threshold is not allowed.</td>
</tr>
<tr>
<td>0x80041011</td>
<td>TMWSERVICE_E_CLASS_NAME_NOT_VALID</td>
<td>The name for data class is not allowed.</td>
</tr>
<tr>
<td>0x80041012</td>
<td>TMWSERVICE_E_CIM_PROPERTY_NOT_FOUND</td>
<td>The property does not belong to the given class.</td>
</tr>
<tr>
<td>0x80041013</td>
<td>TMWSERVICE_E_PARAMETER_BAD_INDEX</td>
<td>The parameter index is out of the bound.</td>
</tr>
<tr>
<td>0x80041014</td>
<td>TMWSERVICE_E_COLLECTOR_UNABLE_TO_USE_PROV</td>
<td>The provider is not able to collect data.</td>
</tr>
<tr>
<td>0x80041015</td>
<td>TMWSERVICE_E_COLLECTOR_CRITICAL_ERROR</td>
<td>It is not possible to collect data.</td>
</tr>
<tr>
<td>0x80041016</td>
<td>TMWSERVICE_E_COLLECTOR_PROV_NOT_READY</td>
<td>The provider is not able to collect data; it may be busy.</td>
</tr>
<tr>
<td>0x80041017</td>
<td>TMWSERVICE_E_DEPENDENCY_NOT_FOUND</td>
<td>It is not possible to find the dependency file.</td>
</tr>
<tr>
<td>0x80041018</td>
<td>TMWSERVICE_E_CANNOT_SAVE_WORKSPACE</td>
<td>It is not possible to save the workspace.</td>
</tr>
<tr>
<td>0x80041019</td>
<td>TMWSERVICE_E_CANNOT_LOAD_WORKSPACE</td>
<td>It is not possible to load the workspace.</td>
</tr>
<tr>
<td>0x8004101a</td>
<td>TMWSERVICE_E_INST_EVENT_NOT_GOT</td>
<td>The definition of the TMW_InstEvent was not found in the CIM Repository.</td>
</tr>
<tr>
<td>0x8004101b</td>
<td>TMWSERVICE_E_INST_EVENT_NOT_FOUND</td>
<td>The TMW_InstEvent cannot be found. The MOF file compilation might have failed.</td>
</tr>
<tr>
<td>0x8004101e</td>
<td>TMWSERVICE_E_INST_EVENT_REDEFINITION</td>
<td>The logging context was already defined.</td>
</tr>
<tr>
<td>0x80041020</td>
<td>TMWSERVICE_E_NOT_VALID_WORKSPACE_FILE</td>
<td>It is not possible to load the workspace because of a bad workspace file format.</td>
</tr>
<tr>
<td>0x80041021</td>
<td>TMWSERVICE_E_ACTION_RULE_NOT_FOUND</td>
<td>The action rule was not found.</td>
</tr>
<tr>
<td>0x80041022</td>
<td>TMWSERVICE_E_ACTION_NOT_FOUND</td>
<td>The action was not found.</td>
</tr>
<tr>
<td>0x80041023</td>
<td>TMWSERVICE_E_SOURCE_NOT_SUPPORTED</td>
<td>The data source is not allowed.</td>
</tr>
</tbody>
</table>
Table 25. Method exception list (continued)

