Tivoli Application Performance Management

User's Guide

Version 2.1

GC32-0415-04
## Installation Locations
- What Is on Your Install Compact Disk
- Installation and Migration Methods
- Installations
  - Installing Tivoli Application Performance Management
  - Installing the Web GUI
  - Installing the HTTP Server
  - Installing Support for EPP
  - Installing Language Support for the Product
  - Installing Monitors for Distributed Monitoring
- Migrating from Version 2.0.1 of the Product
  - What You Need to Carry Out the Migration
  - Migration Patch Software Requirements
  - Migration Processes
- Uninstallation
  - Uninstalling Tivoli Application Performance Management
  - Uninstalling the Web GUI
  - Uninstalling Support for EPP
  - Uninstalling Monitors for Distributed Monitoring

## Chapter 3. Getting Started
- Adding the Notice Group
- Preparing Applications
- Registration
- Creating a Profile Manager and Profiles
- Adding MarProfile as a Managed Resource
- Building and Distributing Profiles
- Setting up Monitors
- Configuring the Web GUI
- Changing Ports

## Chapter 4. Planning What to Measure
- Measurements Used
- Improving Performance
- Deciding on a Method for Measuring

## Chapter 5. Client Capture
- About Client Capture
- Using Client Capture
Syntax Rules .............................................................. 49
Ready-Made Configuration Files ................................. 49
Example 1 (sample.xml) ........................................... 50
Example 2 (advanced_sample.xml) ............................. 50

Chapter 6. Transaction Simulation ................................. 53
  End-to-end Probe Platform ....................................... 53
  Description of the Supplied Probes ......................... 54
  Using the Probes .................................................. 57
  Simulation using Mercury Interactive Scripts ............... 58
  Virtual User Generator .......................................... 58
  Preparing Virtual User Scripts .................................. 59

Chapter 7. Instrumenting an Application for ARM .................. 61
  About ARM .......................................................... 61
  ARM API Call Syntax ........................................... 61
  Return Codes ..................................................... 63
  Case Sensitivity ................................................... 63
  Using the ARM Calls .............................................. 63

Chapter 8. Using ARM Transaction Correlators ..................... 65
  What is ARM Transaction Correlation? ....................... 65
  How Correlation Works ......................................... 66
    Multi-tier Transactions ....................................... 67
  Where to use Correlation ...................................... 68
  Collecting Correlators with Tivoli Application Performance Management .................... 68

Chapter 9. Registering Applications ................................. 71
  Registering Client Capture Applications .................... 72
  Registering Instrumented Applications ...................... 72
  Registering EPP Probe Applications .......................... 73
  Registering Virtual User Applications ....................... 74
    Registering VuGen Scripts ................................... 74
  Advanced Use of the wmarregapp Command .................. 75
    Modifying an Application Definition ....................... 75
  Removing Applications from the Repository ................. 75

Chapter 10. Building Profiles ........................................ 77
  Tivoli Application Performance Management Profiles ....... 77
Chapter 13. Using Performance Data ................................. 133

Appendix A. If You Have Problems ................................. 141

Appendix B. Command Reference ................................. 181
Appendix C. Accessibility ...................................................... 207
   Using Assistive Technologies ............................................. 207
   Keyboard Navigation of the User Interface ............................ 207
   Magnifying What is Displayed on the Screen ....................... 208
   Documentation in Accessible Formats ................................. 208
   Using Alternative Text ..................................................... 208

Appendix D. Authorization Roles .............................................. 209
   Setting Up Profiles ....................................................... 209
   Performing Operations ................................................... 209
   Performing Commands .................................................... 209

Glossary ................................................................................. 211

Index .................................................................................... 223
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Simple representation of Tivoli Application Performance Management</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>Diagram of data flow</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>Functional block diagram</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>Tivoli Application Performance Management Network</td>
<td>5</td>
</tr>
<tr>
<td>5</td>
<td>Choosing a method</td>
<td>7</td>
</tr>
<tr>
<td>6</td>
<td>Client Capture block diagram</td>
<td>7</td>
</tr>
<tr>
<td>7</td>
<td>An ARM API procedure</td>
<td>8</td>
</tr>
<tr>
<td>8</td>
<td>How VuGen works</td>
<td>9</td>
</tr>
<tr>
<td>9</td>
<td>What is in a profile</td>
<td>11</td>
</tr>
<tr>
<td>10</td>
<td>Typical data display</td>
<td>11</td>
</tr>
<tr>
<td>11</td>
<td>Installing the product - first action</td>
<td>16</td>
</tr>
<tr>
<td>12</td>
<td>Installation - selecting product to install</td>
<td>17</td>
</tr>
<tr>
<td>13</td>
<td>Installation - progress dialog</td>
<td>18</td>
</tr>
<tr>
<td>14</td>
<td>Installation - new icon added to confirm installation</td>
<td>19</td>
</tr>
<tr>
<td>15</td>
<td>Installing Web GUI - selecting Web GUI to install</td>
<td>20</td>
</tr>
<tr>
<td>16</td>
<td>Installing Web GUI - selecting install option</td>
<td>21</td>
</tr>
<tr>
<td>17</td>
<td>Installing Web GUI - progress dialog</td>
<td>22</td>
</tr>
<tr>
<td>18</td>
<td>Installation choices for language support</td>
<td>25</td>
</tr>
<tr>
<td>19</td>
<td>Installing a patch - first screen</td>
<td>29</td>
</tr>
<tr>
<td>20</td>
<td>Installing a patch - setting the install path</td>
<td>29</td>
</tr>
<tr>
<td>21</td>
<td>Installing a patch - confirmation that the patch is installed</td>
<td>30</td>
</tr>
<tr>
<td>22</td>
<td>Upgrade language support patch options</td>
<td>32</td>
</tr>
<tr>
<td>23</td>
<td>Adding notes group - set notice groups dialog</td>
<td>38</td>
</tr>
<tr>
<td>24</td>
<td>Adding notes group - specifying the notice group</td>
<td>38</td>
</tr>
<tr>
<td>25</td>
<td>Adding notes group - notice board dialog</td>
<td>39</td>
</tr>
<tr>
<td>26</td>
<td>Create profile manager dialog</td>
<td>40</td>
</tr>
<tr>
<td>27</td>
<td>Adding the product as a managed resource</td>
<td>40</td>
</tr>
<tr>
<td>28</td>
<td>Supplying the network user password</td>
<td>42</td>
</tr>
<tr>
<td>29</td>
<td>Typical response time display</td>
<td>44</td>
</tr>
<tr>
<td>30</td>
<td>EPP information gathering</td>
<td>54</td>
</tr>
<tr>
<td>31</td>
<td>How VuGen works</td>
<td>59</td>
</tr>
<tr>
<td>32</td>
<td>ARM transaction example</td>
<td>64</td>
</tr>
<tr>
<td>33</td>
<td>Example of a distributed transaction</td>
<td>66</td>
</tr>
<tr>
<td>34</td>
<td>Generating a correlator</td>
<td>66</td>
</tr>
<tr>
<td>35</td>
<td>Passing the correlator</td>
<td>67</td>
</tr>
<tr>
<td>36</td>
<td>Network correlation</td>
<td>68</td>
</tr>
<tr>
<td>37</td>
<td>Deselecting correlation</td>
<td>69</td>
</tr>
<tr>
<td>38</td>
<td>Registration</td>
<td>71</td>
</tr>
<tr>
<td>39</td>
<td>Selecting a region</td>
<td>79</td>
</tr>
<tr>
<td>40</td>
<td>Selecting a profile manager</td>
<td>79</td>
</tr>
<tr>
<td>41</td>
<td>Creating a profile, first action</td>
<td>80</td>
</tr>
<tr>
<td>42</td>
<td>Creating a profile dialog</td>
<td>80</td>
</tr>
<tr>
<td>43</td>
<td>Creating a profile, new icon added</td>
<td>81</td>
</tr>
<tr>
<td>44</td>
<td>Starting the product</td>
<td>82</td>
</tr>
<tr>
<td>45</td>
<td>Add Entry to Profile dialog</td>
<td>82</td>
</tr>
<tr>
<td>46</td>
<td>Updating the list of available applications</td>
<td>83</td>
</tr>
<tr>
<td>47</td>
<td>Creating a transaction filter</td>
<td>84</td>
</tr>
<tr>
<td>48</td>
<td>Linear graph of bucket results</td>
<td>85</td>
</tr>
<tr>
<td>49</td>
<td>Setting bucket limits</td>
<td>86</td>
</tr>
</tbody>
</table>
50. Adding schedule information, main dialog .......................................................... 87
51. Setting the schedule period ................................................................................. 88
52. Setting up a collection rule .................................................................................. 89
53. Naming a collection rule ...................................................................................... 89
54. Setting activity days ............................................................................................ 90
55. Setting start and stop times ................................................................................ 90
56. Specifying data records to send to database ...................................................... 91
57. Adding EPP extended values information .......................................................... 93
58. Adding virtual user simulation information ....................................................... 94
59. Edit Entry in the Profile dialog ........................................................................... 95
60. Example of a commented and disabled entry ..................................................... 96
61. Entering server details ......................................................................................... 97
62. Web GUI opening splash screen .......................................................................... 98
63. Supplying a user id and password ....................................................................... 98
64. Web GUI main screen ........................................................................................ 99
65. Choosing a date .................................................................................................. 99
66. List of applications display .................................................................................. 100
67. Choosing a transaction ......................................................................................... 100
68. Selecting the data to display ............................................................................... 100
69. Displayed data .................................................................................................... 101
70. Web GUI status bar .............................................................................................. 101
71. RIM block diagram .............................................................................................. 104
72. Database configuration screen ............................................................................. 105
73. Choosing the managed node for the database .................................................... 106
74. Setting the password ........................................................................................... 106
75. Relative size of collected data ............................................................................. 113
76. Database schema .................................................................................................. 114
77. Data collection ..................................................................................................... 137
78. Data upload .......................................................................................................... 138
79. Tivoli Decision Support ....................................................................................... 138
Preface

This user’s guide describes how to use Tivoli® application performance management (referred to as the product throughout this User’s Guide) to monitor the performance of applications running on a network. It explains how to run specific performance tests and how to collect and store performance data for further use. The guide enables you to install the product on the platforms for which it is intended, configure it on your workstation or network, and prepare and perform test scenarios on your applications.

Who Should Read This Guide

This book is intended for system administrators, performance analysts, and programmers, who need to install the product and configure it, or need to perform the daily system management tasks. Those tasks include setting up and arranging monitoring scenarios for selected applications that are running in your network environment.

Users of this guide should be familiar with the Tivoli desktop system, which encompasses the main network management services and utilities compatible with the product. Users of this guide should also be familiar with the following:

- The UNIX® or Microsoft® Windows® operating systems.
- Network systems and the Tivoli Management Framework environment.

What This Book Contains

This Tivoli Application Performance Management User’s Guide contains the following sections:

- Chapter 1, “Introduction” on page 1
  Provides a general introduction to the product.
- Chapter 2, “Installation and Migration” on page 13
  Identifies the contents of the installation CD-ROM and describes the installation tasks that must be performed.
- Chapter 3, “Getting Started” on page 37
  Provides information on tasks that need to be completed before the product is ready for normal use.
- Chapter 4, “Planning What to Measure” on page 43
  Explains what to monitor to provide performance data and how to choose which method for preparing your application to be run and monitored by the product at an endpoint.
- Chapter 5, “Client Capture” on page 47
  Describes the Client Capture facility of the product and how to use it.
- Chapter 6, “Transaction Simulation” on page 53
  Describes how you can use End-to-end probes to test applications by emulating end user actions or how to use transaction simulations that have been prepared and instrumented using the Mercury Interactive LoadRunner suite of testing applications (MS Windows only) to provide performance data.
- Chapter 7, “Instrumenting an Application for ARM” on page 61
Contains a description of how to insert Application Response Measurement calls into application code so that the product can monitor the performance of that application while it is running on an endpoint.

- **Chapter 8** [Chapter 8, “Using ARM Transaction Correlators” on page 65](#)
  Describes how you can use transaction correlation to provide comparisons between performance data at various endpoints.

- **Chapter 9** [Chapter 9, “Registering Applications” on page 71](#)
  Describes the methods of identifying applications that can be used with the product and how to register them in the repository.

- **Chapter 10** [Chapter 10, “Building Profiles” on page 77](#)
  Describes how you can build a profile, include data collection instructions and distribute it to nominated network endpoints.

- **Chapter 11** [Chapter 11, “The Web GUI” on page 97](#)
  Describes the Web GUI tool for the product and how to gather real-time data on applications or transactions.

- **Chapter 12** [Chapter 12, “Setting up the Database” on page 103](#)
  Explains how to set up the database where measurements data is to be stored.

- **Chapter 13** [Chapter 13, “Using Performance Data” on page 133](#)
  Describes how measurement data is retrieved, moved, stored in a database, and how you can use the Tivoli Decision Support application to sort and display the data.

- **Appendix A** [Appendix A, “If You Have Problems” on page 141](#)
  Describes how to prepare and apply traces, and messages that the product issues. Descriptions of error messages are accompanied by probable cause information and advice on action to take in response.

- **Appendix B** [Appendix B, “Command Reference” on page 181](#)
  Provides a description of the command line interface (CLI) commands, which you can use as an alternative to the GUI.

- **Appendix C** [Appendix C, “Accessibility” on page 207](#)
  Describes the accessibility features in the product documentation, that are provided to assist users who have disabilities.

- **Appendix D** [Appendix D, “Authorization Roles” on page 209](#)
  Contains tables that describe the roles you need to use the product and to perform system administration operations from within the Tivoli environment.

- **Glossary**
  Provides explanation of the terms used in this documentation.

## Publications

This section lists publications in the *Tivoli Application Performance Management* library and any other related documents. It also describes how to access Tivoli publications online, how to order Tivoli publications, and how to make comments on Tivoli publications.
Tivoli Applications Performance Management Library

The following documents are available in the Tivoli Applications Performance Management library:

  Provides information about the product and how to use it.

  Provides specific product information to support the Tivoli Decision Support guide and describes how to use the query facility to access database information from the product.

- Tivoli Applications Performance Management Release Notes, GI10-9260
  Provides late-breaking information about the product and information about known defects, accompanied with work around methods.

Prerequisite Publications

To be able to use the information in this book effectively, you must have some prerequisite knowledge, which you can get from the following books:

- Tivoli Management Framework User’s Guide, GC31-8433
  Provides information about Tivoli desktop, the product used to manage your network and the base product under which Tivoli Application Performance Management runs.

- Tivoli Management Framework Reference, SC31-8434
  Describes the commands used by programmers and administrators used with Tivoli framework.

Related Publications

Tivoli provides the following related documents:

- Tivoli Decision Support for Application Performance User’s Guide.
  Provides specific product information to support the Tivoli Decision Support guide and describes how to use the query facility to access database information from the product.

- Online documentation for Application Response Measurement (ARM), available for downloading from [http://www.cmg.org/regions/cmgarmw](http://www.cmg.org/regions/cmgarmw)
  Describes how to insert ARM instructions into the execution code of an application you want to monitor.

- Software Installation Service User’s Guide
  Describes how to use the Software Installation Service (SIS) to install Tivoli products.

All the documents listed above are available in portable document format (PDF) from the Tivoli customer support web site.

Accessing Publications Online

You can access many Tivoli publications online using the Tivoli Information Center, which is available on the Tivoli Customer Support Web site:


These publications are available in PDF or HTML format, or both. Translated documents are also available for some products.
Ordering Publications

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

http://www.ibm.com/shop/publications/order

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, for a list of telephone numbers, see the following Web site:
  http://www.tivoli.com/inside/store/lit_order.html

Providing Feedback about Publications

We are very interested in hearing about your experience with Tivoli products and documentation, and we welcome your suggestions for improvements. If you have comments or suggestions about our products and documentation, contact us in one of the following ways:

- Send an e-mail to pubs@tivoli.com.
- Complete our customer feedback survey at the following Web site:
  http://www.tivoli.com/support/survey/

Contacting Customer Support

If you have a problem with any Tivoli product, you can contact Tivoli Customer Support. See the Tivoli Customer Support Handbook at the following Web site:

http://www.tivoli.com/support/handbook/

The handbook provides information about how to contact Tivoli Customer 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 you are in
- What information you should gather before contacting support

Conventions Used in This Book

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

Typeface Conventions

The following typeface conventions are used in this book:

**Bold**

Lowercase and mixed-case commands, command options, and flags that appear within text appear like this, in **bold** type.

Graphical user interface elements (except for titles of windows and dialogs) and names of keys also appear like this, in **bold** type.

*Italic*

Variables, values you must provide, new terms, and words and phrases that are emphasized appear like *this*, in *italic* type.

**Monospace**

Commands, command options, and flags that appear on a separate line, code examples, output, and message text appear like this, in monospace type.
Names of files and directories, text strings you must type, when they appear within text, names of Java™ methods and classes, and HTML and XML tags also appear like this, in monospace type.

**Operating System-dependent Variables and Paths**

This book 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.

**Margin Icons**

Some procedures in this guide include icons in the left margin. These icons provide context for performing a step within a procedure. For example, if you have to perform a step in a procedure by double-clicking a policy region icon, that icon appears in the left margin next to the step.
Introduction

This chapter provides a brief description of Tivoli Application Performance Management (the product), what it is and how it is used. Information in this chapter is organized as follows:

- **About the Product**
- **The Component Parts**
- **The Product Environment**
- **Performance Monitoring**
- **How the Product is Used**
- **Starting Work with the Product**
- **Performance Analysis and Reporting**

### About the Product

Tivoli Application Performance Management captures performance data on applications that are running at network endpoints. It gathers data on the response times of client-server applications, while those applications are running and carrying out defined tasks.

A simplistic view of the product is that it is implemented by several tools used to build application simulations and tests, to provide timing data obtained by a sophisticated set of timers and schedules. The data obtained can be stored in a database and presented in many different ways.

![Simple representation of Tivoli Application Performance Management](image)

*Figure 1. Simple representation of Tivoli Application Performance Management*

The product can run and monitor many different applications at a large number of endpoints. The only possible restricting factor is the amount of data to move around the network.

To make such measurements, components of the product run pre-prepared applications or simulated applications. These applications are run to collection schedules and with settings...
for which type of data is to be collected. Applications and schedules are sent to each network endpoint where measurement is required.

You use the product to perform three types of activity:

1. You can perform profile management tasks by using the graphical user interface (GUI) or through the command line interface (CLI).

2. At endpoints you can measure the performance of client capture applications, ARM instrumented applications, simulation scripts and EPP simulation probes.

3. You can collect and present data. Data collections at each endpoint, detailing the progress and results of the application measurement activity at that endpoint, are made by the endpoint engine. Data is gathered for upload to the gateway, either on demand or at specified intervals. From the gateway the data is sent to the database. The database entries can be viewed and manipulated by using the Tivoli Decision Support for application performance. You can view current (todays) data collection activity by using the Web GUI.

The following product configurations on the network enable you to perform these three activities:

- The software application working directly as an element of the Tivoli management region server, enables you to set up and control all application monitoring activities.

- The client endpoints where instrumented applications, client capture applications and application simulations are run. Measurements of their performance are made and recorded under control of the endpoint engine.

- Database management where performance statistics are derived.

- Tivoli Distributed Monitoring on the server and endpoints. You can use Distributed Monitoring to create on-line monitors for certain performance data that is collected by the product.

Figure 1 shows the relationship between the product, the endpoint, the database, and Tivoli Decision Support.

![Figure 2. Diagram of data flow](image)
The Component Parts

Tivoli Application Performance Management components are interconnected as shown in Figure 2.

Note: The configurations shown here are typical examples.

1. The user interface (GUI) is integrated with the Tivoli desktop and can be launched and operated from any workstation of the region network that has the Tivoli desktop installed. The GUI is the main operator interface of the product. You can perform set-up and control operations for gathering performance data by using the GUI.

2. The Tivoli Management Region Server has a Tivoli Application Performance Management server component located within it, which accomplishes the operations that you perform through the graphical interface. In addition it puts into effect a Command Line Interface (CLI) to accomplish administrative tasks. The IBM® HTTP server handles interfacing between the Web GUI and the endpoint that is being examined.

Note: The product server and the IBM HTTP server may be installed on any managed node, or on different nodes within the region system. They are shown as part of the region server for convenience.

3. The Tivoli management gateway interfaces the product server with the endpoint workstations.

4. A mid-level manager resides on each gateway. It responds to data upload requests from the endpoints, stores the performance data from the endpoints, and notifies the controller that it is ready to send data to the database.
5. A product interface resides in each active remote endpoint where you want to manage the customer application. It depends on the services of the Tivoli management agent and receives requests from the customer application for passing on to the engine for action. It also passes communication and results data from the engine to the gateway and the management server.

6. The engine collects performance data from instrumented applications and simulations. To do this, the engine runs the applications and simulations, in accordance with associated configuration commands sent from the Tivoli management server. An engine resides on the same endpoint as the applications to be measured. The engine receives data directly from the application through the application response time measurement (ARM) interface. It gathers and aggregates data to produce information.

7. The library controls the ARM interface that enables the engine to collect performance data from the applications or virtual user scripts.

8. The client capture agent provides an interface for client capture between its files and the engine.

9. The EPP agent provides an interface to the engine to run tests on applications that emulate end user actions.

The Product Environment

The product runs as part of the Tivoli management environment desktop product, which is integrated with other Tivoli enterprise products, to provide an overall function for service-level management. Integration with the two other parts of Tivoli desktop detailed below, provide methods of measuring transaction availability and response times experienced by end users and helps detect service degradation.

You can use the following Tivoli framework products with the product:

**Tivoli Decision Support**

Adds an important element to the solution because historical data can be correlated and analyzed. Data can then be arranged and displayed to pinpoint offending resources, thereby improving availability even further.

**Tivoli Distributed Monitoring**

Enables you to act on any exceptions as they appear. Events are forwarded to the event console in the same way as other Distributed Monitoring components.

**Note:** To gain most from Tivoli Application Performance Management, you need to have a well-developed Tivoli management environment set-up and be familiar with its features.

An overview of a typical implementation of the product in the Tivoli environment is shown in Figure 3, followed by a description and illustration of a typical network arrangement.
The Tivoli Management Region is composed of the following:

**Tivoli server**
A server that holds or refers to the complete set of Tivoli software.

**Managed node**
Any managed resource on which the Tivoli Management Framework is installed. A managed node can connect to an endpoint via a gateway. The product and its server can reside on any managed node.

**Gateway**
Software running on a managed node that provides communication services to endpoints. The mid-level manager resides on a gateway.

**Endpoints**
Systems that are managed from a gateway. The product engine resides at endpoints.

A typical product node of a region, comprises a main server and one or more gateways with several associated endpoints. Additional features may be dedicated machines or endpoints, where various toolkits are installed for the development of applications and scripts before they are used. Some endpoints can be set aside for running simulations and others for client capture activities.

It is not mandatory that the network has a database but it is a recommended feature. Data can be retrieved from the endpoint and examined, but it is much more efficient to store data in a regular and predefined manner in a relational database. In this way, it can be used with Tivoli Decision Support. Because data transfer through the network can be time consuming, it is also more convenient to program your data movements to the database during ‘quiet periods’ of net activity.

---

Figure 4. Tivoli Application Performance Management Network

The Tivoli Management Region is composed of the following:

**Tivoli server**
A server that holds or refers to the complete set of Tivoli software.

**Managed node**
Any managed resource on which the Tivoli Management Framework is installed. A managed node can connect to an endpoint via a gateway. The product and its server can reside on any managed node.

**Gateway**
Software running on a managed node that provides communication services to endpoints. The mid-level manager resides on a gateway.

**Endpoints**
Systems that are managed from a gateway. The product engine resides at endpoints.

A typical product node of a region, comprises a main server and one or more gateways with several associated endpoints. Additional features may be dedicated machines or endpoints, where various toolkits are installed for the development of applications and scripts before they are used. Some endpoints can be set aside for running simulations and others for client capture activities.

It is not mandatory that the network has a database but it is a recommended feature. Data can be retrieved from the endpoint and examined, but it is much more efficient to store data in a regular and predefined manner in a relational database. In this way, it can be used with Tivoli Decision Support. Because data transfer through the network can be time consuming, it is also more convenient to program your data movements to the database during ‘quiet periods’ of net activity.
The product runs preprepared applications, monitors their behavior, collects measurements and stores those measurements. To do this, interaction with and dependence on the Tivoli desktop environment, is as follows:

- Applications and scripts are prepared using client capture or toolkits and then if necessary, moved to an active managed node of the system. You can also select ready made EPP probes that are simulations for specified applications. This is a desktop management or administration activity.
- The desktop command line is used to register the system location of applications or files to be used with the product.
- The product is used to build profiles that contain the registered applications, probes and scripts, with associated information. This information is typically periods of endpoint activity, frequency of runs, and details of data collections to be made during such runs.
- A desktop profile manager, to which your profile or profiles have been subscribed during the profile build session. The profiles are to be distributed to endpoints where measurements are made. This is a desktop activity, not a product function.
- The product starts and controls endpoint application activity runs and data collection tasks, including periodic data upload to the gateway. When data has been collected, summarized and added to a data file, limited inspection of collected data can be monitored through the Web GUI.
- The product sends aggregated data files via the gateway to the database (which may be independent of the system). After data is deposited in the database there is no further data processing or involvement by the product.
- The Tivoli Decision Support component installed as part of the Tivoli desktop extracts and prepares data.

**Performance Monitoring**

The goal of performance monitoring is to perform real-time data collection and analysis, to report the up-to-date health status of the application, the level of resource consumption, and the rate of error detection.

Tivoli Application Performance Management product provides the tools to carry out the following tasks:

- Manage compliance with Service Level Agreements.
- Isolate bottlenecks.
- Balance work loads in order to optimize service levels and cost efficiencies.
- Analyze transactions and their resource consumption, to understand how to redesign and reconfigure the application, or process, to achieve optimal performance.
- Tune resources.

**How the Product is Used**

There are several types of prepared application transactions that may be used with the product. Each one has different advantages and disadvantages. Advice on how to choose a method most appropriate for your requirement, is discussed in greater detail in “Deciding on a Method for Measuring” on page 45 of this User’s Guide.
Client Capture

Client capture enables you to measure application performance at the end user location, without requiring access to the source code for that application. The true GUI processes and activity are intercepted, to recognize events and event patterns. These events and patterns are correlated to produce performance data.

This approach does not require any changes to the application. The events related to the monitored application, are analyzed to determine which type of transaction was run. The response time is measured.

User activity is monitored by the client capture agent, which observes GUI activity of the client component for applications such as web browsers. The agent looks for certain patterns of GUI events that are described in configuration files. When the agent identifies event
patterns, it reports the start and the stop time of the associated business transactions via application response measurement (ARM) application programming interface (API) calls to the engine.

**Application Instrumentation**

This technique requires that ARM API calls are inserted into the application code at the appropriate points. Applications can be *instrumented* at a user’s desktop and on the servers to which their application connects. In this case, the measurements can be correlated to isolate the cause of extended response time.

The application code makes an ARM API call when the user transaction starts, and another call when the transaction ends. The first of these calls starts a timer in the product agent, and the second one stops the timer. The time difference between the two calls is the end-user’s response time. This measurement is collected according to a schedule contained in the associated profile, and the collections are stored in a log file. The product supports ARM APIs versions 1.0 and 2.0.

**Transaction Simulation**

There are two methods of transaction simulation which can be used with the product:

1. End-to-end probe platform (EPP) simulations provided by an IBM technology.
2. Simulation scripts prepared using Mercury Interactive Inc. LoadRunner tools.

**End-to-end probe platform (EPP) Simulations**

EPP interacts with server based applications and gathers data that can be converted into performance and quality of service measurements that reflect the experience of end users. EPP probes are already instrumented with ARM API calls and therefore you do not need to instrument them in any way before use.

EPP’s defining feature is that it conducts and measures end user transactions without requiring any modification to the application or the network. Because transactions generated by EPP probes use the same protocol stack, traverse the same network paths, and interact with servers in the same way as those created by real users, EPP data realistically reflects the end user experience in all these areas.

With EPP, you can:
Measure response and transaction times from the customer’s perspective.

- Get real-time alerts when slowdowns or outages occur.
- Record meaningful data for tracking SLA/SLO performance.
- Understand and demonstrate the effect on performance of hardware and software upgrades.
- Understand daily, weekly, and monthly performance cycles and trends.
- Integrate measurements across all the subsystems involved in delivering applications.

**Simulation Scripts**

The product is designed to work with the Mercury Interactive LoadRunner suite of toolkits. With these, you use a “record and play” technique to record the activity (or selected part of the activity) of a business application. To do this you use a Mercury toolkit to produce the script as you use the selected application. The script that is generated in this way is then instrumented with ARM API calls. These calls are inserted in appropriate parts of the script in the same way as is described in “Application Instrumentation” on page 8. The script is then replayed to simulate the behavior of a “virtual user” (simulated process).

![Figure 8. How VuGen works](image.png)

**Note:** Virtual user simulations may only be used on endpoints with Windows platforms.

**Transaction Simulation Toolkits**

Development toolkits, marketed by Mercury Interactive, may be used to prepare simulated application transactions. These simulations generate performance data as they run. Data collection instructions are inserted for the product engine. The prepared simulated applications are included in profiles that are distributed to an endpoint or several endpoints.

The Mercury toolkits can be installed on a workstation unrelated to the main server. However, if you use a machine that is not a product node, the scripts created on this machine must be moved to a region managed node to be registered.

The Mercury Interactive LoadRunner/Virtual User Generator (VuGen) toolkit may be used with the product.

**LoadRunner**

LoadRunner is the Mercury Interactive core product that enables toolkit scripts and simulation scripts to be developed and to be run on the endpoints.
The virtual user script creator, known as VuGen, is used to create and develop virtual user scripts. VuGen follows the actions of real end users performing typical business processes or transactions and generates a script listing that can then have ARM API instructions inserted.

For example, the accounting department in your company may print monthly salary statements as the payroll is prepared. The operator types employee and account information to a database. To emulate this process, you create a virtual user script that performs the same actions as that operator.

You create virtual user scripts and integrate them into the product profiles to monitor the performance of your distributed applications. By emulating the actions of real end users, virtual user scripts enable you to measure, quantify, and identify performance problems in real time.

As the virtual user scripts perform transactions, the product engine extracts data on application performance (including response times and availability) as well as real-time alerts based on thresholds.

**Starting Work with the Product**

There are five inter dependent main tasks in a Tivoli application performance management project. Only the fourth stage (monitoring) of those in the following list is automatic. The other stages can be carried out at any time, at any managed node location, and they can be considered to be “pencil and paper” tasks:

**Planning**

The first step you take toward monitoring an application, is to decide what to monitor. This includes which transactions to look at within an application, the method to be used, at what time of day these applications are to be run and where they are to be run and monitored.

**Preparing scripts**

When the planning is complete, you must decide which tools would best fit your needs, that is, client capture, EPP, ARM API tools, simulation tools, and so on. You can now start to add API calls to your applications or create simulation scripts. When you have your prepared scripts or ready-made scripts, they need to be registered to a repository. They can then be retrieved and added to profiles together with schedules, to control when and for how long, they are to be active at the endpoints.

**Building profiles**

The product has several GUIs to enable you to build and maintain profiles. You use profiles to define your performance monitoring requirements. For example, in a profile you define the number of applications or simulated applications to run, the actions that they perform, and when they run and are measured. You also define what data is gathered and how it will be recorded.
The completed profiles must be distributed to the endpoints where they are to be used. Distribution is carried out using the Tivoli desktop within the managed node.

**Monitoring and collections**

Profile distribution causes the instrumented applications or simulations, to generate data in the form of response time measurements that are specified in the associated profile. If correlation is active, correlator data and instances are also recorded.

The collected data is gathered, and summarized at each endpoint. This is the active part, where an endpoint engine runs the applications under true network conditions and gathers data. From time to time this data is gathered, aggregated and sent to a database.

**Analyzing data**

With Tivoli Decision Support, you can create reports on the data received. This is another passive and administrative task which can be carried out in isolation. A typical data presentation using Tivoli Decision Support is shown in Figure 10.

**Figure 9. What is in a profile**

**Figure 10. Typical data display**

**Note:** Data analysis is not part of the product.
Performance Analysis and Reporting

Analysis of the gathered data enables you to measure the service level of any application and thus, allows you to tune application and network management more efficiently.

Tivoli Decision Support adds an important element, because historical data can be correlated and analyzed to pinpoint offending resources and thereby improve availability even further. In addition, by using Tivoli Decision Support to analyze both the sources and duration of any problems, you can fine-tune processes they may use to resolve problems. In this way you can improve on the time-to-resolution metric demanded by users.

Tivoli Decision Support Guide

The Tivoli Decision Support guide enables you to view the performance of an application. The Tivoli Discovery Interface presents information in graphical format by using several multidimensional cubes that are “built” with data archived in the database.

The product collects information about application transactions through API calls that you define in the code of the application programs or simulations. Performance data collected at predetermined times is extracted from log files and stored in the database. Data held in the database can be extracted using the Tivoli Decision Support, to provide a variety of statistics that are displayed in many different formats.

For a description of the Tivoli Decision Support Guide and advice on how to use it, refer to the Tivoli Decision Support for Application Performance User’s Guide.

Web GUI

The Tivoli Decision Support guide only shows data up to yesterday. You use the Web GUI to look at significant data from current (today’s) log files, or from older files if they are still held on the endpoint. By default the most recent files are accessed.

The Web GUI accesses the current data files on a specified endpoint using a standard web browser and provides graphic displays of selected data from the chosen endpoint. The display can be set to automatically update after each data collection interval.
The Tivoli Application Performance Management product installation disk, contains packages for the various software components that make up a full implementation of the product. You select installations according to the way your network is arranged, and according to your choice of hardware locations, as described in this chapter.

Information is organized as follows:

- Platform-specific Information
- Software Requirements
- Installation Locations
- Installation and Migration Methods
- Installations
- Migrating from Version 2.0.1 of the Product
- Uninstallation

**Note:** To install products on the Tivoli desktop system you require an authorization role of `senior` and `install-product`.

## Platform-specific Information

The following table identifies the supported platform versions known at the time of publication. For more detailed and up-to-date information, please see the release notes.

<table>
<thead>
<tr>
<th>Platform</th>
<th>Supported Versions</th>
</tr>
</thead>
<tbody>
<tr>
<td>AIX®</td>
<td>Endpoint, Gateway, and Server: IBM RS/6000® series running AIX, Version 4.3.x</td>
</tr>
<tr>
<td>HP-UX</td>
<td>Gateway and Server: HP9000/700 and 800 series running HP-UX, Version 11.00</td>
</tr>
<tr>
<td>Solaris</td>
<td>Endpoint, Gateway, and Server: Sun SPARC series running Solaris, Versions 2.7 and 2.8</td>
</tr>
<tr>
<td>Windows 95</td>
<td>Endpoint:* § IBM-compatible with a Pentium® 133 MHz or faster processor, running Microsoft Windows 95</td>
</tr>
</tbody>
</table>
## Platform Supported Versions

<table>
<thead>
<tr>
<th>Platform</th>
<th>Supported Versions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Windows 98</td>
<td>Endpoint:*&lt;br&gt;IBM-compatible with a Pentium 133 MHz or faster processor, running Microsoft Windows 98</td>
</tr>
<tr>
<td>Windows NT®</td>
<td>Tivoli Desktop for Windows, Endpoint, Gateway, Server:&lt;br&gt;IBM-compatible with a 133 MHz Pentium or faster processor, running Microsoft Windows NT, Version 4.0 SP™ 6.</td>
</tr>
</tbody>
</table>

**Note:**<br> * Distributed Monitoring is not supported by Windows 95 or 98 installations and these platforms do not support EPP transaction simulation.<br> ** Needs Tivoli Framework Version3.7.1 + Patch 3.7.1-TMF-0002.<br> § Needs service pack 1 installed to run Client Capture.

## Software Requirements

You must have the following software installed and running:

- Tivoli Management Framework 3.6.5 + Patch 3.6.5-TMF-0008, or 3.7.1 + Patch 3.7.1-TMF-0002.
- Installation of Tivoli Distributed Monitoring 3.6, 3.6.2 or 3.7 is required if you want to install the product Monitors for Distributed Monitoring.
- Tivoli Decision Support (see your Tivoli Decision Support documentation set) is required if you want to see application performance data stored in the database, and you are using the Tivoli Decision Support for Application Performance 2.0 product.
- Tivoli Enterprise Console® 3.6 is required if you want to view events.
- If you want to use the Web GUI, Windows NT 4.0 or Windows 2000 is required on the managed node where it is installed.
- If you expect to use EPP to measure and monitor Lotus Notes®, the following requirements apply to the Notes™ client on which the EPP notes probe is to be run:
  - The International version of the Notes client is supported.
  - The North American version will not work.
  - Notes Client must be installed on the endpoint.
  - Supported releases of the Notes client are, Release 5.02b (or later), Release 5.03 (preferred)
  - The Notes client must be configured with the Notes ID and the name of the mail server to be used by the probe.
  - CLASSPATH in the System Environment variable must be updated with the full path of the Notes.jar file.
  - PATH in the System Environment variable must be updated with the full path of the Notes installation directory.
Installation Locations

There are two main types of installation location as detailed below:

1. **Install at server.** You must have Tivoli Framework already installed. The product must be fully installed from the product CD-ROM on the region server.

2. **Install at gateway.** The product must be fully installed from the product CD-ROM at all gateways.

What Is on Your Install Compact Disk

The installation CD contains the following directories:

<table>
<thead>
<tr>
<th>Directory</th>
<th>Content</th>
</tr>
</thead>
<tbody>
<tr>
<td>Root</td>
<td>Packed executable files for the core product component, Web GUI, Client Capture, Support for EPP, Distributed Monitoring Monitors and event console class definitions, and files used by the installation software.</td>
</tr>
<tr>
<td>Cfg</td>
<td>Configuration files used by the installation process.</td>
</tr>
<tr>
<td>Tivoli Decision Support for Application Performance 2.0</td>
<td>Tivoli Decision Support Guide for the product. For further details refer to the <em>Tivoli Decision Support for Application Performance User’s Guide</em>.</td>
</tr>
<tr>
<td>IBM_HTTP_Server</td>
<td>Installation file for the HTTP server</td>
</tr>
<tr>
<td>Apm_i10n</td>
<td>Language support files</td>
</tr>
<tr>
<td>Migration_Utilities</td>
<td>Shell command to move existing data from endpoints</td>
</tr>
<tr>
<td>Books/PDF</td>
<td>User’s Guides and Release Notes for the product in PDF format</td>
</tr>
<tr>
<td>Books/HTML</td>
<td>User’s Guides and Release Notes for the product in HTML format</td>
</tr>
</tbody>
</table>

Installation and Migration Methods

There are three methods of installation and migration available:

1. Use the Tivoli Management Framework TME® Desktop graphical user interface (GUI) windows and dialogs.

2. Use Tivoli Command Line Interface (CLI) commands.

3. Use Tivoli Software Installation Service (SIS)

If you are a new user of Tivoli Application Performance Management, you will need to carry out a full install of the product and its other associated components.

If you are already a user of the product, you need to follow instructions for migration to upgrade your current installation to Version 2.1. See “Migrating from Version 2.0.1 of the Product” on page 27.

Installations

This section provides installation procedures to follow for Tivoli Application Performance Management and its associated components.
Installing Tivoli Application Performance Management

The product is installed at the region server and all gateways which may have the product endpoints connected. The following sections describe how to install the product by using any of the following methods:

- From the Tivoli management framework desktop graphical user interface (GUI)
- Using the Tivoli management framework command line interface (CLI)
- Using Software Installation Service (SIS).

