



Tivoli[®] Distributed Monitoring
Supplement
Version 4.1



Tivoli[®] Distributed Monitoring
Supplement
Version 4.1

Tivoli Distributed Monitoring (Advanced Edition) Supplement

Copyright Notice

© Copyright IBM Corporation 2000, 2002. All rights reserved. May only be used pursuant to a Tivoli Systems Software License Agreement, an IBM Software License Agreement, or Addendum for Tivoli Products to IBM Customer or License Agreement. No part of this publication may be reproduced, transmitted, transcribed, stored in a retrieval system, or translated into any computer language, in any form or by any means, electronic, mechanical, magnetic, optical, chemical, manual, or otherwise, without prior written permission of IBM Corporation. IBM Corporation grants you limited permission to make hardcopy or other reproductions of any machine-readable documentation for your own use, provided that each such reproduction shall carry the IBM Corporation copyright notice. No other rights under copyright are granted without prior written permission of IBM Corporation. The document is not intended for production and is furnished “as is” without warranty of any kind. **All warranties on this document are hereby disclaimed, including the warranties of merchantability and fitness for a particular purpose.**

U.S. Government Users Restricted Rights—Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corporation.

Trademarks

Tivoli, the Tivoli logo, Tivoli Enterprise Console, are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both.

Microsoft, Windows, and Windows NT, are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.



Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries.

Other company, product, and service names may be trademarks or service marks of others.

Notices

References in this publication to Tivoli Systems or IBM products, programs, or services do not imply that they will be available in all countries in which Tivoli Systems or IBM operates. Any reference to these products, programs, or services is not intended to imply that only Tivoli Systems or IBM products, programs, or services can be used. Subject to valid intellectual property or other legally protectable right of Tivoli Systems or IBM, any functionally equivalent product, program, or service can be used instead of the referenced product, program, or service. The evaluation and verification of operation in conjunction with other products, except those expressly designated by Tivoli Systems or IBM, are the responsibility of the user. Tivoli Systems or IBM may have patents or pending patent applications covering subject matter 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 the IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, New York 10504-1785, U.S.A.

ISO 9001 Certification

This product was developed using an ISO 9001 certified quality system.

Certification has been awarded by Bureau Veritas Quality International (BVQI) (Certification No. BVQI - 92086 / A).

BVQI is a world leader in quality certification and is currently recognized by more than 20 accreditation bodies.

Contents

Chapter 1. Introduction.....	1
What Is New in This Patch	1
Different Task Behaviour between Windows and UNIX Platforms	2
What This Supplement Contains	2
Chapter 2. Migrating from Tivoli Distributed Monitoring (Classic Edition) to Tivoli Distributed Monitoring (Advanced Edition)	5
Overview	5
Guidelines for Migrating to Tivoli Distributed Monitoring (Advanced Edition)	7
Design Time	7
Deployment Time	8
Run Time	9
Migration Process	10
Sentry Profile Analyzer	10
Compatibility Mode	14
Wizard Process	16
How Everything Fits Together	17
Chapter 3. Creating a Resource Model.....	19
Importing a Monitoring Source with the Wizard	19
Importing a Monitoring Source Manually	22
Importing a Custom Script	24
To Launch a Shell Command Using the Wizard	24
To Import a Script with the Wizard	25
Customizing the Event Messages	26
Chapter 4. Service Object Method Library	27
Basic Object Methods	27
Dynamic Model	27
Events	28
Logging	29
Utilities	29
Mapping Tables	30
Advanced Object Methods	33
Dynamic Model	33
Deprecated Methods	34
Exceptions	34

Appendix A. Error Messages	35
Identifying a Message	35
Notation	35
Messages	36

1

Introduction

This supplement is provided as documentation of the 4.1-DMA0001 patch to Tivoli Distributed Monitoring (Advanced Edition) 4.1.

What Is New in This Patch

The patch includes the following improvements and new features:

- Support of Solaris 2.6 platforms as endpoints for UNIX engines.
- The InstallShield **setup_hp11x** to install the Health Console on **hp-ux11.0** and **hp-ux11i** platforms. The InstallShield is launched issuing the following command:

```
setup_hp11x -is:javahome JVMDIR
```

Where JVMDIR is the installation path of JRE 1.3

Note: To install the Health Console, you have to install JRE 1.3 and all the patches required by the operating system. You can download JRE 1.3 from the following Web site: www.hp.com/products/unix/java/index.html . The list of JRE 1.3 required patches is available at the following Web site: www.hp.com/products1/unix/java/infolibrary/patches.html

- The recovery function of the agent where Tivoli Distributed Monitoring (Advanced Edition) is running. If, accidentally, the monitoring engine stops running, this function automatically restarts it, if enabled. Optionally, this condition can be notified to the heartbeat event subscribers as an event.

Note: To enable the recovery function, issue the following command:

```
wdmconfig -D heartbeat.reboot_engine_if_down = true
```

- Compatibility mode between Tivoli Distributed Monitoring (Classic Edition) monitoring sources and Tivoli Distributed Monitoring (Advanced Edition). It is achieved thanks to the implementation of some wizard procedures to be run on the workbench.
- New wizard implementation to import Tivoli Distributed Monitoring (Classic Edition) monitoring collections and custom scripts. See Chapter 3, “Creating a Resource Model” on page 19 for more details.
- Migration helper script to analyze and migrate Tivoli Distributed Monitoring (Classic Edition) monitors to Tivoli Distributed Monitoring (Advanced Edition) resource models.
- The Autotrace tracing software is now available for Solaris, HP-UX, and AIX Tivoli management region servers. For more information on this feature, see Appendix C of *Tivoli Distributed Monitoring (Advanced Edition) User's Guide, Version 4.1*.

- There is a new UNIX/Linux endpoint log file, *dmxout.log*, that collects information and errors at Java engine startup. It is located at:
\$LCF_DATDIR/LCFNEW/Tmw2k/Unix/data. In particular it checks:
 - The installation of the Java Runtime Environment (JRE)
 - The version of the JRE
 - The platforms supported
 - The correct startup of the JRE.
- The following APARs have been fixed:
 - IY27571: On AIX® and HP systems, the task *DMAE_ReadDataInDB* does not complete successfully because some libraries are missing.
 - IY26682: All numerical fields need to be protected so that the comma sign is correctly handled.
 - IY26927: The man page and the *Tivoli Distributed Monitoring (Advanced Edition) User's Guide, Version 4.1* state that the **wdmloadprf** command can be used to load a file from workbench. This is incorrect, workbench has nothing to do with profile definitions, it regards only resource models.
 - IY26796: The *set_env.sh* script does not source *lcf_env.sh*.
 - IY26792: Serviceability deficiencies found during a failed profile.
 - IY27863: Cannot distribute a *Tmw2kProfile* to an endpoint that is not directly subscribed to the profile's ProfileManager.
 - IY27859: The provider for the resource model *DMXProcess* on AIX does not filter out the "wait/idle CPU" none-process.

Different Task Behaviour between Windows and UNIX Platforms

On Windows platforms, tasks are executed every time the related event is generated, while on UNIX platforms, tasks follow the "clearing event" rule, that is, if an event is generated for more than one cycle time consecutively, then the task is executed only the first time.

What This Supplement Contains

This Supplement is organized into the following chapters and appendix:

- Chapter 2, "Migrating from Tivoli Distributed Monitoring (Classic Edition) to Tivoli Distributed Monitoring (Advanced Edition)" on page 5 describes the migration from Tivoli Distributed Monitoring (Classic Edition) to Tivoli Distributed Monitoring (Advanced Edition), and the considerations to be taken into account to ease the process.
- Chapter 3, "Creating a Resource Model" on page 19 describes the new procedures available for creating a resource model. It also describes how to import a monitoring source manually and how to import a customized script into a new or existing resource model. There is also a section on customizing event messages.
- Chapter 4, "Service Object Method Library" on page 27 describes the Class Object library. It describes both the newly implemented basic and advanced methods for the *TMWService* Object.
- Appendix A, "Error Messages" on page 35 describes the messages you can get when creating or debugging a resource model with the workbench.

More information about Tivoli Distributed Monitoring (Advanced Edition) can be found in *Tivoli Distributed Monitoring (Advanced Edition) User's Guide, Version 4.1* and *Tivoli Distributed Monitoring (Advanced Edition) Workbench User's Guide, Version 4.1*.

2

Migrating from Tivoli Distributed Monitoring (Classic Edition) to Tivoli Distributed Monitoring (Advanced Edition)

This chapter describes the migration from Tivoli Distributed Monitoring (Classic Edition) to Tivoli Distributed Monitoring (Advanced Edition).

Tivoli Distributed Monitoring (Advanced Edition) allows you to monitor availability and performance status of resources on your systems to identify bottlenecks and potential resource problems. It was formerly known as Tivoli Distributed Monitoring for Windows®.

Users of Tivoli Distributed Monitoring (Classic Edition), formerly known as Tivoli Distributed Monitoring, can now easily migrate to using its successor, Tivoli Distributed Monitoring (Advanced Edition).