<table>
<thead>
<tr>
<th>Exceptions</th>
<th>Description</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x80041024</td>
<td>TMWSERVICE_E_PROBE_NOT_FOUND</td>
<td>The Tivoli Distributed Monitoring (Classic Edition) monitoring source has not been defined.</td>
</tr>
<tr>
<td>0x80041025</td>
<td>TMWSERVICE_E_PROBE_NOT_LOAD</td>
<td>The Tivoli Distributed Monitoring (Classic Edition) monitoring source implementation is not available.</td>
</tr>
<tr>
<td>0x80041026</td>
<td>TMWSERVICE_E_NO_INTERP_SUPPORT</td>
<td>The Tivoli Distributed Monitoring (Classic Edition) monitoring source implementation is not available for the given operating system.</td>
</tr>
<tr>
<td>0x80041027</td>
<td>TMWSERVICE_E_NO_DATA</td>
<td>The Tivoli Distributed Monitoring (Classic Edition) monitoring source implementation has returned an empty output.</td>
</tr>
<tr>
<td>0x80041028</td>
<td>TMWSERVICE_E_IMPLIED_ERROR</td>
<td>The Tivoli Distributed Monitoring (Classic Edition) monitoring source implementation is getting an error due to bad arguments.</td>
</tr>
<tr>
<td>0x80041029</td>
<td>TMWSERVICE_E_ERRORVALUE_SCRIPT_ERROR</td>
<td>The Tivoli Distributed Monitoring (Classic Edition) monitoring source implementation contains an internal script error.</td>
</tr>
<tr>
<td>0x8004102a</td>
<td>TMWSERVICE_E_PROBE_WRONG_ARGS_NUM</td>
<td>A wrong number of arguments has been specified in a call to a Tivoli Distributed Monitoring (Classic Edition) monitoring source.</td>
</tr>
<tr>
<td>0x8004102b</td>
<td>TMWSERVICE_E_MAP_HANDLE_NOT_FOUND</td>
<td>The handle of the mapping table is invalid.</td>
</tr>
<tr>
<td>0x8004102c</td>
<td>TMWSERVICE_E_MAP_KEY_NOT_FOUND</td>
<td>The given key is not contained in the mapping table associated with the given handle.</td>
</tr>
<tr>
<td>0x8004102d</td>
<td>TMWSERVICE_E_EVENT_PROP_NOT_FOUND</td>
<td>A property required for sending an event or for logging a call is not contained in the mapping table associated with the given handle.</td>
</tr>
<tr>
<td>0x8004102e</td>
<td>TMWSERVICE_E_CONTEX_PROP_NOT_FOUND</td>
<td>The context property is not available.</td>
</tr>
<tr>
<td>0x8004102f</td>
<td>TMWSERVICE_E_PARAMETER_IS_EMPTY</td>
<td>The parameter does not contain any value.</td>
</tr>
<tr>
<td>0x80041030</td>
<td>TMWSERVICE_E_NOT_WORKSPACE_TOO_NEW</td>
<td>The resource model workspace is too new for the current version of the IBM Tivoli Resource Model Builder.</td>
</tr>
</tbody>
</table>
Appendix C. Instrumentation Library Type Interface

This appendix describes the Instrumentation Library Type (ILT) interface and provides indications to use it. It is composed of three sections: the first section documents the public operations of the ILT interface, the second section documents the support classes, the third and last section provides guidelines and samples for creating a UNIX provider.

ILT Public Operations

This section describes the public operations available with the ILT interface.

Note: Some public operations are not supported by IBM Tivoli Monitoring. These operations are identified as not supported.

**enumerateInstances**

Supported:

YES

Syntax

```java
public java.util.Enumeration enumerateInstances (M12ClassPath classPath,
java.lang.String mappingString,
ParameterSet parms)
throws M12Exception
```

Parameters

- **classPath**
  The M12ClassPath identifying the class whose instances have to be enumerated.

- **mappingString**
  Any string that has been specified in the M12_Instrumentation qualifier for the ENUM operation type for this class.

- **parms**
  A ParameterSet object filled by the client with parameters for ILT.

Description

Returns all M12ObjectIdentity objects that identify all the instances belonging to the class specified in the classPath.

Returns

Enumeration of instances identity (M12ObjectIdentity).

Exceptions Thrown

M12Exception

**getProperty**

Supported:

YES

Syntax

```java
public java.lang.String getProperty (M12ObjectIdentity targetInstance,
java.lang.String propertyName,
java.lang.String mappingString,
ParameterSet parms)
throws M12Exception
```


Description
Gets the value (in String format) of the specified property for the identified object.

Returns
String - the value for property propertyName. Property values have to be CIM standard types and ILT converts them to string format according to the CIM standards.

Exceptions Thrown
M12Exception

getMultipleProperties

Supported:
YES

Syntax
public M12PropertySet getMultipleProperties (M12ObjectIdentity targetInstance,
java.util.Vector propertyList,
java.lang.String mappingString,
ParameterSet parms)
throws M12Exception

Parameters

targetInstance
M12ObjectIdentity that identifies the instance of the resource to be accessed.

propertyList
The list of properties whose value is required.

mappingString
Any string that has been specified in the M12_Instrumentation qualifier for the GET operation type for the class which the specified instance belongs to.

parms A ParameterSet object filled by the client with parameters associated with the class which the specified instance belongs to.