Using the Desktop GUI to Install the Product

To install the product from the Tivoli desktop, perform the following steps:

1. Insert the product CD in the CD-ROM drive of the machine selected for installation. Make a note of the identifying letter (or mount point on UNIX) for the CD-ROM drive because you may need it later when browsing and setting installation paths.

2. Start the Tivoli desktop application.

3. From the Tivoli Manager Install drop-down menu, select the Install Product option.

   ![Figure 11. Installing the product - first action](image)

   The Install Product dialog is displayed.

   **Note:** An error dialog may be displayed. If this happens, it is caused by a minor desktop software conflict, and the dialog can safely be ignored. Click **OK**.

4. If the Select Product to Install list is empty, the install path is not set correctly.
In this case, click the Select Media button, and use the browse facility to set the installation path to the product image directory where the installation files are located. This should be the root directory of your CD-ROM.

When the path is correctly displayed, click Set Path to confirm it and click Set Media and Close, to return to the Install Product dialog.

5. Select the product component you want to install (Tivoli Application Performance Management 2.1) and select the client (Tivoli management region or gateway) where you want to install the product.

Notes:

a. You may notice that when the product is selected, one or more node names move to the right from the Clients to Install On pane into the Available Clients pane. This means that those nodes already have the product, or part of the product installed. These installations or fragments of installation must be removed before continuing.

b. The product must be installed first, before selecting any other component.

c. Before installing Tivoli Application Performance Management 2.1 Monitors for Distributed Monitoring, you must have the Tivoli Distributed Monitoring component installed.

6. When you have made your selection, click Install & Close. The Product Install dialog is displayed, containing a scrolling list of operations.
7. When scrolling has stopped, click **Continue Install** and wait for the message “Finished Product Installation”.

   **Note:** During steps 6 and 7, the installation process may take several minutes. If you notice that scrolling has stopped, check the bottom of the window for a message prompt to continue, or a warning of any problem that requires correction.

8. Click the **Close** button (which is now active) to complete the installation. You are returned to the Tivoli Manager screen.

9. From the View menu select **Refresh** and the window now confirms installation by displaying the **Application Performance Management RIM** database icon.
In this version of the product, the icon provides an alternative method of creating the RIM object. It does not start the product application.

**Using the Framework CLI to Install the Product**

Make sure that the Tivoli environment is started with the `setup_env` command as follows:

On a UNIX machine run:

```
. /etc/Tivoli/setup_env.sh
```

On a Windows NT machine the command is:

```
"%windir%\system32\drivers\etc\tivoli\setup_env.cmd"
```

To install the product from the command line, issue the following command:

```
winstall -c /cdrom/tivoli -i TAPM node_name
```

where:

- `/cdrom/tivoli`
  - Specifies the path to the CD-ROM image where the *.pkt files are held.

- `-i TAPM`
  - Specifies the index file from which the product is installed on the network node.

- `node_name`
  - The identity of the network node where the product is to be installed

**Note:** If the node is not specified, the product is installed throughout the entire region.

See the `winstall` command in the *Tivoli Management Framework Reference Manual* for more information.
Using SIS to Install the Product
Installation using SIS is a standard Tivoli task. See the *Tivoli Software Installation Service User’s Guide* for instructions on installing SIS on your Tivoli installation and using SIS to install products.

Installing the Web GUI
The same installation methods as described in “Installing Tivoli Application Performance Management” on page 16 apply to the Web GUI installation.

**Note:** The Web GUI can only be installed on a Windows NT machine. UNIX is not currently supported.

Using the Desktop GUI to Install the Web GUI
To install the Web GUI option for the product, through the drop down menus of Tivoli desktop, proceed as follows:

Refer to “Using the Desktop GUI to Install the Product” on page 16 and carry out steps 1 to 3. Then continue as follows:

1. Select *Tivoli Application Performance Management 2.1 Web GUI* as the product to be installed.

![Figure 15. Installing Web GUI - selecting Web GUI to install](image)

The Install Options dialog is displayed.
2. The HTTP server used by Web GUI is automatically installed as part of this installation. However you may already have the HTTP server installed on the system and wish to leave it intact.

- If you want to leave an already installed HTTP server undisturbed, check the box and skip to step 3.

- For full installation, type in the Userid and Password under which the product Web GUI server will run. This userid must:
  - Be a local user on the Windows NT machine where the Web GUI server will be installed.
  - Be a member of the Administrators group on the Windows NT machine where the Web GUI server will be installed.
  - Have senior and administrator privileges in the Tivoli Management Region.

3. Click **Set**. You are returned to the Product Install Window.

4. Make a selection from the **Client to Install On**. This is the managed node running Windows NT where you want the Web GUI component to be installed.

5. Click the **Install and Close** button. The Product Install dialog is displayed, containing a scrolling list of operations. Installation may take a few minutes. If you notice that scrolling has stopped, check the bottom of the window for a message prompt to continue or a warning of any problem that requires correction.
6. Click **Continue Install** and wait for the message “Finished Product Installation”.

7. Click the **Close** button (which is now active) to complete the installation. You are returned to the Tivoli Manager screen.

**Configuration** - See “Configuring the Web GUI” on page 41.

**Using the Framework CLI to Install the Web GUI**

Make sure that the Tivoli environment is started with the `setup_env` command as follows:

On a UNIX machine run:
```
. /etc/Tivoli/setup_env.sh
```

On a Windows NT machine the command is:
```
"%windir%\system32\drivers\etc\tivoli\setup_env.cmd"
```

To install Web GUI from the command line do the following:

1. Install the IBM HTTP Server provided separately on the CD-ROM.

2. Issue the command line

   ```
   wininstall -c /cdrom/tivoli -i WEBGUI HTTPSERVINST=1 node_name
   ```

   where:

   **HTTPSERVINST=1**
   Specifies the HTTP server is already installed

   **Note:** The HTTP server must be installed before you can install Web GUI using CLI.

   `node_name`
   Is the identity of the region server.

   **Configuration** - See “Configuring the Web GUI” on page 41.
Using SIS to Install the Web GUI

Installation using SIS is a standard Tivoli task. See the Tivoli Software Installation Service User’s Guide for instructions on installing SIS on your Tivoli installation and using SIS to install products.

**Note:** The HTTP server must be installed first.

**Configuration** - See “Configuring the Web GUI” on page 41

Installing the HTTP Server

The HTTP server can be installed separately by the following procedure:

1. Insert the Tivoli Application Performance Management installation compact disk in your CD-ROM drive. If you are installing from a network drive, connect to that drive.
2. Click **Start** and then **Run**.
3. In the Run dialog, use the browse facility, or type in the following:
   
   ```
   %CD-ROM\IBM_HTTP_Server\setup.exe
   ```
   
   where %CD-ROM is the identity of the drive and top directory for your installation disk.
4. Click **OK**.
5. Follow instructions presented by the install shield application. You are prompted to supply:

   **WEBUSERID**
   
   The userid under which the product Web GUI server will run. This userid must:
   
   - Be a local user on the Windows NT machine where the HTTP server will be installed. Be a member of the Administrators group on the Windows NT machine where the HTTP server will be installed.
   - Have senior privileges in the Tivoli Management Region.

   **WEBPASSWD**
   
   The user’s password

   **Note:** The HTTP server must be installed on the region server.

Installing Support for EPP

The same installation methods as used for the product, described in “Installing Tivoli Application Performance Management” on page 16, apply to the Support for EPP installation.

**Note:** Support for EPP must be installed on the region server.

Using the Desktop GUI to Install Support for EPP

Refer to “Using the Desktop GUI to Install the Product” on page 16 and carry out steps 1 to 3 then proceed as follows:

1. At step 4 select Tivoli Application Performance Management 2.1 Support for EPP.
2. Select the Client to Install On.
3. Click **Install and Close**.

Using the Framework CLI to Install Support for EPP

Make sure that the Tivoli environment is started with the `setup_env` command as follows:

On a UNIX machine run:
On a Windows NT machine the command is:
"%windir%\system32\drivers\etc\tivoli\setup_env.cmd"

To install Support for EPP from the command line, issue the following command:
`winstall -c /cdrom/tivoli -i TAPMEPP node_name`

where:

_/cdrom/tivoli_

  Specifies the path to the CD-ROM image where the *.pkt files are held.

–i TAPMEPP

  Specifies the index file from which the supplied EPP probes are installed on the
  network node.

node_name

  The identity of the region server.

Using SIS to Install Support for EPP

Installation using SIS is a standard Tivoli task. See the Tivoli Software Installation Service
User’s Guide for instructions on installing SIS on your Tivoli installation and using SIS to
install products.

Installing Language Support for the Product

Tivoli Application Performance Management 2.1 language support consists of two
components:

- Four Asian languages. Korean, Japanese, standard Chinese and simplified Chinese. These
  may be separately installed.
- Web GUI language support. One installation covers all four languages.

Where to Install

Language support must be installed on all machines where you have Tivoli Application
Performance Management installed.

Web GUI language support must be installed only on the managed node where Web GUI has
been installed.

Using the Desktop GUI to install Language Support

Files for the language support may be installed as part of the main product 2.1 installation.
The procedure is the same as that described in “Using the Desktop GUI to Install the
Product” on page 16 except that Language support files are contained on their own
APM_L10N directory of the installation CD-ROM:

1. Carry out steps 1 to 3
2. At step 4 you must change the install path by clicking the Set Media button to call the
   File Browser dialog and adding/APM_L10N to the path.
3. Click Set Media and Close. The Install Product dialog is displayed with the Language
   support options displayed.
4. Continue with the installation procedure as if you were installing a product component.

Using the Framework CLI to Install Language Support

Make sure that the Tivoli environment is started with the `setup_env` command as follows:

On a UNIX machine run:

```
. /etc/Tivoli/setup_env.sh
```

On a Windows NT machine the command is:

```
"%windir%\system32\drivers\etc\tivoli\setup_env.cmd"
```

To install Language Support from the command line, issue the following command:

```
winstall -c /cdrom/tivoli -i APM21xx|WG21NLS node_name
```

where:

```
/cdrom/Apm_l10
```

Specifies the path to the CD-ROM image where the *.pkt files are held.

```
– i APM21xx
```

Specifies the index files from which the required support files are installed on the network node. File identities are:

- APM21JA - Japanese
Using SIS to Install Language Support
Installation using SIS is a standard Tivoli task. See the Tivoli Software Installation Service User's Guide for instructions on installing SIS on your Tivoli installation and using SIS to install products.

Installing Monitors for Distributed Monitoring
The same installation methods as used for the product, described in “Installing Tivoli Application Performance Management” on page 16 apply to the Monitors for Distributed Monitoring installation.

Note: Monitors for DM must be installed on the region server.

Using the Desktop GUI to Install Monitors for Distributed Monitoring
Refer to “Using the Desktop GUI to Install the Product” on page 16 and carry out steps 1 to 3 then proceed as follows:

1. At step 4 select Tivoli Application Performance Management 2.1 Monitors for DM.
2. Select the Client to Install On (the region server).
3. Click Install and Close.

Using the Framework CLI to Install Monitors for Distributed Monitoring
Make sure that the Tivoli environment is started with the setup_env command as follows:

On a UNIX machine run:

```bash
/etc/Tivoli/setup_env.sh
```

On a Windows NT machine the command is:

```cmd
"%windir%\system32\drivers\etc\tivoli\setup_env.cmd"
```

To install Monitors from the command line, issue the following command:

```bash
winstall -c /cdrom/tivoli -i TAPMMON node_name
```

where:

- `/cdrom/tivoli`
  Specifies the path to the CD-ROM image where the *.pkt files are held.
- `–i TAPMMON`
  Specifies the index file from which Distributed Monitoring monitors is installed on the network node.

`node_name`
The identity of the region server.
Using SIS to Install Monitors for Distributed Monitoring

Installation using SIS is a standard Tivoli task. See the Tivoli Software Installation Service User’s Guide for instructions on installing SIS on your Tivoli installation and using SIS to install products.

Migrating from Version 2.0.1 of the Product

To migrate your installation of the product, you use patches that will provide new files and overwrite other essential files of your product installation.

Installation of the Tivoli Application Performance Management 2.1 Upgrade Patch allows you to retain all the customers specifications on all the endpoints, the collected data, and the ID of the ARM instrumented applications on the client endpoints. There is no loss of data during the migration from Version 2.0.1 to Version 2.1.

What You Need to Carry Out the Migration

For the product migration you have the following patches and products available on the product CD:

- Tivoli Application Performance Management 2.1 Upgrade Patch.
- Tivoli Application Performance Management 2.1 Web Gui Upgrade Patch.
- Tivoli Application Performance Management 2.1 Monitors for DM Upgrade Patch.
- Tivoli Application Performance Management 2.1 Support for EPP

These images allow you to migrate the Tivoli Application Performance Management product and components to level 2.1 and to install the product 2.1 Support for EPP

On the CD you also have a shell script called save_cache_before_upgrade.sh that is used to send the data currently present on the gateways to the database, before installing the upgrade patch. This script is on the CD-ROM in directory Migration\Utilities

Migration Patch Software Requirements

You must have the following software installed and running:

- Tivoli Management Framework 3.6.5 + patch 3.6.5-TMF-0008 or 3.7.1 + patch 3.7.1-TMF-0002
- Tivoli Application Performance Management 2.0.1
- Tivoli Application Performance Management 2.0.1 Monitors for DM
- Tivoli Application Performance Management 2.0.1 Web GUI

Migration Processes

The following environment configuration and collected data are saved after the Tivoli Application Performance Management 2.1 Upgrade Patch installation:

- The ARM instrumented applications, the Client Capture applications and the simulated applications registered in the Tivoli Application Performance Management repository.
- The MarProfile.
- Application IDs on the endpoints.
- The engine status.
- Aggregated data on the endpoint.
Before installing the patch, you must check whether data files are present on the gateways. The files may be found under the directory $TMPDIR/tapm/cache where TMPDIR is the value returned by the framework command wtemp.

If data files are on the gateway, run the shell script included on the installation CD as follows:

1. Make sure that the Tivoli environment is started with the setup_env command as follows:
   - On a UNIX machine run: 
     
     ```bash
     . /etc/Tivoli/setup_env.sh
     ```
   - On a Windows NT machine the command is:
     
     ```bash
     "%windir%\system32\drivers\etc\tivoli\setup_env.cmd"
     ```

2. Change to the following directory:
   
   ```bash
   $CDroot/Migration_Utilities
   ```
   where CDroot is the top CD-ROM directory.

3. Ensure that the database window is open on the mid-level manager or set it to “always open” with:
   
   ```bash
   wmarsetstatus -g DBUploadStartTime=00:00, DBUploadStopTime=23:59 <mid-level manager_name>
   ```

4. Run the script `Save_cache_before_update.sh`

5. Wait until this operation completes before continuing the migration process

6. Disable any active client capture collections and close the browser windows

7. Apply migration patches as needed, this updates the product code on the region server and managed nodes.

The changes on the product engine at endpoints become operational as soon as you start reusing the endpoint by distributing a MarProfile to that endpoint or by running any of the following commands at that endpoint.

- wmarstarteng, wmarstopeng, wmarcleareng, wmarlsapp, wmarlseng, wmarstartcoll, wmarstopcoll, wmarsetstatus, wmargetdata, wmargetstatus.

After running any of these commands on the endpoint the engine must be restarted using the wmarstarteng command.

After installing the patch, the mid-level manager needs to be restarted on every managed node, using script:

```bash
wmarrestartmlm.sh <managednode-name>
```

**Migrating the Product**

Use one of these three methods to upgrade Tivoli Application Performance Management from 2.0.1 to 2.1.

**Using the Desktop GUI to Migrate the Product**

To install the product patch from the Tivoli desktop, perform the following steps:

1. Start the Tivoli desktop application.

2. From the Desktop drop-down menu of the Tivoli Manager for Administrator window, select the **Install->Install Patch** option.
The Install Patch dialog is displayed.

3. If the Select Patch to Install list is empty, the install path is not set correctly.

Figure 19. Installing a patch - first screen

In this case, click the Select Media button, and use the browse facility to set the installation path to the product image directory where the installation files are located. This should be the root directory of your installation CD ROM.

When the path is correctly displayed, click Set Path to confirm it and click Set Media and Close, to return to the Install Product dialog.

Figure 20. Installing a patch - setting the install path
4. Select the component you want to install (Tivoli Application Performance Management 2.1 xxxx Upgrade Patch) and select the client (Tivoli Management Region or managed node) where you want to install the patch.

5. When you have made your selection, click **Install & Close**. The Patch Install dialog is displayed, containing a scrolling list of operations.

6. When scrolling has stopped, click **Continue Install** and wait for the message “Finished Patch Installation”.

   **Note:** Installation may take several minutes. If you notice that scrolling has stopped, check the bottom of the window for a message prompt to **continue installation**, or a warning of any problem that requires correction.

7. Click the **Close** button (which is now active) to complete the installation. You are returned to the Tivoli Manager screen.

![Figure 21. Installing a patch - confirmation that the patch is installed](image)

**Using the Framework CLI to Migrate the Product**

Make sure that the Tivoli environment is started with the **setup_env** command as follows:

On a UNIX machine run:

```
./etc/Tivoli/setup_env.sh
```

On a Windows NT machine the command is:

```
"%windir%\system32\drivers\etc\tivoli\setup_env.cmd"
```

To install the product patch from the command line, issue the following command:

```
winstall -c /cdrom/tivoli -i TAPM_P21 node_name
```

where:

```
/cdrom/tivoli
```

Specifies the path to the CD-ROM image where the *.pkt files are held.
--i TAPM_P21
   Specifies the index file from which the patch is installed on the network node.

node_name
   The identity of the network node where the product is to be installed

   **Note:** If the node is **not** specified, the product is installed throughout the entire region.

   See the `winstall` command in the *Tivoli Management Framework Reference Manual* for more information.

**Using SIS to Migrate the Product**
Installation using SIS is a standard Tivoli task. See the *Tivoli Software Installation Service User’s Guide* for instructions on installing SIS on your Tivoli installation and using SIS to install products.

**Migrating the Web GUI**
Use one of these three methods to upgrade Web GUI from 2.0.1 to 2.1.

**Using the Desktop GUI to Migrate Web GUI**
To install the Web GUI patch from the Tivoli desktop, use the same procedure as described in “Using the Desktop GUI to Migrate the Product” on page 28 but select Tivoli Application Performance Management 2.1 Web GUI Upgrade Patch at step 5.

**Using the Framework CLI to Migrate Web GUI**
To install the Web GUI patch using the Tivoli CLI, use the same procedure as described in “Using the Framework CLI to Install Language Support” on page 25 except that you use the `wpatch` command instead of `winstall` and substitute the patch name WEBG21 for WEBGUI in the command line.

**Using SIS to Migrate Web GUI**
Installation using SIS is a standard Tivoli task. See the *Tivoli Software Installation Service User’s Guide* for instructions on installing SIS on your Tivoli installation and using SIS to install products.

**Migrating Monitors for DM**
Use one of these three methods to upgrade Web GUI from 2.0.1 to 2.1.

**Using the Desktop GUI to Migrate DM Monitors**
To install the product Monitors patch from the Tivoli desktop, use the same procedure as described in “Using the Desktop GUI to Migrate the Product” on page 28 but select Tivoli Application Performance Management 2.1 Monitors for DM Upgrade Patch at step 5.

**Using Tivoli Framework CLI to Migrate DM Monitors**
The migration process to install the patches using CLI is similar to the “Install using CLI procedures” except that you use the `wpatch` command instead of `winstall` and substitute the patch name MON_P21 for TAPMMON in the command line.

**Using SIS to Migrate Monitors for DM**
Installation using SIS is a standard Tivoli task. See the *Tivoli Software Installation Service User’s Guide* for instructions on installing SIS on your Tivoli installation and using SIS to install products.
Migrating Language Support

Use one of these three methods to upgrade Web GUI from 2.0.1 to 2.1.

Using the Desktop GUI for Migrating Language Support

If you are migrating from 2.0.1 you can install patches for language support. The procedure for installing language patches is the same as that described in “Migration Processes” on page 27 with the same instructions for changing the path detailed above. The Patch install dialog is displayed.

![Patch install dialog](image)

Figure 22. Upgrade language support patch options

Using the Framework CLI to Install Language Support

Make sure that the Tivoli environment is started with the `setup_env` command as follows:

On a UNIX machine run:

```
. /etc/Tivoli/setup_env.sh
```

On a Windows NT machine the command is:

```
"%windir%\system32\drivers\etc\tivoli\setup_env.cmd"
```

To install Language Support from the command line, issue the following command:

```
wpatch -c /cdrom/tivoli -i APM21Pxx|WG21PNLS node_name
```

where:
Specifies the path to the CD-ROM image where the *.pkt files are held.

`-i APM21Pxx`

Specifies the index files from which the required support files are installed on the network node. File identities are:

- APM21PJA - Japanese
- APM21PKO - Korean
- APM21PCN - Simplified Chinese
- APM21PTW - Traditional Chinese

`-i WG21PNLS`

Specifies the index file from which the Web GUI patch is installed on the network node.

`node_name`

The identity of the region server.

Using SIS to Install Language Support

Installation using SIS is a standard Tivoli task. See the Tivoli Software Installation Service User’s Guide for instructions on installing SIS on your Tivoli installation and using SIS to install products.

Migrating Support for EPP

The Support for EPP is a full product not a patch. To install it follow “Installing Support for EPP” on page 23.

Uninstallation

Parts of the product application are installed in several different locations within your network. See “The Product Environment” on page 4. Before attempting a partial uninstall or full uninstall task, it is important to identify from where you want to remove the application or files and the consequences and effects on the network and database.

Uninstall must be carried out in an orderly way so that no remnants are left that could interfere with a future installation. Care must also be taken that files which may be needed at a later stage of uninstall, are not deleted too soon.

You can use the `wuninst` shell script provided by the Tivoli Management Framework to uninstall Tivoli Application Performance Management. Refer to the Tivoli Management Framework Reference Manual for more information about this command.

Notes:

1. To run the wuninst command you must have an authorization role of `policy`.
2. The order in which you uninstall the components from the nodes is important. You must uninstall the components in the following order:
   a. Uninstall the endpoints in any order.
   b. Uninstall the Tivoli framework component.

Note:
Uninstalling Tivoli Application Performance Management

There are situations where you may need to uninstall the product. For example, if something has gone wrong with your install session, and you may want to remove the unsuccessful installation from your machine.

First, identify the region server where Tivoli Application Performance Management is installed. Then issue the following command to synchronize the gateways:

```bash
wep sync_gateways
```

**Note:** To avoid removing prepared simulations, remove the database before uninstalling the product. See “Removing the Product Database” on page 112.

**Windows Endpoints**

Stop the engine on each endpoint by issuing the `wmarstopeng` command accompanied by the correct `endpoint_name`.

At Windows platform endpoints perform the following steps in the directory where the endpoint has been installed:

1. Delete subdirectory `%LCF_DATDIR%\Mar`.
2. Delete subdirectory `%LCF_DATDIR%\cache\bin<interp>\TME\MAR`.
3. Delete subdirectory `%LCFROOT%\bin<interp>tools`.
4. Delete file `%LCF_CATDIR%\<LANG>\marengine.cat`.
5. Delete file `tapmextlib.dll` in `%windir%\system32` or `%windir%\system` depending on the type of Windows system you are running.
6. Delete file `libarm32.dll` in `%windir%\system32` or `%windir%\system` depending on the type of Windows system you are running.
7. Repeat steps 1 to 6 individually, for all Windows endpoints.

You can now start installation again on a “clean” machine.

**UNIX Endpoints**

Stop the engine on the endpoint by issuing the `wmarstopeng` command accompanied by the correct `endpoint_name`.

At UNIX platform endpoints perform the following steps:

1. Delete subdirectory `$LCF_DATDIR/Mar`.
2. Delete subdirectory `$LCF_DATDIR/cache/bin<interp>/TME/MAR`.
3. Delete file `$LCF_CATDIR/<LANG>/marengine.cat`.
4. On AIX installations, also delete file `/usr/lib/libarm.a`
5. On Solaris installations, also delete file `/usr/lib/libarm.so`
6. Repeat steps 1 to 5 individually for all UNIX endpoints where an engine is running. This ensures that the product is removed from all nodes of the region.
Managed Node

1. At the Tivoli command line, issue the following command:

<table>
<thead>
<tr>
<th>Platform</th>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>Windows</td>
<td>sh wuninst TAPM_1.0 &lt;host_name&gt; –rmfiles</td>
</tr>
<tr>
<td>UNIX</td>
<td>wuninst TAPM_1.0 &lt;host_name&gt; –rmfiles</td>
</tr>
</tbody>
</table>

where:
- TAPM_1.0 Is the identity of the product you want to uninstall.
- <host_name> Is the identity of the region server. This ensures the product is removed from all nodes of the region.
- –rmfiles Is the argument that specifies all the product files as well as Tivoli repository entries are removed.

Note: See the documentation for the wuninst shell script in the Tivoli Management Framework Reference Manual for more information.

2. From the command line, issue the command:

```bash
wchkdb –u
```

This is a Tivoli desktop utility that verifies the integrity of the database after the product components are removed.

You can now start installation again on a “clean” region.

Uninstalling the Web GUI

It is possible to uninstall Web GUI and leave the product application intact. However, if you uninstall the product (see “Uninstalling Tivoli Application Performance Management” on page 34), Web GUI is also uninstalled at the same time.

To uninstall Web GUI, type the following command:

```bash
sh wuninst TAPMWEB <host_name> –rmfiles
```

where:
- TAPMWEB Is the identity of the product you want to uninstall.
- <host_name> Is the identity of the managed node where Web GUI is installed.
- –rmfiles Is the argument that specifies all Web GUI component files are removed.

The uninstall procedure given above uses an uninstall shield process. It is recommended to complete system cleanup by restarting Windows after uninstall.

Uninstalling Support for EPP

There is no uninstallation facility provided for EPP support. It is uninstalled when the product is uninstalled.

Uninstalling Monitors for Distributed Monitoring

There is no uninstallation facility provided for Distributed Monitoring monitors. They are uninstalled when the product is uninstalled.
Getting Started

When you have installed Tivoli Application Performance Management, you must perform the following tasks before you can use it:

1. If you want to use Tivoli Management Framework notices, you need to add a notice group
2. Decide which applications and transactions you want to monitor
3. Prepare applications that you want to measure
4. Register all applications and simulations that you want to use, in the product repository
5. Create one or more profile managers for the product
6. Add MarProfiles as a managed resource to your policy region
7. Create profiles, containing the prepared applications and simulations
8. Distribute the profile or profiles to endpoints where you want the applications to run and be monitored
9. If you want to use the Web GUI to interactively check data collections, you need to install it
10. If you want to use Tivoli Management Framework Distributed Monitoring, you must install the monitors
11. You may need to configure the Web GUI installation
12. If you want to run EPP probes, you must install the Support for EPP

Adding the Notice Group

If you intend to receive the product notice messages, you need to add the product Notice Group to the Administrator user account. This must be done to receive and see any notices that contain information about how to configure and run the product. To add this function, perform the following steps:

1. Double click the Administrators icon on the Tivoli management enterprise desktop. The Administrators window opens.
2. Right mouse click on the administrator icon in the Administrators window and select Edit Notice Group Subscription from the pop up menu.
   A Set Notice Group window appears.
3. Select **Tivoli APM** in the **Available Notice Groups** pane and use the left arrow button to add it to the **Current Notice Groups** pane. Click on **Change and Close**.

4. Now all messages concerning the product appear on the notice board of the chosen administrator.
Preparing Applications

There are four types of applications that you prepare for use with the product:

- Create client capture scenarios (see Chapter 5, “Client Capture” on page 47), or
- Instrument application code with ARM API calls (see Chapter 7, “Instrumenting an Application for ARM” on page 61), or
- Use ready made EPP probes to simulate specific application behavior in your network (see Chapter 6, “Transaction Simulation” on page 53).
- Create simulation (virtual user) scripts with ARM API calls embedded in them (see Chapter 6, “Transaction Simulation” on page 53).

Registration

Registration must be done before the product can be used; registration is described in Chapter 9, “Registering Applications” on page 71.

There are three good reasons for registration:

1. The product needs to know which applications are to be monitored and where those applications or simulations are located.
2. The product needs to know what type of applications or simulations you intend to monitor.
3. As an operator, you need to know which applications are available for use. When applications are registered, they are added to a list which is made available to you in the main product GUI.

Note: You can also use the `wmargetapp` CLI command (see “wmargetapp” on page 186), to provide a list of currently registered applications.

Creating a Profile Manager and Profiles

You may already have a profile manager that you intend to use. Creating a profile manager is a standard Tivoli desktop task. Refer to your Tivoli Management Framework User’s Guide for details.
The product only supports dataless endpoints. This means that where the subscribers are intended to be endpoints, profile managers that are used with the product profiles must be created in Dataless Endpoint mode. If subscribers are intended to be other profile managers, they can be created as ordinary profile managers.

To create a dataless profile manager, select the **Dataless Endpoint Mode** check box as shown in the Create Profile Manager dialog, before clicking **Create & Close**. By doing this, you set up rights to distribute to endpoints.

Profile manager names can include any alphanumeric character, an underscore (_), a dash (-), a period (.), or a space.

![Create Profile Manager dialog](image)

*Figure 26. Create profile manager dialog*

The Policy Region window is displayed with an icon representing the new profile manager just created.

**Adding MarProfile as a Managed Resource**

Before you create your first profile, you must first check that the **MarProfile** type is listed as a managed resource of the Policy Region as follows:

1. In the Tivoli desktop Policy Region window, select **Managed Resources** from the **Properties** drop-down menu. The Set Managed Resources dialog is displayed.

![Set Managed Resources dialog](image)

*Figure 27. Adding the product as a managed resource*
2. Verify that the MarProfile resource is present in the Current Resources box. If it is not, locate and select it in the Available Resources box, and use the left arrow to move it into the Current Resources box.

3. If it was necessary to move MarProfile, click Set & Close. If no changes were necessary, click Close.

Building and Distributing Profiles

Profile building and distribution is described in Chapter 10, “Building Profiles” on page 77.

Setting up Monitors

A ready-made set of Distributed Monitoring monitors is included on your installation disk. To install the monitors refer to “Installing Monitors for Distributed Monitoring” on page 26.

Note: You must install Tivoli Distributed Monitoring before installing the product monitors. Refer to “Installing Tivoli Application Performance Management” on page 16 and select Tivoli Application Performance Management 2.1 Monitors for Distributed Monitoring at step 5 of the procedure “Using the Desktop GUI to Install the Product” on page 16.

The product monitors include definitions for enterprise console event notification. If you have enterprise console installed on your desktop and want to take advantage of it, refer to “Sending Results to the Tivoli Enterprise Console” on page 135.

Configuring the Web GUI

The Web GUI installation process installs a web server on the managed node where this component is installed if it is not already present on the machine. There are situations where you may need to configure this web server. The web server installed is registered under Windows NT as two services:

- IBM HTTP Server.
- IBM HTTP Administration.

The IBM HTTP Server is the web server used by the Web GUI. The IBM HTTP Administration service is used to configure the server. This service is set to manual and it is stopped during the Web GUI installation. To start and stop the services you can do one of the following:

- Use the windows shell command net
- Open the Services feature from the windows control panel
- Use Start windows menu -> programs->IBM HTTP Server-> start/stop Administration/Server

If you need to change the configuration, after a full installation start the IBM HTTP Administration and open the URL:

http://<host_name>/index.html

where

<host_name> is the name of the managed node where the Web GUI is installed, and choose Configure server. When the authentication window is displayed, supply the user_id and password that you have used to install the Web GUI.
If you need to change the configuration, after migration (patch installation) start the IBM HTTP Administration and open the URL:

http://<host_name>/indexIBM.html

where

*host_name* is the name of the managed node where the Web GUI is installed, and choose **Configure server**. When the authentication window is displayed, supply the *user_id* and *password* that you have used to install the Web GUI.

![Figure 28. Supplying the network user password](image)

**Changing Ports**

If these services and the application server (tomcat) are installed to work with a port already in use, the Web GUI may not work correctly. In this case, you should change the port numbers.

By default, the ports used are:

- HTTP server uses port 80.
- Http Administration uses port 8008.
- Tomcat uses ports 8007 and 8080.

To change the server port you can use the HTTP Administration Configuration GUI, or you can change the directive **Port** in the file `httpd.conf`. If you need to change the port value to the HTTP Administration, edit the directive **Port** in the file `admin.conf`. You must also change the value 8008 in the file `apadminred.html` to have the correct link to the Administration GUI.

If you need to change ports values for the application server, you edit its configuration files (`tomcat.conf`, `tomcat.properties`, `server.xml`). Refer to the jakarta-tomcat project at: http://www.apache.org/ for information.
Planning What to Measure

An understanding of the way customers experience distributed applications is a valuable management tool for any IT business. In the past, much of this "understanding" has been based on anecdotal evidence, often in the form of vague complaints about "performance". By providing credible and understandable views of application performance the product helps document the reality.

Whether you use application instrumentation, or transaction simulation, you have to decide which transactions you want to measure and monitor. To do this, you have to define the key business transactions that are performance-sensitive, which is the most important step. Define who needs data; what kind of data they need and how the data will be used.

There are two kinds of transactions that generally provide the greatest benefit if they are instrumented:

- Transactions that are visible to users or that represent major business operations. These are the components for service level agreements, early problem detection, and monitoring work load.

- Transactions that are dependent on external services, such as a database operation, a Remote Procedure Call (RPC), or a remote queue operation. These are generally components of a user or business transaction. Knowing how these types of transactions are performing can be invaluable when analyzing problems, tuning applications, and reconfiguring systems and networks.

To assist the analysis, you need to answer the following questions, to help decide which transactions to measure:

- What unit of work does this transaction define?
- Are the transaction counts and response times important?
- Who will use this information?
- If the performance of a transaction is too slow, is there some corrective action that can be taken (for example, off load work from the machine, add memory, or relocate remote files)?

Do not try to measure every transaction; this will only cause unnecessary load.

Measurements Used

The product enables users who are service providers to measure application performance, in order to meet the service level agreements (SLAs) by which they are bound. There are three measures of service delivery, and a complete SLA contains criteria for all three.
1. **Response time** is the measure of how long each business transaction takes to complete. From the typical end-user perspective, it is the most important performance measurement.

A close relationship often exists between response time and transaction availability, for example in web-based electronic commerce applications.

A transaction with a very long response time can often appear to a user as a permanently hung transaction. This can lead the user to give up waiting, convinced that the transaction will not complete. An example of how data on response time can be extracted from the product database and displayed using Tivoli Decision Support (see the Tivoli Decision Support Performance Management User’s Guide) is shown below.

![Figure 29. Typical response time display](image)

2. **Throughput** is the measure of how many transactions complete in a given time period. This is the most relevant service level measurement.

   You should note that the application with the highest throughput may not have the best response times.

3. **Transaction availability** is a measure of whether business transactions complete successfully.

   An excessive delay in several sub-transactions can lead to time-outs in a higher level transaction, thereby causing that transaction to fail, even though no single component is completely unavailable. From the user perspective, in both cases, the application was not available.

Response time and throughput measurements are performance measurements. Although transaction availability is not strictly a performance measurement, it influences overall performance. The techniques used to measure transaction performance can generally be used to measure transaction availability.

**Improving Performance**

There are three steps you can carry out to improve performance:
1. Understand how the application is performing (its service level), how components, including middleware are connected together, and how much each of those components contributes to the overall performance.

2. Use this information to manage the available network and server capacity. This involves correctly sizing the total capacity and balancing the work load between systems.

3. Optimize and tune the individual components, such as servers.

Before spending a lot of time and effort to manage the performance of the networks and servers, it is prudent to determine how much service levels would be improved. The best way to make that determination is to use application performance information. How much of the elapsed time of a transaction, is spent in each network segment or each server? How much of a difference to users would it make if the performance of this component were improved?

**Deciding on a Method for Measuring**

- If end user access to the application is through a web (Internet Explorer or Netscape Navigator), use client capture.

- If you have access to the application source code and the resources to insert API calls into that code, consider building ARM instrumented API applications.

- If you do not have programmer access to the application, use VuGen to build a simulation of the application activities, if the specific protocol used by the application is supported by VuGen.

- If you need to investigate network availability and performance data, use EPP probes.
Deciding on a Method for Measuring
Client Capture

The Tivoli Application Performance Management client capture feature, provides a way to measure response time for real end user transactions. It does this by passively monitoring the end user application activity.

Client Capture is able to detect and measure business transactions by recognizing certain patterns of events that are generated by the application. Client capture has the following features:

- It can sense the application activity. Software agents called sensors accomplish this.
- It can process the resulting stream of events and recognize some patterns as the signature of a specific business transaction.
- Client capture generates its own ARM API calls according to changes in application behavior that is detected by the client capture sensors. It reports data to the product for these business transactions through ARM API calls.

The main feature of client capture is that it can be used to monitor real user activity and measure business transactions for applications, without the need to instrument those applications or access source code. The current sensors exist for Microsoft Internet Explorer 5 and Netscape 4; sensors for other applications will be announced and issued as they become available.

About Client Capture

Client capture recognizes when an end-user starts a transaction, and when the transaction completes. When the client capture agent identifies that the monitored application is carrying out a transaction that it has been configured to monitor, it measures the response time. Client capture uses ARM API calls to report the response time to the engine.

Client capture is able to recognize events in the application, irrespective of whether the state of the application window is maximized, in restore mode, or is minimized. When more than one window is open for a given application (for example, two Netscape windows), client capture is still able to measure response time simultaneously, without getting the windows confused.

To use Client Capture with a Windows 95 platform is necessary to install:

- Microsoft Windows 95 Service Pack 1.
- Internet Explorer 5.0.

For all Windows platforms:

Com registration fails if MFC42.Dll is older than image version 6.0 and product version 6.03.
Using Client Capture

The product uses configuration files in XML format that have been prepared with application and transaction details to recognize the transactions that you are interested in. The configuration files are distributed within profiles to the endpoint or endpoints where a Web application performance is to be measured.

To set up the client capture feature, edit the sample configuration file %BINDIR%\.\generic\Mar\ClientCapture\Sample.xml that was transferred to the managed node during the product installation process.

The first lines of the file must read:

```xml
<?xml version="1.0" ?>
<!DOCTYPE ClientCapture SYSTEM "TapmClientCapture.dtd">
<ClientCapture>
```

The root element in the file must be named "ClientCapture" and contains all the information about which applications are to be monitored. An example for Internet Explorer is:

```xml
<IExplorer_5x>
```

This lines starts configuration of the Internet Explorer sensor.

```xml
<Transaction>
  <FilterInclude>http://*ibm.com*</FilterInclude>
  <Name>$URL$</Name>
  <Details>Page downloaded from an IBM site</Details>
</Transaction>
```

The five lines above define a set of transactions. All URLs that match the specified <FilterInclude> tags are recorded as a transaction. The $URL$ expression used as the value of <name> tag causes the URL to be used as an ARM transaction name.

```xml
<Transaction>
  <FilterInclude>http://*tivoli.com*</FilterInclude>
  <Name>$URL$</Name>
  <Details>Page downloaded from a TIVOLI site</Details>
</Transaction>
```

The six lines above open a similar portion of script for another transaction set. The last line closes the Internet Explorer configuration subsection.

An almost identical example for Netscape, together with the configuration file closing tag is provided without breaks below:

```xml
<Netscape_4x>
```

```xml
<Transaction>
  <FilterInclude>http://*ibm.com*</FilterInclude>
  <Name>$URL$</Name>
  <Details>Page downloaded from an IBM site</Details>
</Transaction>
```

The two examples shown above form a simple XML coded configuration file for client capture.

When a configuration file is completed, you must register it to the product as described in “Registering Client Capture Applications” on page 72 and then build and distribute a profile for it.
Syntax Rules

1. The <ClientCapture> element must contain one <IExplorer_5x> subsection or one <Netscape_4x> subsection, or both. The two subsections can have the same set of children. This means that they can contain the same application transactions.