Overview

Tivoli Distributed Monitoring (Advanced Edition) applies monitoring in a different way from Tivoli Distributed Monitoring (Classic Edition). Whereas the Tivoli Distributed Monitoring (Classic Edition) is based on the concept of monitoring capability collections and monitors, Tivoli Distributed Monitoring (Advanced Edition) applies preconfigured, automated best practices to the automated monitoring of essential system and application resources. Basically, Tivoli Distributed Monitoring (Classic Edition) provides the means for retrieving data through scripts or commands, verifies that the retrieved values do not exceed given thresholds, and finally, based on the user's customization, triggers some response (Tivoli Enterprise Console® events, e-mail, notices, and so on). Tivoli Distributed Monitoring (Classic Edition) leaves the correlation and the problem's root cause analysis to the user.

Tivoli Distributed Monitoring (Advanced Edition) is based on the concept of resource models, implemented through best practice scripts, or reference models, and the definition in Common Information Model (CIM) in terms of monitored resources. See the Web site: http://www.dmtf.org/standards/cim_spec_v22/ for more details. Moreover, Tivoli Distributed Monitoring (Advanced Edition) provides a set of resource models designed to detect run time bottlenecks and other potential problems and to automatically recover from critical situations, eliminating the need for system administrators to manually scan through extensive performance data. For more details see the *Tivoli Distributed Monitoring (Advanced Edition) User's Guide, Version 4.1* and the *Tivoli Distributed Monitoring (Advanced Edition) Workbench User's Guide, Version 4.1*.

Tivoli Distributed Monitoring (Advanced Edition) may use processes that form part of the endpoints' operating systems to obtain resource data. On Windows systems it uses the Windows Management Instrumentation (WMI), which is Microsoft®'s implementation of CIM (see the Web site: <http://msdn.microsoft.com/library/default.asp?url=/library/en->

[us/wmisdk/aboutwmi_1lpl.asp](#)). WMI allows applications to retrieve information about the current status of a system. On UNIX® and Linux platforms the information collection agent is incorporated in the product based on CIM specifications. For more details see the *Tivoli Distributed Monitoring (Advanced Edition) User's Guide, Version 4.1*.

The Tivoli Distributed Monitoring 4.1 bundle provides both Tivoli Distributed Monitoring (Advanced Edition) and Tivoli Distributed Monitoring (Classic Edition) 3.7, so that users can use both and plan their migration to Tivoli Distributed Monitoring (Advanced Edition) in the smoothest possible way. Users should take the following two aspects into consideration:

Coexistence

The two products can coexist as they have two different implementations at the server, gateways, and endpoints. The installation of Tivoli Distributed Monitoring (Advanced Edition) does not change the Tivoli Distributed Monitoring (Classic Edition) environment or configuration. Even when working in Compatibility Mode, see "Compatibility Mode" on page 14, the two products can coexist and run entirely independently.

Migration

To make the migration from Tivoli Distributed Monitoring (Classic Edition) to Tivoli Distributed Monitoring (Advanced Edition) as easy as possible and to save the user's investment on custom scripts and home-developed monitoring collections, the *4.1-DMA-0001 Patch* allows Tivoli Distributed Monitoring (Advanced Edition) to run in "Compatibility Mode", the new working mode that allows Tivoli Distributed Monitoring (Advanced Edition) users to use Tivoli Distributed Monitoring (Classic Edition) monitor collections and custom scripts within resource models.

Due to the different nature of the two versions of Tivoli Distributed Monitoring, the migration can only be semi-automated. In addition to the Compatibility Mode which gives users the possibility of using their scripts or monitors by leveraging Tivoli Distributed Monitoring (Advanced Edition) features, Tivoli Distributed Monitoring (Advanced Edition) also provides a "migration helper" script, *dmae_sentryprofile_analyser.sh*, which provides information on how and where the metric monitored by the current Tivoli Distributed Monitoring (Classic Edition) profiles can be replaced using Tivoli Distributed Monitoring (Advanced Edition). The script will be able to automatically create Tivoli Distributed Monitoring (Advanced Edition) profiles containing resource models that try to address the same problems as the Tivoli Distributed Monitoring (Classic Edition) monitors. More details about this script can be found later in the chapter.

To take full advantage of the Tivoli Distributed Monitoring (Advanced Edition) key features such as instance autodiscovery, metric correlation, aggregation, data logging, and so on, may require some Visual Basic or JavaScript coding skills that might not be immediately available, so to help to introduce Tivoli Distributed Monitoring (Classic Edition) users to the Tivoli Distributed Monitoring (Advanced Edition) monitoring approach the *4.1-DMA-0001 Patch* provides a "wizard-driven process" within the workbench to create resource models and automatically generate all the required code. The wizard allows the user to follow the simple monitoring paradigm in Tivoli Distributed Monitoring (Classic Edition) by leveraging the new functionality offered by Tivoli Distributed Monitoring (Advanced Edition). The wizard approach relieves the user from writing additional code by generating code that applies simple Tivoli Distributed Monitoring (Classic Edition) logic:

```
if (metric value - comparison - threshold) then response action
```

Using the wizard together with the compatibility mode allows users to replicate the same monitoring they already have in place with Tivoli Distributed Monitoring (Classic Edition). More details of the wizard can be found in Chapter 3, “Creating a Resource Model” on page 19.

Guidelines for Migrating to Tivoli Distributed Monitoring (Advanced Edition)

In order to move monitoring solutions from Tivoli Distributed Monitoring (Classic Edition) to Tivoli Distributed Monitoring (Advanced Edition) implementation, it is important to better understand the key features that Tivoli Distributed Monitoring (Advanced Edition) offers at the different phases of its use.

Design Time

Tivoli Distributed Monitoring (Advanced Edition) provides the workbench, an integrated development environment (IDE) to design, create, test, and debug resource models. See the *Tivoli Distributed Monitoring (Advanced Edition) Workbench User's Guide, Version 4.1* for more information. Basically, new Tivoli Distributed Monitoring (Advanced Edition) users should consider the workbench as the place where they can build their own monitoring best practice, where they can modify the current out-of-the-box resource model best practices, and where they can customize and configure the resource models in a more detailed way. In Tivoli Distributed Monitoring (Advanced Edition) users should see the workbench as an additional and more complete GUI where they can customize resource models in addition to the TME[®]-based GUI. The importance of the workbench is mainly related to the possibility of creating and debugging new resource models.

Data sources

Tivoli Distributed Monitoring (Advanced Edition) with the *4.1-DMA-0001 Patch* is able to gather data not only from CIM data sources but also from Tivoli Distributed Monitoring (Classic Edition) monitors and custom scripts (Compatibility Mode). In this way Tivoli Distributed Monitoring Advanced Edition enhances the set of data sources with those already supported by Tivoli Distributed Monitoring (Classic Edition). However, it is still preferable to work in “native mode”, where the monitored data is collected from the underlying CIM implementation. While on UNIX and Linux platforms the CIMOM implementation is embedded on the engine, on Windows platforms the engine is based on WMI implementation (see the Web site: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/wmisdk/aboutwmi_1lpl.asp). Microsoft operating systems and Backoffice applications are delivered with their own WMI providers so that the monitoring of such resources can be implemented very quickly.

Aggregation

Differently from Tivoli Distributed Monitoring (Classic Edition), in Tivoli Distributed Monitoring (Advanced Edition) all events can be consolidated locally at the endpoint before flowing to the upper layers such as Tivoli Enterprise Console or Tivoli Business Systems Manager. This feature is very important from a scalability point of view. Also, the best practice can leverage this functionality to make the troubleshooting analysis smarter and more realistic (for example, whether a process exceeds its CPU usage just once or many times). New Tivoli Distributed Monitoring Advanced Edition users should always take this feature into account when generating an event.

Clearing

As the starting point for Tivoli Distributed Monitoring (Advanced Edition) is providing information for problems that are critical to operating system or application resources, Tivoli Distributed Monitoring (Advanced Edition) notifies users only when the problem arises through Tivoli Enterprise Console or Tivoli Business Systems Manager, and only when the problem has been solved, does it send a warning with a Clearing Event. See the *Tivoli Distributed Monitoring (Advanced Edition) User's Guide, Version 4.1* for more information.

Correlation

The most important feature is definitely having the possibility to write best practices to troubleshoot problems before they happen. That is, the root cause analysis can be performed by the resource model script that, using the Tivoli Distributed Monitoring (Advanced Edition) engine API, defines the data to be collected and, once it has been collected by the engine, retrieves all the metric values and correlates them to establish the final cause of arising problems. The correlation is then achieved by following a programming model that, in simple cases can be implemented by using the wizard-driven process. Metric values collected through the compatibility mode can of course be correlated too.

Multimetric

When starting to write new resource models, new Tivoli Distributed Monitoring (Advanced Edition) users should consider that in each resource model it is possible to collect an undefined number of resources and for each resource an undefined number of metrics. Moreover it gives the possibility to have autodiscovery for all the instances of a resource.

Note: Implementing a resource model that looks just at one metric of a resource may be expensive as Tivoli Distributed Monitoring (Advanced Edition) creates a new thread and a set of objects for each script. Better results can be obtained by designing the resource model to address a specific set of problems (for example, memory and CPU bottlenecks).