Description
Gets the value (in String format) of the specified properties for the identified object.

Returns
M12PropertySet - the values of the requested properties.

Exceptions Thrown
M12Exception

setProperty

Supported:
NO
Syntax
public java.lang.String setProperty (M12ObjectIdentity targetInstance,
java.lang.String propertyName,
java.lang.String propertyValue,
java.lang.String mappingString,
ParameterSet parms)
throws M12Exception

Parameters

targetInstance
M12ObjectIdentity that identifies the instance of the resource to be accessed.

propertyName
The property whose value is to be set.

propertyValue
The property value to be set.

mappingString
Any string that has been specified in the M12_Instrumentation qualifier for the SET™ operation type for this property.

parms
A ParameterSet object filled by the client with parameters associated with this property.

Description
Sets the value (in String format) of the specified property for the identified object.

Returns
String - the new value of the specified property.

Exceptions Thrown
M12Exception

invokeMethod

Supported:
YES

Syntax
public java.lang.String invokeMethod (M12ClassPath classPath,
java.lang.String methodName,
java.lang.String mappingString,
ParameterSet parms,
ParameterSet inParms,
ParameterSet outParms,
throws M12Exception

Parameters

classPath
The M12ClassPath that identifies the class whose method has to be called.

methodName
The name of the method to be called.

mappingString
Any string that has been specified in the M12_Instrumentation qualifier for the INVOKE operation type for this method.
**invokeMethod**

**Supported:**
YES

**Syntax**
```java
public java.lang.String invokeMethod (M12ObjectIdentity targetInstance,
java.lang.String methodName,
java.lang.String mappingString,
ParameterSet parms,
ParameterSet inParms,
ParameterSet outParms,
throws M12Exception)
```

**Parameters**

**targetInstance**
M12ObjectIdentity that identifies the instance whose method has to be called.

**methodName**
The name of the method to be called.

**mappingString**
Any string that has been specified in the M12_Instrumentation qualifier for the INVOKE operation type for this method.

**parms**
A ParameterSet object filled by the client with parameters for this method.

**inParms**
A ParameterSet object filled by the client with parameters to be passed to the method.

**outParms**
A ParameterSet object created by the client and filled by the method with output results.

**Description**
Invokes the specified method on the identified CIM instance.
Returns
String - the result of the method. Result values have to be CIM standard types and ILT converts them to string format according to the CIM standards.

Exceptions Thrown
M12Exception

create

Supported: NO

Syntax
public void create (M12ObjectIdentity targetInstance,
java.lang.String mappingString,
ParameterSet parms)
throws M12Exception

Parameters

  targetInstance
  M12ObjectIdentity that identifies the instance of the resource to be created.

  mappingString
  Any string that has been specified in the M12_Instrumentation qualifier for the CREATE operation type for the class whose instance to be created will belong to.

  parms A ParameterSet object filled by the client with parameters for the ILT.

Description
Creates an instance of the resource that is identified by the specified targetInstance.

Returns
void

Exceptions Thrown
M12Exception

destroy

Supported: NO

Syntax
public void destroy (M12ObjectIdentity targetInstance,
java.lang.String mappingString,
ParameterSet parms)
throws M12Exception

Parameters

  targetInstance
  M12ObjectIdentity that identifies the instance of the resource to be deleted.
mappingString
Any string that has been specified in the M12_Instrumentation qualifier for the DESTROY operation type for the class whose instance to be deleted belongs to.

parms A ParameterSet object filled by the client with parameters for the ILT.

Description
Deletes an instance of the resource identified by the specified targetInstance.

Returns
void

Exceptions Thrown
M12Exception

ILT Support Classes

This section described the set of classes supported by the ILT interface.

M12ClassPath

Syntax Detail:
public M12ClassPath (java.lang.String className)
public M12ClassPath (java.lang.String className, java.lang.String nameSpace)

Method Detail:
public java.lang.String getClassName ()
public java.lang.String getNameSpace ()

M12IdentityElement

Syntax Detail:
public M12IdentityElement (java.lang.String className, M12PropertySet identity)
public M12IdentityElement (java.lang.String className, java.lang.String nameSpace, M12PropertySet identity)

Method Detail:
public java.lang.String getClassName ()
public java.lang.String getNameSpace ()
public M12PropertySet getIdentity ()

M12ObjectIdentity

Syntax Detail:
public M12ObjectIdentity (M12IdentityElement[] scopingPath)

Method Detail:
public M12IdentityElement[] getScopingPath ()

M12PropertySet

Syntax Detail:
public M12PropertySet ()
public M12PropertySet (java.util.Properties prop)
public M12PropertySet (M12PropertySet propertySet)
Writing a provider for UNIX

On UNIX platforms, the provider environment is Java. The provider code is written as Java classes that implement the ILT interface described in this chapter.