2. The <ClientCapture> element subsection can contain one or more <Transaction>. Each transaction tag defines a set of URLs to be captured and logged using <NAME> and <DETAIL> tags as detailed below.

3. The transaction item must have the following children:
   - One or more <FilterInclude> that defines which URLs are included in the transaction.
   - One <Name> that defines the ARM transaction name.
   - One <Details> that defines the ARM transaction details.

4. Both the <Name> and the <Details> tags can contain macros that are automatically expanded. Valid macros are:

   - $PROTOCOL$: the protocol of the URL, which can be: HTTP GET, HTTP POST, HTTPS GET, HTTPS POST.
     - **Note:** Netscape Client Capture is not able to distinguish get and post transactions. You always get HTTP GET or HTTPS GET.

   - $URL$: the URL (truncated to the first 127 characters).
     - **Note:** If the URL has an ending slash (/) it is ignored.

   - $URL_END$: the URL (truncated to the LAST 127 characters).

   - $FILTER$: the include filter that matched the URL.

   - $HOST$: the host part of the URL.

5. If an URL matches more than one transaction, then it is assigned to the first transaction definition it matches.

6. The file extension for client capture configuration files must be .xml.

Example

If you have registered an application "IBM_and_Tivoli_sites" with the above XML file, and an Internet Explorer user goes to http://www.tivoli.com/index.html, then the URL matches with the FilterInclude of the second transaction definition:

- application = IBM_and_Tivoli_sites.
- name = "$URL$" = "http://www.tivoli.com/index.html".
- details = page downloaded from a Tivoli site.

Ready-Made Configuration Files

Two examples are provided:

- A full listing of the file described above.
- A more advanced example including a number of popular Web sites.
Example 1 (sample.xml)

```xml
<?xml version="1.0" ?>
<!DOCTYPE ClientCapture SYSTEM "TAPMClientCapture.dtd">
<ClientCapture>
<!-- Transactions configuration for Internet Explorer 5x -->
<IExplorer_5x>
  <Transaction>
    <FilterInclude>http://*ibm.com*</FilterInclude>
    <Name>$URL$</Name>
    <Details>Page downloaded from an IBM site</Details>
  </Transaction>
  <Transaction>
    <FilterInclude>http://*tivoli.com*</FilterInclude>
    <Name>$URL$</Name>
    <Details>Page downloaded from a TIVOLI site</Details>
  </Transaction>
  <Transaction>
    <FilterInclude>http://*</FilterInclude>
    <Name>$URL$</Name>
    <Details>All transactions (not from IBM or TIVOLI sites)</Details>
  </Transaction>
</IExplorer_5x>

<!-- Transactions configuration for Netscape 4x -->
<Netscape_4x>
  <Transaction>
    <FilterInclude>http://*ibm.com*</FilterInclude>
    <Name>$URL$</Name>
    <Details>Page downloaded from an IBM site</Details>
  </Transaction>
  <Transaction>
    <FilterInclude>http://*tivoli.com*</FilterInclude>
    <Name>$URL$</Name>
    <Details>Page downloaded from a TIVOLI site</Details>
  </Transaction>
  <Transaction>
    <FilterInclude>http://*</FilterInclude>
    <Name>$URL$</Name>
    <Details>All transactions (not from IBM or TIVOLI sites)</Details>
  </Transaction>
</Netscape_4x>
</ClientCapture>
```

Example 2 (advanced_sample.xml)

```xml
<?xml version="1.0" ?>
<!DOCTYPE ClientCapture SYSTEM "TAPMClientCapture.dtd">
<ClientCapture>
<!-- Transactions configuration for Internet Explorer 5x -->
<IExplorer_5x>
  <Transaction>
    <FilterInclude>http://*ibm.com*</FilterInclude>
    <FilterInclude>http://*tivoli.com*</FilterInclude>
    <Name>IBM: $URL$</Name>
    <Details>Page downloaded from the IBM site: $HOST$</Details>
  </Transaction>
  <Transaction>
    <FilterInclude>http://*yahoo.com/*</FilterInclude>
    <FilterInclude>http://*altavista.com/*</FilterInclude>
    <FilterInclude>http://*lycos.com/*</FilterInclude>
    <Name>Search URL: $URL$</Name>
    <Details>Matching filter: $FILTER$</Details>
  </Transaction>
  <Transaction>
    <FilterInclude>http://*</FilterInclude>
    <Name>$URL$</Name>
    <Details>All transactions (not from IBM or TIVOLI sites)</Details>
  </Transaction>
</IExplorer_5x>
</ClientCapture>
```
<Transaction>
  <FilterInclude>http://*amazon.com*</FilterInclude>
  <Name>[$PROTOCOL$] URL: $URL_END$</Name>
  <Details>Amazon transaction</Details>
</Transaction>
</IEExplorer_5x>
<!-- Transactions configuration for Netscape 4x -->
<Netscape_4x>
  <Transaction>
    <FilterInclude>http://*ibm.com*</FilterInclude>
    <Name>IBM: $URL$</Name>
    <Details>Page downloaded from the IBM site: $HOST$</Details>
  </Transaction>
  <Transaction>
    <FilterInclude>http://*yahoo.com/*</FilterInclude>
    <FilterInclude>http://*altavista.com/*</FilterInclude>
    <FilterInclude>http://*lycos.com/*</FilterInclude>
    <Name>Search URL: $URL$</Name>
    <Details>Matching filter: $FILTER$</Details>
  </Transaction>
  <Transaction>
    <FilterInclude>http://*amazon.com*</FilterInclude>
    <Name>[$PROTOCOL$] URL: $URL_END$</Name>
    <Details>Amazon transaction</Details>
  </Transaction>
</Netscape_4x>
</ClientCapture>

Notes:

1. The XML file can contain localized characters in any variable but the file name must not contain any localized characters.

2. **Internet Explorer** - On some Win32 platforms, setting a transaction-include filter of “*” can cause the sensor to capture transactions other than Web page download. Use filters that start with “http:*” or “https:*” or “http*://*”

3. Do not edit the file \TAPMCCLAassID.xml that is installed on the endpoint.

4. Both for Netscape and Internet Explorer sensors, in the URL specified in the <INCLUDE_FILTER> the ending slash is ignored. For example: www.tivoli.com and www.tivoli.com/ defines the same filter.
One way to measure application performance is by transaction simulation. By using simulation you do not have to run the full application or to completion.

Tivoli Application Performance Management (the product) provides two methods of transaction simulation:

1. End-to-end probe platform probes. These are custom made items of code developed by IBM to use real applications such as Lotus Notes in a specific way that simulates end user actions and responses.

2. The product supports Mercury Interactive test scripts so you can set up a 'synthetic user' workstation that will run typical end-user transactions by executing prepared simulation scripts.

End-to-end Probe Platform

End-to-end probes (which currently, can only be used on windows platforms) enables providers of server-based applications to measure and record application performance from an end user’s point of view. EPP probes generate representative user transactions at scheduled intervals to simulate the end user actions and responses involved in using an application. The product supplies eight ready made probes that you can use for measuring and monitoring distributed applications.

An end-to-end method measures transactions across the network infrastructures, thus capturing performance as experienced by end users. (See Figure 30 on page 54). This approach applies what ever the complexity of the environment. Data from both "horizontal" and "vertical" perspectives can be routinely collected, integrated, analyzed and used to understand short and long term trends in performance, for capacity planning and to assist with root cause analysis when problems arise. The advantage of the EPP approach comes from the combination of these data, which when combined and correlated can help answer a range of business questions.
Description of the Supplied Probes

When you install EPP Support from your installation disk the following probes are transferred to your system (see “Installing Support for EPP” on page 23).

Each probe has the following three basic parameters in addition to those listed for the probe:

**execFreq**

specifies the number of probe transaction executions per hour

**Note:** For this first release of EPP, the product supports only one execution frequency per endpoint. This is to prevent having two different simulations executed in the same time frame.

**timeout**

Specifies the maximum number of seconds that a probe transaction can execute. If timeout is not specified a default value is set, which is different for each probe.

**targId**

Specifies the name of the main transaction. Default value is “default”.

The probes are:

<table>
<thead>
<tr>
<th>WebPageAccess</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
</tr>
<tr>
<td><strong>Parameters:</strong></td>
</tr>
<tr>
<td>proxyHost</td>
</tr>
<tr>
<td>proxyPort</td>
</tr>
<tr>
<td>proxyUserId</td>
</tr>
<tr>
<td>proxyPassword</td>
</tr>
<tr>
<td>targetUrl (mandatory)</td>
</tr>
<tr>
<td>negativeTest</td>
</tr>
</tbody>
</table>
WebPageSetAccess

**Description**
Downloads a set of URLs, which may reference pages or page elements, and may reside on one or more servers.

**Parameters:**
- **proxyHost**: Specifies the fully qualified host name of proxy server. If proxy host name is not specified, a direct connection will be used provided that if the probe system is not socksified.
- **proxyPort**: Specifies the port number of the proxy server.
- **proxyUserId**: Specifies the user id for accessing the proxy server. If user id is not specified, user authentication is skipped.
- **proxyPassword**: Specifies the password for proxy user authentication. If user id is not specified, the password is ignored.
- **targetUrl**: One or more target URLs. If more than one URL is specified, you must concatenate the URLs using the ‘+’ sign. For example: targetURL=www.ibm.com+www.tivoli.com Note that the URLs containing the equal character (=) are not supported.
- **timeoutUrl**: The timeout per URL get operation. Default value is the timeout value.

NotesAddrBookLookup

**Description**
Opens the address book located on the Notes mail server, looks up a name, returns the name specified as a parameter in the probe’s configuration file, or an error if name not found.

**Parameters:**
- **sourceIdFile**: Local Notes ID file
- **sourcePasswd**: Password for this id. This parameter is not mandatory, but it is necessary to successfully run a simulation if the specified notes ID has been configured to have a password.
- **sourceServer**: Mail server where the source data base resides (value="local" means perform test on a local DB).
- **sourceDb**: File name of the mail data base on the source Notes server. Default value is “Names.nsf”
- **searchString**: String to look for in the database, for example “Smith”
- **sourceDbView**: Database view to be used, default value is "People"

NotesServerAvail

**Description**
Connects to Notes port on server, issues a Notes command to list the Notes directories, checks the return.

**Parameters:**
- **idFile**: Absolute path to the Notes id file on the local system. This id must have read access to the server
- **passwd**: Password for this id. This parameter is not mandatory, but it is necessary to successfully run a simulation if the specified notes ID has been configured to have a password.
- **server**: Specifies the name of the Notes server that is to be tested. If the probe runs on Notes server, specify "local" to access local system.
### NotesCalendar

**Description**
Logs into server, opens mail on server, creates and deletes a calendar event and signs off.

**Parameters:**
- `sourceIdFile` (mandatory)
  - location of the local Notes user id file on the system
- `sourcePasswd` (mandatory)
  - Password for the above local Notes user id. This parameter is not mandatory, but it is necessary to successfully run a simulation if the specified notes ID has been configured to have a password.
- `sourceServer` (mandatory)
  - Notes server name in which the Notes user’s mail database is located
- `sourceNab` (mandatory)
  - Notes “name and address” book to be used for user name lookup. Default value is “Names.nsf”
- `remoteUser` (mandatory)
  - Notes user id of the remote user whose calendar will be targeted. The remoteUser value must be a valid value for the Database view value specified in NABview.
- `NABview` (mandatory)
  - Database view that is to be used. Default value is “People by Shortname”

### DatabaseAccess

**Description**
Signs on to a Notes server, opens a mail database, creates and saves a message, then deletes the message and closes the database. This test is effective in identifying servers with performance problems, and can also be used to compare performance before and after a server release upgrade.

**Parameters:**
- `idFile` (mandatory)
  - Location of the Notes id file on the system
- `passwd` (mandatory)
  - Password value for the Notes user id. This parameter is not mandatory, but it is necessary to successfully run a simulation if the specified user ID has been configured to have a password.
- `server` (mandatory)
  - Notes server name. If local database will be used for this test, use ‘local’ keyword
- `DB` (mandatory)
  - Name of the database on the Notes server. If a local server is used, make sure the absolute path the database is specified.
- `view` (mandatory)
  - Database view that is to be used. Default value is “Drafts”.

### IMAPAccess

**Description**
creates, deletes and retrieves messages. This probe can be used to assess the performance of a set of interactions with IMAP servers, including Lotus Notes (when enabled for IMAP) and Microsoft Exchange.

**Parameters:**
- `server` (mandatory)
  - The IMAP server name
- `user` (mandatory)
  - User ID for the account on the IMAP server
- `password` (mandatory)
  - User password for the account on the IMAP server. This parameter is not mandatory, but it is necessary to successfully run a simulation if the specified user has been configured to have a password.
- `folder`
  - The set of folders to be accessed on the IMAP server. Default value is “Inbox”. The path of the folder is expressed relative to the root. For example: folder=Inbox+Drafts+Public Folders/My new folder

### IMAPSearch

**Description**
Searches for a text string in a folder. It measures the performance of the search feature on IMAP server. Any IMAP server that supports search function can be measured including MS Exchange.
Parameters:

server (mandatory) the IMAP server name
user (mandatory) user ID on the IMAP server to log on with
password the user password to be used to log on to the IMAP server. This parameter is not mandatory, but it is necessary to successfully run a simulation if the specified user has been configured to have a password.
folder The name of the folder that contains searched messages. Default value is “Inbox”. To include all folders in user’s space, use “/” - default root. Example: folder=Inbox+Public Folders/My folder
searchTerm string that given fields would be searched for

TEST_PROBE Application

This is the sample application automatically installed with EPP support (see “Installing Support for EPP” on page 23). This application is associated with an EPP sample probe that generates a test transaction which is useful for testing or validating EPP. The probe has no application dependencies and produces no network traffic. It remains dormant for a consistent time interval. The only parameter necessary to execute this simulation is execFreq.

Using the Probes

Notes:

1. If you intend using the Lotus Notes probe it is better to manually initiate a Notes session on the target endpoint and send mail to verify that the configuration is correct before starting EPP for the first time.

2. The password parameter is not a mandatory value for Notes Probes.

To register an EPP probe to the product repository, you must specify the –d flag of the wmarregapp command. For example to register a WebAccess probe, you issue the following command:

wmarregapp -a WebPageSimulation -d WebPageAccess

EPP probes are registered (see “Registering EPP Probe Applications” on page 73) and used in profiles in the same way as other applications. When a new EPP simulation is started all the dependencies of the application are sent to the endpoint and new processes that drive and execute the simulation are started on that endpoint.

Note: Tivoli Application Performance Management EPP simulation does not support Windows 95 or 98 endpoints.

To start collecting data on an EPP probe, you issue the wmarstartcoll command specifying the –p flag with the appropriate parameter list. For example to start a simulation of a download of Web page www.ibm.com every five minutes, issue the following command:

wmarstartcoll -a WebPageSimulation -p "execFreq=12,targetUrl=www.ibm.com"

All the probes started on the same endpoint must use the same execFreq value. If you start a EPP with a different execFreq the collection start fails.

The sum of all probe time-outs must be less than the 3600 divided by execFreq. The following example is incorrect because using an execFreq value of 30, the calculation 3600/30 returns 120 and time-outs total 250:
start on the same endpoint:
   a) IMAPAccess with execFreq=30 and timeout=90
   b) IMAPSearch with execFreq=30 and timeout=80
   c) NotesServerAvail with execFreq=30 and timeout=80

The following example is correct because timeouts total 100:
start on the same endpoint:
   a) IMAPAccess with execFreq=30 and timeout=30
   b) IMAPSearch with execFreq=30 and timeout=40
   c) NotesServerAvail with execFreq=30 and timeout=30

If you want to change the execFreq of the unique EPP Probe running on the endpoint, you must stop the collection and restart it.

**Probe Outputs**

When you run an EPP probe it generates data through ARM API calls. The probes distributed with Tivoli Application Performance Management, generate one transaction that represents the overall simulation execution time. The WebPageSetAccess probe also generates one transaction for each URL specified in the targetUrl parameter.

**Simulation using Mercury Interactive Scripts**

ARM API calls are inserted into the scripts. When the transaction starts, an arm_start call is made, and when it completes, an arm_stop call is made. The product agent measures the time between the start and stop call. This is the response time for the transaction.

If the workstation is on the same production network, and uses the same servers as the real end-users, it receives response times that are representative of those received by end-users.

Transaction simulation is easy to set up and has two distinct advantages:

- It establishes a consistent baseline for trend analysis.
- It probes server and application availability during normal operations or at a time when few users may be active, such as in the middle of the night.

You should be aware that the scripts cause real transactions to be run. This may increase the load on your server infrastructure and your network, in the same way as transactions from a real end-user. You should be careful to understand and limit, the load caused by these extra transactions. You also need to carefully plan what you simulate and how you do this.

When you put into effect transaction simulation, you should plan to dedicate a machine in your production environment for the task. Try to use a hardware and software configuration that is identical to a real user machine.

The product supports scripts that are created with test toolkits marketed by Mercury Interactive. It can replay LoadRunner types of script.

**Virtual User Generator**

To develop and create virtual user scripts you use the Mercury Interactive Virtual User Generator (VuGen). These scripts emulate the transactions of human users by performing the same business processes that real end-users perform. VuGen works on the protocol level and communicates directly with the server.
VuGen supports a variety of protocols for which you can create user scripts.

Preparing Virtual User Scripts
After you have recorded a virtual user script, you add lr functions to the script. These functions enable a script to monitor application performance in the product environment.

This section describes:
- Understanding VuGen lr functions.
- ARMing virtual user scripts.
- User-defined functions.
- Custom data.

About Preparing Virtual User Scripts
To use a virtual user script in the product environment, you must add function calls to your script that generate ARM API calls to the product. You insert special lr functions that mark the start and end of the transaction you want to measure and define the name of the transaction.

You define a transaction in your script for each business process whose response time you want to measure.

You manually insert the lr functions into your script after recording. Once a script is “ARMed,” that is ready for use with the product, you integrate it into a profile by registering the virtual user script. For details on registering test cases in the product, refer to Chapter 9, “Registering Applications” on page 71.

Understanding VuGen lr Functions
The application name sent to the product is the name used when you register the simulation (see Chapter 9, “Registering Applications” on page 71). The user name sent to ARM is “*”.

To instrument a virtual user script with ARM calls, add the Mercury Interactive lr functions supported by the product, after recording the script. You can insert these calls into your script using the Step Generator, where you can find the lr functions in the ARM category.

- **lr_start_transaction** marks the start of an instance of an ARM transaction by making an arm_start API call. You define a transaction for each application business process whose response time you want to measure.

```c
int lr_start_transaction(const char *TransactionName);
```
TransactionName must not exceed 128 characters. This function returns 0 if OK, -1 if there is an error.

- **lr_end_transaction** marks the end of an ARM transaction by making an arm_stop API call.

```c
int lr_end_transaction(
    const char *TransactionName,
    const int Value);
```

TransactionName - This name must match the name used on the lr_start_transaction function.

Value - The completion status that you set for the transaction; good and failed transactions are logged separately by the product. This can be set to either LR_PASS, LR_AUTO, or LR_FAIL. If it is set to LR_AUTO this is converted to LR_PASS. This function returns 0 if OK, -1 if there is an error.

**Note:** When you are monitoring more than one transaction in an application, you must insert lr_end_application after the last lr_end_transaction, to mark the end of that collection.

### Instrumenting Virtual User Scripts

You add ARM commands to a virtual user script, to prepare it for use with the product.

To instrument a virtual script with ARM calls proceed as follows:

1. Insert **lr_start_transaction** and **lr_end_transaction** statements to define the business processes, or transactions, whose response time you want to measure.

2. Place the **lr_start_transaction** statement immediately before the action you want to measure.

3. Place the **lr_end_transaction** statement after the last statement of the transaction. For example:

```c
lr_start_transaction("Tivoli_main_page");
URL ("http://www.tivoli.com");
lr_end_transaction("Tivoli_main_page", LR_PASS);
```
The ARM (Application Response Measurement) API is a set of standard Application Programming Interface (API) calls, that enable you to make focused measurements of application performance. Several prominent IT companies have agreed together on these calls.

On-line documentation for ARM is available for download at:
http://www.cmg.org/regions/cmgarmw

You can reach a discussion platform for users at cmgarm@cmg.org. The ARM Software Developer’s Kit available at the current time is version 2.0.

About ARM

The API is designed to allow you to measure the performance of any application. The most likely use of the application monitoring capability is to measure response time. However, analysis of a series of timings of the same application event or process, can also be used to build records of application availability and for application usage.

To measure response times for an application, the application must be instrumented according to the ARM API standard. This means that ARM API calls must be embedded in the application code.

ARM library files and include files that are needed to compile and link your application, are provided automatically during the product installation process. They are located in Tivoli_installation_dir\bin\lcf_bundle\Mar<interp>.

ARM API Version 2 can be used to determine why an application is not performing well and also to report on its availability. It is possible to pass the information regarding the number of bytes that are transferred to ARM 2 API. This information can be used in a graph to display the length of different sizes of transactions.

ARM API Call Syntax

There are six ARM API calls, as follows:

**arm_init**

This call is used to define an application. It must be made before any of the other ARM API calls related to that application.

**Syntax:**

`appl_id=arm_init(appl_name, appl_user_id, flags, data, data_size)`

**Example:**

`appl_handle = arm_init("application", "+",0,0,0);`
The application was defined with a name of *application*, and the *appl_user_id* field was set to *, so that the product will change it to the login user ID of the person running the application.

The handle returned by this call is placed in the variable called *appl_handle*, and this will be used on the **arm_getid** call.

**arm_getid**

This call is used to define a transaction, which must be a child of an application.

Syntax: 
```
tran_id = arm_getid(appl_id, tran_name, tran_detail, flags, data, data_size)
```

Example:
```
getid_handle = arm_getid(appl_handle,"transaction", "W95/NTwks",0,0,0);
```

The transaction was defined with a name of *transaction*, picking up the variable *appl_handle* that was returned by the **arm_init** call. The *tran_detail* field was set to W95/NTwks, to provide additional information on a transaction.

The handle returned by this call is placed in the variable called *getid_handle*, which will be used on the **arm_start** call.

**arm_start**

You make this call when the transaction starts running. It starts the response time clock.

Syntax: 
```
start_handle = arm_start(tran_id, flags, data, data_size)
```

Example:
```
start_handle = arm_start(getid_handle,0,0,0);
```

The call uses the variable *getid_handle* that was returned by the **arm_getid** call.

The handle returned by this call is placed in the variable called *start_handle*, which will be used on the **arm_update** call, if it is made, and on the **arm_stop** call.

**arm_update**

This call is optional. It can be used as a heartbeat, to check the progress of a long-running transaction. The call can be used as many times as you want, between the **arm_start** and **arm_stop** calls. An **arm_update** call example is:

Syntax: 
```
error_status = arm_update (start_handle, flags, data, data_size)
```

Example:
```
updaterc = arm_update(start_handle,0,0,0);
```

The call uses the variable *start_handle* that was returned by the **arm_start** call.

This call returns an error code, rather than a handle. The error code is placed in the variable called *updaterc*.

**arm_stop**

You make this call when the transaction stops running. It stops the response time clock.

Syntax: 
```
error_status = arm_stop (start_handle, tran_status, flags, data, data_size)
```

Example:
```
stoprc = arm_stop(start_handle,ARM_GOOD,0,0,0);
```

The call uses the variable *start_handle* that was returned by the **arm_start** call.
The transaction status field was set to **ARM_GOOD**, for simplicity. The other options are ARM_ABORT and ARM_FAILED. These can be used by Distributed Monitoring to create alarms if the transaction does not complete the way you would like it to.

This call returns an error code, rather than a handle. The error code is placed in the variable called `stoprc`.

`arm_end`

This call should be made when the application terminates. It causes a cleanup of the memory that the ARM Agent has allocated for this application.

**Syntax:**

```
error_status = arm_end(aid, flags, data, data_size)
```

**Example:**

```
endrc = arm_end(appl_handle, 0, 0, 0);
```

The call uses the variable `appl_handle` that was returned by the `arm_init` call.

This call returns an error code, rather than a handle. The error code is placed in the variable called `endrc`.

### Return Codes

Each API call can produce a return code when an error occurs:

- **-1** An invalid value has been used in the API call.
- **-2** The engine is not running.
- **-8** An error occurred when communicating with the product engine.

### Case Sensitivity

ARM API calls are C functions, and are therefore case sensitive. All the ARM API functions use lowercase throughout. If you code any part of the API calls in uppercase, note the following:

- If you are programming in C or C++, then the linker fails to resolve the external functions.
- If you are using a language other than C, the call fails with a return of -1.

### Using the ARM Calls

The benefit of this approach is that you can place the calls that start and stop the response time clock, in exactly the parts of the application that you want to measure. You define individual applications and transactions within a program, and place the ARM API calls at transaction start and transaction end.
Of course you need access to the original code. You also need to invest some programming effort in identifying the places in the application code at which you want to call the ARM API, and then insert the API calls there.

You can provide additional information about the transaction, such as:

- The number of bytes or records being processed.
- The state of the application, for example the length of a work queue, at the moment that the transaction is being processed.

This information (called application-defined metrics) is useful to help understand response times, and how the application can be tuned to better performance.

The product supports ARM 2.0 additional metrics for summary data. When distributing a profile (or starting a collection using the *wmarstartcoll* command), you can decide if metric data must be uploaded. This is done setting the database detail level to 3 (Maximum). See “Database Settings for a Collection Record” on page 91. Note that the ARM 2.0 additional metrics are not supported for instance data.

Aggregated data Numeric Ids and ARM String metric types are not uploaded to the database.
Using ARM Transaction Correlators

This chapter describes a facility known as correlation, that allows you to capture data which provides a “closer look” inside transactions. Information within the chapter is organized as follows:

- What is ARM Transaction Correlation?
- How Correlation Works
- Multi-tier Transactions
- Where to use Correlation
- Collecting Correlators with Tivoli Application Performance Management

Correlation enables you to measure which sub transaction is causing a parent transaction to perform slowly. This is possible since Version 2 of the ARM API allows you to distinguish which sub transaction belongs to which parent transaction.

What is ARM Transaction Correlation?

Most client/server transactions consist of one transaction that is visible to you, and several nested component transactions that are invoked by that single visible transaction. These nested component transactions are the children of the parent transaction (or the child of another child component transaction). It is useful to know how much each component transaction contributes to the total response time of the visible transaction. Similarly, a failure in one of the component transactions can often lead to a failure in the visible transaction, and this information is also useful.

The product supports ARM 2.0 correlators (see Chapter 7, “Instrumenting an Application for ARM”). These correlators enable you to monitor parent-child relationships between transactions on different machines, while measuring applications response time using ARM calls. Transaction monitoring can be as follows:

- Break down a transaction into its component parts. This allows you to see the contribution to the total response time that each component of the application makes.
- Follow the path that the transaction took. Any given transaction can often take more than one possible path.

By correlating response times with the resources allocated, you can get a much better idea of how to configure applications and their environment. Tivoli application performance management can determine whether there are any transactions that are being processed or waiting to be processed. This will help to alleviate any bottlenecks that may exist.
How Correlation Works

It is important to understand the difference between an application and a transaction. An application is generally considered to be a computer program that has been purpose-written, whereas a transaction can be regarded as a specific set of input data that triggers the execution of a specific process or job inside an application. An application can have several independent transactions.

To use transaction correlation the instrumented application needs to “ask” for a correlator to be generated when calling the engine. In fact the default setting for all instrumented applications is they call for correlators and the engine generates them.

Every instance of every transaction of every application gets a unique token or correlator. This is guaranteed, because the correlator contains the handle returned when the arm_start call is made, and the handle returned when a transaction is registered.

Figure 33. Example of a distributed transaction

To use transaction correlation the instrumented application needs to “ask” for a correlator to be generated when calling the engine. In fact the default setting for all instrumented applications is they call for correlators and the engine generates them.

Every instance of every transaction of every application gets a unique token or correlator. This is guaranteed, because the correlator contains the handle returned when the arm_start call is made, and the handle returned when a transaction is registered.

Figure 34. Generating a correlator
A correlator can be passed to a child transaction (the engine sets a flag in the buffer to mark it as a parent correlator). In this case the correlator records transaction time for the child.

![Diagram of passing the correlator](image)

**Figure 35. Passing the correlator**

If the child transaction requests a service, a further correlator is generated by the engine to hold the details of that transaction and the engine sets a flag to mark the new correlator as a child correlator.

### Multi-tier Transactions

Using ARM transaction correlation can provide major benefits in a multi-tiered transaction environment like that shown in [Figure 36 on page 68](#).

You can use transaction correlation when the transaction has components that span several applications or systems. This approach is complex as it requires changes to all the applications involved in processing components of the transaction, but it is the most accurate way to track transaction response time spanning systems.

Using a client server model, ARM can correlate transactions. In this model, the server provides a service to the client through an interface. The client obtains a service from the server by issuing requests. The server may, in turn, be a client to another server in order to obtain further information that it does not have.
In the example illustrated above, a user requests action at client A via a mouse or keyboard action (1). Client A processes the transaction until it needs a service from server B, which in turn requests a service from server C and then later from server D before returning correlators.

- The user sees just one transaction (1 to 14).
- Client A saw one transaction as a client (2 to 13).
- Server B saw one transaction as a server (3 to 12) and two others (4 to 7 and 8 to 11) as a client
- Servers C and D saw just one transaction.

In the case shown in Figure 36, the application must request a new correlator at each transfer.

**Where to use Correlation**

To start monitoring the application, the transactions need to be monitored first. The data that was generated is used for comparison and trend analysis. Correlation analysis will only be done if there are unsatisfactory response times obtained. This will occur at the client where the slow response times are being generated. When logging has been turned on, it is possible to capture correlation information. You can set logging correlation information for every application by using the Data Storage dialog ("Instance Data" information) or the `wmarstartcoll` command with the `-n` option. See “Database Settings for a Collection Record” on page 91 and “wmarstartcoll” on page 201.

**Collecting Correlators with Tivoli Application Performance Management**

When correlator generation is not turned off the agent generates correlators when the instrumented application asks for them. Correlator generation can be turned off in 3 ways:
1. Checking the checkbox on the Add Entry in the Profile dialog. This disables correlator generation for the application specified in the profile.

![Add Entry to the Profile](image)

**Figure 37. Deselecting correlation**

2. Using the `wmarsetstatus` command with `GenerateCorrelator= false ep_label`. This disables correlator generation for all applications running on the specified endpoint.

3. Using the `wmarstartcoll` command with `-k` `-a` application name `ep_label`. This disables correlator generation for the application specified by the `wmarstartcoll` command.

When the agent generates correlators, it logs them into local files at the endpoints which are periodically summarized. See Chapter 13, “Using Performance Data” on page 133. These files are then aggregated and moved to the database.

Generated correlators can be logged locally and stored on the database. Correlation logging can be modified in two ways:

1. Using the Data Storage button in the profile (command line equivalent `wmarstartcoll -n`). This modifies the logging settings for the application specified in the profile.

2. Using the `wmarsetstatus` Instance Logging= true `ep_label`. This modifies the logging settings for all the applications running on the endpoint.
Registering Applications

When you have installed Tivoli Application Performance Management (the product), you must perform the following tasks before you can use it:

1. Prepare applications that you want to measure. They may be:
   - Client capture applications (see Chapter 5).
   - Applications instrumented with ARM API calls (see Chapter 7).
   - End-to-end probes to run tests on applications (see Chapter 6).
   - Simulation (virtual user) scripts with ARM API calls embedded in them, created using Mercury Interactive toolkits (see Chapter 6).

2. Register applications, associated with any of the methods listed above in the product repository.

During the registration process the Tivoli management enterprise files the applications in the region database so that they can be retrieved by the product during the profile build process (see Chapter 10, “Building Profiles” on page 77).

Note: If you are a user of Version 1.1 or earlier of the product you should note that `wmarrregapp` is no longer a shell script and is now a command.
Registering Client Capture Applications

Before you can set up data collection on a client capture configuration file, you must first register it to the product. Registered files then appear in the Add Entry to the Profile dialog as available applications that can be included in profiles.

You generate each configuration file as a *.XML file (see Chapter 5, “Client Capture” and if necessary, you then move the file or files to the managed node where the product is installed.

You can now register the files. This is done using the wmarregapp command at the command line as follows:

1. Open a command window and source the Tivoli Server environment variables using the setup_env command shown here.

<table>
<thead>
<tr>
<th>Platform</th>
</tr>
</thead>
<tbody>
<tr>
<td>Windows</td>
</tr>
<tr>
<td>UNIX</td>
</tr>
</tbody>
</table>

   Platform Command
   Windows "%windir\system32\drivers\etc\tivoli\setup_env.cmd"
   UNIX . /etc/Tivoli/setup_env.sh

2. Run

   wmarregapp -a <client_capture> -f
   C:...\tapm\client capture\file.xml

Notes:

1. The XML file can contain localized characters in any variable but the file name must not contain any localized characters.
2. At registration time, the syntax of a configuration (XML) file is checked before it is made available in the repository.
3. <client_capture>, the application name specified by the –a flag, is the name that will appear in the Add Entry to the Profile window (See “Adding an Application to a Profile” on page 83). It is the name recognized by the whole product system. You can use any name of your choice.
4. The –f parameter is the path name of the configuration file on the managed node where you run the command. If the path includes blanks (the space character) then the command must be enclosed inside quotation marks. Both absolute and relative path names are accepted.
5. On UNIX systems, the user who runs the wmarregapp command needs to have the permission to write in the product repository directory ($BINDIR/../generic/Mar/RepositoryData) and in the temporary directory (the directory you get by issuing the wtemp command in the Tivoli environment) on the TMR server.

Registering Instrumented Applications

Before you can set up data collection on an application, you must first register that application to the product. Registered applications then appear in the Add Entry to the Profile dialog as a list of valid applications that can be included in profiles. You register an application as follows:
1. Open a command window and source the Tivoli Server environment variables using the `setup_env` command shown here.

<table>
<thead>
<tr>
<th>Platform</th>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>Windows</td>
<td><code>&quot;%windir%\system32\drivers\etc\tivoli\setup_env.cmd&quot;</code></td>
</tr>
<tr>
<td>UNIX</td>
<td><code>. /etc/Tivoli/setup_env.sh</code></td>
</tr>
</tbody>
</table>

2. Run the `wmarregapp` command, to register the application. At the command line type the following command:

   `wmarregapp -a application_name`

   Where `application_name` matches the name used on the `arm_init` call within the application.

3. Repeat Step 2 for each application you want to register.

### Registering EPP Probe Applications

Before you can set up data collection on an EPP probe, you must first register the application associated with that probe to the product. Registered applications then appear in the Add Entry to the Profile dialog as a list of valid applications that can be included in profiles. You register EPP applications as follows:

1. Open a command window and source the Tivoli Server environment variables using the `setup_env` command shown here.

<table>
<thead>
<tr>
<th>Platform</th>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>Windows</td>
<td><code>&quot;%windir%\system32\drivers\etc\tivoli\setup_env.cmd&quot;</code></td>
</tr>
<tr>
<td>UNIX</td>
<td><code>. /etc/Tivoli/setup_env.sh</code></td>
</tr>
</tbody>
</table>

2. Run the `wmarregapp` command, to register the application. At the command line type the following command:

   `wmarregapp -a epp_application_name -d <probe_name>`

   Where `probe_name` is the name of one of the probes supported by the product. See “Description of the Supplied Probes” on page 54.

3. Repeat Step 2 for each application you want to register.

**Note:** The `epp_application_name` specified by the `-a` flag is the name that will appear in the Add Entry to the Profile window (See “Adding an Application to a Profile” on page 83). You can use any name of your choice.

A confirmation message is displayed, to identify the item that has been registered. You can use the `wmargetapp` command to obtain a list of applications and scripts that are registered.

**Note:** On UNIX systems, the user who runs the wmarregapp command needs to have the permission to write in the product repository directory ($BINDIR/../generic/Mar/RepositoryData) and in the temporary directory (the directory you get by issuing the wtemp command in the Tivoli environment) on the TMR server.
Registering Virtual User Applications

Before you can set up data collection on a simulation script, you must first register that script to the product. Registered scripts then appear in the Add Entry to the Profile dialog as a list of valid applications that can be included in profiles.

You generate each script with the corresponding toolkit. Each simulation comprises a number of different files that the product needs in order to manage them.

Note: The toolkit generates files with an extension .usr in the same directory as the other script files. The full path to that .usr file must be supplied for registration.

If the scripts have been prepared on a workstation that is not a product managed node, you must map the location of the scripts during the registration process described below.

Registering VuGen Scripts

If you use the VuGen toolkit to create a script named vugen_simulation and save it in the directory c:\tapm, a new sub directory called c:\tapm\vugen_simulation is created. This directory contains the file vugen_simulation.usr which identifies the script.

The product needs all the files under c:\tapm\vugen_simulation. This means that if the script has to be moved to a different machine for registration, the entire c:\tapm\vugen_simulation directory and its sub directories must be moved.

You can now register the script.

1. Open a command window and source the Tivoli Server environment variables using the setup_env command shown here.

<table>
<thead>
<tr>
<th>Platform</th>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>Windows</td>
<td>&quot;%windir%\system32\drivers\etc\tivoli\setup_env.cmd&quot;</td>
</tr>
<tr>
<td>UNIX</td>
<td>. /etc/Tivoli/setup_env.sh</td>
</tr>
</tbody>
</table>

2. Run the wmarregapp command, to register the script. At the command line type the following command:

   wmarregapp -a <vugen_application_name> -f C:\tapm\vugen_simulation\vugen_simulation.usr

Notes:

1. <vugen_application_name> is the name that will appear in the Add Entry to the Profile window (See “Adding an Application to a Profile” on page 83). You can use any name of your choice.

2. The -f parameter is the path name of the configuration file on the managed node where you run the command. If the path includes blanks (the space character) then the command must be enclosed inside quotation marks. Both absolute and relative path names are accepted.

3. A confirmation message is displayed, to identify the item that has been registered. You can use the wmaorgetapp command to obtain a list of applications and scripts that are registered.

4. On UNIX systems, the user who runs the wmarregapp command needs to have the permission to write in the product repository directory.
($BINDIR/../generic/Mar/RepositoryData) and in the temporary directory (the directory you get by issuing the wtemp command in the Tivoli environment) on the TMR server.

Advanced Use of the wmarregapp Command

**Reusing scripts** - It is possible that a script or file needs to be used by more than one application being monitored by the product. In this case, it is desirable that the physical copy of the set of files on the managed node occurs only once. Make sure of this by registering them with only the first application that uses them.

Registration is carried out by running the `wmarregapp` command as follows:

```
  wmarregapp –a <application_name1> -l label_1 -f <full_path_name>
```

This command copies the files, registers `application_name_1` and applies `label_1` to the copied files for future reuse.

To register other applications that use the same script files run:

```
  wmarregapp –a <application_name_2> -l label_1
```

to register the fact that `appl_name_2` uses a set of files already copied to the managed node as `label_1`.

**Modifying an Application Definition**

If you need to change the specification for a registered application, by changing the file set associated with it, you must run the `wmarregapp` command specifying the `-w` flag as follows:

```
  wmarregapp –a <application_name> -l label -f file_path -w
```

This deletes `appl_name` from the set of registered applications if it exists and performs a new registration.