Automation

When building new resource models, or while modifying existing ones, Tivoli Distributed Monitoring (Advanced Edition) users can set recovery actions called "built-in actions" to be triggered whenever an indication is consolidated. The built-in actions are the running of CIM methods (see *Tivoli Distributed Monitoring (Advanced Edition) Workbench User's Guide, Version 4.1*) against a CIM class or CIM class instance.

Logging

Tivoli Distributed Monitoring (Advanced Edition) allows users to log data to a local database. Data stored in such a database can be retrieved and viewed through the Health Console. Data retrieved through the "compatibility mode" can also be logged.

Deployment Time

The Tivoli Distributed Monitoring (Advanced Edition) resource models are configured and deployed through Tivoli Management Framework just as they are in Tivoli Distributed Monitoring (Classic Edition). Instead of using a Monitoring Collection and monitors, Tivoli Distributed Monitoring (Advanced Edition) users use resource models, and instead of creating a SentryProfile create a Tivoli Distributed Monitoring (Advanced Edition) profile, *Tmw2kProfile*. As the configuration and deployment procedures of Tivoli Distributed

Monitoring (Advanced Edition) follow the paradigm followed by all profile-based Tivoli monitoring applications, we will describe only the things that differ from Tivoli Distributed Monitoring (Classic Edition) usage.

Resource model installation

Once Tivoli Distributed Monitoring (Advanced Edition) users have built their own resource models they can build the package with the workbench (see *Tivoli Distributed Monitoring (Advanced Edition) Workbench User's Guide, Version 4.1*) and then install it on the Tivoli management region, using the **wdmrm** command.

Tivoli Enterprise Console and Tivoli Business Systems Manager Customization

Tivoli Distributed Monitoring (Advanced Edition) allows users to set a unique destination Tivoli Enterprise Console server per profile. This means that all resource models belonging to the same profile will send events to the Tivoli Enterprise Console server. The target Tivoli Enterprise Console can be specified from the properties dialog (see *Tivoli Distributed Monitoring (Advanced Edition) User's Guide, Version 4.1*).

Task configuration

Tivoli tasks can be run as recovery actions. Differently from Tivoli Distributed Monitoring (Classic Edition) they can be triggered only when an indication has been consolidated.

Note: In the Task environment there are the Event properties in the form of environment variables, (see *Tivoli Distributed Monitoring (Advanced Edition) User's Guide, Version 4.1*).

Profile distribution

Tivoli Distributed Monitoring (Advanced Edition) supports endpoints only. This means that there is no managed node Tivoli Distributed Monitoring (Advanced Edition) monitoring engine, and implies some rework to be done when replacing Tivoli Distributed Monitoring (Classic Edition) profiles with Tivoli Distributed Monitoring (Advanced Edition) profiles because Tivoli Distributed Monitoring (Advanced Edition) cannot be distributed on managed nodes.

Run Time

Once a Tivoli Distributed Monitoring (Advanced Edition) profile and its resource models have been distributed to endpoints, the following considerations must be made for Tivoli Distributed Monitoring (Advanced Edition):

Multithread

Tivoli Distributed Monitoring (Advanced Edition) operates in a multithread environment. This means that each script runs in a separate thread and no processes are generated except those needed to run in compatibility mode. Tivoli Distributed Monitoring (Advanced Edition) users should take this into account when creating a resource model that looks at only one metric.

Health Console

Tivoli Distributed Monitoring (Advanced Edition) provides a stand-alone Java™ GUI called Health Console to browse the Tivoli Distributed Monitoring (Advanced Edition) monitored endpoints. Details of the Health Console features can be found in the *Tivoli Distributed Monitoring (Advanced Edition) User's Guide, Version 4.1*.

Tivoli Enterprise Console rule for clearing event

Tivoli Distributed Monitoring (Advanced Edition) provides a Tivoli Enterprise

Console rule (see *Tivoli Distributed Monitoring (Advanced Edition) User's Guide, Version 4.1*) to automatically close events for which a Clearing event is generated.

Migration Process

As described above, Tivoli Distributed Monitoring (Advanced Edition) provides a set of facilities to help and facilitate Tivoli Distributed Monitoring (Classic Edition) users to migrate their monitoring solutions into a Tivoli Distributed Monitoring (Advanced Edition) environment. There is no tool to completely automate the migration process because, as described above, the two products are really different.

So what should new Tivoli Distributed Monitoring (Advanced Edition) users do? How can they migrate to the new monitoring infrastructure? Here is a description of the facilities that, used together, will help users in their migration task. Tivoli Distributed Monitoring (Advanced Edition) users who do not plan to use the Sentry Profile analyzer script should still read the following section because it describes the thought process to be followed when planning to migrate.

Sentry Profile Analyzer

The first facility is the Sentry Profile analyzer script that analyses the contents of all the Sentry Profiles and, based on a mapping table provided with Tivoli Distributed Monitoring (Advanced Edition), produces a report suggesting how the monitors can be replaced with resource models or how new resource models can be created to collect the same data.

Installation Path

The installation of the *4.1-DMA-0001 Patch* on TME puts the script, "dmae_sentryanalyser.sh", under \$BINDIR/TME/Tmw2k/migration_helper. The mapping table file, "monitors_rm_table", is also installed in the same directory.

Mapping Table Content

The mapping table defines:

- If the metric values returned by each monitor provided by Tivoli Distributed Monitoring (Classic Edition) are collected by a Tivoli Distributed Monitoring (Advanced Edition) resource model, for example, the monitor "AvailBytes" of the NT_Memory monitoring collection is collected by the Memory resource model.
- If a metric is defined in a Tivoli Distributed Monitoring (Classic Edition) CIM class: that is, the corresponding CIM class.

Note: many metrics usually belong to the same class, for example, the monitor "AvailBytes" of NT_Memory can be found as a property of the CIM class TMW_Memory, and the monitor "Committed Bytes" can be found as a property of the same TMW_Memory class.

- For the metrics defined in a CIM class, could there be a Tivoli Distributed Monitoring (Advanced Edition) MOF file where they are defined. See the Web site:
<http://www.dmtf.org/education/cimtutorial/extend/spec.php#MOFLanguage> for more details. For example, the class TMW_Memory is defined in the file TMW_Resources10.mof.

Note: the TMW_Resources10.mof file is automatically installed on WMI with the endpoint engine when the first push is performed.

All the other mof files are only installed in the related CIM implementation when the resource model containing these files as dependencies gets downloaded. This means that creating a resource model that uses a resource defined in `TMW_Resource10.mof` there is no need to add that file to the dependency, while for all other cases the user must add the mof file to the dependency of the resource model.

- If a metric is made available by any CIM provider, independently of whether or not it is a property of a CIM class, for example, the "TransitionFaults" metric is made available by the PerfProv WMI provider, but there is not any CIM class already available on Windows NT that indirectly uses that provider to collect that counter.

At the Web site:

http://msdn.microsoft.com/library/default.asp?url=/library/en-us/wmisdk/us_perfcount users can find more information on how to use performance data providers and at the Web site:

http://msdn.microsoft.com/library/default.asp?url=/library/en-us/wmisdk/us_prov_33mx.asp for all other WMI providers. All providers used on Windows platforms are provided by default with WMI core, so there is no need to install or download these on the endpoints.

- Could there be a case such as "TMW_NetworkPortInfo" whose provider has been developed by Tivoli and then in order to be used on an endpoint must be added to the dependencies of a resource model. All providers used for UNIX and Linux platforms, as there is no native CIM implementation, have been developed by Tivoli and their entire implementation must be added to the resource model dependencies. The implementations of all UNIX and Linux providers consist of:
 - a tar file containing a set of Java class files
 - for each supported platform, a native shared library responsible for gathering data.

Based on the above considerations, in order to reuse a UNIX or Linux CIM class inside a resource model, users must add to the dependencies the related mof file and the tar file (they are common to all platforms) and the related shared library for each platform to be supported.

The Sentry Profile analyzer script must be run from the Tivoli environment on any Server or Managed Node. Based on the content of the mapping table described above, the Sentry Profile analyzer analyzes all Sentry Profiles present in the Tivoli management region and produces a report that suggests the way to proceed in the migration process.

Optionally, using "-p" option it can also create Tivoli Distributed Monitoring (Advanced Edition) profiles with the resource models covering as much as possible the resources monitored by the Tivoli Distributed Monitoring (Classic Edition) monitors.

The Analyzer Script Process

The script proceeds with the following paradigm.

- It verifies whether the metrics of the monitors set in a SentryProfile are collected by any Tivoli Distributed Monitoring (Advanced Edition). If they are, the script generates a section in the report describing which monitor can be

replaced with which resource model. Optionally a Tmw2kProfile with the same name as that containing the monitor will be created. Other resource models can also be added to that profile. Note that the configuration of a monitor in terms of arguments, responses, schedules, and so on, cannot be replicated because of the different product object models (see the section above). Usually the arguments in Tivoli Distributed Monitoring (Classic Edition) monitors are used to identify a specific resource instance. This is in conflict with the format of the out-of-the-box Tivoli Distributed Monitoring (Advanced Edition) resource models which try to autodiscover the failing instances at run time. Current resource models in fact do not accept resource instances as input. Nevertheless they can easily be modified with the workbench by using the parameters that only look and perform their analyses on specific instances. Also, other customization related to responses and schedules cannot be automatically recreated within a resource model but instead existing resource models can be modified in order to simulate the Tivoli Distributed Monitoring (Classic Edition) configuration.