There are two technologies used to create data providers: the first technology is writing the code that interacts with the managed resource, the second is writing the CIM classes that model the managed resource. The CIM classes definitions specify what code to invoke to retrieve the desired data.

The following sections show simplified elements for creating a provider.

Creating a MOF file for UNIX

The first step consists in writing a MOF file for UNIX providers (in other words a textual definition of a CIM class) and in loading the MOF file to WMI so that the IBM Tivoli Resource Model Builder can be used to create resource models for UNIX. Note that the MOF file must have DOS style CR/LF newlines, otherwise the classes do not show up in WMI.
The usual namespace in WMI is root\cimv2, but for UNIX, load the MOF files into root\default:

mofcomp -N:root\default Sample01.mof

M12_Instrumentation is a non-standard qualifier which applies to Class, Property and Method. Its format is:

[M12_Instrumentation("Java.<path to ILT>|<mappingstring>|<operation>"].

This qualifier is used to identify which ILT ("path to ILT") is able to perform the specified operation.

<path to ILT> is the Java class that implements the ILT and that must be specified with the complete package (with no .class extension).

<mappingstring> is a string whose meaning is known to the ILT.

The allowed operations are:
- Enum (only for the qualifier associated with the class)
- Get (for the qualifier associated with class and property)
- Invoke (for the qualifier associated with class and method)

It is required that all classes have an M12_Instrumentation qualifier for the Enum operation. It is also required that an M12_Instrumentation qualifier for the Get operation be present for every non-key property. As an alternative, when the instrumentation string is the same for every property, a single M12_Instrumentation qualifier for the Get operation can be set at class level. It is also required that an M12_Instrumentation qualifier for the Invoke operation be present for every method. As an alternative, when the instrumentation string is the same for every method, a single M12_Instrumentation qualifier for the Invoke operation can be set at class level.

The following is an example of MOF file for UNIX:

```
/******************* Description("Unix File Systems info"),
provider("com.tivoli.dmunix.ep.touchpoint.cimom.ifc.M12JavaProvider"),
M12_Instrumentation("Java.com.tivoli.dmunix.ep.ilts.DMXFileSystemIlt|ENUM"),
M12_Instrumentation("Java.com.tivoli.dmunix.ep.ilts.DMXFileSystemIlt|GET")
]}
class DMXFileSystem
{
[key]string mountPoint;
sint32 usedKBytes;
sint32 availKBytes;
[ provider("com.tivoli.dmunix.ep.touchpoint.cimom.ifc.M12JavaProvider"),
M12_Instrumentation("Java.com.tivoli.dmunix.ep.ilts.DMXFileSystemIlt|GET")]
sint32 totalKBytes;
};
```

Properties not associated with any "provider" or "M12_Instrumentation" qualifier are collected by calling the getMultipleProperties method, while all other properties are collected through a getProperty method.

**ILT Sample**

The following is a sample of ILT:
package com.tivoli.dmunix.ep.ilts;
import java.util.*;
import com.tivoli.javaultils.Trace;
import com.tivoli.dmunix.ep.touchpoint.base.*;
import com.tivoli.dmunix.ep.providers.DMXFileSystem;
public class DMXFileSystemIlt implements ILTInterface {
	public String getProperty(M12ObjectIdentity targetInstance, String propertyName,
	String mappingString, ParameterSet parms)
	throws M12Exception {
	try {
	M12IdentityElement idElem = (targetInstance.getScopingPath())[0];
	M12PropertySet propSet = idElem.getIdentity();
	String mountPoint = propSet.getProperty("mountPoint");
	if (propertyName.equals("totalKBytes")) {
	Integer res = new Integer(DMXFileSystem.getTotalKBytes(mountPoint));
	return res.toString();
	}
	} catch (Exception e) {
	...................
	public M12PropertySet getMultipleProperties (M12ObjectIdentity