**Removing Applications from the Repository**

You use the `wmardelapp` CLI command (see “wmardelapp” on page 185) to remove an application from the product repository.
Building Profiles

This chapter provides a description of the product profile. It describes how you use the product and how you can build up a profile and distribute it to network endpoints. Information is organized as follows:
- **Tivoli Application Performance Management Profiles**
- **Adding an Entry to a Profile**
- **Adding an Application to a Profile**
- **Adding Schedule Information**
- **Database Settings for a Collection Record**
- **Setting Simulation Extended Parameter Values**
- **Changing Collection Details**
- **Editing an Entry in a Profile**
- **Distributing a Profile**

This chapter uses extensive reference to the product GUIs. To improve clarity, often only the relevant portion of the window or dialog is illustrated.

**Tivoli Application Performance Management Profiles**

A Tivoli Application Performance Management profile (MarProfile) lists a series of product applications with all the parameters needed to start a collection on a generic endpoint. You use the product profiles to define your performance monitoring requirements. For example, in a profile you define which applications are to be monitored, when they are to be monitored, and by selective distribution of profiles, the machines on which they should run.

Each profile prepared with the product is a special data container that holds such items as:
- List of applications to measure.
- Timings:
  - When to measure,
  - Data collection periods,
  - How often to capture data.
- Simulation settings (parameters).
- Data:
  - Retrieval instructions.
- Storage instructions.

Profiles are stored centrally and can be distributed to endpoints.

The product profiles, called MARProfiles, are similar to other Tivoli profiles that are associated with other products.

Before you integrate applications of any type, supported by your installation into profiles, you must register them to the Tivoli Application Performance Management repository. For details on registering any type of application, refer to Chapter 9, “Registering Applications”.

You distribute profiles to endpoints (target machines) within your region where you want to measure activity of applications.

Key Points About Profiles

A Tivoli profile is a collection of information. There is a relationship between profiles, profile managers, policy regions and endpoints:

- Profile managers contain profiles; they also administer profiles and subscriber lists (group of endpoints).
- Profile managers are created within a policy region.
- MarProfiles must be a managed resource of that region
- A profile manager can not contain two identical profiles.
- Different profiles can contain the same applications to be measured, but the last profile to arrive at the endpoint overwrites the others.

Creating an Empty Profile

You create an empty profile from the Profile Manager window, after starting Tivoli Management Framework. Empty profiles must be of the type MarProfile. Check that this is among the Managed Resources of your region as described in “Adding MarProfile as a Managed Resource” on page 40.

Authorization The required authorization role for creating a profile is Senior.

To create a profile:

1. In the main Tivoli desktop GUI, double click the region in which you will be using the product.
The Policy region window is displayed.

2. In the Policy Region window, select the profile manager you wish to work with.

The Profile Manager window is displayed.

3. In the Profile Manager window, select **Profile** from the **Create** menu.
The Create Profile dialog is displayed.

4. Verify that MarProfile is present in the Type box. If it is not, follow instructions for “Adding MarProfile as a Managed Resource” on page 40 before continuing with the next step.

5. Enter a name for your profile in the Name/Icon Label box.
   Profile names can include any alphanumeric character, an underscore (_), a dash (-), a period (.), or a space.

6. Click Create & Close. The Profile Manager window is displayed with your new profile icon.
Adding an Entry to a Profile

Before you can add applications to a profile you must:

- Prepare the application (see “How the Product is Used” on page 6).
- Transfer and register the prepared files to a managed node of the system (see Chapter 9, “Registering Applications”).

Adding application collections into a profile is a multi-stage task in which you must complete the following actions before clicking the Add button or the Add & Close button:

1. Decide whether you want correlators to be collected.
2. Accept the default bucket limits or redefine them.
3. Add a data collection schedule for the application.
4. Accept the default database storage settings for the data collection file, or redefine them.
5. If the application has a script, then you must set the run interval and add simulation parameters if necessary.
6. If the application requires other specific parameters (for example for an EPP Probe), you need to define them.

To add an application you need to open the profile in which you want your application to be contained.

Figure 43. Creating a profile, new icon added
When you first start adding application collections to an empty profile, the Application Availability Profile window application list is empty and only the **Add Entry** button is enabled.

1. In the Application Availability Profile window, click **Add Entry**.
2. The Add Entry to the Profile window is displayed.

**Figure 44. Starting the product**

**Figure 45. Add Entry to Profile dialog**

**Note:** The Simulations Settings button is not active in the view above, because the application selected is not a simulation.
Adding an Application to a Profile

The Application panel contains a list of applications that are registered in the application repository. Select the entry that you want to add, by highlighting it in the list.

Notes:

1. If the Application display field is empty, there are no applications registered in the repository and the only available option is to register an application (see Chapter 9, “Registering Applications”), and then click the Refresh button.

2. If another person sharing the product on the network registers an application after you have selected the Add Entry to the Profile dialog, it is not displayed in the Application panel until the Refresh button is selected.

Figure 46. Updating the list of available applications

The following subsections describe the various tasks associated with building a profile. When it is completed, changes can be made either by using the Edit an Entry in the Profile dialog (see “Editing an Entry in a Profile” on page 95), or by using the wmarstartcoll command (see “wmarstartcoll” on page 201 for full syntax). Both of these methods change application settings at an endpoint. For each of the steps described below, the applicable portion of the wmarstartcoll command is given to produce the same setting as described and displayed in the GUI screen capture.

Command Line

The application selected above as our chosen example is identified as armAppl2, therefore any changes made to that application through the CLI must start as follows:

```
wmarstartcoll -a armAppl2.....
```

followed by appropriate flags and values.

Creating a Transaction Filter

You can use the transaction filter to specify which transactions should be monitored.

A transaction is identified by its arm_getid (see Chapter 7, “Instrumenting an Application for ARM” on page 61) call inserted into the application code, or the lr_start_transaction (see “Instrumenting Virtual User Scripts” on page 60) call inserted into a simulation script.

Best use is made of the transaction filter when transactions are identified in such a way that similar transactions contain similar names.
To set up a filter proceed as follows:

1. In the Transaction Filter box, type one or more characters accompanied by the * or ? wild card symbols to form a filter pattern.

2. Any transaction matching the given pattern is monitored. For example, the entry "*tn", filters in all transaction names that end with the letters tn. If the filter contained the entry "tranf??x", all eight-character transaction names beginning with the letters tranf and ending with the letter x would be monitored.

In the example below, collections for all transactions that have names beginning with the characters “armapt” are included.

![Transaction Filter](image)

*Figure 47. Creating a transaction filter*

**Command Line**

The equivalent part of the CLI for setting the values above is:

```
wmarsstartcoll -a armAppl2... -t armapt*....
```

**Choosing Bucket Limits**

A bucket is an interval of time. You can specify buckets and bucket limits as a way of gathering extra information about application response times. An application transaction may run many times during a collection period. Response times recorded during that collection interval are averaged to provide a single average response time. For closer analysis you can specify bucket limits which record how many responses fall inside a bucket.

For example an application runs ten times and response times are as follows:

\[
3.2, 3.7, 2.9, \ldots, 3.5 \quad \text{Response time (Average) = 3.1.}
\]

By setting bucket limits so that the buckets are clustered around the time 2.5 to 3.7 at 200 millisecond limits you capture the number of times the response time was within the bands 2.5 to 2.7, 2.7 to 2.9, 2.9 to 3.1, 3.1 to 3.3, 3.3 to 3.5, and 3.5 to 3.7 as shown below.
This illustrates that although the average response time was 3.1, the greatest number of response times were 3 seconds and there was an interesting cluster of responses that took 3.4 seconds.

You can specify up to six intermediate bucket limits between the fixed lower limit (zero) and upper limit (infinity). This allows you to define seven buckets.

The default bucket limits, in milliseconds, are as follows:

<table>
<thead>
<tr>
<th>Bucket</th>
<th>Lower boundary</th>
<th>Upper boundary</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>100</td>
</tr>
<tr>
<td>2</td>
<td>100</td>
<td>500</td>
</tr>
<tr>
<td>3</td>
<td>500</td>
<td>1000</td>
</tr>
<tr>
<td>4</td>
<td>1000</td>
<td>2000</td>
</tr>
<tr>
<td>5</td>
<td>2000</td>
<td>5000</td>
</tr>
<tr>
<td>6</td>
<td>5000</td>
<td>10000</td>
</tr>
<tr>
<td>7</td>
<td>10000</td>
<td>infinity</td>
</tr>
</tbody>
</table>

To change the bucket boundaries:

1. Use the GUI slide bar to move to and display the bucket range you want to change.
2. Select the boundary box to be changed and overwrite the limit contained in that box.
3. Repeat steps 1 and 2 as often as necessary.

Notes:

1. At least one set of bucket limits must be specified.
2. In the example below, the default bucket limits (0 to 100 and 100 to 500) have been accepted, but content of the other limits boxes have been deleted so that a set of two buckets has been specified. However; you will still get an unwanted third bucket with limits 500 to infinity.
Command Line

The equivalent part of the CLI for setting the values above is:
\[ \text{wmarstartcoll –a armAppl2...–b 100,500..} \]

Enabling and Disabling Correlators

The product supports ARM 2.0 correlators (See Chapter 8, “Using ARM Transaction Correlators” on page 65). By default, the collection of correlators is active for any ARM instrumented applications. You must select the Disable Correlation Generation checkbox on the Add Application to a Profile dialog to deselected correlation for that application.

Adding Schedule Information

Adding schedule information is a two-part process:

- The data collection times for the application, control the overall period for monitoring an application and the frequency with which transaction data is to be collected.
- The schedule rules set more specific time spans within the overall monitoring period when you want data collections to occur.

In the Add Entry to the Profile window, click Schedule. The Add Scheduling Information dialog appears.
This window has three panels:

1. A panel for setting data collection period and frequency.
3. A panel for creating and editing collection rules.

To work in any of the three panels, the Add Scheduling Information window must be called.

**Setting the Data Collection Period**

To set the collection period and also specify how often you want to collect data during this period, proceed as follows:

1. In the top left of the Add Scheduling Information dialog, choose a start and stop date to define the monitoring period.
The stop date must be later than, or can be the same as the start date.

2. Choose a data collection interval (how often you want monitor data to be collected).

3. Add one or more schedule rules.

**Command Line**

The equivalent part of the CLI for setting the values above is:

```
wmstartcoll -a armApp12 -s 2001-01-20 -e 2001-01-31 -i 1...
```

**Managing Schedule Rules for an Application Collection**

The application schedule controls the overall period for monitoring an application and the frequency with which transaction data is to be collected. Schedules can also contain information to break up the overall collection period into active and inactive periods; this is applied as one or more rules.

You create a rule as follows:

1. Call up the Add Scheduling Information window.

2. Click the **New Rule** button in the **Schedule Rules** panel.
3. In the **Rule Editor** group box, type in a name for the rule.

4. You should now decide the day or days on which you want the collections to be active during the collection period. You may select one or more items in the list of weekdays.

   **Note:** Use the **Shift** key or **Ctrl** key as necessary, to select more than one day from the list.

   **Figure 52. Setting up a collection rule**

   **Figure 53. Naming a collection rule**

   **Note:** If you do not supply a name, a pop-up reminder is displayed when you try to save.
5. Set start and stop times for the collection activity. Select the All Day check box for continuous data collection.

6. Click the Set Rule button. Your new rule is added to the Schedule rules list. You can create another rule for addition to the list or delete an existing rule from the list.

**Note:** To check the details of any rule, select the rule in the Schedule Rules list. Its settings are displayed in the Rule Editor panel.

**Command Line**

The equivalent part of the CLI for setting the values above is:
Synchronizing Data Collection

When you define the collection interval and collection start time, remember that transaction data is recorded at the end of the first collection interval. For example, if you define a collection interval of 20 minutes, starting at 8:50, a transaction that occurs at 8:53 is not recorded in the data record until 9:10.

This means that when the data files are aggregated to hourly boundaries and uploaded to the database (see “Uploading Data from the Endpoint to the Database” on page 136), the data for the 8:53 transaction, is part of the data related to following hour (9:00 to 9:59). This appears as VALUE_9 in the product database.

To have the transaction recorded closer to the time it occurs, set the collection interval at the minimum value (1 minute).

Database Settings for a Collection Record

The amount of information sent to the database can become quite large, because it is affected by:

- How many users there are.
- How many endpoints are included.
- How many applications are being monitored at each endpoint.
- Data collection frequency.
- How much instance data is being collected.

You may alter the settings for an individual record as follows:

1. In the Add Entry to the Profile, or the Edit Entry in a Profile window, click Data Storage. The Data Storage dialog is displayed.

![Data Storage dialog](image)

*Figure 56. Specifying data records to send to database*

2. Choose whether or not, you want data to be summarized at the endpoint by checking Yes or No in the Summary Data Local Logging panel.
3. Choose whether or not, you want data collection records to be stored in the database by checking **Yes** or **No** in the Summary Data Database Settings Store Data? panel

4. If you select **Yes**, choose a level of detail to be saved.

   **minimum**
   
   Is the default setting. It stores average response time (ms), max. response time (ms), min. response time (ms), successful, failed and aborted transactions.

   **medium**
   
   Stores minimum detail plus buckets.

   **maximum**
   
   Stores medium-level details plus ARM 2 user defined metrics.

5. Choose what level of instance data you want to be logged on the endpoint. In the Instance Data Local Logging panel choose:

   - **No Data** if you do not need to log any instance data.
   - **Correlated Data** if you want to log only instance data containing correlator information.
   - **All Data** if you want to log all the instance data.

6. In the Instance Data Database Settings, Choose what level of instance data you want to send to the database. This panel is not activated if Instance Data Local Logging No Data is checked. Choose:

   - **No Data** if you do not need to log any data in the database.
   - **Correlated Data** if you want to log only instance data containing correlator information in the database.
   - **All Data** if you want to log all the instance data in the database.

7. Click the **Add & Close** button

**Command Line**

The equivalent part of the CLI for setting the values above is:

```
wmarsstartcoll -a armAppl2...-d 1...
```

**Setting Simulation Extended Parameter Values**

Simulation applications often require extended parameters to be passed. This section describes how to add or edit parameters.

**Adding or Editing EPP Extended Parameter Values**

EPP simulations are subject to the same schedule control as applications, but have additional simulation parameters. Details of these parameters are given in “Description of the Supplied Probes” on page 54.

Set the extended parameters as follows:

1. In the Add Entry to the Profile, or the Edit Entry in a Profile dialog click **Simulation Settings**. The Add Extended Parameter Values dialog is displayed.
2. In the panel type the list of parameters and values divided by commas, required by the EPP Probe you are working with. See "Using the Probes" on page 50 for further information.

3. Click the Add & Close button.

Command Line
The equivalent part of the CLI for setting the values above is:

```
wmarsstartcoll -a ... -p "execFreq=30,timeout=80,targId=test" ...
```

Adding or Editing Virtual User Script Parameter Values
Virtual user simulations are subjected to the same schedule control as applications, but have additional simulation interval parameters. The Simulation Interval is the time between successive runs of a virtual user script (VUS).

Set the simulation interval for a virtual user script as follows:

1. In the Add Entry to the Profile, or the Edit Entry in a Profile dialog click **Simulation Settings**. The Add VUS Parameter Values dialog is displayed.
2. In the Hours and Minutes boxes, set the interval you want between successive simulation runs.

3. Check any parameter boxes that have been automatically filled and correct them if necessary. Add any missing parameters you require.

4. Click the Add & Close button.

When you have finished configuring collection details for the application, click the Add button. Select another application for addition, or click Add & Close to return to the Application Availability Profile window.

Note: All collection settings are inactive until the profile is distributed to an endpoint.

Command Line
The equivalent part of the CLI for setting the values above is:

\texttt{wmarstartcoll -a armAppl2...-r 10 -p \"BAAN Company=570,User Name=baan, Server Name=hazard,Path to BAAN Application=f:users\baan, BAAN Shell=bshell\"...}

Changing Collection Details
Collection details for an application may be changed at any time.

- If changes are the same for several copies or all copies of an application, it is better to use the GUI. With this, you can edit the details for that application in the profile or profiles and then redistribute the updated profile to chosen endpoint or endpoints. When the profile arrives at the endpoint, the engine will discard any \textit{unchanged} application collections in the profile. The engine stops application collections that are different, overwrites with the new information, and restarts collection.

- If changes (especially if they are temporary) are targeted at a particular endpoint, it may be simpler to issue a new set of collection details via a \texttt{wmarstartcoll} command.
Editing an Entry in a Profile

You can change the settings for any application in a profile by selecting it in the Application Availability Profile window and clicking the **Edit Entry** button. The Edit Entry in the Profile window is displayed.

![Figure 59. Edit Entry in the Profile dialog](image)

Use this window to change existing entry settings in a profile. It has identical features and actions to the Add an Entry in the Profile window and enables you to recall a profile entry that has already been configured. Once all changes have been made, you may save those changes by clicking the **Change & Close** button on the dialog. You can decide to abandon the changes you have made by clicking the **Cancel** button.

Command Line

```
wmarrstartcoll -a netscape...... -b 0,100,500..............
```

Disabling an Entry

When an entry (collection) has been added to a profile, it is automatically marked as enabled and will be active as soon as it reaches an endpoint. To disable an entry in a profile, highlight it in the Add an Entry to a Profile window and click the **Disable** button. You can put an entry in a profile back into use, by using the **Enable** button in the same way.
Note: After changing enable setting for any application, the profile must be redistributed to the endpoint or endpoints before the changes are valid.

You may add comment to an entry in the Add Application to a Profile or Edit Entry in a Profile dialogs.

Command Line

Use the `wmarstopcoll` (see “wmarstopcoll” on page 205) and `wmarstartcoll` (see “wmarstartcoll” on page 201) CLIs to disable and enable a collection at an endpoint.

Deleting an Entry from a Profile

In the Application Availability Profile window, select the entry you want to delete and click the Delete button.

Distributing a Profile

When a profile has been built or changed, it can be pushed to an endpoint or to several endpoints. Profile distribution to subscribers is a standard “drag and drop” action in Tivoli desktop. Refer to your Tivoli Management Framework User’s Guide for details.

Notes:

1. If this is the first profile to be sent to the endpoint, the product detects that the endpoint has not been used before and includes the product engine with the profile.
2. When a profile arrives at an endpoint, it automatically starts the engine if it is not running.
The Web GUI

The Web GUI is part of the product. It enables you to have access to the application performance data that the product collects, on the day that it is collected. When you use Web GUI, data from the data files that are held on the endpoint is displayed in simple graphical form through your web interface.

In this way, the browser provides a real time easy-to-use interface to current application performance data collected on the product endpoints.

The product online browser only displays data from one endpoint at a time, and does not perform any special processing on the data. Data processing functions are provided by the Tivoli Decision Support Guide. See Tivoli Decision Support for Application Performance User's Guide.

Using the Web GUI

To use the Web GUI for viewing recent data:

1. Start your Internet browser (MS Internet Explorer or Netscape), and enter the following URL, but substitute the address of the managed node that has the IBM HTTP server (Web GUI component) installed. See Step 5 of “Using the Desktop GUI to Install the Web GUI” on page 20.

2. Press the Enter key. The first page of the Web GUI is displayed.

Figure 61. Entering server details

2. Press the Enter key. The first page of the Web GUI is displayed.
3. Click **Start Tivoli APM Web GUI**, which is a hotspot. An identity dialog is displayed. The current version of Web GUI supports the user identity `tapm` and password `tapmtapm`. Your administrator may change this during or after installation (see “Configuring the Web GUI” on page 41).

4. The main page is displayed.
5. Type in the identity of the endpoint where you want to examine data.

   **Note:** You may provide the endpoint_name or the host_name of the computer. If host_name is used, a previous data upload to the database must have occurred, so that host_name has been registered within the database.

6. The default data which is presented is today’s data that is collected since midnight, or since collections were started. The collection period may be specified, and confirmation of it is displayed on the screen.

   If you want to specify a previous day, click the calendar icon and select a day from the calendar that is presented.

   **Note:** The number of previous days data files available on an endpoint, depends on the engine settings sent to the engine with a wmarstatus command (see “wmarstatus” on page 197). Default value is three previous days.

7. Click the GO> button located at the upper right hand corner of the window. It changes color to green. The product now interrogates the selected data file at the endpoint. This may take a minute or two.
8. When interrogation has completed, a drop down list of active applications is available for you to make a selection. Click the reload button to reload this list at any time.

![List of applications display](image1)

**Figure 66. List of applications display**

9. When you have chosen your application, select the transaction you want to examine from the associated drop down list.

![Choosing a transaction](image2)

**Figure 67. Choosing a transaction**

**Note:** Remember that the endpoint engine is still running and carrying out collections. You use the refresh button to obtain latest information on available applications and transaction data.

10. Select the type of data display you require from the drop down list and click the **GO** button.

![Selecting the data to display](image3)

**Figure 68. Selecting the data to display**

A graphic data display is now presented.
As more data is collected, the display is automatically updated if you pressed the Start button. A time countdown to the next collection is displayed in the Windows status bar and a Stop button replaces the Start button.

Figure 69. Displayed data

Figure 70. Web GUI status bar
Setting up the Database

Tivoli Application Performance Management Version 2.1 (the product) uses an updated database schema to that of the previous Version. If you are migrating from 2.0.1 with the database installed you do not need to remove or delete data from your database, but you must upgrade it. For further information refer to "Upgrading the Database Schema 2.0.1" on page 111.

A component of the Tivoli Framework called the RDBMS Interface Module (RIM) in the database, is used to store the product data. It shields access to the RDBMS from the Tivoli application.

When the product is installed, you must carry out configuration actions to make sure that your data is collected and stored in a database, as follows:

1. Install the RDBMS client/server software.
2. Configure the RDBMS client.
3. Create the product RIM object.
4. Create the product database.
5. Verify the RIM to database connection.

Authorization Role

Your database administrator, who holds the required system authority and access role carries out Installation. Refer to the documentation for the database you are using, for usage information and installation. Check the Tivoli Application Performance Management Release Notes 2.1 for any last-minute changes in this procedure. Setting-up the database may also require other authorization roles such as super.

Installing the RDBMS Client/server Software

The prerequisite for setting up the product database components, is to have your relational database management system (RDBMS) client/server software already installed and working correctly.

The versions of databases that are supported in this release of the product are detailed in the Tivoli Application Performance Management Release Notes 2.1. Please refer to that document for the latest versions of databases supported, and for useful information on configuring the database client.
RDBMS Interface Module (RIM) Object

The Tivoli RIM is the interface that enables Tivoli applications (such as this product) to support different RDBMS products. The RIM provides the Tivoli applications with a common application programming interface to these database products.

Useful information on installation of the RDBMS on several platforms and information on the RIM usage, can be found in the IBM Redbook “Using Databases with Tivoli Applications and RIM” SG24-5112. This book can be downloaded from http://www.redbooks.ibm.com/.

Refer to Tivoli Management Framework Reference Manual for details of the RIM commands.

The Tivoli managed node through which the application communicates with the relational database is known as the RIM host. The RIM daemon and RIM agent processes run on this system, and must therefore have the database client installed.

The region server, the product RIM host, and database server can run on the same machine. However, the recommended configuration is for the region server, RIM host, and database server to be on separate machines. This separation creates an effective balance of the processing load across the region.

Figure 71. RIM block diagram
Creating the RIM Object

You need to create a RIM object in order to enable the product to communicate with the database. Only a Tivoli management region administrator with the required authorization of senior or super, can create and maintain the RIM object. This authorization is required whatever operating system being used. By default, a region root administrator has such authority.

The product provides two methods for creating the RIM object:

- By using the GUI
- By a shell script (CLI)

Each of these methods creates a RIM object named “tapm”.

Creating the RIM Object by Using the GUI

To create the RIM object by using the GUI:

1. On the Tivoli Desktop window, double-click the APM RIM icon. The Set APM RIM configuration dialog is displayed.

2. Select your database type from the Database Vendor list.

3. Click the Managed Nodes button to obtain a list of available nodes from which you can choose a RIM Host. Select a node and click Select and Close.

Figure 72. Database configuration screen
4. The Set APM RIM configuration dialog is displayed again with your selected node now occupying the RIM Host box. Type in the remaining database details in the same way as explained in “Creating the RIM Object by Using the CLI” below.

5. Click **Set and Close** to save the values. You are prompted to set a password.

6. When you have set and confirmed the password, click **Set and Close**. A confirmation dialog is displayed.

**Creating the RIM Object by Using the CLI**

Tivoli Framework applications use the *wcrtrim* command to create a RIM object on a specific managed node. The product provides you with a shell script that prompts you for the required input. You can run this script from any managed node as a Tivoli administrator. You can run it on any UNIX shell or Tivoli bash shell (Windows NT) with the Tivoli environment variables set (see Step 1 of “Registering Instrumented Applications” on page 72). The *cr_apm_rim.sh* script attempts to retrieve your RIM object attributes. If *cr_apm_rim.sh* cannot find the RIM object (as in the case of a first time installation), you are prompted to enter the required information.

Run the script as follows:

1. Change to the following directory:

   `$BINDIR/TME/MAR/SQL`. 
2. Run the appropriate RIM creation script as follows:

<table>
<thead>
<tr>
<th>Platform</th>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>Windows</td>
<td>sh cr_apm_rim.sh</td>
</tr>
<tr>
<td>UNIX</td>
<td>cr_apm_rim.sh</td>
</tr>
</tbody>
</table>

The information that you need to supply, depends on what the vendor product requirements are for the specific database. You can also use this script to change the RIM object parameters. If a RIM object tapm already exists, you are asked to remove it and create a new one by supplying the following information:

**Database Vendor:**
The vendor name for the RDBMS required to manage the product data. The supported databases are Sybase, Oracle, MS-SQL, DB2®, and Informix®.

**RIM Host:**
The machine where the RIM processes are running. It must be a Tivoli management region server or a managed node where the RDBMS client software is installed.

**Database Home:**
The directory on the RIM host where the database software is installed.

**Instance Home (DB2 only):**
DB2 instance install directory.

**Database Server ID:**
The name of the RDBMS server. This is an alias to enable the client/server connection.

**Database ID:**
A unique name for the database.

**Database User ID:**
ID of the user who is authorized to access the database.

**Database User ID Password:**
The database user password. After the RIM creation you can change the password by using the Tivoli Management Framework command wsetrmpw.
- For DB2 users, the password must match the password of the DB2 instance owner.
- For Sybase users, the password must be at least six characters.
- For Informix users, the password must match the Informix NT/UNIX user password.

In the following table there is specific database information for the RIM parameters. Included with each title, there is the related parameter for the wcrtrim command used by the cr_apm_rim.sh script.

Note that some of the parameters use database-specific environment variables that are highlighted in bold.
Verifying the RIM Object Creation

You can issue the command `wgetrim tapm`, to verify that the RIM object has been created correctly. If any of the settings are incorrect, you can use `wsetrim` with the appropriate options to change the RIM object labeled `tapm` or you can run the script `cr_tapm_rim.sh` again to remove the existing RIM object and create a new one.

Creating the Product Database

If you have never installed the product database, you need to create the database schema 2.1. To do this, run the script `cr_apm_db.sh`. If you are migrating from 2.0.1 or you already have the product database schema 2.0.1 installed, you do not need to run the `cr_apm_db.sh` script. Use the `upg_apm21_db.sh` script instead. Refer to “Upgrading the Database Schema 2.0.1” on page 111 for further information.

**Note:** To run these, and other, scripts you need an authorization role Tivoli administrator.

Refer to the Tivoli Application Performance Management Release Notes Version 2.1 for the latest information on database requirements. In this document you can find all the supported database versions and the prerequisites for creating the product database.

A successful installation of the product, copies the script files specific to each RDBMS, to the $BINDIR/TME/MAR/SQL directory of your managed nodes, where the product is installed. The file name extensions identify which database vendors they are written for:

- `syb` = Sybase.
- `ora` = Oracle.
- `mssql` = MS-SQL 6.x.
- `mssql7` = MS-SQL 7.0.
- `ifmx` = Informix.
- `db2` = DB2.
RDBMS client and server software, should already be installed and configured beforehand, to create the database structure. You have two different ways to create the product tables for the database:

1. Use the script `cr_apm_db.sh` to create the database schema. The method of use differs slightly.

   If you run the database creation script from a shell with the Tivoli environment set on the RIM host, the script `cr_apm_db.sh` attempts to obtain database information from the RIM object attributes. If it cannot find the product RIM object, you are prompted to enter the required information.

   - **Oracle, Sybase, or MS-SQL.** For these databases, you can run the shell script `cr_apm_db.sh` from the RIM host where the database client part is installed. You should run the command from a bash shell (Windows NT) or from any UNIX shell (UNIX), with the Tivoli environment already set.

   - **DB2 and Informix.** For these databases the procedure is different from that described. This is because you cannot execute the DB2 database creation script from the DB2 Client Command Line Processor without an existing database connection. Informix uses the `dbaccess` function which is shipped as part of the Informix server. The script `cr_apm_db.sh` prompts you to move to the database server, the script `cr_db.db2.sql` or `cr_db.ifmx.sql`, located in `$BINDIR/TME/MAR/SQL`. The same script must be then processed by the database specific SQL processor.

     - **DB2.** Use a shell with the DB2 environment already set. Then run:
       ```
       db2 –td; -f $script_DIR/cr_db.db2.sql
       ```

     - **Informix.** Use a shell with the Informix environment already set. Then run:
       ```
       dbaccess - $script_DIR/cr_db.ifmx.sql
       ```

2. After customizing the `cr_db.xxx` and `cr_tbl.xxx` scripts, where `xxx` = `syb`, `ora`, `mssql`, `mssql7`, `ifmx`, or `db2`, use the interactive SQL processor. You can run these SQL scripts on the RDBMS client or server.

### Running the Database Creation Script from a Shell

Before running the RDBMS product scripts you are recommended to back up your RDBMS database. See your RDBMS documentation for backup procedures and restore procedures.

If you run the database creation script from a shell with the Tivoli environment set on the RIM host, you need to run the script `SBINDIR/TME/MAR/SQL/cr_apm_db.sh`.

If you do not have the RIM object already created, you are asked for the parameter identified in Creating Application Performance Management RIM object by CLI (See page [106]). The other configuration parameters are:

- **Database Location (Sybase and MS-SQL only):**
  You are requested to specify an existing directory on the database server where the database and transaction log files are placed.

- **Database dbspace (Informix only):**
  Do not use the `rootdbs dbspace` for the product database. You must first create two db spaces for this database, one for the aggregated data, the other for the transaction data. You must enter names for these dbspaces.

- **Database Size: (All database vendors except Informix)**
  The size (in MB) of the database to be created.
For Sybase, MS-SQL you are prompted to enter the sizes for the database and the transaction log. Usually 40 Mb.

For DB2 allow 140 Mb and for Oracle 30 Mb.

When prompted to do so, enter the database administrator passwords.

**Running the RDBMS Configuration Scripts Through the SQL Processor**

Your database administrator can customize the SQL templates such as `cr_db.xxx` and `cr_tbl.xxx`, and use the interactive SQL processor to run these SQL scripts on the RDBMS client or server.

<table>
<thead>
<tr>
<th>RDBMS Vendor</th>
<th>Configuration File</th>
<th>Interactive SQL Processor</th>
<th>RDBMS Administrator</th>
</tr>
</thead>
<tbody>
<tr>
<td>Oracle</td>
<td>tnsnames.ora</td>
<td>sqlplus</td>
<td>sys</td>
</tr>
<tr>
<td>Sybase</td>
<td>interfaces</td>
<td>isql</td>
<td>sa</td>
</tr>
<tr>
<td>MS-SQL</td>
<td>interfaces</td>
<td>isql</td>
<td>sa</td>
</tr>
<tr>
<td>DB2</td>
<td></td>
<td>db2</td>
<td>instance owner</td>
</tr>
<tr>
<td>Informix</td>
<td>sqlhost</td>
<td>dbaccess</td>
<td>informix</td>
</tr>
</tbody>
</table>

To use the SQL processor, run the RDBMS configuration scripts on the RDBMS client or server. This is an example for Oracle:

1. On the managed nodes where the product is installed, customize the `cr_db.ora` and `cr_tbl.ora` scripts to meet your needs and then save them as `cr_db.ora.sql` and `cr_tbl.ora.sql`.

2. If the current managed node where the product is installed is the RIM host with the RDBMS client software installed, skip to step 4.

3. Copy the `cr_db.ora.sql` and `cr_tbl.ora.sql` files from `$BINDR/TME/MAR/SQL` directory on the managed nodes where the product is installed, to a temporary directory on the RIM host where the RDBMS client software is installed.

4. As a prerequisite, no Tivoli Application Performance Management related data files are expected to be found on the RDBMS server machine, i.e. no product database is present there. On the RIM host launch:

   `sqlplus sys@lab15069 @o:/Tivoli/bin/w32-ix86/TME/MAR/SQL/cr_db.ora.sql`

   where `lab15069` is the RDBMS server and the sql script is located in the specified path. The password for the user ‘sys’ is requested. This sql script creates the structure of the product database (data tablespaces, index tablespace, temporary tablespace, rollback segments).

5. Install the layout by entering the following command:

   `sqlplus tapm@lab15069 @o:/Tivoli/bin/w32-ix86/TME/MAR/SQL/cr_tbl.ora.sql`

   The password for the user “tapm” is requested. It is the same password specified at RIM creation.

6. Open a sqlplus session specifying the user “tapm”:

   `sqlplus tapm@lab15069`

7. In the sql session, type the following to check that the product repository was installed:
select * from <table >;
> go

where <table> is the name of the created table.

Results should indicate that 0 rows were found, except for the tables TAPM_ID and TAPM_VERSION where 1 row should be found. If results indicate that <table> is unknown, the repository was not installed.

8. Log out of sqplusl by entering the following command:
> quit

Verifying the Database Client/Server Connections

After you have installed the product database, use the interactive SQL processor provided by your database Vendor, to verify the connection to the database created from the RIM host. See steps 6 to 8 of “Running the RDBMS Configuration Scripts Through the SQL Processor” on page 110.

For further details about the database Client/Server connection parameters for the different platforms, refer to the IBM Redbook Using Databases with Tivoli Applications and RIM number SG24-5112. This book can be downloaded from:

Upgrading the Database Schema 2.0.1

To support the transaction correlation feature, the product database schema has been updated by adding new tables for transaction data.

Note: Migration: Before upgrading, you should check whether data files are present on the gateways. Refer to “Migration Processes” on page 27.

You must take note that where there is a reference to cr_apm_db.sh, it should be substituted with upg_apm21_db.sh and when there are references to cr_tbl.xxx and cr_db.xxx, they should be substituted with upg21_db.xxx, where xxx = syb, ora, mssql, mssql7, ifmx or db2.

If you are migrating from 2.0.1 and have historical data stored in the database, you must run the upg_apm21_db.sh script.

The procedure and recommendations described in “Creating the Product Database” on page 108 are still valid.

Informix and DB2 RDBMS are handled the same way as the creation step. The script: upg21_db.db2.sql/upg21_db.ifmx.sql must be moved on the database server and processed there with the SQL processor.

Verifying the RIM to Database Connection

When the database has been created, test the RIM connection to the database by using the following Tivoli Framework command:
wriment -l tapm

Example:
C:\Tivoli\bin\w32-ix86\TME\MAR\SQL>wriment -l tapm
Resource Type : RIM
Resource Label : tapm
Host Name : amadeus
User Name : informix
Vendor : Informix
Removing the Product Database

Use the script `rm_apm_db.sh` to remove the database schema.

1. Go to the following directory:
   
   `$BINDIR/TME/MAR/SQL`

2. Run the uninstall script:
   
   `rm_apm_db.sh`

- **Oracle, Sybase, or MS-SQL.** To use these databases, you can run the shell script `rm_apm_db.sh` from the RIM host where the database client part is installed. Run the command from a Tivoli bash shell (Windows NT) or from any UNIX shell (UNIX) with the Tivoli environment already set.

- **DB2 and Informix**
  
  - **DB2.** Use a shell with the DB environment already set. Then run:
    
    `db2 -td; -f $script_DIR/cr_db.db2.sql`

  - **Informix.** Use a shell with the Informix environment already set. Then run:
    
    `dbaccess - $script_DIR/cr_db.ifmx.sql`

---

Oracle Notes

After the database has been removed and before any further creation actions, you must physically delete data files on the server. These are normally located in `$ORACLE_HOME/dbs` or `$ORACLE_HOME/database`, and are named `tapm_xxx.dat`. Perform these steps:

1. Using the server manager application, connect to the database as **internal**.

2. Shut down the instance.

3. Delete the identified files.

4. Start up the instance.

---

Database Details

This section contains information on the database architecture.

**Data Storage**

Some consideration must be made of the data to be stored in the database and hence the derived network traffic load. In the product there are several levels of data manipulation before the data is stored in the databases. The engine captures the ARM events on the endpoint and provides performance information as response times. In a well defined collection interval the average, minimum and maximum response times are computed, then an aggregator process aggregates those data at hourly steps. In order to limit the amount of data collected, three different types of data configuration can be deployed ("Database Settings for a Collection Record" on page 91) to the endpoints:
They only differ by the number of metrics to be measured.

The table shown below details the amount of data produced by the Tivoli application performance management engine on the endpoint for each set of application, or user and transaction data, for different configurations (minimum, medium, maximum). The collection interval can be set from 1 to 60 minutes, so the number of intervals range from 1440 to 24. From this table you can see that reducing the amount of performance data to be stored is important.

<table>
<thead>
<tr>
<th>Tivoli Management for Transaction Performance Configuration</th>
<th>Metric Type</th>
<th>Data Size per Collection Interval (bytes)</th>
<th>MAX/MIN data size per day (bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minimum</td>
<td>Average Response Time</td>
<td>4</td>
<td>576 / 34560</td>
</tr>
<tr>
<td>Minimum</td>
<td>Maximum Response Time</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>Minimum</td>
<td>Minimum Response Time</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>Minimum</td>
<td># Transactions GOOD</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>Minimum</td>
<td># Transactions FAILED</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>Minimum</td>
<td># Transactions ABORTED</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>Maximum</td>
<td>Buckets</td>
<td>4 x 7</td>
<td>1248 / 74880</td>
</tr>
<tr>
<td>Maximum</td>
<td>Metric #1</td>
<td>8</td>
<td>2400 / 144000</td>
</tr>
<tr>
<td>Maximum</td>
<td>Metric #2</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>Maximum</td>
<td>Metric #3</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>Maximum</td>
<td>Metric #4</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>Maximum</td>
<td>Metric #5</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>Maximum</td>
<td>Metric #6</td>
<td>8</td>
<td></td>
</tr>
</tbody>
</table>

Figure 75. Relative size of collected data

The amount of measured numeric data for one day, to be transferred into the database from the gateway is estimated from the formula:

\[
\text{Transaction} \times \text{Application} \times \text{User} \times \text{Host} \times \text{Data}
\]

In a typical Tivoli hardware configuration you can probably expect to collect performance data from only 100 hosts that have one user connected, running three applications that produce 20 different class of transactions. Hence for the minimum product configuration case, we will have to transfer into the database (without taking in account of all the strings relative to the host_name, tx_name and so on) at least 100x1x3x20x576 = 3.4 MB per day.

**Example:** Consider the network bandwidth that can range from 64 Kbit/s to 2 Mbit/s in a very optimistic condition. This means if all the customer network bandwidth is reserved for data transfer (and it will not), the transfer of only the measured data on a 64 Kbit/s network takes at least 435 s = 7 minutes.

**Schema**

The following diagram shows the product 2.1 database schema with the logical relationship between tables.
Data Dictionary

The following tables give a comprehensive description of each field in the database. This description includes:

- Field name.
- Key.
  - The Primary (P) key is a column or set of columns that uniquely identifies the rest of the data in any given row of the table.
  - The Foreign (F) key is a column in a table that is a primary key of another table, which means that any data in a foreign key column must have corresponding data in the other table where that column is the primary key. For databases, this correspondence is known as referential integrity.
  - The NOT NULL (NN) value means that the column must have a value in each row. If NULL was used, that column may be left empty in a given row.
- Data type.
- Field size.
- A description of the purpose of each field.