- If no resource models that collect a specific metric are found in the mapping table then the script tries to identify whether a CIM class exists which provides that value. If one does then a section is added to the report. This only happens for Windows Platforms because there are several providers and CIM classes already made available by Microsoft. This information is really important because users, using the workbench (with or without the wizard), can easily take advantage of these classes to create a more sophisticated resource model running in "native mode".
- Sometimes there might be the case where the script finds a metric that is not implemented by any CIM class, but that is provided by a provider such as PerfProv. Microsoft provides with WMI core the WMI performance counter provider that can be used to define CIM classes representing the objects and their counters as they appear in the Windows performance monitor. Instructions on how to use the provider can be found at the Web site:http://msdn.microsoft.com/library/default.asp?url=/library/en-us/wmisdk/aboutwmi_1lpl.asp and good examples can be seen in the TMW_Resources10.mof that can be found at **\$BINDIR/./lcf_bundle40/Tmw2k/Mof**. Most Tivoli Distributed Monitoring (Advanced Edition) Windows resource models are based on CIM classes using the PerfProv WMI provider. Here is an example:

On Windows platforms the provider for instances or properties of a given class is specified through the "Provider" CIM qualifier. For example the provider for the following class NTProcesses is the "PerfProv".

```
[dynamic, provider("PerfProv"), ClassContext("local|Process")]
class NTProcesses
{

    [key]
    String Process;
    [PropertyContext("ID Process")]
    uint32 ID;
    [PropertyContext("Working Set")]
    uint32 WorkingSet;

};
```

The script will add a section to the report suggesting that the user considers creating a new resource model based on new CIM classes that can be created using the indicated provider.

- If the monitor examined cannot be remapped in any CIM class without the creation of a new provider (see the Web site: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/wmisdk/aboutwmi_1lpl.asp for how to create WMI providers), the script suggests using the "Compatibility Mode" in conjunction with the Wizard-driven process and choosing the Tivoli Distributed Monitoring Classic Monitoring Collection as the data source. Note that in order to optimize the use of monitors users should consider using more than one monitor of the same Monitoring collection inside the same resource model wherever possible.
- If the monitor is a custom script (string or numeric) and cannot be remapped in any CIM class without the creation of a new provider (see the Web site: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/wmisdk/aboutwmi_1lpl.asp for how to create WMI providers), the script suggests using the "Compatibility Mode" in conjunction with the Wizard-driven process and choosing "Custom Script" as the data source.
- If the monitor is an async monitor then users should ask Tivoli services to replace that functionality as it has not yet been implemented.

Examples

Here is an example of report generated by the Sentry Profile analyzer script:

```
#      Tivoli Distributed Monitoring (Advanced Edition),
#      Version 4.1 (C) Copyright IBM Corporation 2001.
#      All rights reserved.
```

The metrics collected by the following monitors are collected by the by the following RESOURCES MODELS:

	COLLECTION	MONITOR	RESOURCE MODEL	CIM CLASSES(PROVIDER)
UnixProfile#MyRegion	Unix_Sentry	File size	DMXFile	DMXFile(DMXFileIlt)
UnixProfile#MyRegion	Unix_Sentry	File size	DMXSecurity	DMXFileSec(DMXFileSecIlt)
UnixProfile#MyRegion	Unix_Sentry	Space free	DMXFileSystem	DMXFileSystem(DMXFileSystemIlt)
WinNTProfile#MyRegion	NT_NetworkMonitor	Network utilization	TMW_NetworkIntCard	TMW_NetworkSegment(PerfProv)
WinNTProfile#MyRegion	NT_LogicalDisk	Avg Disk sec/Transfer	TMW_LogicalDisk	TMW_LogicalDisk(PerfProv)

=====

The metrics collected by the following monitors can be found in the following CIM CLASSES:

	COLLECTION	MONITOR	CIM CLASSES(PROVIDER)
--	------------	---------	-----------------------

=====

Please consider to create Cim class to use the performance provider.
For more information visit:
[www.http://msdn.microsoft.com/library/default.asp?url=/library/en-us/wmisdk/r_prov_2joy.asp](http://msdn.microsoft.com/library/default.asp?url=/library/en-us/wmisdk/r_prov_2joy.asp).

	COLLECTION	MONITOR	PROVIDER
--	------------	---------	----------

DummyProfile#NewPolicy	NT_Cache	Async Pin Reads/sec	PerfProv
DummyProfile#NewPolicy	NT_Cache	Pin Reads/sec	PerfProv
DummyProfile#NewPolicy	NT_Cache	Sync Pin Reads/sec	PerfProv
DummyProfile#NewPolicy	NT_System	File Control Bytes/sec	PerfProv
WinNTProfile#MyRegion	NT_Cache	Async Pin Reads/sec	PerfProv
WinNTProfile#MyRegion	NT_Cache	Pin Reads/sec	PerfProv
WinNTProfile#MyRegion	NT_Cache	Sync Pin Reads/sec	PerfProv

=====

The metrics collected by the following monitors are not collected by any resource model and can not be found in any CIM class. Please consider to use WORKBENCH WIZARD choosing 'DM CLASSIC MONITORING COLLECTION' to import the monitor in a resource model

	COLLECTION	MONITOR
DummyProfile#NewPolicy	Unix_Sentry	daemonct
DummyProfile#NewPolicy	Unix_Sentry	diskusedpct
DummyProfile#NewPolicy	Unix_Sentry	diskused
OS400Profile#EMEA-region	OS/400 Job	SubType
OS400Profile#EMEA-region	OS/400 Object	ObjectOwner
UnixProfile#MyRegion	Unix_Sentry	daemonct

=====

The metrics collected by the following monitors are not collected by any resource model and can not be found in any CIM class. Please consider to use WORKBENCH WIZARD choosing 'CUSTOM SCRIPTS' to import the monitor in a resource model

	COLLECTION	MONITOR
--	------------	---------

=====

Please consider to ask Tivoli services for replacing the following monitors:

	COLLECTION	MONITOR
UnixProfile#MyRegion	Unix_Sentry	sasync

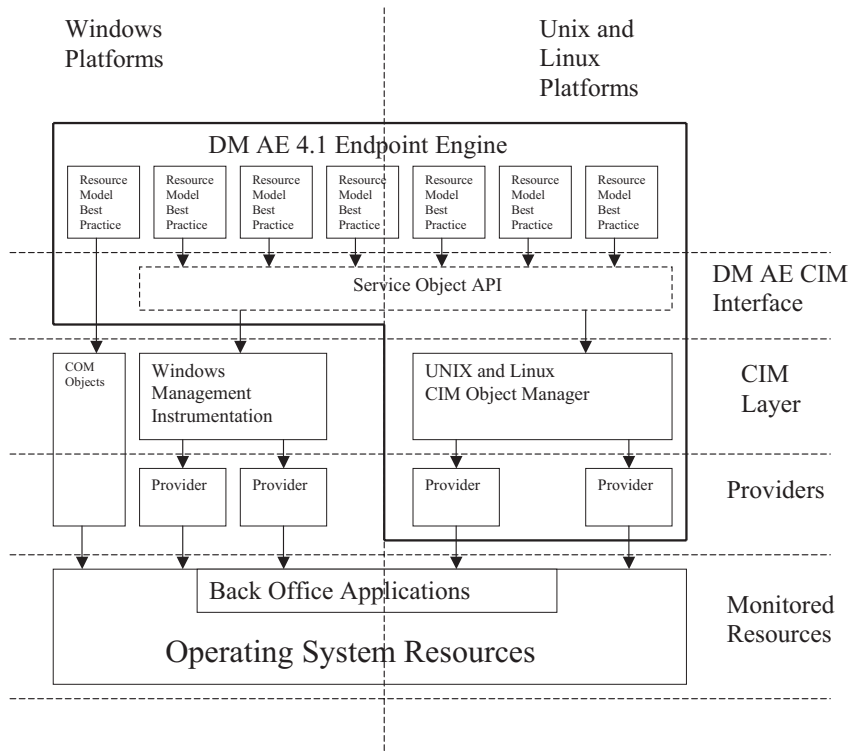
Compatibility Mode

The compatibility mode is a new working mode allowing Tivoli Distributed Monitoring (Advanced Edition) users to use Tivoli Distributed Monitoring (Classic Edition) monitors inside a Tivoli Distributed Monitoring (Advanced Edition) resource model. In this way, Tivoli Distributed Monitoring (Advanced Edition) can collect data to analyze, not only from the CIM data source, but also from Tivoli Distributed Monitoring (Classic Edition) monitoring sources (also known as probes). This means that Tivoli Distributed Monitoring (Classic Edition) users can recycle their existing customized Tivoli Distributed Monitoring (Classic Edition) monitoring collections into the new Tivoli Distributed Monitoring (Advanced Edition) resource models.