targetInstance,

Vector propertyList,

String mappingString, ParameterSet parms)
	throws M12Exception {
	try {
	M12IdentityElement idElem = (targetInstance.getScopingPath())[0];
	M12PropertySet propSet = idElem.getIdentity();
	String mountPoint = propSet.getProperty("mountPoint");
	M12PropertySet result = new M12PropertySet();
	for (int i = 0; i < propertyList.size(); i++) {
	String propertyName = (String)propertyList.elementAt(i);
	if (propertyName.equals("usedKBytes")) {
	Integer res = new Integer(DMXFileSystem.getUsedKBytes(mountPoint));
	result.setProperty(propertyName, res);
	}
	(result.equals("availKBytes")) {
	Integer res = new Integer(DMXFileSystem.getAvailKBytes(mountPoint));
	result.setProperty(propertyName, res);
	return result;
	} catch (Exception e) {
	...................
	public Enumeration enumerateInstances (M12ClassPath classPath, String mappingString,

ParameterSet parms)
	throws M12Exception {
	try {
	Vector result = new Vector();
	String[] mountPoints = DMXFileSystem.getMountPoints();
	if (mountPoints == null) {
	trace.log(1, "DMXFileSystemIlt", "enumerateInstances: no instances found");
	return null;
	}
	for (int i = 0; i < mountPoints.length; i++) {
	M12IdentityElement idElem;
	M12PropertySet propSet = new M12PropertySet();
	propSet.setProperty("mountPoint", mountPoints[i]);
	nidElem = new M12IdentityElement(classPath.getClassName(),

classPath.getNameSpace(),

propSet);
	result.add(new M12ObjectIdentity(new M12IdentityElement[] {idElem}));
	return result.elements();
	} catch (Exception e) {
	...................
	...........................
	}
If the ILT interfaces with native libraries (through JNI, the Java Native Interface), these libraries must be added to the resource model as a dependency.

ILTs must be packaged into .jar files and added to the resource model as dependencies.

**Supported CIM Types**

The following list specifies the supported types for the attributes of CIM classes:

- string
- boolean
- sint8
- uint8
- sint16
- uint16
- sint32
- uint32
- sint64
- real32
- real64
Appendix D. Resource Model Troubleshooting

This appendix describes resource model troubleshooting methods, including states of the resource models, causes of the states of the resource models and actions taken by the engine.

When a resource model is run through the Tivoli Monitoring engine can take depends on the state of the resource model. The various possible states are either assigned automatically by the engine, or specified by the user in the decision tree script. The resource model states are displayed together with the monitoring results, by the command line, Tivoli Monitoring Web Health Console, or Tivoli Business Systems Manager, when enabled.

The following table shows the states automatically assigned by the engine, the external causes from which the states result, and the related actions taken by the engine.

<table>
<thead>
<tr>
<th>Resource model state</th>
<th>Cause of the state</th>
<th>Action taken by engine</th>
</tr>
</thead>
<tbody>
<tr>
<td>Running</td>
<td>Resource model running successfully.</td>
<td>None</td>
</tr>
<tr>
<td>Stopped</td>
<td>User stopped the resource model, or it has successfully finished running.</td>
<td>None</td>
</tr>
<tr>
<td>Disabled</td>
<td>User action.</td>
<td>None</td>
</tr>
<tr>
<td>Scheduled</td>
<td>User has scheduled resource model for a later time.</td>
<td>None</td>
</tr>
<tr>
<td>Error</td>
<td>An error is preventing the resource model from running.</td>
<td>Every 3 minutes the engine automatically tries to rerun it.</td>
</tr>
<tr>
<td>Missing prerequisites</td>
<td>Required prerequisites are missing.</td>
<td>None</td>
</tr>
<tr>
<td>Not compiled</td>
<td>An error in Visual Basic or Java Script code.</td>
<td>None</td>
</tr>
</tbody>
</table>

You can program resource model states so that they derive from the return codes generated by the monitoring algorithm you write. You can use the VisitTree, in the Init, or in the SetDefaultConfiguration functions to specify what return codes are given for particular states of the resource models.

All resource model states listed in the following table are caused by return codes as specified in the monitoring algorithm. Therefore when you write the monitoring algorithm, you can foresee specific situations that might cause the resource model to fail. You can specify a return code that must be returned in these situations. Return codes match with specific resource model states and cause the engine to take some actions.