Note: The “Datatype” and “Field Size” depend on the way the database vendor was implemented.
### Table 1. Data Dictionary for HOSTS

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Key</th>
<th>Data Type</th>
<th>Field Description</th>
<th>Field Size (bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST_ID</td>
<td>P</td>
<td>Long</td>
<td>A unique sequential (incremented by 1) number or random number, assigned by Tivoli Application Performance Management whenever a new record is added to the table.</td>
<td>4</td>
</tr>
<tr>
<td>HOST_NAME</td>
<td>NN</td>
<td>String</td>
<td>Official name of the host. If using the DNS or similar resolution system, it is the Fully Qualified Domain Name (FQDN)</td>
<td>255</td>
</tr>
<tr>
<td>HOST_IP_ADDRESS</td>
<td>NN</td>
<td>String</td>
<td>Last known IP network address of the host (string in Internet standard dotted format)</td>
<td>32</td>
</tr>
<tr>
<td>TME_OBJECT_LABEL</td>
<td></td>
<td>String</td>
<td>The Tivoli Framework Endpoint label</td>
<td>32</td>
</tr>
<tr>
<td>HOST_TIME_ZONE</td>
<td></td>
<td>Long</td>
<td>Difference in minutes of the Host Time zone from the Greenwich Time Zone</td>
<td>4</td>
</tr>
<tr>
<td>DST</td>
<td></td>
<td>Long</td>
<td>Daylight Saving Time active (1 or 0)</td>
<td>4</td>
</tr>
</tbody>
</table>

### Table 2. Data Dictionary for USERS

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Key</th>
<th>Data Type</th>
<th>Field Description</th>
<th>Field Size (bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>USER_ID</td>
<td>P</td>
<td>Long</td>
<td>A unique sequential (incremented by 1) number or random number, assigned by Tivoli Application Performance Management whenever a new record is added to the table.</td>
<td>4</td>
</tr>
<tr>
<td>USER_NAME</td>
<td>NN</td>
<td>String</td>
<td>The name of the application user (is the appl_user_id passed in the arm_init API call)</td>
<td>128</td>
</tr>
</tbody>
</table>

### Table 3. Data Dictionary for APPLICATIONS

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Key</th>
<th>Data Type</th>
<th>Field Description</th>
<th>Field Size (bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>APPL_ID</td>
<td>P</td>
<td>Long</td>
<td>A unique sequential (incremented by 1) number or random number, assigned by Tivoli Application Performance Management whenever a new record is added to the table.</td>
<td>4</td>
</tr>
<tr>
<td>APPL_NAME</td>
<td>NN</td>
<td>String</td>
<td>The name to identify the application user (is the appl_user_id passed in the arm_init API call)</td>
<td>128</td>
</tr>
</tbody>
</table>

### Table 4. Data Dictionary for TRANSACTIONS

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Key</th>
<th>Data Type</th>
<th>Field Description</th>
<th>Field Size (bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>TX_ID</td>
<td>P</td>
<td>Long</td>
<td>A unique sequential (incremented by 1) number or random number, assigned by Tivoli Application Performance Management, whenever a new record is added to the table.</td>
<td>4</td>
</tr>
<tr>
<td>TX_NAME</td>
<td>NN</td>
<td>String</td>
<td>The unique name of the transaction class. It is defined for each transaction class by the application developer (is the tran_name passed in the arm_getid API call)</td>
<td>128</td>
</tr>
<tr>
<td>TX_DETAIL</td>
<td></td>
<td>String</td>
<td>Additional information about a transaction class (is the tran_detail passed in the arm_getid API call).</td>
<td>128</td>
</tr>
</tbody>
</table>
### Table 5. Data Dictionary for ID_INFO

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Key</th>
<th>Data type</th>
<th>Field Description</th>
<th>Field Size (bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>TX_CLASS_ID</td>
<td>P</td>
<td>Long</td>
<td>A unique sequential (incremented by 1) number or random number assigned by Tivoli Application Performance Management whenever a new record is added to the table. Identify a transaction of an application run from a user on a given host</td>
<td>4</td>
</tr>
<tr>
<td>HOST_ID</td>
<td>F</td>
<td>Long</td>
<td>The unique ID from the HOSTS table</td>
<td>4</td>
</tr>
<tr>
<td>USER_ID</td>
<td>F</td>
<td>Long</td>
<td>The unique ID from the USERS table</td>
<td>4</td>
</tr>
<tr>
<td>APPL_ID</td>
<td>F</td>
<td>Long</td>
<td>The unique ID from the APPLICATIONS table</td>
<td>4</td>
</tr>
<tr>
<td>TX_ID</td>
<td>F</td>
<td>Long</td>
<td>The unique ID from the TRANSACTIONS table</td>
<td>4</td>
</tr>
</tbody>
</table>

### Table 6. Data Dictionary for METRICDATA

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Key</th>
<th>Data type</th>
<th>Field Description</th>
<th>Field Size (bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATA_ID</td>
<td>P</td>
<td>Long</td>
<td>A unique sequential (incremented by 1) number or random number, assigned by Tivoli Application Performance Management whenever a new record is added to the table.</td>
<td>4</td>
</tr>
<tr>
<td>DT_STAMP</td>
<td>NN</td>
<td>Long</td>
<td>The timestamp UTC of the aggregated data collected (used only for the date trigger)</td>
<td>4</td>
</tr>
<tr>
<td>COLLECTION_DATE</td>
<td>NN</td>
<td>Date</td>
<td>Date (day) of the aggregated data collected</td>
<td>8</td>
</tr>
<tr>
<td>METRIC_NAME</td>
<td>NN</td>
<td>String</td>
<td>Metric name (rt_avg, rt_min, rt_max, n_tx_good, n_tx_failed, n_tx_aborted, buckets and the user defined metric name as set in the data buffer of the arm_getid call)</td>
<td>44</td>
</tr>
<tr>
<td>METRIC_TYPE</td>
<td>NN</td>
<td>Long</td>
<td>Metric type. This field is used to address the proper data table (0 = DATA32, 1 = DATA64, 2 = DATABUCKETS)</td>
<td>4</td>
</tr>
<tr>
<td>TX_CLASS_ID</td>
<td>F</td>
<td>Long</td>
<td>The unique ID from the ID_INFO table</td>
<td>4</td>
</tr>
<tr>
<td>PARENT_CLASS_ID</td>
<td></td>
<td>Long</td>
<td>The parent data transaction ID used to correlate different aggregated data (not used)</td>
<td>4</td>
</tr>
<tr>
<td>IP_ADDRESS</td>
<td>NN</td>
<td>String</td>
<td>IP address of the Host at aggregation time (string in Internet standard dotted format)</td>
<td>32</td>
</tr>
</tbody>
</table>
### Table 7. Data Dictionary for DATA32

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Key</th>
<th>Data Type</th>
<th>Field Description</th>
<th>Field Size (bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATA_ID</td>
<td>P</td>
<td>Long</td>
<td>The unique ID from the METRICDATA table</td>
<td>4</td>
</tr>
<tr>
<td>VALUE_00</td>
<td></td>
<td>Long</td>
<td>Metric value for the specified data</td>
<td>4</td>
</tr>
<tr>
<td>VALUE_01</td>
<td></td>
<td>Long</td>
<td>Metric value for the specified data</td>
<td>4</td>
</tr>
<tr>
<td>VALUE_02</td>
<td></td>
<td>Long</td>
<td>Metric value for the specified data</td>
<td>4</td>
</tr>
<tr>
<td>VALUE_03</td>
<td></td>
<td>Long</td>
<td>Metric value for the specified data</td>
<td>4</td>
</tr>
<tr>
<td>VALUE_04</td>
<td></td>
<td>Long</td>
<td>Metric value for the specified data</td>
<td>4</td>
</tr>
<tr>
<td>VALUE_05</td>
<td></td>
<td>Long</td>
<td>Metric value for the specified data</td>
<td>4</td>
</tr>
<tr>
<td>VALUE_06</td>
<td></td>
<td>Long</td>
<td>Metric value for the specified data</td>
<td>4</td>
</tr>
<tr>
<td>VALUE_07</td>
<td></td>
<td>Long</td>
<td>Metric value for the specified data</td>
<td>4</td>
</tr>
<tr>
<td>VALUE_08</td>
<td></td>
<td>Long</td>
<td>Metric value for the specified data</td>
<td>4</td>
</tr>
<tr>
<td>VALUE_09</td>
<td></td>
<td>Long</td>
<td>Metric value for the specified data</td>
<td>4</td>
</tr>
<tr>
<td>VALUE_10</td>
<td></td>
<td>Long</td>
<td>Metric value for the specified data</td>
<td>4</td>
</tr>
<tr>
<td>VALUE_11</td>
<td></td>
<td>Long</td>
<td>Metric value for the specified data</td>
<td>4</td>
</tr>
<tr>
<td>VALUE_12</td>
<td></td>
<td>Long</td>
<td>Metric value for the specified data</td>
<td>4</td>
</tr>
<tr>
<td>VALUE_13</td>
<td></td>
<td>Long</td>
<td>Metric value for the specified data</td>
<td>4</td>
</tr>
<tr>
<td>VALUE_14</td>
<td></td>
<td>Long</td>
<td>Metric value for the specified data</td>
<td>4</td>
</tr>
<tr>
<td>VALUE_15</td>
<td></td>
<td>Long</td>
<td>Metric value for the specified data</td>
<td>4</td>
</tr>
<tr>
<td>VALUE_16</td>
<td></td>
<td>Long</td>
<td>Metric value for the specified data</td>
<td>4</td>
</tr>
<tr>
<td>VALUE_17</td>
<td></td>
<td>Long</td>
<td>Metric value for the specified data</td>
<td>4</td>
</tr>
<tr>
<td>VALUE_18</td>
<td></td>
<td>Long</td>
<td>Metric value for the specified data</td>
<td>4</td>
</tr>
<tr>
<td>VALUE_19</td>
<td></td>
<td>Long</td>
<td>Metric value for the specified data</td>
<td>4</td>
</tr>
<tr>
<td>VALUE_20</td>
<td></td>
<td>Long</td>
<td>Metric value for the specified data</td>
<td>4</td>
</tr>
<tr>
<td>VALUE_21</td>
<td></td>
<td>Long</td>
<td>Metric value for the specified data</td>
<td>4</td>
</tr>
<tr>
<td>VALUE_22</td>
<td></td>
<td>Long</td>
<td>Metric value for the specified data</td>
<td>4</td>
</tr>
<tr>
<td>VALUE_23</td>
<td></td>
<td>Long</td>
<td>Metric value for the specified data</td>
<td>4</td>
</tr>
</tbody>
</table>

**Note:** This table is addressed by a value of 0 in the METRIC_TYPE column of the METRICDATA table.
Table 8. Data Dictionary for DATA64

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Key</th>
<th>Data Type</th>
<th>Field Description</th>
<th>Field Size (bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATA_ID</td>
<td>P</td>
<td>Long</td>
<td>The unique ID from the METRICDATA table</td>
<td>4</td>
</tr>
<tr>
<td>VALUE_00</td>
<td></td>
<td>Float</td>
<td>Metric value for the specified data</td>
<td>4</td>
</tr>
<tr>
<td>VALUE_01</td>
<td></td>
<td>Float</td>
<td>Metric value for the specified data</td>
<td>4</td>
</tr>
<tr>
<td>VALUE_02</td>
<td></td>
<td>Float</td>
<td>Metric value for the specified data</td>
<td>4</td>
</tr>
<tr>
<td>VALUE_03</td>
<td></td>
<td>Float</td>
<td>Metric value for the specified data</td>
<td>4</td>
</tr>
<tr>
<td>VALUE_04</td>
<td></td>
<td>Float</td>
<td>Metric value for the specified data</td>
<td>4</td>
</tr>
<tr>
<td>VALUE_05</td>
<td></td>
<td>Float</td>
<td>Metric value for the specified data</td>
<td>4</td>
</tr>
<tr>
<td>VALUE_06</td>
<td></td>
<td>Float</td>
<td>Metric value for the specified data</td>
<td>4</td>
</tr>
<tr>
<td>VALUE_07</td>
<td></td>
<td>Float</td>
<td>Metric value for the specified data</td>
<td>4</td>
</tr>
<tr>
<td>VALUE_08</td>
<td></td>
<td>Float</td>
<td>Metric value for the specified data</td>
<td>4</td>
</tr>
<tr>
<td>VALUE_09</td>
<td></td>
<td>Float</td>
<td>Metric value for the specified data</td>
<td>4</td>
</tr>
<tr>
<td>VALUE_10</td>
<td></td>
<td>Float</td>
<td>Metric value for the specified data</td>
<td>4</td>
</tr>
<tr>
<td>VALUE_11</td>
<td></td>
<td>Float</td>
<td>Metric value for the specified data</td>
<td>4</td>
</tr>
<tr>
<td>VALUE_12</td>
<td></td>
<td>Float</td>
<td>Metric value for the specified data</td>
<td>4</td>
</tr>
<tr>
<td>VALUE_13</td>
<td></td>
<td>Float</td>
<td>Metric value for the specified data</td>
<td>4</td>
</tr>
<tr>
<td>VALUE_14</td>
<td></td>
<td>Float</td>
<td>Metric value for the specified data</td>
<td>4</td>
</tr>
<tr>
<td>VALUE_15</td>
<td></td>
<td>Float</td>
<td>Metric value for the specified data</td>
<td>4</td>
</tr>
<tr>
<td>VALUE_16</td>
<td></td>
<td>Float</td>
<td>Metric value for the specified data</td>
<td>4</td>
</tr>
<tr>
<td>VALUE_17</td>
<td></td>
<td>Float</td>
<td>Metric value for the specified data</td>
<td>4</td>
</tr>
<tr>
<td>VALUE_18</td>
<td></td>
<td>Float</td>
<td>Metric value for the specified data</td>
<td>4</td>
</tr>
<tr>
<td>VALUE_19</td>
<td></td>
<td>Float</td>
<td>Metric value for the specified data</td>
<td>4</td>
</tr>
<tr>
<td>VALUE_20</td>
<td></td>
<td>Float</td>
<td>Metric value for the specified data</td>
<td>4</td>
</tr>
<tr>
<td>VALUE_21</td>
<td></td>
<td>Float</td>
<td>Metric value for the specified data</td>
<td>4</td>
</tr>
<tr>
<td>VALUE_22</td>
<td></td>
<td>Float</td>
<td>Metric value for the specified data</td>
<td>4</td>
</tr>
<tr>
<td>VALUE_23</td>
<td></td>
<td>Float</td>
<td>Metric value for the specified data</td>
<td>4</td>
</tr>
</tbody>
</table>

**Note:** This table is addressed by a value of 1 in the METRIC_TYPE column of the METRICDATA table.
### Table 9. Data Dictionary for DATABUCKETS

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Key</th>
<th>Data type</th>
<th>Field Description</th>
<th>Field Size (bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATA_ID</td>
<td>P</td>
<td>Long</td>
<td>The unique ID from the METRICDATA table</td>
<td>4</td>
</tr>
<tr>
<td>BUCKET RANGE</td>
<td>P</td>
<td>Long</td>
<td>Bucket number (1-7)</td>
<td>4</td>
</tr>
<tr>
<td>BUCKET_ULB</td>
<td>NN</td>
<td>Long</td>
<td>Bucket Upper Limit Boundary</td>
<td>4</td>
</tr>
<tr>
<td>VALUE_00</td>
<td></td>
<td>Float</td>
<td>Metric value for the specified data</td>
<td>4</td>
</tr>
<tr>
<td>VALUE_01</td>
<td></td>
<td>Float</td>
<td>Metric value for the specified data</td>
<td>4</td>
</tr>
<tr>
<td>VALUE_02</td>
<td></td>
<td>Float</td>
<td>Metric value for the specified data</td>
<td>4</td>
</tr>
<tr>
<td>VALUE_03</td>
<td></td>
<td>Float</td>
<td>Metric value for the specified data</td>
<td>4</td>
</tr>
<tr>
<td>VALUE_04</td>
<td></td>
<td>Float</td>
<td>Metric value for the specified data</td>
<td>4</td>
</tr>
<tr>
<td>VALUE_05</td>
<td></td>
<td>Float</td>
<td>Metric value for the specified data</td>
<td>4</td>
</tr>
<tr>
<td>VALUE_06</td>
<td></td>
<td>Float</td>
<td>Metric value for the specified data</td>
<td>4</td>
</tr>
<tr>
<td>VALUE_07</td>
<td></td>
<td>Float</td>
<td>Metric value for the specified data</td>
<td>4</td>
</tr>
<tr>
<td>VALUE_08</td>
<td></td>
<td>Float</td>
<td>Metric value for the specified data</td>
<td>4</td>
</tr>
<tr>
<td>VALUE_09</td>
<td></td>
<td>Float</td>
<td>Metric value for the specified data</td>
<td>4</td>
</tr>
<tr>
<td>VALUE_10</td>
<td></td>
<td>Float</td>
<td>Metric value for the specified data</td>
<td>4</td>
</tr>
<tr>
<td>VALUE_11</td>
<td></td>
<td>Float</td>
<td>Metric value for the specified data</td>
<td>4</td>
</tr>
<tr>
<td>VALUE_12</td>
<td></td>
<td>Float</td>
<td>Metric value for the specified data</td>
<td>4</td>
</tr>
<tr>
<td>VALUE_13</td>
<td></td>
<td>Float</td>
<td>Metric value for the specified data</td>
<td>4</td>
</tr>
<tr>
<td>VALUE_14</td>
<td></td>
<td>Float</td>
<td>Metric value for the specified data</td>
<td>4</td>
</tr>
<tr>
<td>VALUE_15</td>
<td></td>
<td>Float</td>
<td>Metric value for the specified data</td>
<td>4</td>
</tr>
<tr>
<td>VALUE_16</td>
<td></td>
<td>Float</td>
<td>Metric value for the specified data</td>
<td>4</td>
</tr>
<tr>
<td>VALUE_17</td>
<td></td>
<td>Float</td>
<td>Metric value for the specified data</td>
<td>4</td>
</tr>
<tr>
<td>VALUE_18</td>
<td></td>
<td>Float</td>
<td>Metric value for the specified data</td>
<td>4</td>
</tr>
<tr>
<td>VALUE_19</td>
<td></td>
<td>Float</td>
<td>Metric value for the specified data</td>
<td>4</td>
</tr>
<tr>
<td>VALUE_20</td>
<td></td>
<td>Float</td>
<td>Metric value for the specified data</td>
<td>4</td>
</tr>
<tr>
<td>VALUE_21</td>
<td></td>
<td>Float</td>
<td>Metric value for the specified data</td>
<td>4</td>
</tr>
<tr>
<td>VALUE_22</td>
<td></td>
<td>Float</td>
<td>Metric value for the specified data</td>
<td>4</td>
</tr>
<tr>
<td>VALUE_23</td>
<td></td>
<td>Float</td>
<td>Metric value for the specified data</td>
<td>4</td>
</tr>
</tbody>
</table>

**Note:** This table is addressed by a value of 2 in the METRIC_TYPE column of the METRICDATA table.
### Table 10. Data Dictionary for TAPM_ID

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Key</th>
<th>Data Type</th>
<th>Field Description</th>
<th>Field Size (bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOST_ID</td>
<td>NN</td>
<td>Long</td>
<td>The last (max.) assigned ID from the HOSTS table</td>
<td>4</td>
</tr>
<tr>
<td>USER_ID</td>
<td>NN</td>
<td>Long</td>
<td>The last (max.) assigned ID from the USERS table</td>
<td>4</td>
</tr>
<tr>
<td>APPL_ID</td>
<td>NN</td>
<td>Long</td>
<td>The last (max.) assigned ID from the APPLICATIONS table</td>
<td>4</td>
</tr>
<tr>
<td>TX_ID</td>
<td>NN</td>
<td>Long</td>
<td>The last (max.) assigned ID from the TRANSACTIONS table</td>
<td>4</td>
</tr>
<tr>
<td>TX_CLASS_ID</td>
<td>NN</td>
<td>Long</td>
<td>The last (max.) assigned ID from the ID_INFO table</td>
<td>4</td>
</tr>
<tr>
<td>DATA_ID</td>
<td>NN</td>
<td>Long</td>
<td>The last (max.) assigned ID from the HOSTS table</td>
<td>4</td>
</tr>
<tr>
<td>DEF_ID</td>
<td>NN</td>
<td>Long</td>
<td>The last (max.) assigned ID from the METRIC_DEF table</td>
<td>4</td>
</tr>
<tr>
<td>METRIC_ID</td>
<td>NN</td>
<td>Long</td>
<td>The last (max) assigned ID from the METRIC_ID table</td>
<td>4</td>
</tr>
<tr>
<td>INFO_ID</td>
<td>NN</td>
<td>Long</td>
<td>The last (max) assigned ID from the INFO_ID table</td>
<td>4</td>
</tr>
<tr>
<td>INSTANCE_ID</td>
<td>NN</td>
<td>Long</td>
<td>The last (max) assigned ID from the TX_INSTANCE table</td>
<td>4</td>
</tr>
<tr>
<td>CORRELATOR_ID</td>
<td>NN</td>
<td>Long</td>
<td>The last (max) assigned ID from the CORRELATOR table</td>
<td>4</td>
</tr>
</tbody>
</table>

**Note:** This table is unrelated to the other tables because its purpose is to hold the last IDs assigned by Tivoli Application Performance Management to the main tables.

### Table 11. Data Dictionary for TAPM_VERSION

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Key</th>
<th>Data Type</th>
<th>Field Description</th>
<th>Field Size (bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB_SCHEMA</td>
<td>NN</td>
<td>Long</td>
<td>Identifies the version of the database schema. Version 2.1 is identified as 210</td>
<td>4</td>
</tr>
</tbody>
</table>

### Table 12. Data dictionary for the UUID_INFO table

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Key</th>
<th>Data Type</th>
<th>Field Description</th>
<th>Field Size (bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>INFO_ID</td>
<td>P</td>
<td>Long</td>
<td>The unique ID from the UUID_INFO table</td>
<td>4</td>
</tr>
<tr>
<td>TX_UUID</td>
<td></td>
<td>String</td>
<td>Universal Unique Identifier as defined in the ARM 3.0 specifications. This uniquely identifies the type of transaction. The transaction ID is optionally associated to two strings: the name of the transaction and the name of the application. If ARM 2.0 is used, then the UUID is built using the handle assigned to the instance of a transaction (is the start_handle returned from the arm_start API call)</td>
<td>32 (*)</td>
</tr>
<tr>
<td>APPL_ID</td>
<td>F</td>
<td>Long</td>
<td>The unique ID from the APPLICATIONS table</td>
<td>4</td>
</tr>
<tr>
<td>TX_ID</td>
<td>F</td>
<td>Long</td>
<td>The unique ID from the TRANSACTIONS table</td>
<td>4</td>
</tr>
<tr>
<td>METRIC_ID</td>
<td></td>
<td>Long</td>
<td>The unique ID from the METRIC_INFO table (a zero value means no metrics)</td>
<td>4</td>
</tr>
</tbody>
</table>

**Note:** (*) This is a character field, caused by the RIM limitation on big integer fields (> 4 bytes). This field represents a hexadecimal string such as “001E01018235063E0015019A000000040000000F00089254B54BD6ABB13B” where each character represent 4 bits (multiplied by 32 is equivalent to 16 bytes information).
### Table 13. Data dictionary for the TX_INSTANCE table

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Key</th>
<th>Data type</th>
<th>Field Description</th>
<th>Field Size (bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>INSTANCE_ID</td>
<td>P</td>
<td>Long</td>
<td>A unique sequential (incremented by 1) number assigned by Tivoli Application Performance Management whenever a new record is added to the table.</td>
<td>4</td>
</tr>
<tr>
<td>INFO_ID</td>
<td>F / NN</td>
<td>Long</td>
<td>The unique ID from the UUID_INFO table</td>
<td>4</td>
</tr>
<tr>
<td>STOP_DATE</td>
<td>NN</td>
<td>Date</td>
<td>The transaction stop date (local time)</td>
<td>7</td>
</tr>
<tr>
<td>STOP_TIME_S</td>
<td>NN</td>
<td>Long</td>
<td>The transaction stop time in seconds (timestamp in UTC format)</td>
<td>4</td>
</tr>
<tr>
<td>STOP_TIME_NS</td>
<td>NN</td>
<td>Long</td>
<td>The transaction stop time nanoseconds detail (timestamp in UTC format)</td>
<td>4</td>
</tr>
<tr>
<td>RT_MS</td>
<td>Long</td>
<td></td>
<td>Milliseconds part of Response Time. The total response time is given from the formula RT_MS*1000 + RT_NS</td>
<td>4</td>
</tr>
<tr>
<td>RT_NS</td>
<td>Long</td>
<td></td>
<td>Nanoseconds part of Response Time. The total response time is given from the formula RT_MS*1000 + RT_NS</td>
<td>4</td>
</tr>
<tr>
<td>STATUS</td>
<td>Long</td>
<td></td>
<td>Completion status (0=ARM_GOOD, 1=ARM_ABORT, 2=ARM_FAILED, 3=ARM_UNKNOWN)</td>
<td>4</td>
</tr>
<tr>
<td>HOST_ID</td>
<td>F / NN</td>
<td>String</td>
<td>The unique ID from the HOSTS table</td>
<td>4</td>
</tr>
<tr>
<td>IP_ADDRESS</td>
<td>Long</td>
<td></td>
<td>IP address of the Host at instance stop time (integer)</td>
<td>4</td>
</tr>
<tr>
<td>USER_ID</td>
<td>F</td>
<td>Long</td>
<td>The unique ID from the USERS table. Value can be NULL.</td>
<td>4</td>
</tr>
<tr>
<td>METRIC_TYPE</td>
<td>NN</td>
<td>Long</td>
<td>Metric type (0, 1, 2, 3). This field address to the proper data table. (**))</td>
<td>4</td>
</tr>
<tr>
<td>CORRELATOR_ID</td>
<td>Long</td>
<td></td>
<td>The unique ID from the CORRELATOR table</td>
<td>4</td>
</tr>
<tr>
<td>P_CORRELATOR_ID</td>
<td>Long</td>
<td></td>
<td>The parent correlator ID used to correlate different client/server transactions</td>
<td>4</td>
</tr>
</tbody>
</table>

**Notes:**

1. (*) This is a char field, because of RIM limitation on big integer fields (> 4 bytes). This field will represent an hexadecimal string such as “937de6c4751511d5a8090002557453d2” where each char represent 4 bits (multiplied by 32 is equivalent to 16 bytes).

2. (**) This field has to be interpreted as a mask: if the bit 0 is set (=1), a integer data type (32 or 64-bit) is present (stored in the TX_INT table), if the bit 1 is set a float data type (32 bit) is present (stored in the TX_FLOAT table), if the bit 2 is set a string data type is present (stored in the TX_STR table).

### Table 14. Data dictionary for the CORRELATOR table

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Key</th>
<th>Data type</th>
<th>Field Description</th>
<th>Field Size (bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CORRELATOR_ID</td>
<td>P</td>
<td>Long</td>
<td>A unique sequential (incremented by 1) number assigned by Tivoli Application Performance Management whenever a new record is added to the table.</td>
<td>4</td>
</tr>
<tr>
<td>TX_CORRELATOR</td>
<td>String</td>
<td></td>
<td>The correlator info passed by the TAPM engine</td>
<td>60 (*)</td>
</tr>
</tbody>
</table>

**Note:** (*) This field will represent an hexadecimal string where each char represent 4 bits. For the meaning of the TX_CORRELATOR string, please refer to appropriate documentation (Tivoli Application Performance Management Correlator Design).
**Table 15. Data dictionary for the METRIC_DEF table**

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Key</th>
<th>Data type</th>
<th>Field Description</th>
<th>Field Size (bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEF_ID</td>
<td>P</td>
<td>Long</td>
<td>A unique sequential (incremented by 1) number assigned by Tivoli Application Performance Management whenever a new record is added to the table</td>
<td>4</td>
</tr>
<tr>
<td>METRIC_UUID</td>
<td>P</td>
<td>String</td>
<td>UUID of metric definition. A metric definition defines the name and format of data values (named “metrics” in ARM), either numeric or string, that represent other interesting data related to a transaction instance.</td>
<td>32 (*)</td>
</tr>
<tr>
<td>METRIC_NAME</td>
<td></td>
<td>String</td>
<td>The Application-Defined Metric name</td>
<td>44</td>
</tr>
<tr>
<td>DATA_TYPE</td>
<td></td>
<td>Long</td>
<td>ARM Data type (***)</td>
<td>4</td>
</tr>
</tbody>
</table>

**Notes:**

1. (*) This field represents a hexadecimal string where each character represents four bits.
2. (***) The store able ARM 2.0 data types are:
   1 = ARM_Counter32 (32-bit)
   2 = ARM_Counter64 (64-bit)
   3 = ARM_CntrDivr32 (32-bit)
   4 = ARM_Gauge32 (32-bit)
   5 = ARM_Gauge64 (64-bit)
   6 = ARM_GaugeDivr32 (32-bit)
   7 = ARM_NumericID32 (32-bit)
   8 = ARM_NumericID64 (64-bit)
   9 = ARM_String8
   10 = ARM_String32

**Table 16. Data dictionary for the METRIC_INFO table**

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Key</th>
<th>Data type</th>
<th>Field Description</th>
<th>Field Size (bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>METRIC_ID</td>
<td>P</td>
<td>Long</td>
<td>UUID of metric definition. A metric definition defines the name and format of data values (named “metrics” in ARM), either numeric or string, that represent other interesting data related to a transaction instance.</td>
<td>32 (*)</td>
</tr>
<tr>
<td>METRIC_CTR</td>
<td>P</td>
<td>Long</td>
<td>Metric counter (1 .. 7)</td>
<td>4</td>
</tr>
<tr>
<td>DEF_ID</td>
<td></td>
<td>Long</td>
<td>The unique ID from the METRIC_DEF table</td>
<td>4</td>
</tr>
</tbody>
</table>

**Table 17. Data dictionary for the TX_INT table**

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Key</th>
<th>Data type</th>
<th>Field Description</th>
<th>Field Size (bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>INSTANCE_ID</td>
<td>P / F</td>
<td>Long</td>
<td>The unique ID from the INSTANCE table</td>
<td>4</td>
</tr>
<tr>
<td>METRIC_CTR</td>
<td>P</td>
<td>Long</td>
<td>Metric counter</td>
<td>4</td>
</tr>
<tr>
<td>VALUE_HIGH</td>
<td></td>
<td>Long</td>
<td>High part of the metric value for the specified data</td>
<td>4</td>
</tr>
<tr>
<td>VALUE_LOW</td>
<td></td>
<td>Long</td>
<td>Low part of the metric value for the specified data</td>
<td>4</td>
</tr>
</tbody>
</table>

**Note:** If the value to be stored is 32-bit, only the VALUE_LOW field is meaningful.
### Table 18. Data dictionary for the TX_STR table

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Key</th>
<th>Data type</th>
<th>Field Description</th>
<th>Field Size (bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>INSTANCE_ID</td>
<td>P / F</td>
<td>Long</td>
<td>The unique ID from the INSTANCE table</td>
<td>4</td>
</tr>
<tr>
<td>METRIC_CTR</td>
<td>P</td>
<td>Long</td>
<td>Metric counter</td>
<td>4</td>
</tr>
<tr>
<td>VALUE</td>
<td></td>
<td>String</td>
<td>The strings data types</td>
<td>255 (*)</td>
</tr>
</tbody>
</table>

**Note:** (*) This is a variable char field with a maximum of 255.

### Table 19. Data dictionary for the TX_FLOAT table

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Key</th>
<th>Data type</th>
<th>Field Description</th>
<th>Field Size (bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>INSTANCE_ID</td>
<td>P / F</td>
<td>Long</td>
<td>The unique ID from the INSTANCE table</td>
<td>4</td>
</tr>
<tr>
<td>METRIC_CTR</td>
<td>P</td>
<td>Long</td>
<td>Metric counter</td>
<td>4</td>
</tr>
<tr>
<td>VALUE</td>
<td></td>
<td>Float</td>
<td>Value for the specified data</td>
<td>4</td>
</tr>
</tbody>
</table>

### RDBMS Supported Versions

This section provides database product specific information to aid configuration of the database. To store the performance data on a relational database management system (RDBMS), the product uses the Tivoli Framework RIM as the interface.

For information about relational databases supported by the current desktop, refer to *Tivoli Management Enterprise Framework Release Notes*.

A RDBMS must be installed and configured. Refer to your RDBMS documentation for database installation instructions and requirements. Refer also, to the *Tivoli Application Performance Management Release Notes 2.1* for any late breaking information about configuring the RDBMS and the RDBMS Interface Module (RIM) host.

In addition to installing the RDBMS, you must designate a managed node as the RIM host in the Tivoli management region.

### RDBMS Client Configuration

When the RDBMS software has been installed you must ensure that it is accessible through TCP/IP. To verify the database client/server connection, the following tips can be followed. The indicated operating system platform has the corresponding database clients it supports in parenthesis.

**DB2 Client - V5.2 for Windows NT/95, V6.1 for Windows NT/95/98**

**Note:** Do not use the 6.1 client to connect to version 5.0 or 5.2 DB2 database servers because the client is not backward compatible.

**Defining Your Database Connection information**

Start the client configuration assistant by clicking **Start->Programs->DB2 for Windows NT ->Client Configuration Assistant**

Next, click **Add->Add Database SmartGuide**

Type in the required details as follows:

1. Source: Select **Manual configure connection to a DB2 database**.
2. Protocol: Select **TCP/IP**.

3. TCP/IP: Type in the *host_name* where the server is located and the *port_number* (example, 50000) where the service can be located or type in the *service_name* (example, db2cDB2) that is listed in the `c:/winnt/system32/drivers/etc/services` file.

4. Database: Type in the *database_name* you want to connect to (example, SAMPLE; which should already exist)

5. ODBC: Select **Register this database for ODBC** and select **As a system data source**.


7. Select **Done**

8. from **Client Configuration Assistant** choose **Test option**.

9. On **Test the connection**, type in *user_id* and *password*. Select **OK** and the test should return a successful message similar to:

   The connection test was successful.
   Database product = DB2/NT 6.1.0
   SQL authorization ID = db2
   Database alias = SAMPLE

10. Reboot.

**Client/Server Connection Test**

   The connection between client and server is tested as follows:

   1. Start a DB2 command line window
      
      `db2cmd`

   2. Start the DB2 interactive session:
      
      `db2`

   3. Within the DB2 Interactive and using `<db user password>`, type:
      
      `connect to <db name> user <db user>`

**DB2 Client - V5.2 for UNIX (HPUX 10.20, AIX4.3) V6.1 for UNIX (Solaris 2.6, HPUX 10.20, AIX 4.3)**

After installed the DB2 CAE (Client Application Enabler) without creating an instance during the initial install process, do the following:

1. Create a DB2 instance
   
   `db2icrt -u <db2 user> -a server <instance name>`

   **Note:** If this instance is for use with RIM, the `<instance name>` must match the DB2 user name. Additionally, with 3.6.1 and below, the username on the client and server DB2 instances must be the same.

2. Log into the system as the DB2 user.

3. Source in the DB2 environment, if not sourced:
   
   `. $HOME/sqlib/db2profile`

4. Start the DB2 interactive session:
   
   `db2`

5. Register the server node with the instance
   
   `catalog tcpip node <nodename> remote <node dns name> server <port # or svc name>`
6. Register the database
catalog database <db name> at node <nodename> authentication server

7. To see the available databases on a DB2 server, start a DB2 interactive session on the
server and enter the following command:
list database directory

**Client/Server Connection Test**
The connection between client and server is tested as follows:
1. Log into the system as the DB2 user.
2. Source in the DB2 environment:
   . $HOME/sqlib/db2profile
3. Start the DB2 interactive session:
db2
4. Inside the DB2 Interactive, type in:
   connect to <db name> user <db user>
   using <db2 user password>

**Oracle Client - V2.3 for Windows NT**

**Configuring the SQL*Net**
Configure SQL*Net by running ‘SQL*Net Easy Configuration’ from the Oracle for
Windows NT/95 program group.
1. Select **OK** to Add Database Alias.
2. Type in a Database Alias name.
3. Select **OK** to continue.
4. Type in the TCP/IP Host Name.
5. Type in the Database Instance name: for example ORCL, OR73, PRCL.
6. Select **OK** to continue.
7. Select **Yes** to add this database alias.
8. Select **Cancel** to exit if done or you can Add, Modify, Delete Database Alias or View
   Configuration Information.

**Notes:**
1. The configuration file TNSNAMES.ORA is stored in the Oracle install directory on
   your hard drive (if you installed on ‘c:\orant\c:\orant\network\admin\tnsnames.ora).
2. Once you have edited the tnsnames.ora file with an editor, you cannot run the
   'SQL*Net Easy Configuration’ utility again to add other Oracle entries. If you do,
   it will wipe out all your previous changes.

**Database Connectivity Test**
If you have an ID, you can test the connection to the database as follows:
1. Select SQL*Plus 3.3 from the Oracle for Windows NT/95 program group.
2. Type in your username, password, and the Host String (Database Alias name from above).

3. Select OK to continue.

4. If the connection is successful, connection information is displayed, type in exit to quit SQL*Plus.

5. If the connection is not successful, the Oracle ERROR: "ORA-12154" will be displayed. If this occurs, press Return twice to exit SQL*Plus and retry the connection. Verify the information that you have typed in.

Oracle Client - Net 8.0.5 for Windows NT/95

Configuring the SQL*Net

Run ‘Oracle Net8 Easy Config’ from the Oracle for Windows NT/95 program group to configure SQL*Net

Note: If you get a java.exe error while trying to run Easy Config, then, you can edit the tnsnames.ora file manually to type in the Oracle server configuration information.

1. Select the Add new service radio button; Type in a New Service Name (Database Alias) name for example: cafaron805; Select Next to continue.

2. Select TCP/IP as the network protocol, and click Next to continue.

3. Type in the TCP/IP Host Name.

4. Type in the appropriate Port Number and click Next to continue.

5. Type in the Database Instance name: for example ORCL, OR73, PRCL; click Next to continue.

6. Select TEST SERVICE to verify the entry is correct.

7. Type in your username and password and select TEST.

8. Select DONE when successful.

9. Select Next, then Finish.

Note: The configuration file TNSNAMES.ORA is stored in the Oracle install directory on your hard drive (if you installed on 'c:\orant') in c:\orant\network\admin\tnsnames.ora.

Oracle Client - V2.3 for UNIX (Solaris 2.6, HPUX 10.20, HPUX 11.0, AIX 4.3)

To configure SQL*Net on UNIX, you must manually add an entry in the tnsnames.ora for the alias that is used by SQL*Net to connect to the Oracle server. This instance for the alias can be inserted anywhere in the file. The alias we added was nt. As you can see, you must add the .world extension to the alias name to specify the domain.

The nt.world entry is for an HOST = london (an Oracle NT server), and SID = ORCL (the Oracle database instance name on NT). The following is an extract from the tnsnames.ora file for this alias.

```plaintext
nt.world =
  (DESCRIPTION =
   (ADDRESS_LIST =
    (ADDRESS =
     (COMMUNITY = tcp.world)
     (PROTOCOL = TCP)
     (Host = london)
     (Port = 1526)
    )
   )
```
The PORT number is used by SQL*Net to connect between the Oracle server and the client. The PORT for the alias must be the same as the PORT that the TNS listener is using on the Oracle for NT server, and the SID must be the database instance name on the Oracle for NT server.

Database Connectivity Test
After the alias is created, you can use either the \texttt{tnsping} command to see if you can contact the TNS listener or you use SQL*Plus to connect to the database instance.

The following example use the Oracle TNS ping utility and the Oracle SQL*Plus utility to verify that an Oracle for AIX client can connect to the Oracle for NT server.