Tivoli Distributed Monitoring (Advanced Edition) uses "native mode" to collect data, relying on CIM Object Manager implementation (WMI on Windows or an endpoint Tivoli Distributed Monitoring (Advanced Edition) engine itself on UNIX and Linux), to collect data about system resources. The following picture shows the endpoint Tivoli Distributed Monitoring (Advanced Edition) engine architecture and how the Tivoli Distributed Monitoring (Advanced Edition) uses CIM Object Manager implementations in the DM 4.1

release, that is, before applying the *4.1-DMA0001Patch*. The Tivoli Distributed Monitoring (Advanced Edition) engine works only in "native mode" and resource model best practice scripts interact with the underlying CIMOM implementation through a set of API. The CIMOM is responsible for loading the providers that, in turn, get performance and availability data from system and application resources. Windows resource models may interact with COM objects directly from the scripts. In this case the CIM layer is not used.

The following picture illustrates the Tivoli Distributed Monitoring 4.1 native mode:

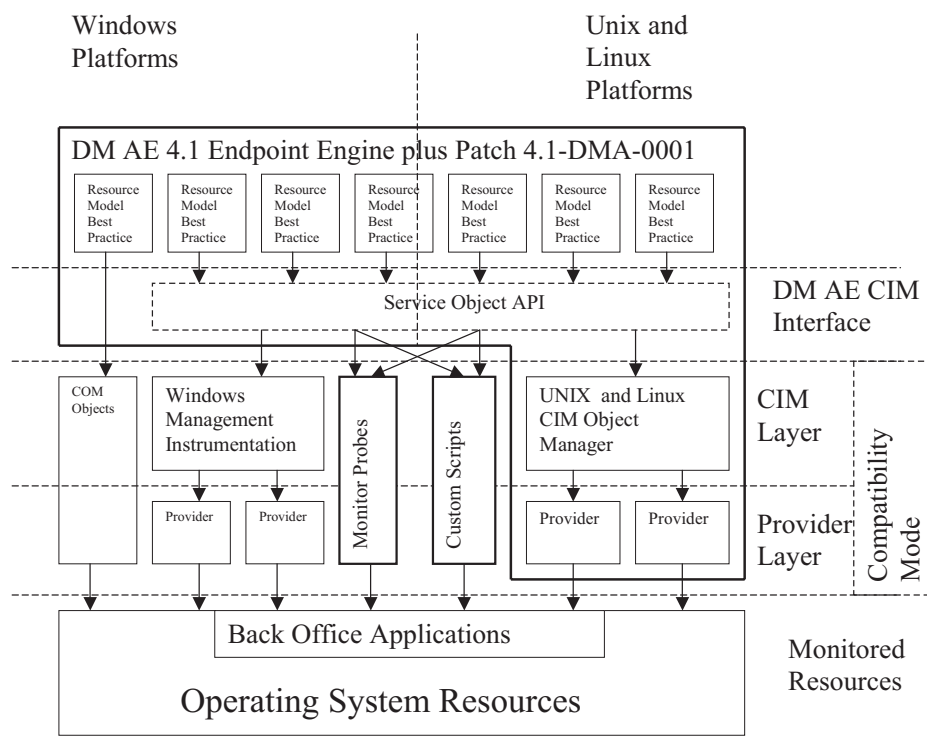


The compatibility mode can be used in addition to the native mode, so it is possible to have resource models that use both these technologies in their implementation. The compatibility mode provides an easy way to import Tivoli Distributed Monitoring (Classic Edition) probe implementation into a resource model without the user having to write any additional code, by supplying the workbench with a new wizard that imports Tivoli Distributed Monitoring (Classic Edition) monitoring sources. The Wizard process is described in Chapter 3, "Creating a Resource Model" on page 19. The following picture shows the endpoint Tivoli Distributed Monitoring (Advanced Edition) engine architecture with the *4.1-DMA0001Patch*

applied. It can be noted the Service Object API now enables the resource model scripts to use monitoring collection probes and custom scripts and CIM in any combination.

Note: Asynchronous monitors are not supported.

The following picture illustrates Tivoli Distributed Monitoring 4.1 and *4.1-DMA0001Patch* native and compatibility modes:



Wizard Process

The workbench provides a wizard, that is a GUI- driven process, to create new resource models. Basically, the wizard, starting from a selected CIM class taken from the WMI repository, from a monitoring collection, or from a custom script, displays a sequence of dialogs with default values already filled in to drive the user to create simple resource models. At the end of the wizard process all the needed Visual Basic or JavaScript code will

be automatically generated. This code will implement a monitoring logic very similar to that used by Tivoli Distributed Monitoring (Classic Edition), that is:

```
if (metric value - comparison - threshold) then indication
```

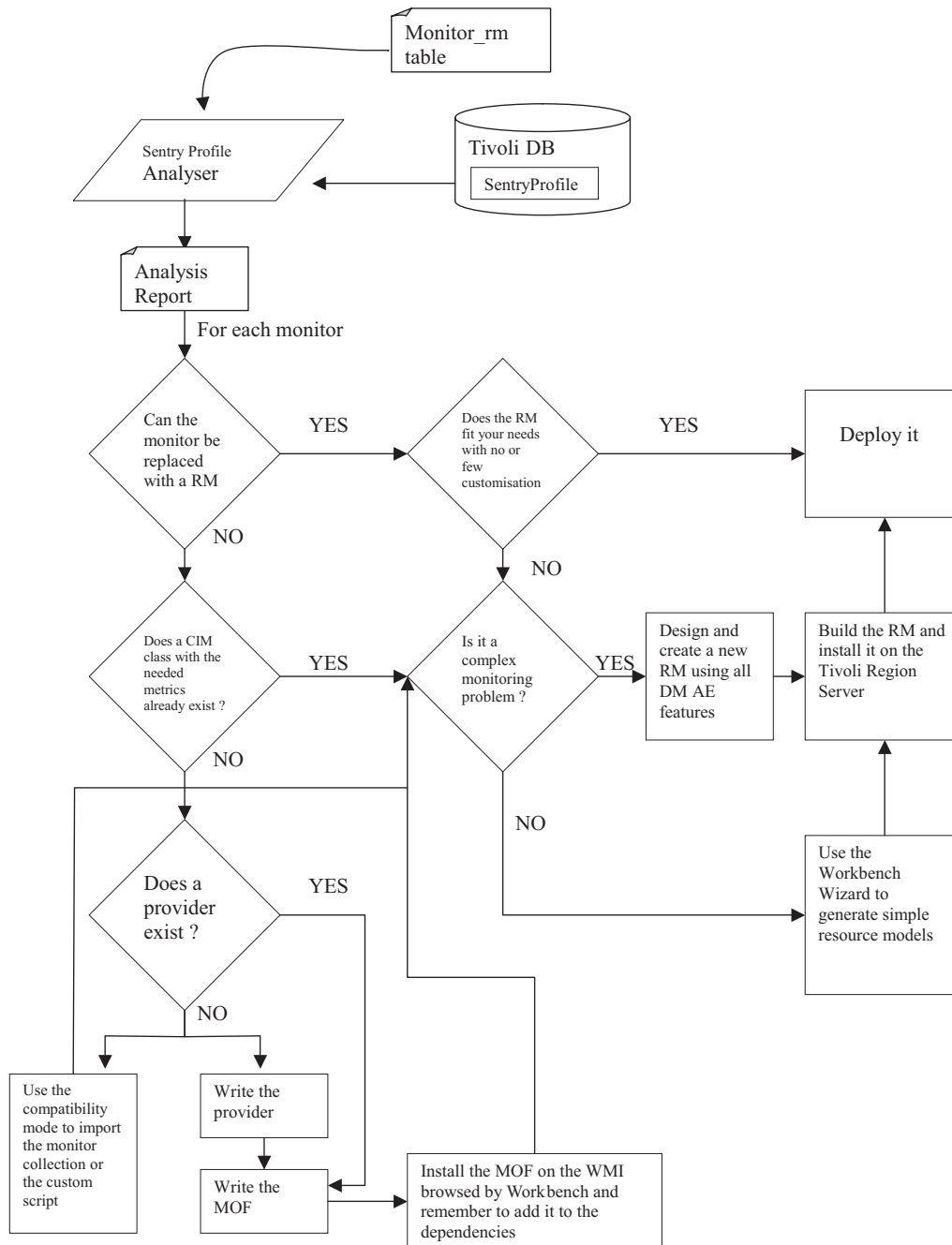
Obviously the generated code will take into account the Tivoli Distributed Monitoring (Advanced Edition) features described in earlier sections so it will be optimized to fit into the new monitoring paradigm. The wizard process is described in detail in Chapter 3, “Creating a Resource Model” on page 19.