**Example:** You have created a resource model that monitors some files. You know that sometimes one of these files can be temporarily locked, which might prevent the resource model from running successfully. You can bypass this problem adding a few lines to the VisitTree function, specifying that if one file is not accessible, the
VisitTree must return a specific code included between 601 and 800. This return code means that the resource model state is Retrying and the engine keeps trying to run it until it finds the file unlocked.

In the following table, the first column lists the resource model states. The second column lists the return code ranges. The third column lists how Tivoli Monitoring interprets resource model states and the consequent actions it takes.

<table>
<thead>
<tr>
<th>Resource model state</th>
<th>Cause of the state</th>
<th>Action taken by the engine</th>
</tr>
</thead>
<tbody>
<tr>
<td>Failed 1-200. The resource model has failed.</td>
<td>VisitTree return code: 201-400</td>
<td>None</td>
</tr>
<tr>
<td>Failing 1-200. The resource model has an error.</td>
<td>VisitTree return code: 401-600</td>
<td>The engine automatically retries every 3 minutes to run the resource model.</td>
</tr>
<tr>
<td>Retrying 1-200. The resource model is running</td>
<td>VisitTree return code: 601-800</td>
<td>Retries 3 times in each cycle time, indefinitely, to run the resource model.</td>
</tr>
<tr>
<td>Unable to start 1-200. The resource model is unable to start. Assumed missing prerequisites.</td>
<td>Init, or SetDefaultConfiguration return code: 801-1000</td>
<td>None</td>
</tr>
<tr>
<td>Recovering 1-200. The resource model is running</td>
<td>VisitTree return code: 1001-1100</td>
<td>Once per cycle, for three cycles only, tries to run the resource model. After three unsuccessful attempts, interprets the resource model as failed.</td>
</tr>
<tr>
<td>Failed after recovery</td>
<td>The resource model failed in three successive cycles. Interpreted as failed.</td>
<td>None</td>
</tr>
</tbody>
</table>

**Note:** If in the monitoring algorithm you specify a return code that is not included in the supported ranges, the IBM Tivoli Resource Model Builder assumes this return code to be 0 (successful).

The return codes that are displayed by the Tivoli Monitoring Web Health Console are not the same return codes you enter in the monitoring algorithm. Those displayed by the Tivoli Monitoring Web Health Console are the result of the following formula:

\[
\text{modulus of return code} / 200
\]

That means the remainder of the return code divided by 200. **Example:** if the VisitTree returns 1051, the Tivoli Monitoring Web Health Console displays: **Recovering 51.**

You can use different return codes to better specify different causes for the same resource model state. For example, a resource model that monitors some files can be in Retrying state because it finds a file locked (one return code), or because this file does not exist (another return code).
Appendix E. Accessibility

Accessibility features help a user who has a physical disability, such as restricted mobility or limited vision, to use software products successfully. The major accessibility features in IBM Tivoli Resource Model Builder enable users to:

- Use assistive technologies such as screen-reader software and a digital speech synthesizer to hear what is displayed on the screen
- Operate specific or equivalent features using only the keyboard
- Magnify what is displayed on the screen

In addition, the product documentation has been modified to include features to aid accessibility:

- All documentation available in both HTML and convertible PDF formats to give the maximum opportunity for users to apply screen-reader software.
- All images provided with alternative text so that users of the documentation with vision impairments can understand the contents of the images.

Using assistive technologies

Assistive technology products such as screen-readers, function with both the text-based and graphical user interfaces found in IBM Tivoli Resource Model Builder. Consult the assistive technology product documentation for specific information about using it to access command line or graphical interfaces.

Magnifying what is displayed on the screen

In all components of IBM Tivoli Resource Model Builder other than the Web Health Console, users can magnify the screens used by the product’s user interfaces using facilities provided by the operating systems on which the product is run. For example, in a Windows environment you can change the screen settings to a lower resolution to enlarge the font sizes of the text on the screen. Information about these facilities is provided in the relevant operating system documentation.

Documentation in accessible formats

All user documentation is provided in HTML format, which can be read directly by assistive tools such as screen readers, or in convertible PDF format. Convertible PDF files are those that can be converted from PDF to HTML by the Adobe PDF to HTML converter. For information about converting PDF documents to HTML, refer to the Adobe book Optimizing Adobe PDF Files for Accessibility.