\texttt{# tnsping nt 5}
\texttt{TNS Ping Utility for IBM/AIX RISC System/6000:}
\texttt{Version 2.3.3.0.0 - Production on 28-APR-98 17:05:43}
\texttt{Copyright (c) Oracle Corporation 1995. All rights reserved.}
\texttt{Attempting to contact (ADDRESS=(COMMUNITY=tcp.world)(PROTOCOL=TCP)(Host=london)(Port=1526))}
\texttt{OK (1290 msec) OK (60 msec) OK (60 msec) OK (60 msec) OK (30 msec)}

The following display shows the SQL*Plus connection from the Oracle for AIX client to the Oracle for NT server.

\texttt{# sqlplus system/manager@nt}
\texttt{SQL*Plus: Release 3.3.3.0.0 - Production on Tue Apr 28 17:08:47 1998}
\texttt{Copyright (c) Oracle Corporation 1979, 1996. All rights reserved.}
\texttt{Connected to: Oracle7 Server Release 7.3.2.2.0 - Production}
\texttt{At the SQL> prompt, you can execute SQL commands.}

Informix Client - SDK 2.30 for Windows NT/95
After the client installation and rebooting you need to setup the ODBC data source as follows:

1. To create the ODBC data source select \texttt{Start} \rightarrow \texttt{Settings} \rightarrow \texttt{Control Panel} \rightarrow \texttt{ODBC}
2. Highlight the System DSN tab and choose \texttt{Add}.
3. Select \texttt{INFORMIX 3.30 32 BIT} and select \texttt{Finish}.
4. Type in the Data Source Name on the General tab (for example, stores7).
5. On the Connection tab enter the following:
   a. Server Name -- The server name can be either the value defined for the \texttt{DBSERVERNAME} or \texttt{DBSERVERALIASES} parameter in the \texttt{onconfig} file for the Informix server.
   b. Host Name of the Informix Server.
   c. Service name or Port number where service is available (example, turbo). This should be the same port number that is defined in the \texttt{etc/sqlhosts} file on the Informix.

---

12. Setting up the Database
server. If you use the service name instead of the port number, make sure there is an entry for the service name in the \winnt\system32\drivers\etc\services file on the client machine.

d. Select a connection protocol - onsoctcp
e. User ID
f. Password
g. Select the Test Connection button to ensure connectivity.

Informix Client - SDK 2.30 for UNIX (Solaris 2.5 - 2.5.1 - 2.6 - 7, HPUX 10.2 - 11.0, AIX 2.1 – 4.3 – 4.3.1 – 4.3.2 - 4.3)

After installing the Informix CLI product, copy the $INFORMIXDIR/.odbc.ini and $INFORMIXDIR/etc/sqlhosts file from the Informix server to the newly installed client. If the .odbc.ini file and sqlhosts file you just copied do not contain the connection information for your database, perform the next three steps:

1. Edit the ODBC Copy the ODBC Data Sources in $INFORMIXDIR/.odbc.ini file. Add an entry for the new database:
   
   inv1=Informix 9.x ODBC Driver
   or
   inv1=Informix 7.x ODBC Driver

2. Define the ODBC Database driver in $INFORMIXDIR/.odbc.ini file.
   
   [inventory]
   Driver=<your $INFORMIXDIR>/cli/dlls/
   IXinf708.so;
   Please refer to other entries for exact driver file
   Description=Informix 9.x ODBC Driver
   or
   Description=Informix 7.x ODBC Driver
   Database=inv1@<informix server name>;
   Server name is located in $INFORMIXDIR/etc/sqlhosts
   HostName= <yourhostname>
   LogonID=
   Password=

3. Edit the $INFORMIXDIR/etc/sqlhosts file. Add an entry for the new informix server. example,.
   
   <db server name> <comm method> <hostname> <port #>
   chip730 onsoctcp chip 1617

   Alternatively, you can make an entry in the /etc/services file for the database service. You can then specify the database service name instead of the port number in.

Client Connectivity Test

Perform a client connection as follows:

1. Source in the CLI environment
   
   . $INFORMIXDIR/cli/.cli.sh

2. Run adhoc
   
   $INFORMIXDIR/cli/demo2/adhoc

3. Enter the connection info and a SQL statement to execute.
If you have installed the stores7 sample database and have an entry in the .odbc.ini file for stores7, you can use the following sample adhoc session:

Enter Data Source Name: stores7
Enter userid: informix
Enter password: informix
Enter SQL string: select * from customer

Note: On AIX, the $INFORMIXDIR/.odbc.ini file must be copied to the Informix home directory for adhoc to work.

Sybase Client for Windows NT
After installing the Sybase 11.1.1 Client for Windows, apply EBF 8550. This software is located under the Win95-NT directory from which you installed the 32 bit client (directory name is EBF8550). Follow the directions in the Cover.ROLL.8550MS Word document located in the EBF8550 directory. Basically you just copy the files from each folder (bin, dll, etc.) to your sybase folders (bin, dll, etc.). Reboot afterwards.

Make sure the SYBASE and DSQUERY variables are set correctly before using isql. SYBASE should be set to the directory Sybase is installed in. DSQUERY should be set to the name of the Sybase server. The interfaces file can be consulted for a list of available servers.

Defining the Servers
Define the servers as follows:
1. Run DSEDIT from the Sybase program group.
2. Select OK on the 'Select Directory Service'
3. Select Server Object-> Add
4. Enter a server name from the list below, select OK when done.
5. Double Click on Server Address field
6. Select Add.
7. From the Protocol drop down box, select NLWNSCK.
8. Enter the network address that applies to the server name.
9. Select OK to save.

Test Pinging the Server
Ping the server as follows:
1. From the DSEDIT screen, Select Server Object->Ping Server->Ping
2. The display should read "Succeeds". If not, check your parameters again and retry.
3. Repeat Server Objects->Add to add additional servers.

Running SQL Queries
To run SQL Queries:
1. Run ‘SQL Advantage’ from the Sybase program group.
2. Select Server/Connect
3. Type in the server name or select it from the drop down box.
4. Type in your login and password information, then select Connect.
Client Connectivity Test
Perform a client connection as follows:

1. Make sure the SYBASE and DSQUERY variables are set correctly before using isql. SYBASE should be set to the directory Sybase is installed in. DSQUERY should be set to the name of the Sybase server. The interfaces file can be consulted for a list of available server.

2. Change directory:
   
   cd /data/sybase/bin

3. Run isql:
   
   ./isql -U <username> -P <password> -S <server name>
   
   example
   
   ./isql -U tivoli -P tivoli -S CAFARON

Microsoft SQL Client 6.5 for Windows NT/95

Running SQL Client Configuration Utility
Run the SQL client configuration utility as follows:

1. Run the SQL Client Configuration Utility in the 'SQL Server Tools’ group.

2. Select Net Library, Select TCP/IP sockets from the 'Default Network’ drop down box.

3. Select Done.
   
   If an error occurs during this step, update your path according to the information in step 1 of the connectivity test below and try again.

Client Connectivity Test
Test the connection by running ISQL/w from the SQL Server Tools group as follows:

1. Make sure that c:/mssql/binn is in your path

2. If connecting to a MS Sql 6.0 server, make sure you have your network protocol set to TCP/IP. If set to IPX/SPX, then you may get errors during the EA build (the create table part).

Microsoft SQL Client 7.0 for Windows NT/95

Running SQL Client Configuration Utility
Run the SQL client configuration utility as follows:

1. Run the ‘Client Network Utility’ from the Microsoft Sql Server 7.0 program group.

2. Select TCP/IP from the 'Default Network’ drop down box.

3. Select Add to add a server alias.

4. Add an entry for the Server alias and computer name. The port address is 5000. Select OK to add this new server alias.

Client Connectivity Test
Test the connection by running ‘Query Analyzer’ from the Microsoft Sql Server 7.0 Program group as follows:
1. To test connect and run queries, select your *clientserver* from the Sql Server drop down box and enter your database ID and password.
Tivoli Application Performance Management summarizes transaction data at individual endpoints to produce a single summary record at the end of the collection interval and logs individual transaction instance data. The summary record is aggregated to hourly records and transaction instance data is grouped together at intervals that can be customized, before being uploaded to the database.

**Note:** Transaction instance data logging could result in a very large amount of data. You will need to choose whether you want to enable the logging activity (See Chapter 8, “Using ARM Transaction Correlators” on page 65) and the amount of data to be logged locally and uploaded to the database.

The product writes performance data onto the endpoint where it has made measurements. Distributed Monitoring can be used at the endpoint to compare data against thresholds. The data is automatically aggregated into hourly units and uploaded, using the Tivoli RDBMS Interface Module (RIM), into a relational database.

Once the data is stored in the database, it can be extracted and built into cubes, and the Tivoli Decision Support can be used for detailed analysis. See *Tivoli Decision Support for Application Performance User’s Guide*.

**Collecting Data**

Data collection is controlled by the collection settings for an application delivered to the endpoint within the product profile. Data collection schedules in the profile, enable you to choose whether to collect data from the application daily, or on specified days of the week, and for how long each day.

You can also choose the collection interval. At the end of the collection interval, a record is written into a pair of binary log files on the endpoint where you are running the ARM instrumented application, the EPP probe or the Client Capture scenario. See “Setting the Data Collection Period” on page 87 for details.

**Instance Data Logging**

Transaction instances data -with or without correlators- are logged in a pair of files on the endpoint `marrawdata<date>.dat` and `marrawinfo<date>.dat`. The file `marrawdata<date>.dat` contains the raw data (the data for each `arm_start` and `arm_stop` pair) in terms of response time (ns), completion status and eventually the correlator and the parent correlator. The file `marrawinfo<date>.dat` contains information about the application for which data is being collected:

- The application name.
- The transaction name.
The UserId

The files are stored locally until they are sent to the database at fixed intervals. The maximum size of instance data files can be defined using `wmarsetstatus` command with the `TransInstanceFilesMaxSize (range 100 - 100000000)` parameter. This parameter sets the maximum size (in byte) occupied for transaction instance data on the endpoint for each day.

If `MaxTransInstanceFiles` is set to `m` and `TransInstanceFilesMaxSize` is set to `n`, the maximum space needed on the endpoint (for transaction instance data) is: `m * n`. After the maximum size of data is reached, the agent stops writing transaction instance data for the current day and restarts writing after midnight.

Another parameter (`MaxTransInstanceFiles (range 2 - 30)`) controls how many days data files can be stored locally on the endpoint before they are to be deleted. This controls how many files are cached if the mid-level manager cannot be reached.

At intervals that can be customized, a merging process is run. In this process the contents of two files are combined into a single file named `marinst<date><hour>.dat`.

Instance data tracing can be controlled at endpoint level by setting the `InstanceEnabled` (true, false) property with the `wmarsetstatus` command or at collection level using the `-n` flag parameter of `wmarstartcoll` command.

Endpoints with a large number of transactions should be carefully tuned on an application-by-application basis in order to keep amount of performance data at a low level. For the same reason, data sent to the database can be controlled on an application-by-application basis using the `wmarstartcoll` command.

**Summary Data Logging**

Each application has its summary data logged serially into two binary files `marinfo<date>.dat` and `mardata<date>.dat`.

The file `marinfo<date>.dat` contains information about the application for which data is being collected: the application name, transaction name, user ID, for example.

The file `mardata<date>.dat` contains the collected values for each application. A new record is created for each interval in which data is collected. The file contains a summary record for each transaction in each collection interval, which includes the average minimum and maximum response times.

At the end of each day, the two files are closed, and a new pair of files is opened for input. The string `<date>` in the file names has the format yyyymmdd, and so it is different each day.

At intervals that can be customized, an aggregation process is run. In this process the contents of the two log files are merged and aggregated into a single file `maraggr<date><hour>.dat`. This process is described in “Uploading Data from the Endpoint to the Database” on page 136.

**File Clean-up**

The product agent runs a daily cleanup procedure to remove from memory all dangling transactions that issued start calls, without completing them with a stop. The cleanup also removes applications and transaction type entries, for those applications that have not made any arm call in a certain period of time.
Monitoring Data Against Thresholds

If you have Tivoli Distributed Monitoring installed, you can monitor application performance through the product Monitoring Collection.

The Monitoring Collection

The product monitoring collection is shipped on the installation CD-ROM as a separate install component. As soon as you have installed it on the region server, the monitoring collection appears in Tivoli Distributed Monitoring. The product monitors and their descriptions are listed here.

<table>
<thead>
<tr>
<th>ARM Monitor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transaction Count</td>
<td>Number of times the transaction completed successfully during the last collection interval</td>
</tr>
<tr>
<td>Failed Transactions</td>
<td>Number of times the transaction failed during the last collection interval</td>
</tr>
<tr>
<td>Abended Transactions</td>
<td>Number of times the transaction ended abnormally during the last collection interval</td>
</tr>
<tr>
<td>Average Transaction</td>
<td>Average response time of the transaction during the last collection interval (in milliseconds)</td>
</tr>
<tr>
<td>Average Transaction</td>
<td></td>
</tr>
<tr>
<td>Time</td>
<td></td>
</tr>
<tr>
<td>Minimum Transaction</td>
<td>The minimum response time (in milliseconds) of the transaction during the last collection interval</td>
</tr>
<tr>
<td>Response Time</td>
<td></td>
</tr>
<tr>
<td>Time</td>
<td></td>
</tr>
<tr>
<td>Maximum Transaction</td>
<td>The maximum response time (in milliseconds) of the transaction during the last collection interval</td>
</tr>
<tr>
<td>Response Time</td>
<td></td>
</tr>
<tr>
<td>Time</td>
<td></td>
</tr>
<tr>
<td>Bucket count #1</td>
<td>Number of times the transaction response time was below bucket boundary 1 during the last collection interval</td>
</tr>
<tr>
<td>Bucket Count #2</td>
<td>Number of times the transaction response time fell between bucket boundaries 1 and 2 during the last collection interval</td>
</tr>
<tr>
<td>Bucket Count #3</td>
<td>Number of times the transaction response time fell between bucket boundaries 2 and 3 during the last collection interval</td>
</tr>
<tr>
<td>Bucket Count #4</td>
<td>Number of times the transaction response time fell between bucket boundaries 3 and 4 during the last collection interval</td>
</tr>
<tr>
<td>Bucket Count #5</td>
<td>Number of times the transaction response time fell between bucket boundaries 4 and 5 during the last collection interval</td>
</tr>
<tr>
<td>Bucket Count #6</td>
<td>Number of times the transaction response time fell between bucket boundaries 5 and 6 during the last collection interval</td>
</tr>
<tr>
<td>Bucket count #7</td>
<td>Number of times the transaction response time was above bucket boundary 6 during the last collection interval</td>
</tr>
</tbody>
</table>

See *Distributed Monitoring Collection Reference Guide* for details of how to set up and use the Distributed Monitoring Monitors listed above.

Sending Results to the Tivoli Enterprise Console

The product includes class definitions for use with the enterprise console. If you create Application Performance Management monitors using the DM Monitoring collection, and those monitors trigger, you can create enterprise console events. The class definitions enable enterprise console to understand the format of those events when they arrive.

When you install the DM Monitoring Collection for the product from the Install disk, the class definitions file is automatically copied to the region server as a file called
tapmmon.baroc. This file is copied to the %BINDIR%\...\generic\SentryMonitors directory on Windows, and to the $BINDIR/../generic/Sentry Monitors directory on UNIX platforms.

To define the product event classes to the Tivoli enterprise console event server, do the following:

1. Import the Tivoli Distributed Monitoring event classes to the rule base by issuing the following command:
   
   ```sh
   wimprbclass $BINDIR/../generic/SentryMonitors/
   Sentry.baroc $rule_base
   ```

   where: $rule_base is the name of the rule base to which you want to add the Tivoli Distributed Monitoring event classes.

   **Note:** If the rule base is the default rule base you cannot add other event classes to it. You must create a new rule base by issuing the `wcrtbr` command.

2. Import the product event classes to the rule base by issuing the following command:
   ```sh
   wimprbclass $BINDIR/../generic/SentryMonitors/
   tapmmon.baroc $rule_base
   ```

   where: $rule_base is the name of the rule base to which you want to add the Tivoli Application Performance Management event classes.

3. Recompile the rule base by issuing the following command:
   ```sh
   wcomprules $rule_base
   ```

   where: $rule_base is the name of the rule base to be compiled.

4. Load the rule base by issuing the following command:
   ```sh
   wloadrb $rule_base
   ```

   where: $rule_base is the name of the rule base to be loaded.

5. Stop and restart the enterprise console server by issuing the following commands:
   ```sh
   wstopesvr
   wstartesvr
   ```

   For more information about enterprise console events and classes, see *Tivoli Enterprise Console User’s Guide*.

   For more information about Tivoli enterprise console commands, see *Tivoli Enterprise Console Reference Manual*.

**Retrieving Results**

At intervals that can be customized, the data log files are aggregated into a single data file at each active endpoint. The aggregated file is detailed into records of hourly data and is sent to the gateway for storage in the database.

You can force the data aggregation and immediate upload to the gateway and then to the database at any time, by using the `wmarsenddata` command.

**Uploading Data from the Endpoint to the Database**

Performance data is moved to the database in two independent steps:

1. At predefined times each day the data is aggregated or merged, and the data file is uploaded to the gateway.
2. From the gateway the data is sent via RIM to a database, from which it can be retrieved and used.

Some engine properties can be set at the endpoint for customizing the aggregation and upload process by issuing the `wmarsetstatus` command.

The time and frequency that aggregation takes place at an endpoint and when upload takes place from the gateway to the database, can be specified with the `wmarsetstatus` command.

The aggregation and upload process from the endpoint to the gateway, comprises several stages as detailed below. These stages are controlled by the engine at the endpoint and by the mid-level manager at the gateway.

1. After midnight, the binary files are switched, so that a new pair of files is opened for input.
2. At the time specified by the engine settings, the engine aggregates the files to hourly boundaries, to produce a log file identified with a name containing the date and the time interval to which the data belongs.
3. The endpoint upcalls its gateway, and sends the aggregated log file or files to it.

**Note:** Aggregated files are deleted from the endpoint as soon as they are stored at the gateway. Oldest non-aggregated data files are deleted as the number of files exceeds the `MaxBinaryFiles` setting.

Once the aggregated files are uploaded from the endpoints and stored at the gateway, they are transferred by the RIM object to the database, at predefined times.
The mid-level manager properties for customizing the database upload, can be set and changed at the gateway. These properties are set by the `wmarsetstatus` command accompanied by the `-g` flag and `managed_node` name.

You can force the data aggregation and immediate upload to the database at any time by using the `wmarsenddata` command.

**Data Analysis**

Measurement results that are stored in the database can be compiled and presented in many different ways. To select and construct specific reports from the information stored in the database, you can use the Tivoli Decision Support interface, taking advantage of the On-line Analytical Process (OLAP) technology. See *Tivoli Decision Support for Application Performance User’s Guide* for details.

---

**Figure 78. Data upload**

The mid-level manager properties for customizing the database upload, can be set and changed at the gateway. These properties are set by the `wmarsetstatus` command accompanied by the `-g` flag and `managed_node` name.

You can force the data aggregation and immediate upload to the database at any time by using the `wmarsenddata` command.

**Figure 79. Tivoli Decision Support**
Looking at Todays Data

The database holds historical data that is, aggregated data collected before today. When you want more recent information, such as when you are finding and correcting problems on the net, use the Web GUI (see Chapter 11, “The Web GUI” on page 97).

The following data displays are available using Web GUI:

- Average Response Time
- Minimum Response Time
- Maximum Response Time
- Average Min. and Max. Response Time
- Number of Good Transactions
- Number of Aborted Transactions
- Number of Failed Transactions
- Number of Good, Aborted and Failed Transactions

Getting Information from Instance Data

This section details queries that may be used to extract further information from instance data that has already been uploaded to the database.

Get all Instances for an Application and User

This query is used to get all instances for a specified application and user. The markups `appl_name` and `user_name` must be replaced by existing application name and user name, obtained by querying tables APPLICATIONS and USERS.

```sql
SELECT E.TX_NAME, A.INSTANCE_ID, A.STOP_DATE, A.STOP_TIME_S, A.STOP_TIME_NS,
       A.RT_MS, A.RT_NS, A.STATUS
FROM TX_INSTANCE A,
     UUID_INFO B,
     APPLICATIONS C,
     USERS D,
     TRANSACTIONS E
WHERE A.INFO_ID=B.INFO_ID AND
     B.APPL_ID=C.APPL_ID AND
     A.USER_ID=D.USER_ID AND
     B.TX_ID=E.TX_ID AND
     C.APPL_NAME=’appl_name’ AND
     D.USER_NAME=’user_name’
ORDER BY E.TX_NAME;
```

Get all Instances for a Specified Transaction

This query is used to get all instances for a specified transaction class, when the application and the user are specified.

```sql
SELECT A.INSTANCE_ID, A.STOP_DATE, A.STOP_TIME_S, A.STOP_TIME_NS,
       A.RT_MS, A.RT_NS, A.STATUS
FROM TX_INSTANCE A,
     UUID_INFO B,
     APPLICATIONS C;
```
**Get all Instances running on a Specified Host**

This query is used to get all instances running on a host for a specific user (first query the HOSTS table to get the user name).

```sql
SELECT A.INSTANCE_ID, A.STOP_DATE, A.STOP_TIME_S, A.STOP_TIME_NS,
A.RT_MS, A.RT_NS, A.STATUS
FROM TX_INSTANCE A,
    HOSTS B,
    USERS C
WHERE A.HOST_ID=B.HOST_ID AND
    A.USER_ID=C.USER_ID AND
    B.HOST_NAME='host_name' AND
    C.USER_NAME='user_name';
```

**Get How Many Instances in the Data Base are Correlated to Each Other**

This query is used to know how many instances there are in the database which are correlated among each other. The result of the query is a set of rows ordered by transaction, application, user and host.

```sql
SELECT F.TX_NAME, A.INSTANCE_ID, A.STOP_DATE, A.STOP_TIME_S, A.STOP_TIME_NS,
A.RT_MS, A.RT_NS, A.STATUS,
C.APPL_NAME, D.USER_NAME, E.HOST_NAME
FROM TX_INSTANCE A,
    TX_INSTANCE B,
    APPLICATIONS C,
    USERS D,
    HOSTS E,
    TRANSACTIONS F,
    UUID_INFO G
WHERE A.P_CORRELATOR_ID<>0 AND
    A.P_CORRELATOR_ID=B.CORRELATOR_ID AND
    A.INFO_ID=G.INFO_ID AND
    G.APPL_ID=C.APPL_ID AND
    G.TX_ID=F.TX_ID AND
    A.USER_ID=D.USER_ID AND
    A.HOST_ID=E.HOST_ID
ORDER BY F.TX_NAME,
    C.APPL_NAME,
    D.USER_NAME,
    E.HOST_NAME;
```
If You Have Problems

This appendix provides information to help you avoid problems and gather information to analyze and help solve problems that may arise when using the product. It contains the following sections:

- Checking the Product Notice Group
- Trouble Shooting
- Error Messages

Checking the Product Notice Group

The product notice group lists information that is generated by an error or exception. The following type of information is posted to the notice group:

- Error messages that are too long or complex to be viewed online as exceptions.
- Information messages that can provide an audit trail of activity in Tivoli Distributed Monitoring, for example, when the Device Manager database is backed up or restored.
- Errors generated from other applications for which Tivoli Distributed Monitoring cannot display a message, for example, inconsistencies in the Device Manager database that are detected after you run the `wchkdb` command on device groups.
- Problems that arise during upload to the database

Refer to Tivoli Management Framework User’s Guide for information about notice groups, including the authorization role required to perform operations on notice groups.

Trouble Shooting

This section is a summary of common problems that have been found during verification and test of the product. They are provided to help isolate and solve the current problems, and help you determine whether there is a definite problem before contacting customer support. It also helps you gather information on problem symptoms that would be helpful to your Tivoli representative.

Before reading this section, you should be familiar with the products and latest Tivoli Application Performance Management Release Notes. The documents and online help that you should read before proceeding are:

- This User’s Guide Version 2.1
- Tivoli Application Performance Management online help
- Tivoli Application Performance Management 2.1 Release Notes
Using Log Files

Log files are created by the product as it operates. In these log files you can find information on any problems that have occurred.

You can set traces for:

- The engine
- Mid-level manager
- RDBMS Interface Module (RIM)

Note: The following instructions apply to Tivoli Application Performance Management product traces. For details on how to set traces and logs on Tivoli Management Framework, refer to the Framework documentation.

Engine Trace Settings

The engine provides several traces (engine, back-end, EPP, Client Capture).

Engine

Set Maximum Engine level with CLI Tivoli Application Performance Management command:

```
wmarsetstatus LogLevel=3 <endpoint_name>
```

where `<endpoint_name>` is the name of the machine on which you need to trace. This setting logs activity to `tapm_engine.log`.

Back-end

Set Maximum back-end level by stopping the engine, manually editing the `tapm_ep.cfg` configuration file using the following values:

```
[ENDPOINT::LOG]
LogLevel=1 Minimum backend trace level

[ENDPOINT::LOG]
LogLevel=2 Medium backend trace level

[ENDPOINT::LOG]
LogLevel=3 Maximum backend trace level
```

You must now restart the engine. This logs back-end activity to the file `tapm_ep.log` file on the endpoint.

EPP Controller

Log file `Tectrl.log` is created on the endpoint after the first EPP Probe collection starts. The trace level of the file can be set by stopping the engine and manually adding the following lines in the `tapm_ep.cfg` file:

```
[TECTRL::LOG]
FileName=Tectrl.log
LogLevel=1 Minimum Tectrl.log trace level
FileMaxSize=50000
DirName=logs Default directory
```
where:
LogLevel can assume the following values:
   1 = Minimum Tectrl.log trace level
   2 = Medium Tectrl.log trace level
   3 = Maximum Tectrl.log trace level

Notes:
1. These lines can be added only when no EPP Probe is running on the endpoint.
2. If you want DEBUG information for the EPP probes running on the endpoint, you must set the consDebugEnabled parameter to true in the sysCfg.tapm configuration file.
3. The sections on EPP probes above are available only when at least one collection has been started on the endpoint.

The EPP probes log files can be found under the following path (this path can be changed by setting the DirName value in the tapm_ep.log):
path: $LCF_DATDIR/Mar/logs/

EPP probe system configuration file: sysCfg.tapm
path: $LCF_DATDIR/Mar/Simulation/Epp/

Information returned by the endpoint engine to a logfile is the same for all EPP probes. It is the overall transaction time for each URL in the form:
<PeId>::<Date>::<TimeStamp>::<SubTestID>::<TrxId>::<STrxId>::<TrxType>::<StatusCode>::<TrxDuration>::<Extended Attributes>

PeId
The PeId (probe ID) is generated to identify a running instance of a probe. The probe id is analogous to a process id, and the probe ID space is used to identify the probe in the logfile, and also when executing remote commands, such as start, stop and list probe status.

Note: The probe ID associated with a probe is only persistent while the endpoint engine is running, and may change when the engine is restarted.

Date
The date of the (sub) transaction execution. The date has the form
<year><month><day>.

Timestamp
The time of day of the (sub) transaction execution in the following format:
<hour><minute><second>.

SubTestID
Identifies a logical data stream produced by a probe transaction. This field is used when a transaction generates multiple, logically independent, tests. The SubTestID is used to identify the records associated with each test. This ID, which is optional, can only be defined for sub transaction records. ("*" is a reserved ID, and represents a non initialized SubTestID.)

TrxId
A unique ID that identifies the parent transaction. If the STrxId == 0, then the record represents the parent or overall transaction duration.
STrxId
A unique ID that identifies the sub transaction record. The STrxId will be > 0 in sub transaction records, In which case the TrxId identifies the parent transaction which contains this sub transaction.

TrxType
Represents the type of the transaction. This value is fixed for the main transaction record, where "__main" is used to define the transaction type. For each sub transaction record, the probe transaction must define a non null transaction type.

Status Code
An integer which represents the results of the (sub) transaction execution. A Status Code of "1" indicates that the (sub) transaction completed without error. A Status Code < 0 corresponds to the default codes defined by the EPP framework. Status Codes > 1, are defined by, and are specific to, a particular test transaction.

TrxDuration
The duration of the (sub) transaction, in milliseconds. If the Status Code indicates that an error was encountered, the EPP framework sets the duration to the configured timeout interval. The TrxDuration reported in the main transaction record always represents the duration of the synchronous execution of the transaction. In sub transaction records, the duration may represent a synchronous or asynchronous interaction.

Extended Attributes
Probe specific attributes which describe the sub transaction.
- Extended attributes are associated only with sub transaction records.
- If multiple extended attributes may be associated with a sub transaction record, they are delimited by the one character string ":".
- Each extended attribute consists of a key/value pair, which is formatted as "key=value".

Client Capture
Client Capture log files are only generated on machines where client capture collections have been started. Log file TAPMBridge.log is located as follows:
path: $LCF_DATDIR\Mar\ClientCapture

LoadRunner
LoadRunner simulation script log files are only generated on machines where simulation collections have been started. Log file ScriptCtrl.log is located as follows:
path: $LCF_DATDIR\Mar\ScriptCtrl This file is generated only when an error occurs.

Endpoint Log File Locations
The Tivoli Application Performance Management engine and backend log file can be found on the following paths:
Engine: tapm_engine.log
path: $LCF_DATDIR/Mar/

Backend: tapm_ep.log
path: $LCF_DATDIR/Mar/
Configuration file: tapm_ep.cfg
    path: $LCF_DATDIR/Mar/

EPP Controller: Tectrl.log
    path: $LCF_DATDIR/Mar/logs/  by default

Tivoli Application Performance Management EPP probe log files:
    cons.cfg.XXX.YYYYMMDDHHMMSS.X.log
    cons.ctrl.XXX.YYYYMMDDHHMMSS.X.log
    cons.msg.XXX.YYYYMMDDHHMMSS.X.log

Mid-level Manager Settings

Set trace level
    Set maximum mid-level manager level with CLI command:
    "wmarssetstatus -g LogLevel=3 <Managed_Node_label>"

Managed node log file tapm.log
    The log file can be found in the tapm directory in the path indicated by the wtemp
    command.

RIM Settings

Set trace level
    The wrimtrace command enables or disables tracing information for RIM objects.
    The communication between the RIM object and client program, and RDBMS errors
    are printed to the RIM log. You can set and change the location of the RIM log file
    with the following command:
    odadmin environ get > environ.cfg
    Edit the environ.cfg file you created by adding the following line:
    RIM_DB_LOG=/path_name/file_name
    Set the environment with
    odadmin environ set < environ.cfg
    Note: Ensure that the path name you specify exists, because it will not be created.
    Executing the wrimtrace command without the trace_level argument prints the
    current trace level to a standard output. The tracing function is intended for
    debugging purposes. If enabled for extended periods of time, tracing can decrease
    performance and slow the processing of RIM calls considerably.
    Set the RIM trace to information and error with:
    wrimtrace tapm "INFORMATION|ERROR"
    Now kill all the RIM_* processes running on the RIM Host with the UNIX kill
    command or with the Win NT Task Manager, the processes is respawned
    automatically with the new trace level and log output you set.
    For more information refer to the Tivoli Management Framework Documentation.

Common Problems

This section lists common causes of problems.

1. Distributing a profile on a Windows 95 operating system.
For the correct operation of the Tivoli Application Performance Management 2.1 on Windows 95 operating system platform it is necessary to install the Microsoft Windows 95 Service Pack 1 and the Internet Explorer 5.x application.

2. Attempting to uninstall the Tivoli Application Performance Management 2.1 and Tivoli Application Performance Management 2.1 Web GUI from a Tivoli Framework 3.6.5. If you have installed the Tivoli 2.0 Web GUI both on the region server and on the managed node, you must also uninstall Tivoli Application Performance Management 2.0. Use the following workaround:

- First uninstall WEB GUI with the command:
  
  "wuninst TAPMWEB <managed_node_label> -rmfiles"

- Second uninstall the Tivoli Application Performance Management product with the command:
  
  "wuninst TAPM_1.0 <managed_node_label> -rmfiles"

3. Attempting to distribute a Client Capture collection after having changed the endpoint directories location.

After changing the endpoint location on a Windows operating system the system is not aware of the change until the endpoint is restarted.

4. Attempting to start any Tivoli Application Performance Management command after the last command was in time out.

If you try to start any command line Tivoli Application Performance Management command after that the last command was in time out, it is likely that a mar_ep process is hung. You must manually kill the process on the endpoint to continue.

5. Under some conditions, scheduling fields in the Add Application to the Profile GUI display incorrect information.

If you change the endpoint label without stopping and restarting the engine, it continues to treat endpoint label as the old one. When the AggregatorManager starts the aggregation and issues the upload request to its gateway, it calls the mar_up process with a wrong label. Suggested solution is that the AggregatorManager should resolve the endpoint label before starting the upload process.

6. If the IBM HTTP Server is installed separately (see "Installing the HTTP Server" on page 23), the HTTP Administration service is stopped, but service is set to start automatically if you reboot.

After the Tivoli Application Performance Management 2.1 Web GUI installation you must change the startup properties of the HTTP Administration Server in the Windows service panel, to avoid automatic start of the service you reboot.

7. If you follow the following step the Rule Editor panel of the Adding Schedule Information dialog is disabled.

   a. Open the Add Scheduling Information dialog (see “Adding Schedule Information” on page 86) and “Managing Schedule Rules for an Application Collection” on page 88
   
b. Click the New Rule button and type in a rule name
   
c. Click the Set Rule button, an error message is prompted
   
d. Check the All Day box

   Suggested solution is to close the Scheduling Information dialog and open it again.
8. Using the `wmargetapp` command or the desktop GUI Profile Manager, a maximum number of 1000 applications can be shown. The applications are not listed in the Add Entry to the Profile. This means using the GUI, it is not possible to start a collection for all the applications registered in product repository.

9. Concurrent running probes time-out or are not run. This is probably because they are executed sequentially and one of them or both, are timing out. A solution is to decrease the exec frequency or decrease the time out setting.

**Messages**

This section explains the messages issued by the server, client and gateway when you are using the product.

Messages are listed in ascending numeric order.

**Identifying a Message**

The product 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.

<table>
<thead>
<tr>
<th>Identity</th>
<th>Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>APF0005Z</td>
<td>Usage: wmarlsapp endpoint_name</td>
</tr>
</tbody>
</table>

**APF**  This prefix identifies the message as belonging to the product.

**0005**  The unique serial number of the message.

**Z**  Represents the type of message and may 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 a user or system error has occurred. 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 with the changeable string identity displayed in italics at the appropriate part of the message.

**System Messages**

Sometimes a message without an identifier is returned when you are using the product. These messages are generated by the Tivoli management software, for example Tivoli management framework or desktop or some other system component. Such messages are normally only for user information and advice and are not detailed in this document.

**User Messages**

The messages issued by the product graphical user interface (GUI), or command line interface (CLI) instructions from the Tivoli server, are displayed at the server. For further information, check the following log files:
Messages

- `tapm.log` on the server
- `tapm_engine.log` on the endpoint
- `tapm_ep.log` on the endpoint

**APF001I**

Usage: `wmarstarteng endpoint_name`

Explanation: This is a command syntax prompt that is displayed when a CLI command is incomplete or is incorrect.

User Response: Verify the command against the given syntax and retype the command.

**APF002I**

Usage: `wmarstopeng endpoint_name`

Explanation: This is a command syntax prompt that is displayed when a CLI command is incomplete or is incorrect.

User Response: Verify the command against the given syntax and retype the command.

**APF003I**

Usage: `wmarcleareng endpoint_name`

Explanation: This is a command syntax prompt that is displayed when a CLI command is incomplete or is incorrect.

User Response: Verify the command against the given syntax and retype the command.

**APF004I**

Usage: `wmarlseng endpoint_name`

Explanation: This is a command syntax prompt that is displayed when a CLI command is incomplete or is incorrect.

User Response: Verify the command against the given syntax and reenter the command.

**APF005I**

Usage: `wmarlsapp [-a app_name] [-t tx_name] [-u user_id] endpoint_name`

Explanation: This is a command syntax prompt that is displayed when a CLI command is incomplete or is incorrect.

User Response: Verify the command against the given syntax and reenter the command.

**APF006I**


Explanation: This is a command syntax prompt that is displayed when a CLI command is incomplete or is incorrect.

User Response: Verify the command against the given syntax and reenter the command.

**APF007I**

Usage: `wmarstopcoll -a app_name endpoint_name`

Explanation: This is a command syntax prompt that is displayed when a CLI command is incomplete or is incorrect.

User Response: Verify the command against the given syntax and reenter the command.
APF0008I Usage: `wmargetdata -a app_name [-t tx_name] [-u user_id] [-i interval] [-s start_time [-e end_time]] [-m metric_name] endpoint_name`

Explanation: This is a command syntax prompt that is displayed when a CLI command is incomplete or is incorrect.

User Response: Verify the command against the given syntax and reenter the command.

APF0009I Usage: `wmargetstatus [-g] host_name`

where:
host_name is either an endpoint containing a Tivoli Application Performance Management engine or, if the -g option is specified, a managed node containing a Tivoli Application Performance Management mid-level manager

Explanation: This is a command syntax prompt that is displayed when a CLI command is incomplete or is incorrect.

User Response: Verify the command against the given syntax and reenter the command.

APF0010E `command_name: You must supply an argument -argument`

Explanation: You did not supply a mandatory argument with the command you have just issued at the command line.

System Action: The command is not performed.

User Response: Check the message usage information that accompanied this warning message for correct syntax and retry the command using the required argument.

APF0011E `command_name: There are extraneous arguments`

Explanation: You included an argument that is not compatible with the command you have just issued at the command line.

System Action: The command is not performed.

User Response: Check the message usage information that accompanied this warning message for correct syntax and retry the command using the required argument.

APF0012E Unrecognized command `command_name`

Explanation: The command just issued at the command line has not been recognized by the system. It may just have been mistyped.

System Action: None.

User Response: Use a command from the list of valid commands given in Appendix B of this guide.

APF0013E `command_name: You must supply an endpoint name argument`

Explanation: The endpoint where the last command was to have been run, was not identified.

System Action: The command is not performed.