How Everything Fits Together

The following picture summarizes the migration approach that new Tivoli Distributed Monitoring (Advanced Edition) users should follow after taking into consideration what has been described above:

1. The Sentry Analyzer script analyzes the Sentry Profile based on a monitor-resource model mapping table
2. The Sentry Analyzer script generates a report
3. For each monitor in the report the user can:
 - a. If a resource model collecting the related metric exists:
 - 1) If the resource model addresses the monitoring problems controlled by the monitor, customize the resource model and deploy it
 - 2) If the resource model does not address the monitoring problems controlled by the monitor, go to step 4.
 - b. If a resource model collecting the related metric does not exist, but a CIM class and its provider do, then go to step 4.
 - c. If a resource model collecting the related metric does not exist and neither does a CIM class, but a provider does (usually this happens for WMI performance data providers):
 - 1) Write the MOF
 - 2) Then go to step 4.
 - d. If a resource model collecting the related metric does not exist, and nor do a CIM class nor a provider:
 - 1) Write the provider
 - 2) Write the MOF as indicated, OR,
 - 3) Use the compatibility mode
 - 4) Then go to step 4.
4. If the monitoring problem is complex and needs correlation and aggregation:
 - a. Create a new resource model or modify an existing one using all features provided by Tivoli Distributed Monitoring (Advanced Edition)
 - b. Go to step 6.
5. If the monitoring problem is a simple one that follows the Tivoli Distributed Monitoring (Classic Edition) monitoring paradigm, then use the wizard process in the workbench to create a new resource model

- Version 4.1



3

Creating a Resource Model

This chapter describes new procedures to create a resource model.

In particular, it explains how to import monitoring sources created with Tivoli Distributed Monitoring (Classic Edition) into a resource model. Only synchronous monitors can be imported; asynchronous monitors cannot.

In addition, it describes the procedure for importing a customized script into a new or an existing resource model.

Importing a Monitoring Source with the Wizard

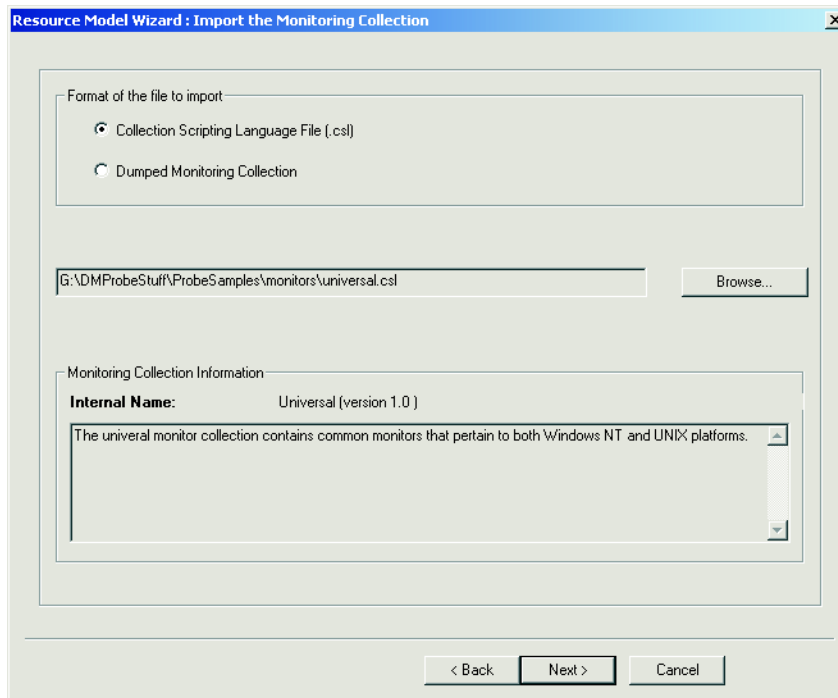
To import a monitoring collection created with Tivoli Distributed Monitoring (Classic Edition), perform the following steps:

1. Open the **File** menu and select **New** to create a new resource model.
2. From the displayed dialog, select the language of your resource model and click **OK**.
3. From the New Resource Model Workspace, select **Resource Model Wizard** and click **OK**.

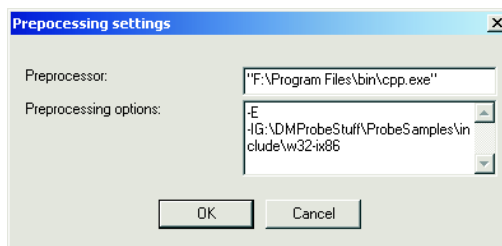
The Select data source type dialog opens:



4. Select **Distributed Monitoring (Classic Edition) collection** and click **Next**.
The Import the Monitoring Collection dialog opens.



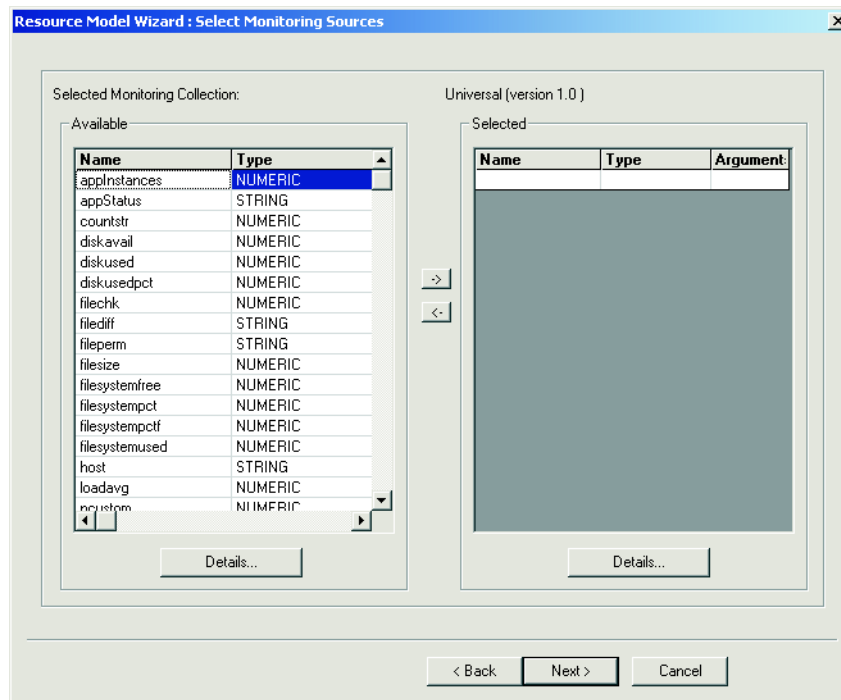
5. Select the required file format to import and browse to the file location.
If you are importing a .csl file, specify the preprocessing settings in the displayed dialog:



- a. In the **Preprocessor** text-box, specify the preprocessor you want to use to resolve the preprocessing guidelines used in the .csl file. The specified preprocessor must be in the system path. The workbench automatically installs a default preprocessor and points to it.
- b. In the **Preprocessing options** text-box, specify the same options used by the mcs1 command with Tivoli Distributed Monitoring (Classic Edition).
These options are passed verbatim to the C preprocessor. In addition to these preprocessor arguments, the workbench also passes the value of the MCSLCPPARGS environment variable. Arguments from the environment variable follow on the preprocessor command line those arguments that are given explicitly as preprocessor options. For more information, see the *Tivoli Distributed Monitoring User's Guide, Version 3.6.2*.
- c. Click **OK**, and, if prompted, save the preprocessing options so that they are available next time you import a .csl file.

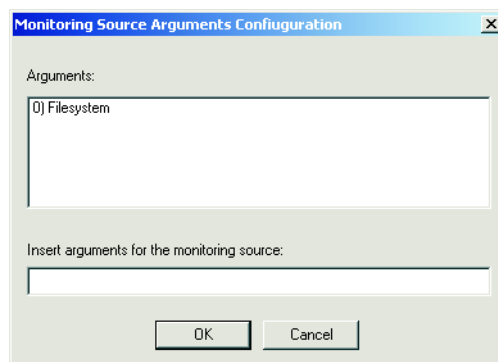
After the operation is completed successfully, some information about the selected monitoring collection appears in the displayed dialog.

6. Click **Next**. The Select Monitoring Sources dialog opens.



7. To see the details about a particular monitoring source, select it in the **Available** list and click **Details**.
8. From the **Available** list, double-click the monitoring sources you want to import in your resource model. This list displays all the sources that belong to the selected monitoring collection. You can import either all or a subset of them.

If you select a source that has arguments, the Monitoring Source Argument Configuration dialog opens.



9. The **Arguments** text box displays a list of all the mandatory arguments to be specified for the selected source.
10. In the **Insert arguments for the monitoring source** text box, specify the required arguments, separated by blanks.

The arguments you specify here identify the monitored resource. They are included in the resource model structure as parameters of the resource model itself, therefore they

represent instances of the monitored resource. So, you can modify a resource model at a later time, adding new parameters to it to monitor more instances of the same resource.

11. Click **Next** and follow the remaining dialogs of the wizard to configure triggering conditions, logging, and cycle time. These steps are the same as those described in the resource model wizard creation, in the *Tivoli Distributed Monitoring Workbench User's Guide, Version 4.1*.

When you have completed the procedure, the workbench 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.