Using alternative text

All documentation images are provided with an alternative text that can be read by assistive tools such as screen readers.
Appendix F. Notices

This information was developed for products and services offered in the U.S.A. IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user’s responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785 U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement might not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.
IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
2Z4A/101
11400 Burnet Road
Austin, TX  78758   U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

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.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. You may copy, modify, and distribute these sample programs in any form without payment to IBM for the purposes of developing, using, marketing, or distributing application programs conforming to IBM’s application programming interfaces.

If you are viewing this information in softcopy form, the photographs and color illustrations might not display.
Trademarks

AIX, DB2, IBM, the IBM logo, IBMLink, Informix, OS/2, OS/400, Tivoli, the Tivoli logo, Tivoli Enterprise Console, and TME are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both.

Intel, the Intel Inside logos, MMX, and Pentium are trademarks of Intel Corporation in the United States, other countries, or both.

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

Microsoft and Windows NT are registered trademarks of Microsoft Corporation in the United States, other countries, or both.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

SET and the SET Logo are trademarks owned by SET Secure Electronic Transaction LLC.

Other company, product, and service names may be trademarks or service marks of others.
Index

A
accessibility xi
Accessibility 155
actions 15
    adding
        CIM data source 26
        Custom Script data source 27
        DM Classic data source 27
        addInterp command 93
aggregating indications 14
asynchronous data collection 14
attributes 15

B
Basic Object Methods 111
Basic Resource Model Wizard 23
    using 24
bldamepkg command 94
bldpkg command 95
bldtecbaroc command 96
books
    see publications x
building
    Tivoli Enterprise Console BAROC 87

C
category descriptive name 46
category internal name 46
CIM
    implementation 2
    CIM class edits 47
    CIM class methods 15
    CIM data source 26
    CIM/WMI class 61
    Class Object library 111
    class selection 28
collecting data
    asynchronously 14
    synchronously 14
command summary 91
commands
    addInterp 93
    bldamepkg 94
    bldpkg 95
    bldtecbaroc 96
    convert 97
    expconf 98
    expmsgscat 99
    expmsgcat 100
    exptmsxml 101
    extract 102
    imptmsxml 103
    pack 104
    setcopyright 105
    setectree 106
    setdep 107
    setsendtobsm 108
commands (continued)
    setver 109
    summary 92
    Common Information Model 13
    Common Information Model implementation 2
    configuration file generation 87
    context, logging 17
    conventions
        typeface xii
        convert command 97
    creating a resource model 23
    custom script
        create resource model from 67
        Custom Script data source 27
        custom script specification 33
    customer support
        see software support xi
    cycle time 14, 46
    setting 39
    cycle time definition example 20

D
data
    filtering 14
    data collection
        asynchronous 14
        synchronous 14
    data logging 16
    data sorting 14
    decision tree script 18
    definition 60
    Init function 18
    main subroutine 18
    SetDefaultConfiguration function 18
    VisitTree function 18
    dependencies 18
    editing 59
    descriptive name 46
    directory names, notation xii
    disable TBSM events 46
    Distributed Monitoring Collection
        create resource model 65
    DM Classic data source 27
    DM classic probe edits 50
    Dynamic model 112

e
edit
    CIM classes 47
    data to log 57
    dependencies 59
    DM classic probes 50
    events 51
    general settings 45
    parameter 56
    source 60
edit (continued)
    threshold 55
    edit resource model 45
    empty resource model wizard 41
    Empty Resource Model Wizard 23
    environment variables, notation xii
    error
        viewing in script 73
        event
            editing 51
            viewing 74
        event trigger condition selection 35
        events 14
            actions 15
            attributes 15
            events and indications 14
            expconf command 98
            expmsgscat command 99
            expmsgcat command 100
            export package 83
            exporting
                message catalog 86
            exptmsxml command 101
            extract command 102

F
filter conditions 34
filtering 14
filtering definition example 20
function
    Init 18
    SetDefaultConfiguration 18
    VisitTree 18

G
general setting edits 45
    generate
        HTML file 89
        IBM Tivoli Monitoring configuration file 87
        java message catalog 85
        Managed Object Format file 88
        MOF file 88
        resource model package 83
        resource model settings 84
        script file 88
        getProperties 143

H
holes 14
HTML file 89
W

wizard
creating resource model  23