User Response: Retype the command including the missing `endpoint_name`.
<table>
<thead>
<tr>
<th>Code</th>
<th>Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>APF0014E</td>
<td><strong>The command command_name failed</strong></td>
</tr>
<tr>
<td></td>
<td><strong>Explanation:</strong> The command just issued, did not complete properly.</td>
</tr>
<tr>
<td></td>
<td><strong>System Action:</strong> The command is not performed.</td>
</tr>
<tr>
<td></td>
<td><strong>User Response:</strong> Try the command again and if the problem persists contact your system programmer or Tivoli representative.</td>
</tr>
<tr>
<td>APF0015I</td>
<td><strong>The engine is already running</strong></td>
</tr>
<tr>
<td></td>
<td><strong>Explanation:</strong> A start engine command was issued to an endpoint where the engine is already running. An incorrect endpoint_name may have been used or a start command may have been issued by mistake for a stop command.</td>
</tr>
<tr>
<td></td>
<td><strong>User Response:</strong> Verify the endpoint identity and retry the command.</td>
</tr>
<tr>
<td>APF0016I</td>
<td><strong>The command command_name was successfully executed</strong></td>
</tr>
<tr>
<td></td>
<td><strong>Explanation:</strong> The command has been completed and the system is waiting for a fresh command.</td>
</tr>
<tr>
<td></td>
<td><strong>User Response:</strong> None.</td>
</tr>
<tr>
<td>APF0017E</td>
<td><strong>The engine is not running</strong></td>
</tr>
<tr>
<td></td>
<td><strong>Explanation:</strong> A command has been issued to an endpoint where the engine is not running.</td>
</tr>
<tr>
<td></td>
<td><strong>System Action:</strong> The command is not performed.</td>
</tr>
<tr>
<td></td>
<td><strong>User Response:</strong> Verify you specified the correct endpoint identity. Use wmarstarteng command to start the engine.</td>
</tr>
<tr>
<td>APF0018E</td>
<td><strong>Incorrect parameter specified.</strong></td>
</tr>
<tr>
<td></td>
<td><strong>Explanation:</strong> The command just issued contained an invalid parameter or a parameter containing a value that is outside limits.</td>
</tr>
<tr>
<td></td>
<td><strong>System Action:</strong> The command is not performed.</td>
</tr>
<tr>
<td></td>
<td><strong>User Response:</strong> Verify the command against the given syntax and Retype the command.</td>
</tr>
<tr>
<td>APF0019E</td>
<td><strong>command_name Collection not active</strong></td>
</tr>
<tr>
<td></td>
<td><strong>Explanation:</strong> A wmarstopcoll command tried to stop a collection that was not active.</td>
</tr>
<tr>
<td></td>
<td><strong>System Action:</strong> The command is not performed.</td>
</tr>
<tr>
<td></td>
<td><strong>User Response:</strong> Verify the application-name (-a value) is correct. You can use the wmarlseng command to check which collections are specified.</td>
</tr>
<tr>
<td>APF0020E</td>
<td><strong>Incorrect parameter specified: application name too long</strong></td>
</tr>
<tr>
<td></td>
<td><strong>Explanation:</strong> An incorrect application name was specified.</td>
</tr>
<tr>
<td></td>
<td><strong>System Action:</strong> The command is not performed.</td>
</tr>
<tr>
<td></td>
<td><strong>User Response:</strong> Verify that you are not exceeding the 127-byte limit for the app_name string.</td>
</tr>
<tr>
<td>Code</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>APF0021E</td>
<td>Incorrect parameter specified: the number of bucket limits must be greater than zero and in increasing order.</td>
</tr>
<tr>
<td>APF0022E</td>
<td>Incorrect parameter specified: too many bucket boundaries</td>
</tr>
<tr>
<td>APF0023E</td>
<td>Incorrect parameter specified: invalid collection interval</td>
</tr>
<tr>
<td>APF0024E</td>
<td>Incorrect parameter specified: invalid time format</td>
</tr>
<tr>
<td>APF0025E</td>
<td>Incorrect parameter specified: invalid detail level</td>
</tr>
<tr>
<td>APF0026E</td>
<td>The application app_name is not registered to the application repository</td>
</tr>
</tbody>
</table>
The application app_name is a VuGen simulation please specify the run interval using the r flag.

Explanation: An attempt has been made to start a collection using the wmarstartcoll command. You must specify a value for the –r flag which controls the interval between simulation runs.

System Action: The command is not performed

User Response: Retype the command with a valid –r argument.

Incorrect parameter specified: invalid run interval

Explanation: A run interval specified for a simulation by the –r argument was invalid.

System Action: The command failed.

User Response: Retype the command supplying a valid –r argument. Valid values are greater than 60 seconds and less than one day (expressed in seconds).

The application app_name is a simulation that does not need parameters (script name: script_name).

Explanation: A collection was started using the wmarstartcoll command. The identified simulation included in the profile does not need to be accompanied by arguments.

System Action: The command failed.

User Response: Retype the command without the –p parameter.

The application app_name is a simulation (script name: script_name) that needs parameters.

Explanation: A collection was started using the wmarstartcoll command. The identified simulation included in the profile needs to be accompanied by arguments.

System Action: The command failed.

User Response: Retype the command supplying the required arguments using the –p flag.

The application app_name is a not a VuGen simulation: you should not specify –r flag.

Explanation: The command contained extraneous arguments.

System Action: The command failed.

User Response: Verify that the command you tried to execute has the correct arguments. Retype the command omitting those arguments not needed.

The number of parameters you specified for application app_name (script name: simulation script_name) does not match the number of required parameters.

Explanation: The –p argument contains several input parameters for the script. The correct number of parameters were not supplied.

System Action: The command failed.

User Response: Retype the command supplying the valid input parameters for the simulation script.

You have specified a stop date that is earlier than the current date.

Explanation: Incorrect stop date using the wmarstartcoll command. It must be sometime in the past.

System Action: The command failed.

User Response: Retype the command supplying a correct date.
<table>
<thead>
<tr>
<th>Message Code</th>
<th>Description</th>
<th>Explanation</th>
<th>System Action</th>
<th>User Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>APF0034E</td>
<td>You have specified a stop date that is earlier than the start date.</td>
<td>The <code>wmastartcoll</code> command contains an incorrect interval date.</td>
<td>The command failed.</td>
<td>Retype the command supplying the correct date.</td>
</tr>
<tr>
<td>APF0035E</td>
<td>There is no data available on the endpoint. Possible reasons are:</td>
<td></td>
<td>The command failed.</td>
<td>Verify that the <code>mardatayyyymmdd.log</code> file exists and verify the endpoint path(s) for binary data file location are correct and collection is active. The simple answer may be lack of disk space. If the failure persists, contact your system programmer or Tivoli representative.</td>
</tr>
<tr>
<td></td>
<td>a data collection is not active, or</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>a data collection is active but the related application is not running, or</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>a data collection is active but the first collection interval is not finished yet.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>APF0036E</td>
<td>The binary info file was not found on the endpoint.</td>
<td>A <code>wmargetdata</code> command has been issued, or a Distributed Monitoring monitor has activated. The product has tried to fetch an information data file at the endpoint and could not find it. This may be because no collection has been made.</td>
<td>The command failed.</td>
<td>Verify that the <code>marinfoyyymmd.log</code> file exists and verify the endpoint path(s) for information data file location are correct. The simple answer may be lack of disk space. If the failure persists, contact your system programmer or Tivoli representative.</td>
</tr>
<tr>
<td>APF0037E</td>
<td>An error occurred while reading data.</td>
<td>Possible network or modem failure.</td>
<td>The command failed.</td>
<td>Retype the command. If the failure persists, contact your system programmer or Tivoli representative.</td>
</tr>
<tr>
<td>APF0038I</td>
<td>There are no collections on the specified endpoint.</td>
<td>The command <code>wmarlsapp</code> has been issued but no collection has been started on the specified endpoint.</td>
<td>The command failed.</td>
<td></td>
</tr>
<tr>
<td>APF0039E</td>
<td>An error occurred while retrieving data from the aggregated file on the endpoint.</td>
<td>The product has tried unsuccessfully to fetch the aggregated file that was to be sent to the gateway from the endpoint. There may be a network communication problem.</td>
<td>The command failed.</td>
<td>Verify the endpoint path(s) for information data file location are correct. The simple answer may be lack of disk space. If the failure persists, contact your system programmer or Tivoli representative.</td>
</tr>
<tr>
<td>Code</td>
<td>Message</td>
<td>Explanation</td>
<td>System Action</td>
<td>User Response</td>
</tr>
<tr>
<td>----------</td>
<td>-------------------------------------------------------------------------</td>
<td>----------------------------------------------------------------------------</td>
<td>------------------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>APF0040E</td>
<td>The endpoint did not establish a connection back to the gateway.</td>
<td>The endpoint has tried to open a connection circuit to send a data stream of data to its gateway.</td>
<td>The command failed.</td>
<td>Verify that there are no network problems. If the problem persists contact your system programmer or Tivoli representative.</td>
</tr>
<tr>
<td>APF0041I</td>
<td>Aggregated data successfully loaded on database.</td>
<td>The mid-level manager at the gateway reports that data files have been stored.</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td>APF0042E</td>
<td>Unable to find the RIM object for Tivoli Application Performance Management.</td>
<td>The RDBMS Interface Module (RIM) object for The product has not been found.</td>
<td>The command failed.</td>
<td>Check that the specification of the RIM object is correct by typing the <code>wgetrim</code> command and/or run the RIM creation script again on the gateway.</td>
</tr>
<tr>
<td>APF0043E</td>
<td>Unable to connect to the database.</td>
<td>The RDBMS Interface Module (RIM) failed to connect to the database.</td>
<td>The command failed.</td>
<td>Check that there are no network problems and/or try to run the client SQL interface in order to check the client/server database connection.</td>
</tr>
<tr>
<td>APF0044E</td>
<td>Error inserting row in the database.</td>
<td>The RDBMS Interface Module (RIM) failed to insert a row in the database.</td>
<td>The command failed.</td>
<td>Verify the product database tables have been created with the right access. If the problem persists contact your system programmer or Tivoli representative.</td>
</tr>
<tr>
<td>APF0045E</td>
<td>Error updating row in the database.</td>
<td>The RDBMS Interface Module (RIM) failed to update a row in the database.</td>
<td>The command failed.</td>
<td>Verify the product database tables have been created with the correct access. If the problem persists contact your system programmer or Tivoli representative.</td>
</tr>
<tr>
<td>APF0046W</td>
<td>No rows found in the database.</td>
<td>The RDBMS Interface Module (RIM) failed to retrieve some rows in the database.</td>
<td>The command failed.</td>
<td>Verify the product database tables have been created with the correct access. If the problem persists contact your system programmer or Tivoli representative.</td>
</tr>
<tr>
<td>Message ID</td>
<td>Description</td>
<td>Explanation</td>
<td>System Action</td>
<td>User Response</td>
</tr>
<tr>
<td>------------</td>
<td>-------------</td>
<td>-------------</td>
<td>---------------</td>
<td>---------------</td>
</tr>
<tr>
<td>APF0047E</td>
<td>Error retrieving rows in the database.</td>
<td>The RDBMS Interface Module (RIM) failed to retrieve some rows in the database.</td>
<td>The command failed.</td>
<td>Verify the product database tables have been created with the correct access. If the problem persists contact your system programmer or Tivoli representative.</td>
</tr>
<tr>
<td>APF0048E</td>
<td>Retrieved too many entries from Database.</td>
<td>The RDBMS Interface Module (RIM) retrieved more rows than expected from database.</td>
<td>The command failed.</td>
<td>Verify the product database tables have not been corrupted. Check that there are no entries with the same ID. If the problem persists contact your system programmer or Tivoli representative.</td>
</tr>
<tr>
<td>APF0049E</td>
<td>Data from daily aggregated file has NOT been loaded on database.</td>
<td>The process of data loading on the database has failed.</td>
<td>The command failed.</td>
<td>Verify the previous message that explains the details of the failure. If the problem persists contact your system programmer or Tivoli representative.</td>
</tr>
<tr>
<td>APF0050I</td>
<td>Aggregation process successfully completed</td>
<td>The engine invoked the aggregator and generated a new data file.</td>
<td>None</td>
<td></td>
</tr>
<tr>
<td>APF0051E</td>
<td>Aggregation process failed</td>
<td>The engine invoked the aggregator but it could not generate the aggregated file.</td>
<td>The command failed.</td>
<td>Contact your Tivoli representative.</td>
</tr>
<tr>
<td>APF0052E</td>
<td>Wrong binary files format</td>
<td>The binary files generated by the engine are not in the correct format and cannot be read by the aggregator or by using the <code>wmagetdata</code> command.</td>
<td>The command failed.</td>
<td>Contact your Tivoli representative.</td>
</tr>
<tr>
<td>APF0054E</td>
<td>Transaction Filter too long: transaction filter can only be up to 127 characters</td>
<td>Transaction filter is too long.</td>
<td>The command failed.</td>
<td>Retype the CLI command with a string containing fewer than 128 characters.</td>
</tr>
</tbody>
</table>
APF0055E  *string*: These characters are not allowed in the string parameter

**Explanation:** The engine received a non-ASCII parameter.

**System Action:** The command is not performed.

**User Response:** Retype the CLI command with only ASCII characters in the string.

APF0056E  Incorrect parameter specified: invalid aggregation interval

**Explanation:** An invalid aggregation interval has been specified.

**System Action:** The command failed.

**User Response:** Retype the CLI command using correct values.

APF0057E  Incorrect parameter specified: invalid filter values

**Explanation:** A `-t` flag has been specified incorrectly in the `wmarrstartcoll` command.

**System Action:** The command failed.

**User Response:** Retype the CLI command using correct filter values with (`-t`).

APF0058E  Incorrect parameter specified: invalid schedule values

**Explanation:** An attempt has been made to start a collection using the `wmarrstartcoll` command with one or more invalid values for the `-c` flag.

**System Action:** The command failed.

**User Response:** Retype the CLI command with correct schedule values.

APF0059E  Incorrect parameter specified: invalid metric name

**Explanation:** An unknown metric has been called with the `-m` flag using the `wmargetdata` command.

**System Action:** The command failed.

**User Response:** Select a metric from the valid list and repeat the command. If the problem persists contact your system programmer or Tivoli representative.

APF0060E  Flag `-i` cannot be used with flags `-s` and `-e`

**Explanation:** An interval with the `wmargetdata` command can use the `-i` flag or `-s` and `-e`, but not both options.

**System Action:** The command failed.

**User Response:** Retype the `wmargetdata` command use the correct flag option.

APF0061E  Flag `-e` cannot be used without flag `-s`

**Explanation:** A `wmargetdata` command was issued but the end time flag `-e` for collection interval was not accompanied with a valid start time.

**System Action:** The command failed.

**User Response:** Retype the `wmargetdata` command using the correct flag option.
<table>
<thead>
<tr>
<th>Code</th>
<th>Message</th>
<th>Explanation</th>
<th>System Action</th>
<th>User Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>APF0062E</td>
<td>Error in retrieving environment variables</td>
<td>A software problem has occurred.</td>
<td>The command failed</td>
<td>Contact your Tivoli representative.</td>
</tr>
<tr>
<td>APF0063E</td>
<td>Error in updating environment variables</td>
<td>A software problem has occurred.</td>
<td>The command failed</td>
<td>Contact your Tivoli representative.</td>
</tr>
<tr>
<td>APF0064E</td>
<td>Incorrect parameter specified: invalid number of intervals</td>
<td>A <code>wmargetdata</code> command was issued with an incorrect value for the <code>–i</code> flag.</td>
<td>The command failed</td>
<td>Retype the CLI command with correct intervals (&gt; 0) specified.</td>
</tr>
<tr>
<td>APF0065E</td>
<td>Invalid interval specified: end time is older than start time</td>
<td>A collection command specified incompatible start and stop times.</td>
<td>The command failed</td>
<td>Retype the CLI command making sure the time specified by the <code>–e</code> flag is later than that of the <code>–s</code> flag.</td>
</tr>
<tr>
<td>APF0066E</td>
<td>Invalid interval specified: start time is newer than current time</td>
<td>A collection command specified a start time in the future.</td>
<td>The command failed</td>
<td>Retype the CLI command making sure the time specified by the <code>–s</code> flag is before now.</td>
</tr>
<tr>
<td>APF0067E</td>
<td>Incorrect parameter specified: invalid script name</td>
<td>The endpoint is attempting to run a script it does not have.</td>
<td>The command failed</td>
<td>Verify the profile contains the script you want to run. If the problem persists contact your Tivoli representative.</td>
</tr>
<tr>
<td>APF0068E</td>
<td>Error during file decompression</td>
<td>The engine failed to start a script.</td>
<td>The command failed</td>
<td>Contact your Tivoli representative.</td>
</tr>
<tr>
<td>Code</td>
<td>Message</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>--------</td>
<td>---------------------------------------------------------------------------------------------</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>APF0069E</td>
<td>The scheduling information specified by the --c flag is incorrect.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Explanation:</strong> An application collection schedule has been specified incorrectly.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>System Action:</strong> The command failed.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>User Response:</strong> Retype the CLI command with correct schedule values.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>APF0070E</td>
<td>The endpoint is not equipped to run simulations. Install the simulation agent and try again.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Explanation:</strong> LoadRunner is not installed on the endpoint where you want to run your simulation script.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>System Action:</strong> The command failed.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>User Response:</strong> Install the required Mercury toolkit at that endpoint.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>APF0071E</td>
<td>There are no applications matching the specified criteria.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Explanation:</strong> A <code>wmarsapp</code> command has been issued but the product reports that a specified application has not run for that engine</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>System Action:</strong> The command failed.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>User Response:</strong> Verify the correct application name is being used for that endpoint.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>APF0072E</td>
<td>Incorrect parameter specified: invalid script parameter name, <code>parameter_name</code>.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Explanation:</strong> A script name that is invalid has been specified.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>System Action:</strong> The command failed.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>User Response:</strong> Retype the CLI command with correct script name and path.</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Usage: wmarsetstatus [-g] PROPERTY=value, PROPERTY=value
......, PROPERTY=VALUE host_name
Where: host_name is either an endpoint containing a Tivoli Application Performance Management engine or, (if the -g option is specified), a managed node (gateway) containing a Tivoli Application Performance Management mid-level manager.

Possible properties that can be set on the engine are:
LogLevel (range 0 - 3)
LogComponents (NONE, ARMSERVER, COMMAND COLLECTION, AGGREGATOR, ALL_ENGINE), FileMaxSize (range 100 - 100000000)
GenerateCorrelator (true, false)
AggregationEndTime (HH:MM, where 00:00 <= HH:MM <= 23:59),
GwUploadRandTimeRange (HH:MM, where 00:00 <= HH:MM <= 23:59),
CleanupStartTime (HH:MM, where 00:00 <= HH:MM <= 23:59),
CleanupApplicationTimeout (HH:MM, where 00:05 <= HH:MM <= 672:00)
CleanupTransactionTimeout (HH:MM, where 00:01 <= HH:MM <= 24:00), (4 weeks),
SummarizationOn (true, false)
MaxSumaryDataFiles (range 2 - 30),
AggregationStartTime (HH:MM, where 00:00 <= HH:MM <= 23:59),
CleanupStartTime (HH:MM, where 00:00 <= HH:MM <= 23:59),
CleanupApplicationTimeout (HH:MM, where 00:05 <= HH:MM <= 672:00 i.e. 4 weeks)
AggregationEnabled (true, false)
AggregationTimeStep (range 1 - 24)
AggregationStartTime (HH:MM, where 00:00 <= HH:MM <= 23:59)
AggregationEndTime (HH:MM, where 00:00 <= HH:MM <= 23:59)
AggregationRandTimeRange (HH:MM, where 00:00 <= HH:MM <= 23:59)
InstanceLogging (true, false)
MaxTransInstanceFiles (range 2 - 30)
TransInstanceFilesMaxSize (range 100 - 100000000)
InstanceEnabled (true, false)
InstanceTimeStep (range 1 - 24)
InstanceStartTime (HH:MM, where 00:00 <= HH:MM <= 23:59)
InstanceEndTime (HH:MM, where 00:00 <= HH:MM <= 23:59)
InstanceRandTimeRange (HH:MM, where 00:00 <= HH:MM <= 23:59)

Possible properties that can be set on the mid-level manager are:
LogLevel (range 0 - 3),
DBUploadStartTime (HH:MM, where 00:00 <= HH:MM <= 23:59)
DBUploadStopTime (HH:MM, where 00:00 <= HH:MM <= 23:59)

Explanation: This is a command syntax prompt that is displayed when a CLI command is incomplete or is incorrect.

User Response: Verify the command against the given syntax and retype the command.

wmarsestatus: You must supply at least one PROPERTY=value pair.

Explanation: A wmarsetstatus command was issued without any property values.

System Action: The command failed.

User Response: Resend the command with correction.
<table>
<thead>
<tr>
<th>Message Code</th>
<th>Issue Description</th>
<th>Explanation</th>
<th>System Action</th>
<th>User Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>APF0075E</td>
<td>Invalid parameter name specified.</td>
<td>A <code>wmarsstatus</code> command was issued with an incorrect parameter or parameters.</td>
<td>The command failed.</td>
<td>Resend the command with correction.</td>
</tr>
<tr>
<td>APF0076E</td>
<td>Tivoli Application Performance Management log file could not be opened.</td>
<td>The product interface on the Endpoint did not create its log file. The log file was not created and no information about the commands sent to the the product engine will be written on the Endpoint.</td>
<td>The command failed.</td>
<td>Verify that you have sufficient disk space available at that endpoint and retry the <code>wmarsstarteng</code> command. If the problem persists, contact your Tivoli representative.</td>
</tr>
<tr>
<td>APF0077E</td>
<td>Invalid parameter value specified</td>
<td>A <code>wmarsstatus</code> command was issued with an incorrect parameter value.</td>
<td>The command failed.</td>
<td>Resend the command with correction.</td>
</tr>
<tr>
<td>APF0078E</td>
<td>Missing <code>=</code> between parameter and value</td>
<td>A <code>wmarsstatus</code> command was issued with parameter value that is not separated from the parameter by an equals ((=)) character.</td>
<td>The command failed.</td>
<td>The correct format is PROPERTY=&lt;value&gt;. Resend the command with correction.</td>
</tr>
<tr>
<td>APF0079E</td>
<td>Too many parameters. Maximum allowed = 100</td>
<td>A <code>wmarsstatus</code> command was issued with too many parameters.</td>
<td>The command failed.</td>
<td>Resend the command with correction.</td>
</tr>
<tr>
<td>APF0080E</td>
<td>The command is too long</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>APF0081E</td>
<td>The aggregated file <code>file_name</code> on the endpoint was not found.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Code</td>
<td>Message</td>
<td>Explanation</td>
<td>System Action</td>
<td>User Response</td>
</tr>
<tr>
<td>----------</td>
<td>-------------------------------------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>---------------------</td>
<td>---------------------------------------------</td>
</tr>
<tr>
<td>APF0082E</td>
<td>Transaction not found.</td>
<td>A <code>wmarlsapp</code> command was issued, but the transaction specified with the <code>-t</code> option does not exist.</td>
<td>The command failed.</td>
<td>To verify which transactions are valid, you can use the same command specifying only <code>-a</code> and <code>-u</code> option. In this way you will see all the transactions for a certain application and a certain user.</td>
</tr>
<tr>
<td>APF0083I</td>
<td>Send file process successfully completed</td>
<td>A <code>wmarsenddata</code> command was issued to force an aggregation.</td>
<td>The command executed correctly.</td>
<td></td>
</tr>
<tr>
<td>APF0084E</td>
<td>Send file process failed.</td>
<td>A <code>wmarsenddata</code> command was issued to force an aggregation but failed.</td>
<td>The command failed.</td>
<td>Contact your Tivoli representative.</td>
</tr>
<tr>
<td>APF0085E</td>
<td>The endpoint is not equipped to run WinRunner scripts. Install the RoboticClient and WinRunner and try again.</td>
<td>A command has been issued and although notification of execution (APF0016I) is generated, the system has so far only checked syntax and IDs to make sure the command is valid. It may not have finished all actions.</td>
<td>Wait a short while before issuing a new command.</td>
<td></td>
</tr>
<tr>
<td>APF0086I</td>
<td>Usage: <code>wmarsenddata [-i] [-d YYYY-MM-DD] [-s HH] [-e HH] endpoint_name.</code></td>
<td>This is a command syntax prompt that is displayed when a CLI command is incomplete or is incorrect.</td>
<td>Retype the command correctly.</td>
<td></td>
</tr>
<tr>
<td>APF0087E</td>
<td>Upcall process failed: <code>&lt;message&gt;</code>.</td>
<td>The endpoint failed to upload aggregated data to the gateway. There could be a network communication problem. Also refer to the message generated by the Tivoli Management Framework application.</td>
<td>The command failed.</td>
<td>Refer to the message generated by the Tivoli Management Framework application. Try to run the command entering <code>wgateway &lt;gateway_name&gt; restart</code>.</td>
</tr>
<tr>
<td>APF0088E</td>
<td>One of the properties has been specified twice.</td>
<td>A <code>wmarsetstatus</code> command was issued in which one of the properties is repeated.</td>
<td>The command failed.</td>
<td>Resend the command string with only one occurrence of that property.</td>
</tr>
<tr>
<td>Code</td>
<td>Message</td>
<td>Explanation</td>
<td>System Action</td>
<td>User Response</td>
</tr>
<tr>
<td>---------</td>
<td>---------------------------------------------------------------------------------------------</td>
<td>------------------------------------------------------------------------------</td>
<td>-------------------------------</td>
<td>--------------------------------------</td>
</tr>
<tr>
<td>APF0089E</td>
<td>One of the properties values has been specified twice.</td>
<td>A <code>wmarsetstatus</code> command was issued containing repeated values for one of the properties.</td>
<td>The command is not executed.</td>
<td>Resend the command with the correct single value.</td>
</tr>
<tr>
<td></td>
<td><strong>Explanation:</strong> A <code>wmarsetstatus</code> command was issued containing repeated values for one of the properties.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>System Action:</strong> The command is not executed.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>User Response:</strong> Resend the command with the correct single value.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>APF0090E</td>
<td>One of the properties contains conflicting values (example ALL, NONE).</td>
<td>A <code>wmarsetstatus</code> command was issued containing two conflicting values for one of the properties.</td>
<td>The command failed.</td>
<td>Resend the command with the correct single value.</td>
</tr>
<tr>
<td></td>
<td><strong>Explanation:</strong> A <code>wmarsetstatus</code> command was issued containing two conflicting values for one of the properties.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>System Action:</strong> The command failed.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>User Response:</strong> Resend the command with the correct single value.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>APF0091E</td>
<td>One of the string properties contains an invalid value.</td>
<td>A <code>wmarsetstatus</code> command was issued with an invalid property value.</td>
<td>The command failed.</td>
<td>Resend the command with correction.</td>
</tr>
<tr>
<td></td>
<td><strong>Explanation:</strong> A <code>wmarsetstatus</code> command was issued with an invalid property value.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>System Action:</strong> The command failed.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>User Response:</strong> Resend the command with correction.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>APF0092E</td>
<td>One of the properties that contains multiple values has wrong values</td>
<td>A <code>wmarsetstatus</code> command was issued with a property that takes multiple values, but a value is incorrect.</td>
<td>The command failed.</td>
<td>Resend the command with correction.</td>
</tr>
<tr>
<td></td>
<td><strong>Explanation:</strong> A <code>wmarsetstatus</code> command was issued with a property that takes multiple values, but a value is incorrect.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>System Action:</strong> The command failed.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>User Response:</strong> Resend the command with correction.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>APF0093E</td>
<td>One of the numeric properties contains a non numeric value.</td>
<td>A <code>wmarsetstatus</code> command has been issued with a corrupt numeric value.</td>
<td>The command failed.</td>
<td>Resend the command with correction.</td>
</tr>
<tr>
<td></td>
<td><strong>Explanation:</strong> A <code>wmarsetstatus</code> command has been issued with a corrupt numeric value.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>System Action:</strong> The command failed.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>User Response:</strong> Resend the command with correction.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>APF0094E</td>
<td>One of the numeric properties contains a value outside of the possible boundary.</td>
<td>A <code>wmarsetstatus</code> command was issued with a numeric value that is too large.</td>
<td>The command failed.</td>
<td>Verify the limit for the specified property and resend the command with correction.</td>
</tr>
<tr>
<td></td>
<td><strong>Explanation:</strong> A <code>wmarsetstatus</code> command was issued with a numeric value that is too large.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>System Action:</strong> The command failed.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>User Response:</strong> Verify the limit for the specified property and resend the command with correction.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>APF0095E</td>
<td>Invalid time format. Correct format is hh:mm.</td>
<td>A <code>wmarsetstatus</code> command was issued containing a time that the product does not understand.</td>
<td>The command failed.</td>
<td>Resend the command with correction.</td>
</tr>
<tr>
<td></td>
<td><strong>Explanation:</strong> A <code>wmarsetstatus</code> command was issued containing a time that the product does not understand.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>System Action:</strong> The command failed.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>User Response:</strong> Resend the command with correction.</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
APF0096E The command has timed out

**Explanation:** A command was issued but after an interval, the product did not respond to it. There may be a network communication problem.

**System Action:** The command failed.

**User Response:** Resend the command. If the problem persists, contact your system administrator.

APF0097I The engine is running.

**Explanation:** A `wmargetstatus` command was issued. The product has reported back that the engine is active.

**User Response:** None

APF0098I You must supply a host_name argument

**Explanation:** A `wmargetstatus` command was issued without specifying an endpoint or mid-level manager.

**User Response:** Retype the command and supply the endpoint or gateway name from where you want an engine status report.

APF0099I There is no data because no application has run on the endpoint

**Explanation:** A `wmarlsapp` command was issued and the product reported back that no applications matching those specified has run since the engine was started.

**User Response:** None

APF0100I One of the time properties contains a value outside of the possible boundary.

**Explanation:** A `wmarsetstatus` command was issued with an invalid time value.

**System Action:** The command failed.

**User Response:** Resend the command with correction. If the problem persists, contact your system administrator

APF0101W You must specify a name for this profile

**Explanation:** The profile name was omitted.

**User Response:** Resend the command and supply the profile name.

APF0102E Could not create the Tivoli Application Performance Management profile database.

**Explanation:** A failure occurred trying to create the profile partition in the server. The server database may be locked or otherwise in a state that prevents it being written to.

**System Action:** The creation of an instance of a product profile and the associated profile database is terminated.

**User Response:** Contact your system programmer or Tivoli representative.
<table>
<thead>
<tr>
<th>Code</th>
<th>Message</th>
<th>Explanation</th>
<th>System Action</th>
<th>User Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>APF0103E</td>
<td>The instance @MarProfile:TivoliDefaultMarProfile is missing. This instance should have been created at application install time. Use wcrtpf to create an instance of this profile in the context of @ProfileManager:TivoliDefaultMarProfileMgr</td>
<td>The profile TivoliDefaultMarProfile is missing. This is an indication that installation of the Tivoli Application Performance Management software product was not completed correctly.</td>
<td>The creation of an instance of the MarProfile is terminated.</td>
<td>Manually create the missing profile or contact your system programmer or Tivoli representative.</td>
</tr>
<tr>
<td>APF0104E</td>
<td>An internal error has occurred within the Tivoli Application Performance Management product. Please contact your support representative for assistance.</td>
<td>An unknown error has occurred in the product.</td>
<td>The GUI is terminated.</td>
<td>Contact your Tivoli representative.</td>
</tr>
<tr>
<td>APF0105W</td>
<td>An entry must have at least one application name allocated.</td>
<td>An application name was not supplied.</td>
<td>The addition of an entry into your product profile is terminated.</td>
<td>Retype the data including an application name.</td>
</tr>
<tr>
<td>APF0106E</td>
<td>The following records could not be moved or copied: name: error msg.</td>
<td>An attempt was made to move or copy some entries from one product profile to another.</td>
<td>The entries identified by “name” are not moved or copied. The reason is given by the displayed error message.</td>
<td>Verify that the destination profile does not already contain the entries and try the move or copy again.</td>
</tr>
<tr>
<td>APF0107E</td>
<td>Could not finish moving the selected records.</td>
<td>An internal error occurred while moving entries from one profile to another.</td>
<td>The move is terminated.</td>
<td>Contact your system programmer or Tivoli representative.</td>
</tr>
<tr>
<td>APF0108E</td>
<td>Could not locate the validation policy for this profile.</td>
<td>The validation policy entry is missing from the profile database.</td>
<td>The current operation is terminated.</td>
<td>Contact your system programmer or Tivoli representative.</td>
</tr>
<tr>
<td>APF0109E</td>
<td>Could not locate the default policy for this profile.</td>
<td>The default policy entry is missing from the profile database.</td>
<td>The current operation is terminated.</td>
<td>Contact your system programmer or Tivoli representative.</td>
</tr>
</tbody>
</table>
APF0110I Removed the following entries from the profile_name profile entry name, entry name, etc.
Explanation: The listed entries were deleted from the named profile.
System Action: The profile entries are removed from the database.
User Response: None.

APF0111W An entry already exists for the specified app_name.
Explanation: An attempt was made to add an application that is already present in this profile.
System Action: The entry is not added into the profile.
User Response: Remove the existing application from the profile or verify that the correct application was chosen for addition.

APF0112I Added an entry for application app_name to the profile_name profile.
Explanation: The application “application_name” was successfully added to profile “profile_name”.
System Action: An entry is added to the profile database.
User Response: None.

APF0113E Could not add app_name to the profile for the following reason:
Explanation: The application entry that was selected could not be added to the profile for the given reason.
System Action: The entry is not added into the profile database.
User Response: Correct the problem described by “reason” and try again, or contact your system programmer or Tivoli representative if the problem persists.

APF0114E You cannot retrieve policy records through this method.
Explanation: This is an internal error involving the policy entry in the profile database.
System Action: The current action is terminated.
User Response: Contact your system programmer or Tivoli representative.

APF0115E There is no entry for the specified application_name: app_name
Explanation: This is an internal error involving the profile database.
System Action: The current action is terminated.
User Response: Contact your system programmer or Tivoli representative.

APF0116E An internal application error has occurred. A method passed an invalid iterator ID to the profile iterator.
The invalid iterator was: reference_id.
Explanation: This is an internal error involving a profile iterator.
System Action: The current action is terminated.
User Response: Contact your system programmer or Tivoli representative.
APF0117E  Could not finish adding the requested entries because a communication error occurred.

Explanation: An attempt was made to add one or more entries into the profile database but the attempt failed because of a communication failure.

System Action: The current action is terminated.

User Response: Contact your system programmer or Tivoli representative.

APF0118I  Added the following entries to the profile_name profile:

Explanation: The system confirms that the additional entries are now part of the identified profile.

User Response: None.

APF0119W  You cannot edit an entry’s application name. You must delete the entry and recreate it with the new fields.

Explanation: This is an internal error that should not occur under normal circumstances. You may have tried to alter an existing application_name. All fields of an entry in the profile database are editable except for the application name.

System Action: The current operation is terminated.

User Response: Contact your system programmer or Tivoli representative.

APF0120W  You cannot edit the specified entry in a subscriber context. It is marked as locked in the original profile.

Explanation: An attempt was made to edit a read-only entry in the profile.

System Action: The current operation is terminated.

User Response: Contact your system programmer or Tivoli representative.

APF0121E  Could not save the app_name entry. There was an error writing to the CCMS database.

Explanation: An internal error occurred while writing the entry to the CCMS database.

System Action: The save operation is terminated.

User Response: Contact operation is terminated.

Contact your system programmer or Tivoli representative.

APF0122E  Could not distribute profile profile_name because of a fatal error.

Explanation: An internal error occurred while distributing the profile to the endpoint.

System Action: The push operation is terminated.

User Response: Contact your system programmer or Tivoli representative.

APF0123I  Removed the profile_name profile.

Explanation: The profile and all the entries it contained were successfully removed from the database.

System Action: Removes the profile and entries from the database.

User Response: None.
APF0124W First select the entry you want to remove.
Explanation: Delete was clicked on the GUI without first choosing an entry to delete.
System Action: None.
User Response: Choose the entry to delete and try again.

APF0125W You cannot edit this entry right now. This entry is currently being edited by another user.
Explanation: The entry cannot be modified because the file is locked while it is being edited by another administrator.
System Action: None.
User Response: Try again at a later time when the entry is not being edited by another administrator.

APF0126W Could not free edit lock on the current row because of the following error:
Explanation: An internal error occurred while removing a lock on a profile entry.
System Action: The current operation is terminated.
User Response: Contact your system programmer or Tivoli representative.

APF0127W You can only delete one entry at a time. Select the entry you want to delete and press Delete.
Explanation: More than one entry was selected when the “delete entry” button was clicked.
System Action: The delete operation is not performed.
User Response: Choose only one entry and try again.

APF0128W First select the entry you want to edit.
Explanation: “Edit Entry” was selected on the GUI without first choosing an entry.
System Action: None.
User Response: Choose an entry first then click the “Edit Entry” button.

APF0129W You can only edit one entry at a time.
Explanation: “Edit Entry” was selected on the GUI but more than one entry was selected.
System Action: None.
User Response: Verify only one entry is chosen then try again.

APF0130W Bad gadget definition type found.
Explanation: This is an internal error indicating that invalid information has been passed to the GUI from the application registry.
System Action: The VUS dialog of the GUI is closed.
User Response: Verify the application being edited has been properly registered into the application registry and try again. If the problem persists contact your system programmer or Tivoli representative.
<table>
<thead>
<tr>
<th>Message Code</th>
<th>Message Content</th>
</tr>
</thead>
<tbody>
<tr>
<td>APF0131W</td>
<td>A rule name must be specified.</td>
</tr>
<tr>
<td></td>
<td><strong>Explanation:</strong> No name was allocated to this new scheduling rule.</td>
</tr>
<tr>
<td></td>
<td><strong>System Action:</strong> None.</td>
</tr>
<tr>
<td></td>
<td><strong>User Response:</strong> Supply the missing information and retry the command.</td>
</tr>
<tr>
<td>APF0132W</td>
<td>This dialog is already open.</td>
</tr>
<tr>
<td></td>
<td><strong>Explanation:</strong> An attempt was made to open a dialog that is already opened.</td>
</tr>
<tr>
<td></td>
<td><strong>System Action:</strong> None.</td>
</tr>
<tr>
<td></td>
<td><strong>User Response:</strong> Locate the dialog. It may be minimized or hidden by another dialog.</td>
</tr>
<tr>
<td>APF0133W</td>
<td>Rule Name contains illegal characters.</td>
</tr>
<tr>
<td></td>
<td><strong>Explanation:</strong> The name used for a scheduling rule may not contain this character.</td>
</tr>
<tr>
<td></td>
<td><strong>System Action:</strong> None.</td>
</tr>
<tr>
<td></td>
<td><strong>User Response:</strong> Remove any non-alphanumeric character from the scheduling rule name and try again.</td>
</tr>
<tr>
<td>APF0136I</td>
<td>Usage: <code>wmregapp -a app_name [-l file_label] [-f script_full_path_name] [-d data_provider_name] [-w]</code></td>
</tr>
<tr>
<td></td>
<td><strong>Explanation:</strong> This is a command syntax prompt that is displayed when a CLI command is incomplete or is incorrect.</td>
</tr>
<tr>
<td></td>
<td><strong>User Response:</strong> Verify the command against the given syntax and retype the command. If the failure persists, contact your system programmer.</td>
</tr>
<tr>
<td>APF0137I</td>
<td>Usage: <code>wmargetapp [-a app_name]</code></td>
</tr>
<tr>
<td></td>
<td><strong>Explanation:</strong> This is a command syntax prompt that is displayed when a CLI command is incomplete or is incorrect.</td>
</tr>
<tr>
<td></td>
<td><strong>User Response:</strong> Verify the command against the given syntax and retype the command. If the failure persists, contact your system programmer.</td>
</tr>
<tr>
<td>APF0138W</td>
<td>The type code for entry <code>application_name</code> in the <code>registry_name</code> registry does not match TC_AppDef.</td>
</tr>
<tr>
<td></td>
<td><strong>Explanation:</strong></td>
</tr>
<tr>
<td></td>
<td><strong>System Action:</strong></td>
</tr>
<tr>
<td></td>
<td><strong>User Response:</strong></td>
</tr>
<tr>
<td>APF0139W</td>
<td>No applications were found in the <code>registry_name</code> registry.</td>
</tr>
<tr>
<td></td>
<td><strong>Explanation:</strong> The <code>wmargetapp</code> command could not find any registered applications. Applications must first be registered for the product. In other words the product must know what applications are available to it. Registration is part of the installation procedure.</td>
</tr>
<tr>
<td></td>
<td><strong>System Action:</strong> None.</td>
</tr>
<tr>
<td></td>
<td><strong>User Response:</strong> Register at least one instrumented application or prepared simulation script.</td>
</tr>
</tbody>
</table>
APF0140W  No applications were found in the `registry_name` registry.
Explanation: The “Add Application to a Profile” GUI could not find any registered applications to list. Applications must first be registered for the product. In other words the product must know what applications are available. Registration is part of the installation and customizing procedure.
System Action: The GUI “Add Entry” operation is terminated.
User Response: Register at least one instrumented application or prepared simulation script.