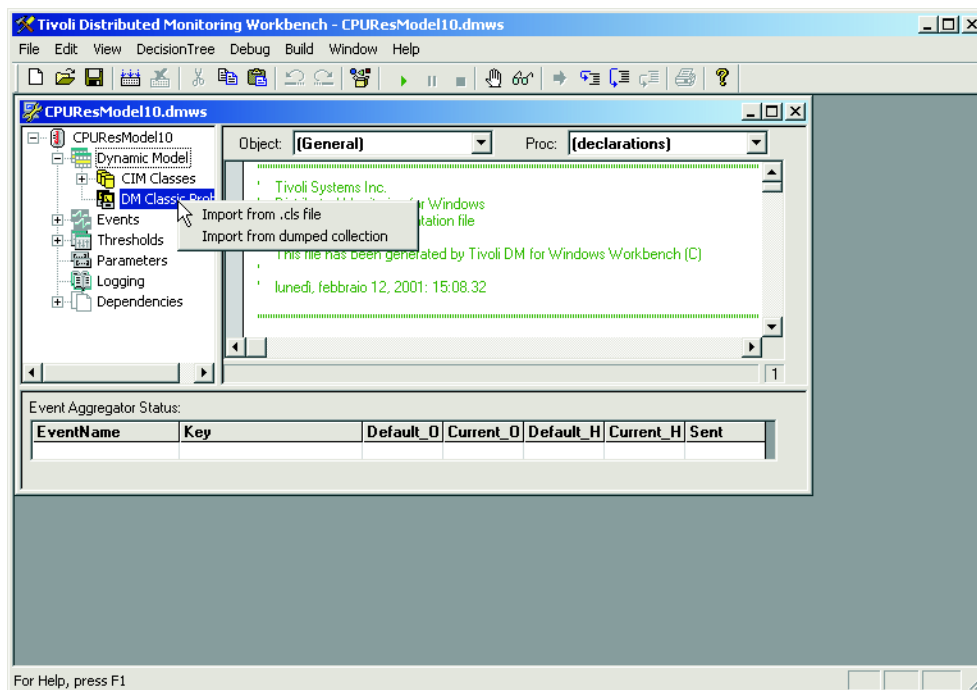
Importing a Monitoring Source Manually

You can import a monitoring source (or a whole monitoring collection) also in a resource model that already exists, opening the resource model tree structure. You can also import, in the same resource model, monitoring sources that belong to different monitoring collections.

In this procedure, only the definitions of the imported sources are included in the decision tree script automatically. However, you must write the monitoring algorithm that governs the resource model.

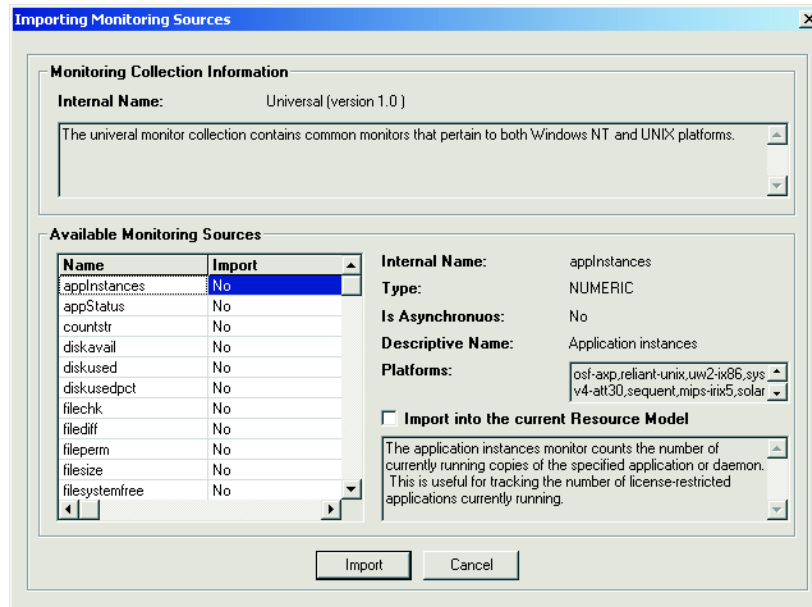
To import a monitoring source into a resource model, perform the following steps:

1. From the workbench main dialog, open the resource model tree structure.
2. Under the **Dynamic Model** element, right-click **DM Classic Probes**, and select the required file format:



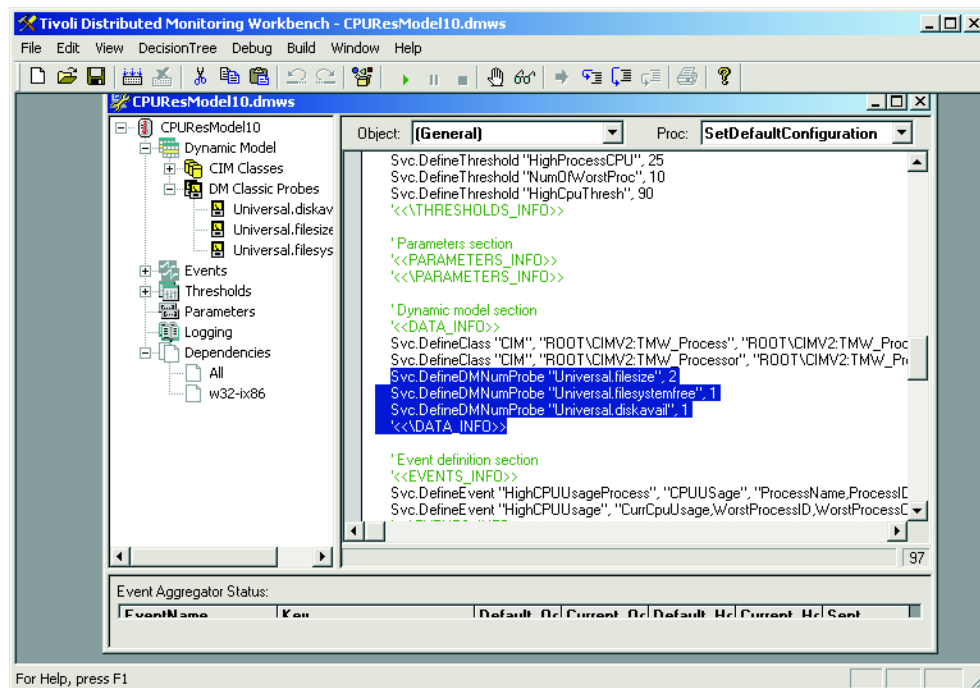
3. Perform the procedure described in step 5 on page 20.

4. In the Importing Monitoring Sources dialog, from the **Available Monitoring Sources** list, double-click the sources you want to import, switching **No** to **Yes**.



5. Select the required sources, and click **Import**.

In the decision tree script, the definitions of the imported sources are automatically specified under the dynamic model section, as shown in the following window:



The selected monitoring sources are automatically added to the resource model. Now, you need only to write the monitoring algorithm in the decision tree script, as explained in the *Tivoli Distributed Monitoring Workbench User's Guide, Version 4.1*.

Importing a Custom Script

Now that it is more fully integrated with Tivoli Distributed Monitoring (Classic Edition), the workbench enables you to use resource models to launch shell commands or scripts and retrieve the output.

You need only to include in the resource model the commands or the scripts you want to launch, and they will be embedded in the resource model, together with its other components.

To Launch a Shell Command Using the Wizard

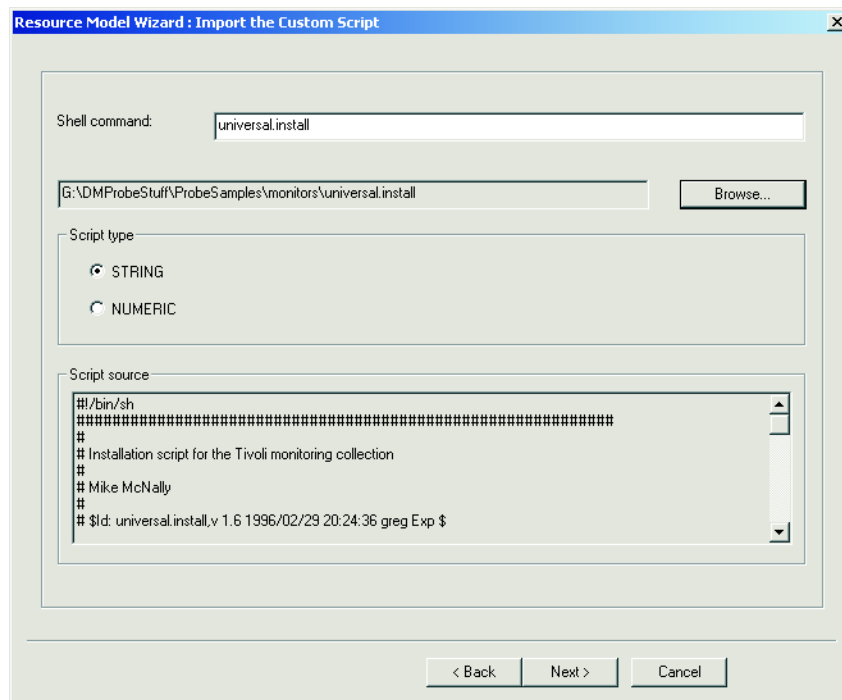
To include a shell command in a resource model, perform the following steps:

1. Open the **File** menu and select **New** to create a new resource model.
2. From the New dialog, select the language of your resource model and click **OK**.
3. From the New Resource Model Workspace dialog, select **Resource Model Wizard** and click **OK**.

The Select data source type dialog opens:



4. Select **Custom Script**. The Import the Custom Script dialog opens.



5. In the **Shell command** text box, type the command you want to launch when you run this resource model. This could be, for example, a command to determine whether a file is contained in the specified directory.
6. In the **Script type** group-box, select the kind of output you expect from the command.
7. Click **Next** and follow the remaining dialogs of the wizard to configure triggering conditions, logging, and cycle time. These steps are the same that are described in the resource model wizard creation, in the *Tivoli Distributed Monitoring Workbench User's Guide, Version 4.1*

In the triggering dialog, for example, you can specify that you want an event to be generated if the command output is equal to the given string, that is, if the file is found in the directory.

To Import a Script with the Wizard

With the workbench, you can also import a custom script (like a bash, or perl script) into a resource mode, so that the script is launched when you run the resource model.

To import a custom script, follow the procedure described in “To Launch a Shell Command Using the Wizard” on page 24, from step 1, to step 3.

From the Import the Custom Script dialog, browse to the file that contains your script.

The workbench automatically imports that file and adds it to the dependencies of the resource model. Therefore, the file containing the script is automatically installed on the endpoint, together with the resource model.

Customizing the Event Messages

When you create a resource model, you can specify that the generated events be notified to Tivoli Enterprise Console server, and to the Tivoli Business Systems Manager. The event notifications come with messages that specify the generated event. These messages are generic, but the workbench now enables you to customize them and make them more detailed and specific.

Previously, the event notification messages did not identify specifically the single events, but gave general information that indicated that an event had occurred. Now, when you define an event, you can associate its notification message to one or more of the event attributes, so that the message specifies more in detail what kind of event has occurred.

To customize an event notification message, perform the following:

- Open the Events dialog of your resource model.
- In the **Message** text box type a message, specifying one or more event attributes, between @ symbols. For example: "This process @ProcessName@, @ProcessID@ is consuming too much CPU."

When the event is generated, the corresponding message is displayed identifying univocally the process through the process name and ID.

4

Service Object Method Library

This chapter, describes the Class Object library. This library contains the syntax required to express all the main functions for creating an object in your resource model. It also contains a list of possible errors users can get when they run a resource model.

The following methods enable compatibility between Tivoli Distributed Monitoring (Classic Edition) monitoring collections and the Tivoli Distributed Monitoring (Advanced Edition) application.

Basic Object Methods

This section describes newly implemented basic methods for the TMWService object. You can refer to it for the syntax required for calling the specific methods.

Dynamic Model

The following methods help you to express functions for dynamic model configuration.

Method CallDMNumProbe

Syntax

```
Object.CallDMNumProbe(ProbeKey As String, Arguments As String) As Double
```

Parameters

Probekey

A key that uniquely identifies the monitoring source; 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 is generated.

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.CallDMNumProbe(ProbeKey As String, Arguments As String) As Double

Parameters

Probekey

A key that uniquely identifies the monitoring source; 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 is generated.

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.

Events

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 to 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_SPAWN_EVENT_FAILED
 TMWSERVICE_E_EVENT_NOT_DECLARED
 TMWSERVICE_E_EVENT_PROP_NOT_FOUND
 TMWSERVICE_E_EVENT_INDICATION_FAILED S_OK.

Logging**Method LogInstEx****Syntax**

Object.LogInstEx(contextName As String, Resource As String, mapHndl As Integer)

Parameters**context**

The logging context.

pResource

The resource that the attributes refer to.

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 DefineLogInst in the SetDefaultConfiguration Subroutine. The mapping table associated to 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**Method Shell****Syntax**

Object.Shell(shell As String) As String

Parameters

shell The command line to launch the new process.

Returns

The standard output of the launched process.

Description

The Shell method is used to run a new program on the monitored host and to retrieve its output.

Remarks

It is possible to launch customized scripts that are available on the endpoint, by issuing, for example,

```
sh -c myscript.sh
```

The launched process must return within 60 seconds, otherwise an error is generated.

Error codes

TMWSERVICE_E_NO_INTERP_SUPPORT TMWSERVICE_E_NO_DATA

TMWSERVICE_E_IMPLIED_ERROR

TMWSERVICE_E_ERRORVALUE_SCRIPT_ERROR S_OK.

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 to 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 to 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 to the given key.

Description

This method retrieves the value associated to 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 to the given key.

Description

This method retrieves the value associated to the key in the mapping table linked to 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 to 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 to 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 to 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 a the mapping table associated to the handle hndl.

Error codes

TMWSERVICE_E_MAP_HANDLE_NOT_FOUND S_OK.

Advanced Object Methods

This section contains a description of the advanced methods of the TMWService object. You should never call these methods directly, because the calls to these methods are always generated by the workbench.

Dynamic Model

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.

Deprecated Methods

Some methods used with Tivoli Distributed Monitoring (Advanced Edition), Version 4.1 are being replaced by new ones.

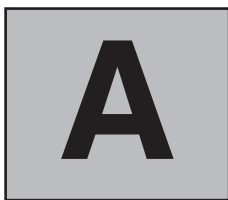
The SendEvent method is deprecated. To send an event, use the SendEventEx method.

The LogInst **method** is deprecated. To log the attributes of a resource, use the LogInstEx method, instead.

Exceptions

This table contains a list of exceptions you might encounter, with the corresponding message.

Exceptions	Description	Remarks
0x80041024	TMWSERVICE_E_PROBE_NOT_FOUND	The Tivoli Distributed Monitoring (Classic Edition) monitoring source has not been defined.
0x80041025	TMWSERVICE_E_PROBE_NOT_LOAD	The Tivoli Distributed Monitoring (Classic Edition) monitoring source implementation is not available.
0x80041026	TMWSERVICE_E_NO_INTERP_SUPPORT	The Tivoli Distributed Monitoring (Classic Edition) monitoring source implementation is not available for the given platform.
0x80041027	TMWSERVICE_E_NO_DATA	The Tivoli Distributed Monitoring (Classic Edition) monitoring source implementation has returned an empty output.
0x80041028	TMWSERVICE_E_IMPLIED_ERROR	The Tivoli Distributed Monitoring (Classic Edition) monitoring source implementation is getting an error due to bad arguments.
0x80041029	TMWSERVICE_E_ERRORVALUE_SCRIPT_ERROR	The Tivoli Distributed Monitoring (Classic Edition) monitoring source implementation contains an internal script error.
0x8004102a	TMWSERVICE_E_PROBE_WRONG_ARGS_NUM	A wrong number of arguments has been specified in a call to a Tivoli Distributed Monitoring (Classic Edition) Monitoring source.
0x8004102b	TMWSERVICE_E_MAP_HANDLE_NOT_FOUND	The handle of the mapping table is invalid.
0x8004102c	TMWSERVICE_E_MAP_KEY_NOT_FOUND	The given key is not contained in the mapping table associated to the given handle.
0x8004102d	TMWSERVICE_E_EVENT_PROP_NOT_FOUND	A property required for sending an event or for logging a call is not contained in the mapping table associated to the given handle.



Error Messages

This appendix explains the messages that can you can get when creating or debugging a resource model with the workbench.

The messages are listed in ascending numeric order.

Identifying a Message

Messages are of different types but are all identified in the same way. The following example shows a typical message and explains its identifying components.

Identity	Message
AMW0500E	The Action Browser is unable to get the class definition.

AMW This prefix identifies the message as belonging to the workbench.

0XXX The unique serial number of the message.

E Is the type of message and can be:

- I** **Information messages** provide feedback about something that has happened in the product or system that may be important. These messages also give guidance when you are requesting a specific action from the product.
- W** **Warning messages** call your attention to an exception condition that is not necessarily an error but may cause problems if not attended to.
- E** **Error messages** indicate that an action cannot be completed because of a user or system error. These error messages always require user response.

Notation

Some messages, especially information and warning messages, are multi-purpose. The same basic text can contain different strings such as different command names or application names, according to the way the application was behaving when the message was generated. These messages are shown in the following sections with the string identity displayed in *italics* at the appropriate part of the message.

Messages

The following messages can be displayed.

AMW0643I It is not possible to remove the class property.

Explanation: The specified property could not be removed, because it is a key for the selected class.

User Response: Do not remove that property.

AMW0644E The monitoring collection cannot be imported.

Explanation: The monitoring collection cannot be imported from the file *FileName* due to the following error:

User Response: Make sure that the preprocessor and the preprocessing options are correctly set. Ensure that the *Path* environment variable contains the folder where the preprocessor is located. Check also if the .csl file contains errors.

AMW0645E Asynchronous monitors cannot be imported.

Explanation: You have tried to import an asynchronous monitor source. The workbench does not asynchronous monitors to be imported.

User Response: Import a synchronous monitor.

AMW0646W The monitor *MonitorName* is not available for the platform *Platform*.

Explanation: The monitor *MonitorName* is not available for the platform *Platform*, therefore it cannot be used by this resource model when running on the given platform. Anyway the package building will proceed.

User Response: Do not use the monitor *MonitorName* on the platform *Platform*, or import a monitor source that supports the given platform.

AMW0647E Probe *ProbeName* collection test failed.

Explanation: Error *ErrorDescription* occurred during probe collection test.

User Response: Ensure that the probe is available on w32-ix86.



Printed in the United States of America
on recycled paper containing 10%
recovered post-consumer fiber.