APF0141W  No definition was found for application `app_name` in the `registry_name` registry.
Explanation: The application had not been registered.
System Action: The current operation is terminated.
User Response: Register the instrumented application or prepared simulation script.

APF0142W  A bad type code was detected in the parameter list for application `app_name`.
Explanation: This is an internal error indicating that invalid information was received from the application registry.
System Action: The current operation is terminated.
User Response: Contact your system programmer or Tivoli representative.

APF0143E  The registration of application `app_name` was terminated. for the following reason:
Explanation: The application had not been registered.
System Action: The current operation is terminated.
User Response: Note the reason and correct it before attempting to register the application again. If the problem persists, contact your system programmer or Tivoli representative.

APF0144W  A schedule cycle name must be specified.
Explanation: You have attempted to add or close an incomplete schedule.
System Action: The current operation is terminated.
User Response: Verify that you have completed the rule name field and that rule timings have been correctly allocated. If the problem persists, contact your system programmer or Tivoli representative.

APF0145E  The command failed for the following reason:
Explanation: The command just issued, failed to complete properly.
System Action: The command is not performed.
User Response: Verify the reason and correct it. Try the command again and if the problem persists contact your system programmer or Tivoli representative.

APF0146W  The application name is invalid.
Explanation: An incorrect application name has been specified.
System Action: The command is not executed.
User Response: Verify that you are not exceeding the 127-byte limit for the `application-name` string.
<table>
<thead>
<tr>
<th>Message Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>APF0147W</td>
<td>The schedule for this entry is not correct. Ensure the stop date is after the start date and at least one scheduling rule is present.</td>
</tr>
<tr>
<td></td>
<td><strong>Explanation:</strong> The timing rules have not been specified correctly or perhaps you did not set at least one timing rule.</td>
</tr>
<tr>
<td></td>
<td><strong>System Action:</strong> The current operation is terminated. The schedule is not applied.</td>
</tr>
<tr>
<td></td>
<td><strong>User Response:</strong> Identify the faulty timing rule and correct it. If the problem persists contact your system programmer or Tivoli representative.</td>
</tr>
<tr>
<td>APF0148W</td>
<td>The schedule interval is invalid. It must be greater than zero.</td>
</tr>
<tr>
<td></td>
<td><strong>Explanation:</strong> A collection interval was not set or was incorrect.</td>
</tr>
<tr>
<td></td>
<td><strong>System Action:</strong> The current operation is terminated. The schedule is not applied.</td>
</tr>
<tr>
<td></td>
<td><strong>User Response:</strong> Set a collection interval or accept the 10 minute default value. If the problem persists contact your system programmer or Tivoli representative.</td>
</tr>
<tr>
<td>APF0149W</td>
<td>At least one bucket limit must be specified. A maximum of six limits may be specified in increasing order.</td>
</tr>
<tr>
<td></td>
<td><strong>Explanation:</strong> An invalid bucket limit was specified.</td>
</tr>
<tr>
<td></td>
<td><strong>User Response:</strong> Edit the bucket limits ensuring at least one bucket limit is specified and repeat the command.</td>
</tr>
<tr>
<td>APF0150W</td>
<td>A transaction filter must be specified. Use '*' for all transactions.</td>
</tr>
<tr>
<td></td>
<td><strong>Explanation:</strong> A transaction filter must be added to the entry before closing the dialog.</td>
</tr>
<tr>
<td></td>
<td><strong>User Response:</strong> Add a value for the transaction filter and repeat the command.</td>
</tr>
<tr>
<td>APF0151W</td>
<td>Schedule rule <code>schedule_rule</code> has an invalid stop time. Ensure the stop time is after the start time.</td>
</tr>
<tr>
<td></td>
<td><strong>Explanation:</strong> A valid period of time was not specified for data collections.</td>
</tr>
<tr>
<td></td>
<td><strong>System Action:</strong> The command failed.</td>
</tr>
<tr>
<td></td>
<td><strong>User Response:</strong> Edit the values and repeat the command.</td>
</tr>
<tr>
<td>APF0154E</td>
<td>The application name may not exceed 127 characters.</td>
</tr>
<tr>
<td></td>
<td><strong>Explanation:</strong> The application name is too long.</td>
</tr>
<tr>
<td></td>
<td><strong>System Action:</strong> The command failed.</td>
</tr>
<tr>
<td></td>
<td><strong>User Response:</strong> Shorten the application name to within the limit and retype the CLI command.</td>
</tr>
<tr>
<td>APF0158E</td>
<td>One or more buckets contain invalid values.</td>
</tr>
<tr>
<td></td>
<td><strong>Explanation:</strong> The bucket limits are incorrect.</td>
</tr>
<tr>
<td></td>
<td><strong>System Action:</strong> The command failed.</td>
</tr>
<tr>
<td></td>
<td><strong>User Response:</strong> Verify that you have specified limits in ascending order and they do not overlap (each successive bucket boundary should be greater than the preceding one). Retype the CLI command.</td>
</tr>
</tbody>
</table>
APF0159E An invalid value was specified for the transaction filter switch. It must be either 'T' or 'F'.

Explanation: The transaction filter switch must be specified as True or False. True sets the switch so that parameters are used. False sets the default (all).

System Action: The command failed.

User Response: Retype the CLI command.

APF0160E An invalid value was specified for the database activation switch. It must be either 'T' or 'F'.

Explanation: The database transaction switch must be specified as True or False. True sets the database activation switch to save information to the RIM database. False does not save.

System Action: The command failed.

User Response: Retype the CLI command.

APF0161E An invalid value was specified for the database detail level. It must be either 'L', 'M' or 'H'.

Explanation: An invalid database detail level has been specified for application performance data storage. L, M, and H represent Low, Medium, High.

System Action: The command failed.

User Response: Retype the CLI command.

APF0162E An invalid value was specified for the simulation interval.

Explanation: The simulation interval (time between each successive run) was set incorrectly.

System Action: The command failed.

User Response: Retype the CLI command with correction.

APF0167I Usage: wmaradelapp -a app_name

Explanation: This is a command syntax prompt that is displayed when a CLI command is incomplete or is incorrect.

User Response: Verify the command against the given syntax and retype the command.

APF0168E A value for the simulation parameter parameter is invalid or missing.

Explanation: 

System Action: The command failed.

User Response: Retype the CLI command and supply a correct value for the identified parameter.

APF0169W The profile is being used by another administrator: user_name.

Explanation: The profile cannot be opened because it is currently being used by another administrator.

System Action: The command failed.

User Response: Try again after the profile has been closed by the other administrator.
<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
<th>Explanation</th>
<th>System Action</th>
<th>User Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>APF0170W</td>
<td>The specified schedule rules do not overlap with the date range specified. At least one day from one rule must fall within the scheduled start and stop dates.</td>
<td>The scheduled start and stop dates must coincide with the specified schedule rules. At least one day of the week in the schedule rules, must fall within the start and stop dates.</td>
<td>The command failed.</td>
<td>Edit the start and stop dates and the schedule rules and try again.</td>
</tr>
<tr>
<td>APF0173E</td>
<td>Too many bucket values were specified. Please specify a maximum of 6 values.</td>
<td>The CLI contained too many bucket values.</td>
<td>The command failed.</td>
<td>Retype the CLI command and supply no more than six bucket values</td>
</tr>
<tr>
<td>APF0174E</td>
<td>The application <em>app_name</em> cannot be deleted from the registry because it exists in one or more profiles. Delete this application from all MarProfiles and try again.</td>
<td>A <code>wmardelapp</code> command was issued to delete an application from the repository but the application has already been added to one or more profiles.</td>
<td>The command failed.</td>
<td>Identify the profiles containing the application and delete it from them. Resend the command to check that all instances are deleted.</td>
</tr>
<tr>
<td>APF0175I</td>
<td>The application <em>app_name</em> was deleted from the registry.</td>
<td>A <code>wmardelappreg</code> command has been issued, and the product reports that it was executed.</td>
<td>The command completed successfully.</td>
<td>None.</td>
</tr>
<tr>
<td>APF0176E</td>
<td>Profile push ended with errors on the following Endpoints: <em>endpoint_names</em>.</td>
<td>There was an attempt to push a profile or profiles to one or more endpoints but some, failed. The most likely cause is a network communication problem.</td>
<td>The command failed.</td>
<td>Verify that you can communicate with the endpoints and try again. If the problem persists contact your system administrator.</td>
</tr>
<tr>
<td>APF0201E</td>
<td>Simulation and Client Capture collections need Microsoft Windows to run. You cannot run them on a UNIX machine.</td>
<td>Endpoints where Client Capture and Simulated applications are to be run, must be running Windows.</td>
<td>The profile or profiles containing the client capture or simulated application has been distributed to the correct endpoint. If necessary, use another endpoint.</td>
<td>None.</td>
</tr>
<tr>
<td>Code</td>
<td>Message</td>
<td>Explanation</td>
<td>System Action</td>
<td>User Response</td>
</tr>
<tr>
<td>--------</td>
<td>-------------------------------------------------------------------------</td>
<td>-------------------------------</td>
<td>---------------------------</td>
<td>-------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>APF0202E</td>
<td>Unable to retrieve endpoint information from the EndpointManager.</td>
<td>A software error has occurred.</td>
<td>The current operation is terminated.</td>
<td>Contact your local Tivoli representative.</td>
</tr>
<tr>
<td>APF0203I</td>
<td>Usage: wmarep –e endpoint_name</td>
<td>–h host_name</td>
<td>This is a command syntax prompt that is displayed when a CLI command is incomplete or is incorrect.</td>
<td>Verify the command against the given syntax and retype the command.</td>
</tr>
<tr>
<td>APF0204E</td>
<td>Unable to find the host</td>
<td>endpoint.</td>
<td>The given endpoint name or host name, is not found from the managed node where you ran the command.</td>
<td>The current operation is terminated. Verify that there are no network communication problems between managed node and relative endpoint. If the network is functioning correctly, then contact your system programmer or Tivoli representative.</td>
</tr>
<tr>
<td>APF0205E</td>
<td>Flag -f can not be used without flag -x.</td>
<td>This is an internal message.</td>
<td>The current operation is terminated.</td>
<td>Contact your Tivoli representative.</td>
</tr>
<tr>
<td>APF0206E</td>
<td>XML output format requires -a -u -t flags.</td>
<td>This is an internal message.</td>
<td>The current operation is terminated.</td>
<td>Contact your Tivoli representative.</td>
</tr>
<tr>
<td>APF0207E</td>
<td>Unknown output format specified.</td>
<td>This is an internal message.</td>
<td>The current operation is terminated.</td>
<td>Contact your Tivoli representative.</td>
</tr>
<tr>
<td>APF0208E</td>
<td>Transaction details are not supported for XML output format.</td>
<td>This is an internal message.</td>
<td>The current operation is terminated.</td>
<td>Contact your Tivoli representative.</td>
</tr>
</tbody>
</table>
**Messages**

**APF0209E** Can’t open XML output file.

**Explanation:** This is an internal message.

**System Action:** The current operation is terminated.

**User Response:** Contact your Tivoli representative.

**APF0303W** The Tivoli Application Performance Management mid-level manager on *(gateway_name)* reports: The file *(file_name)* from endpoint *(endpoint_name)* was not uploaded. It was not found on the endpoint or is empty.

**Explanation:** An attempt was made by the mid-level manager to upload a file from the endpoint but the file was not found.

**System Action:** The command failed.

**User Response:** Examine the mid-level manager log *(TMP/Mar/tapm_gw.log)* to see if the file was previously uploaded. It could be the case that more than one request to upload the file was sent to the mid-level manager. The first request will upload the file while the second will fail because the file no longer exists. If this is not the case, note the name of the missing file and contact Tivoli representative.

**APF0306W** The Tivoli Application Performance Management mid-level manager on *(gateway_name)* reports: The aggregation information in file *(file_name)* could not be sent to the RIM database.

**Explanation:** A failure occurred while sending performance data to the database.

**System Action:** The data is not sent and is cached on the mid-level manager.

**User Response:** Ensure RIM is properly connected to a database and restart the mid-level manager.

**APF0307W** The Tivoli Application Performance Management mid-level manager on *(gateway_name)* reports: All further attempts to send the file *(file_name)* to the RIM database will be deferred.

**Explanation:** The performance data in file *(file_name)* is deferred from further attempts to be sent to the database.

**System Action:** The file is cached on the mid-level manager.

**User Response:** Ensure RIM is properly connected to a database and restart the mid-level manager to cause the file to be resent to the database.

**APF0308E** The Tivoli Application Performance Management mid-level manager on *(gateway_name)* reports: The MarController distinguished object was not found. The mid-level manager is shutting down.

**Explanation:** The MarController distinguished object was not found. It is needed by the mid-level manager to synchronize RIM database access.

**System Action:** The mid-level manager shuts down.

**User Response:** Contact your Tivoli representative.

**APF0309E** The Tivoli Application Performance Management mid-level manager on *(gateway_name)* reports: The following exception occurred sending *(file_name)* to the RIM database: <exception>.

**Explanation:** A RIM exception occurred while sending data to the database.

**System Action:** The command fails and the data is cached on the mid-level manager.

**User Response:** Ensure RIM is properly configured and try again.
APF0310E  The Tivoli Application Performance Management mid-level manager on (gateway_name) reports: The endpoint (endpoint_name) was not found.

Explanation: Possible network or modem problems.
System Action: The command failed.
User Response: Retype the CLI command and supply a correct value for the identified parameter.

APF0311E  The Tivoli Application Performance Management mid-level manager on (gateway_name) reports: The following exception occurred while trying to open an IOM connection to the endpoint (endpoint_name): <exception>.

Explanation: An IOM exception occurred between the mid-level manager and an endpoint. This could be caused by network problems between the mid-level manager and the endpoint.
System Action: The command failed.
User Response: Ensure the endpoint is running and accessible and try again.

APF0312E  The Tivoli Application Performance Management mid-level manager on (gateway_name) reports: The following exception occurred while receiving a file from the endpoint (endpoint_name): <exception>.

Explanation: An IOM exception occurred between the mid-level manager and an endpoint. This could be caused by network problems between the mid-level manager and the endpoint.
System Action: The command failed.
User Response: Ensure the endpoint is running and accessible and try again.

APF0313E  The Tivoli Application Performance Management mid-level manager on (gateway_name) reports: The following exception occurred while trying to create the cache directory (cache_name): <exception>.

Explanation: An IOM exception occurred between the mid-level manager and an endpoint. This could be caused by network problems between the mid-level manager and the endpoint.
System Action: The command failed.
User Response: Ensure the endpoint is running and accessible and try again.

APF0315E  Tivoli Application Performance Management installation is corrupted on the specified endpoint. Client Capture collections cannot be started. Reinstall Tivoli Application Performance Management and try again.

Explanation: An executable file has become corrupted. Client Capture collections are not possible unless the product is reinstalled.
System Action: The command failed.
User Response: Reinstall Tivoli Application Performance Management.

APF0401I  The RIM object 'tapm' has been created with the following parameters:
- Database Vendor:
- RIM Host:
- Database Home:
- Database ID:
- Database User ID:
- Database Server ID:
- DB2 Instance Name:

User Response: none
The RIM object 'tapm' has been modified with the following parameters:
- Database Vendor:
- RIM Host:
- Database Home:
- Database ID:
- Database User ID:
- Database Server ID:
- DB2 Instance Name:
User Response: none

Too many options
Explanation: Too many parameters were specified in a wmarstartcoll command.
System Action: The command failed.
User Response: Check the command syntax

L parameter conflicts with D parameter
Explanation: The wmarstartcoll command has been issued with both –d and –l flags. The equivalent of asking to move non existant data to the database that you have not
System Action: The command failed.
User Response: Retype the command. If you require data to be stored, do not use the –l flag.

Instance data settings are invalid
Explanation: The wmarstartcoll values accompanying the –n flag are invalid.
System Action: The command failed.
User Response: Refer to "wmarstartcoll" on page 201 and check the values for –n.

Memory allocation error
Explanation: There is no more memory available
System Action: The command failed.
User Response: Contact your system programmer.

No local instance logging specified
Explanation: The wmarstartcoll values accompanying the –n flag are invalid.
System Action: The command failed.
User Response: Refer to "wmarstartcoll" on page 201 and check the values for –n.

Local logging of instance data is disabled. No data will arrive in the database.
Explanation: The wmarstartcoll values accompanying the –n flag are invalid.
System Action: The command partially failed.
User Response: Refer to "wmarstartcoll" on page 201 and check the values for –n.
APF0457E  An error occurred while retrieving data from the instance file on the endpoint.

Explanation: The product has tried unsuccessfully to fetch the instance file that was to be sent to the gateway from the endpoint. There may be a network communication problem.

System Action: The command failed.

User Response: Verify the endpoint path(s) for information data file location are correct. The simple answer may be lack of disk space. If the failure persists, contact your system programmer or Tivoli representative.

APF0459E  The data file <file_name> on the endpoint was not found.

Explanation: The gateway tried to upload the specified file from the endpoint but did not find it.

System Action: The upload failed.

User Response: Lost data can be recovered by using the wmarsenddata command.

APF0460E  Data from instance file has NOT been loaded on database.

Explanation: The process of data loading on the database has failed.

System Action: The command failed.

User Response: Verify the previous message that explains the details of the failure. If the problem persists contact your system programmer or Tivoli representative.

APF0461W  Only correlated data will arrive in the database.

Explanation: The wmarstartcoll values accompanying the –n flag are invalid.

System Action: The command partially failed.

User Response: Refer to “wmarstartcoll” on page 201 and check the values for –n.

APF0462W  No metrics will arrive in the database.

Explanation: The wmarstartcoll values accompanying the –n flag are invalid.

System Action: The command partially failed.

User Response: Refer to “wmarstartcoll” on page 201 and check the values for –n.

APF0463W  Local logging of instance data/metrics is disabled.

Explanation: The wmarstartcoll values accompanying the –n flag are invalid.

System Action: The command partially failed.

User Response: Refer to “wmarstartcoll” on page 201 and check the values for –n.

APF0464E  <application_name> is a ClientCapture application: flags -r, -p or -x are not supported.

Explanation: The specified application has been registered as a Client Capture application. Stated flags are not supported for client capture.

System Action: The command failed.

User Response: Retype the command using correct syntax.
APF0465E  File &lt;file_name&gt; not found.
Explanation: The file name specified by the -x parameter of the wmarstartcoll command was not found.
System Action: The command failed.
User Response: Check that the file has been registered and distributed to that endpoint. Retype the command supplying the correct file name.

APF0466E  Generic error reading file &lt;file_name&gt;.
Explanation: A file specified in the wmarstartcoll command may be corrupted.
System Action: The command failed.
User Response: Check and replace the file if neccessary.

APF0467E  The -x flag is not supported for application application_name.
Explanation: The specified application belongs to a class that does not support the -x flag.
System Action: The command failed.
User Response: Retype the command using correct syntax. Refer to "wmarstartcoll" on page 201

APF0468E  Data Provider for specified application is unknown to this APM installation. Command failed.
Explanation: An attempt was made to start a collection on an endpoint that does not support that type of collection
System Action: The command failed.
User Response: Check the endpoint environment and use a different endpoint if neccessary.

APF0469W  Some collections that ran during last engine activation cannot be started now. See the log files for more details. Command completed succesfully.
Explanation: The engine has been restarted successfully on an endpoint but some collections that were active before cannot now be started.
System Action: The command succeeded but not all collections are available.
User Response: Redistribute any missing collections.

APF0470E  Endpoint platform is not supported
Explanation: An attempt has been made to perform an action on the endpoint but the endpoint platform is not supported by Tivoli Application Performance Management.
System Action: The command failed.
User Response: If the action is required, use another and appropriate endpoint.

APF0471W  Local logging of summary data is disabled. No data will arrive in the database.
Explanation: The wmarstartcoll values accompanying the -n flag are invalid. You cannot move aggregated data to the database if they are not logged locally.
System Action: The command partially failed.
User Response: Refer to "wmarstartcoll" on page 201 and check the values for -n.
APF0472E Incorrect parameter specified: invalid date format.

Explanation: A wmarsenddata command has been sent with the wrong date parameter.
System Action: The command failed.
User Response: Retype the command with correct parameter.

APF0473E <application_name> is a ARM instrumented application: flags -r, -p or -x are not supported.

Explanation: The specified application has been registered as an ARM application. Indicated flags are not supported.
System Action: The command failed.
User Response: Retype the command using the correct syntax.

APF0474E Extended collection data provider cannot start collection. Check input parameters.

Explanation: A wmarstartcoll on a ClientCapture, simulation or Epp collection failed because one or more of the parameters requested by the specific collection type is not correct (either extra, wrong or missing). The parameters that can cause the above error are those specified with flags -r, -x and -p
System Action: The command failed.
User Response: Retype the command with correct parameters.
Command Reference

This appendix describes the purpose and syntax of the command line interface (CLI) commands, that are used to set up and control the Tivoli Application Performance Management application.

Tivoli Application Performance Management Commands

The following table lists the product commands and their purpose statements for engine or profile control. You can run these from the command line of the management server in a distributed environment:

Notes:

1. To execute each command you must have the requisite authorization or role.

2. In the table below, the commands are listed in the order you would logically use them.
### Tivoli Application Performance Management Commands

#### Server Command Purpose Page

<table>
<thead>
<tr>
<th>Server Command</th>
<th>Purpose</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Application registration commands</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>wmarregapp</td>
<td>Registers an application in the product repository.</td>
<td>194</td>
</tr>
<tr>
<td>wmargetapp</td>
<td>Returns a list of all registered applications in the product repository.</td>
<td>186</td>
</tr>
<tr>
<td>wmardelapp</td>
<td>Removes a registered application from the product repository.</td>
<td>185</td>
</tr>
<tr>
<td><strong>Application commands</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>wmarlsapp</td>
<td>Lists all applications and transactions at the given endpoint_name (endpoint).</td>
<td>191</td>
</tr>
<tr>
<td>wmarlseng</td>
<td>Lists all active collections and simulations running at the given endpoint_name (endpoint).</td>
<td>193</td>
</tr>
<tr>
<td><strong>Engine commands</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>wmarstarteng</td>
<td>Starts the engine at the given endpoint_name (endpoint).</td>
<td>204</td>
</tr>
<tr>
<td>wmarstartcoll</td>
<td>Defines a new data collection on specified applications.</td>
<td>201</td>
</tr>
<tr>
<td>wmarstopcoll</td>
<td>Stops a data collection.</td>
<td>205</td>
</tr>
<tr>
<td>wmarstopeng</td>
<td>Stops the engine at the given endpoint_name (endpoint).</td>
<td>206</td>
</tr>
<tr>
<td>wmargetstatus</td>
<td>Retrieves data about the engine at a specific endpoint or about the mid-level manager at the gateway.</td>
<td>189</td>
</tr>
<tr>
<td>wmarcleareng</td>
<td>Stops all active collections and simulations.</td>
<td>184</td>
</tr>
<tr>
<td>wmarsetstatus</td>
<td>Enables you to set properties that modify the behavior of the engine at the given endpoint_name (endpoint), or the mid-level manager at an identified gateway.</td>
<td>197</td>
</tr>
<tr>
<td>wmarsenddata</td>
<td>Immediately (disregards scheduled upload) sends aggregated data from endpoint to gateway.</td>
<td>196</td>
</tr>
<tr>
<td>wmargetdata</td>
<td>Retrieves performance data on transactions in an application.</td>
<td>187</td>
</tr>
</tbody>
</table>

CLI commands enable you to perform system operations from a UNIX or PC command line instead of using the Tivoli desktop and the product graphical user interface (GUI). All the Tivoli product commands begin with **wmar** and vowels are often omitted to shorten the name of a command.

Commands are developed using the **wmar+verb+object** syntax, which matches the way you might think of the action. For example, to start a data collection, you use the **wmarstartcoll** command. This command contains scheduling information and repeats at pre-defined intervals until a stop command **wmarstopcoll** is issued. In each case the command is supported by an argument supplying the identity of the robotic client or endpoint where the data collection needs to take place.

It is often necessary or more convenient, to invoke an application operation from the command line rather than from the desktop. For example:

- You may not have access to a desktop, or perhaps you are connected to a slow network where CLI commands would probably be faster than using the GUI.
- You want to group several operations in a shell script or batch file.
- An operation is not available using the desktop.
- You prefer to invoke a command from a shell.
CLI Commands

The product system supports composite command line instructions and command messages. These command strings are composed automatically from information you have entered into the Graphical User Interface (GUI) screens or are typed in from the command line. Either method may be used or you can use a combination of both. In each case the commands are automatically parsed to make sure all required information is present and valid.

Before issuing a command (executable or script), make sure the Tivoli environment is started on the server using the **setup_env** command as follows:

On a UNIX machine run:
```
. /etc/Tivoli/setup_env.sh
```

On a Windows NT machine the command is:
```
"%windir%\system32\drivers\etc\tivoli\setup_env.cmd"
```

All the CLI commands take the following form, in which a command is followed by an argument or series of arguments where:

- **command** specifies the action to be taken.
- **argument** is a single letter or series of letters that is case-insensitive and represents a parameter for the command. An argument is always preceded by a hyphen (–). Arguments can be mandatory (required) or optional.

**Note:** When an argument is used to pass values, quotes should always be placed around them to prevent the command shell interpreting any special character in the string as an argument.

Command Line Syntax

This chapter uses the following special characters to define the syntax of commands:

- `[ ]` Identifies optional arguments. Arguments not enclosed in brackets are required.
- `...` Indicates that you can specify multiple values for the previous argument.
- `|` Indicates mutually exclusive information. You can use the argument to the left of the separator or the argument to its right. You cannot use both arguments in a single use of the command.
- `{ }` Delimits a set of mutually exclusive arguments when one of the arguments is required. If the arguments are optional, they are enclosed in brackets ([ ]).

For example:

```
wmargetapp [-a app_name]
```

If you do not specify the `–a` argument, all applications registered to the product registry are listed.

**Note:** When you are using the command line interpreter (CLI), you should enclose characters intended to be passed as part of an input parameter, in quotation marks.

Getting Help on Commands

The following section lists the product commands, with syntax and descriptions of their functions. You can access help information by typing `-` on the command line of any platform - the command name without any parameters.
wmarcleareng

Clears the engine configuration. All active collections and simulations are stopped.

Syntax

wmarcleareng endpoint_name.

Description

Use the wmarcleareng command to reset an endpoint. When you use this command you delete all collection information associated with the endpoint so that when you restart the engine, it is clean. Old values are not reinstated. This means that after using wmarcleareng and restarting the engine you must send new collection information either by using a wmarstartcoll command or by distributing original or updated versions of profiles on the endpoint.

Options

endpoint_name

Identity of the endpoint that you want to reset.

Authorization

Senior.

Examples

The command below stops all active collections at the endpoint named Idaho.

wmarcleareng Idaho

See Also

wmarstartcoll, wmarstarteng.
wmardelapp

Removes a registered application from the product application repository.

Syntax

wmardelapp –a app_name.

Description

This command enables you to remove a registered application from the repository so that it is taken out of use. You would issue this command for instance, if you inadvertently typed its name incorrectly and needed to reregister it. If the application is being used by a profile, the command fails.

Options

–a app_name

The name of the application entry to delete.

Authorization

Senior.

Notes

When you are using the Add Application to the Profile window and another operator deletes an application, that application will remain in the Available Applications panel until you click the Refresh button. If you try to add that application to a profile, an error message is generated.

Examples

1. The example below removes the application vusAppl and all dependencies to it from the repository.

wmardelapp –a vusAppl

See Also

wmarregapp, wmargetapp.
wmargetapp

wmargetapp

Returns a list of all applications registered to the product application repository.

Syntax

wmargetapp [–a app_name]

Description

This command provides a list of all entries that are registered in the application repository which can be included in profiles. You can also use the command to check if a specific application is registered by using the –a flag.

Note: It does not imply that such entries are valid applications.

Options

–a app_name

The name of the application.

Authorization

Admin or Senior.

Examples

1. The example below lists all applications found in the product application repository.

   wmargetapp

2. The example below lists a specific application found in the application repository. If supplementary information on type of application and dependency is available, it is also displayed.

   wmargetapp –a vusscript

See Also

wmarregapp, wmardelapp
wmargetdata

Retrieves valid current and historical performance data about a transaction.

Syntax

wmargetdata –a app_name [–t tx_name] [–u user_id] [–i interval] [–s start_time [–e end_time]] [–m metric_name] endpoint_name

Description

You use the wmaregdata command to fetch current run information for an application on a chosen endpoint.

Options

–a app_name

The name of the application.

–u user_id

The user identifier. If the parameter is omitted, data is aggregated for all users running the application specified by application_name.

–t tx_name

The name of the transaction. If this argument is omitted, data is retrieved for all transactions that the application has performed.

–m metric

The name of the metric to be returned. If the parameter is omitted, values for the first six metrics are returned. Valid metric names are:

<table>
<thead>
<tr>
<th>Metric</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TxRTAvg</td>
<td>Average response time</td>
</tr>
<tr>
<td>TxRTMin</td>
<td>Minimum response time</td>
</tr>
<tr>
<td>TxRTMax</td>
<td>Maximum response time</td>
</tr>
<tr>
<td>TxGood</td>
<td>Number of good transactions</td>
</tr>
<tr>
<td>TxFailed</td>
<td>Number of failed transactions</td>
</tr>
<tr>
<td>TxAborted</td>
<td>Number of terminated transactions</td>
</tr>
<tr>
<td>Bucket1...Bucket7</td>
<td>Number of entries in specified bucket</td>
</tr>
<tr>
<td>Buckets</td>
<td>Reports all the previous metrics</td>
</tr>
<tr>
<td>All</td>
<td>Reports all the previous metrics and the user defined metrics</td>
</tr>
</tbody>
</table>

–i interval

Specifies the number of collection records that are retrieved. For instance if –i is set to 10 and collection interval is every 60 seconds, ten data collections are retrieved, representing activity during the last ten minutes.

The records are displayed in reverse order, that means the most recent record is displayed first.

–s [[yyyy-mm-dd],[hh:mm]]

The start_time of the intervals to be retrieved for which the default is current day. If the “yyyy-mm-dd” value prefixes the time, then a previous day can be specified. If a time is not specified, then default is start of collections on that day.

–e [[yyyy-mm-dd],[hh:mm]]

The end_time of intervals period. The date value can be used to specify that more than one days intervals are to be retrieved. If the time is not specified, all intervals for that day or the last day are returned.
**endpoint_name**

Specifies the endpoint from where the transaction data should be retrieved.

**Authorization**

Admin or Senior.

**Notes**

- If `-m` is not specified, then the first six metrics are retrieved.
- If the collection for the specified transaction is not active, the command returns a message.
- If neither `-s` nor `-i` are specified then the record for the last collection interval is retrieved.
- If `-e` is not specified it is assumed to be same as current day and time.
- If `-e` is specified then `-s` must have a date or time or both, older than the `-e` values.

**Examples**

1. This example fetches stored data for the transaction `my_transaction` which is part of application `my_application` while it ran on endpoint LCF98 between 12.30 and 18.00 today.
   ```
   wmargetdata -a my_application -t my_transaction -s 12:30 -e 18:00 LCF98
   ```

2. This example fetches stored data for the transaction `my_transaction` which is part of application `my_application` while it ran on endpoint LCF98 between 12.30 and 18.00 yesterday (date is specified).
   ```
   wmargetdata -a my_application -t my_transaction -s 2001-03-23,12:30 -e 2001-03-23,18:00 LCF98
   ```

**See Also**

`wmarsenddata`
wmargetstatus

Retrieves the current engine status, logfiles for the endpoint and other information

Syntax

wmargetstatus [-g] host_name.

Description

Use this command to find out what is happening at an endpoint or gateway mid-level manager.

Options

-g Specifies that you want to retrieve status information from a gateway mid-level manager.

host_name Identifies the endpoint from where you want to retrieve engine status, or the managed node where a gateway mid-level manager is installed from where MLM status information should be retrieved.

Authorization

Admin or Senior.

Examples

1. This example retrieves the status of endpoint LCF364

wmargetstatus LCF364

APF0097I The engine is running
Log files: /opt/Tivoli/lcf_sbellucc/dat/11/Mar/tapm_ep.log
          /opt/Tivoli/lcf_sbellucc/dat/11/Mar/tapm_engine.log

Engine settings:

FileMaxSize = 5000000
LogComponents = all_engine
LogLevel = 1
AggregationStartTime = 00:00
AggregationEndTime = 00:00
CleanupStartTime = 20:00
CleanupApplicationTimeout = 168:00
CleanupTransactionTimeout = 01:00
MaxSummaryDataFiles = 3
AggregationTimeStep = 24
SummarizationOn = true
MaxTransInstanceFiles = 3
TransInstanceFilesMaxSize = 5000000
GenerateCorrelator = true
InstanceLogging = false
AggregationEnabled = true
AggregationRandTimeRange = 01:00
InstanceStartTime = 00:00
InstanceEndTime = 00:00
InstanceTimeStep = 24
InstanceEnabled = true
InstanceRandTimeRange = 01:00

2. This example retrieves the status of the mid-level manager on gateway ohio.

   `wmargetstatus –g ohio`

   LogLevel = 3
   DBUploadStartTime: = 00:00
   DBUploadStopTime: = 06:00

See Also

`wmarstarteng, wmarstopeng wmarsetstatus,`
**wmarlsapp**

Lists all applications and transactions that have made an ARM call to the engine since the engine started at the endpoint.

**Syntax**

```
wmarlsapp [-a app_name] [-t tx_name] [-u user_id] endpoint_name
```

**Description**

This command is used to identify which applications and transactions are active, or have been active at a specified endpoint.

**Options**

- `-a app_name`
  The name of the application.
- `-t tx_name`
  The name of the transaction. If this argument is omitted, data is retrieved for all transactions defined by the `app_name`.
- `-u user_id`
  The user identity. If the parameter is omitted, data is retrieved for all users running the application specified by `app_name`.

`endpoint_name`
Specifies the endpoint from where the application and transaction data should be retrieved.

**Authorization**

Admin or Senior.

**Notes**

Depending on which flags are specified, the command returns information with different granularity. There are four possible cases:

- No flags are being passed. The command returns a list of application names and `user_id`.
- Only `-a` flag is specified. The command returns a list of transaction names, for the specified application, for all users currently running.
- `-a` and `-u` flags are specified. The command returns a list of transaction names, for the specified application and user.
- `-a`, `-u` and `-t` flags are passed. The command returns metadata description of the specified transaction.

**Examples**

1. This example returns a list of all applications with `user_id`, running or dormant after running, since the engine started at the endpoint “Bank”.
   
   `wmarlsapp Bank`

2. This example lists all the transactions associated with the application named `dummy_application` running on endpoint LCFNT.
   
   `wmarlsapp -a dummy_application LCFNT`

3. This example provides information associated with the transaction1 within `dummy_application` running on endpoint LCFNT for user vusUser.
wmarsapp –a dummy_application -t transaction1 -u vusUser LCFNT
wmarlseng

Lists all active collections at the specified endpoint.

Syntax

wmarlseng endpoint_name

Description

Lists all active collections at the specified endpoint.

Options

endpoint_name

Name of the endpoint or node where collections are being made.

Authorization

Admin or Senior.

Return Values

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Collection active on application</td>
<td>appl</td>
</tr>
<tr>
<td>Transaction filter</td>
<td>*</td>
</tr>
<tr>
<td>bucket limits (milliseconds)</td>
<td>100 500 1000 2000 5000 10000</td>
</tr>
<tr>
<td>collection interval (seconds)</td>
<td>60</td>
</tr>
<tr>
<td>collection active on</td>
<td>always=Everyday()</td>
</tr>
<tr>
<td>start date (yyyy:mm:dd)</td>
<td>2001 06 13</td>
</tr>
<tr>
<td>stop date (yyyy:mm:dd)</td>
<td>not specified</td>
</tr>
<tr>
<td>Correlator Generation</td>
<td>enable</td>
</tr>
<tr>
<td>Summary Data:</td>
<td></td>
</tr>
<tr>
<td>Local Logging:</td>
<td>ACTIVE</td>
</tr>
<tr>
<td>Data base storage:</td>
<td>active</td>
</tr>
<tr>
<td></td>
<td>Minimum(MAX, MED)</td>
</tr>
<tr>
<td>Instance Data:</td>
<td></td>
</tr>
<tr>
<td>Local Logging:</td>
<td>NO_DATA</td>
</tr>
<tr>
<td>data-base storage:</td>
<td>NO_DATA</td>
</tr>
<tr>
<td>Available Extended Information</td>
<td></td>
</tr>
<tr>
<td>Data Provider Name</td>
<td>&lt;data_provider_name&gt;</td>
</tr>
<tr>
<td>Start Collection Parameters</td>
<td>r=1, &lt;ULRname&gt; Param1=qw (string passed at collection start time)</td>
</tr>
</tbody>
</table>

Examples

1. This example lists details of all active collections that are taking place on the endpoint identified as LCFNT.

   `wmarlseng LCFNT`

See Also

wmarstartcoll, wmarstopcoll

Tivoli Application Performance Management User’s Guide
wmarregapp

Registers an application to the application registry.

Syntax

```
wmregapp -a app_name [-f file_name] [-l label] [-d data_provider] [-w]
```

Description

The `wmarregapp` command creates an entry in the application repository. Each application that you want to collect data for, has to be previously defined by the `-a` flag to the product repository. For an ARM instrumented application only the `application_name` is needed. Other applications are defined by:

- Application name
- One or no associated file
- Reference to the data provider, a process that can collect data and make ARM calls to the product.

**Note:** On UNIX systems, the user who runs the wmarregapp command to register an application (except for the instrumented applications) needs to have the permission to write in the product repository directory (`$BINDIR/../generic/Mar/RepositoryData`) and in the temporary directory (the directory you get by issuing the wtemp command in the Tivoli environment) on the TMR server.

Options

- `-a app_name`
  The name of the application that is to be registered. If the application already exists in the repository and the `-w` flag has not been specified, the command returns an error.

- `-f file_name`
  The name of the file associated with the application. If this parameter is specified, and flag `-l` is not specified, the file is registered using the application name passed by the `-a` flag.

  **Note:** You should enclose the string following the `-l` and `-f` flags in single quotation marks.

- `-l file_label`
  Denotes a label for the file if it needs to be used by more than one application.

- `-d data_provider`
  Specifies the data provider name, which must match an entry in a data provider look-up table in the product. The current supported data providers are probes defined for EPP (`DataBaseAccess | IMAPAccess | IMAPSearch | NotesAddrBookLookup | NotesServerAvail | NotesCalendar | WebPageAccess | WebPageSetAccess`)

- `-w replace`
  Is an optional flag that switches ON replace mode and allows you to overwrite a registered application with another version of it.

Authorization

Senior.
Examples

1. The example below shows how to use the `wmarregapp` command to register an ARM instrumented application named `armApplication` that does not have any associated simulation script.
   
   ```
   wmarregapp -a armApplication
   ```

2. The example below shows how to use the `wmarregapp` command for an extended application collection that does not require any extra associated file.
   
   ```
   wmarregapp -a app1 -d WebPageAccess
   ```

3. The example below shows how to use the `wmarregapp` command to record an application with a new dependency. The dependency uses the same name as the application but it cannot be used by other applications.
   a. if a .usr file is associated, the product assumes it is a simulation and the reference to the data provider table is ScriptController.
   b. if a .xml file is associated, the product assumes it is a client capture. It opens the xml file and reads the reference to the sensor. That reference becomes the reference to the data provider table.

   ```
   wmarregapp -a appl -f file_name
   ```

4. The examples shown below registers an application with the associated file `<file_name>` using the label ll.

   ```
   wmarregapp -a appl -f file_name -l ll
   ```

5. The examples shown below registers application ll from the example above reusing the file pointed to by label ll.

   ```
   wmarregapp -a appl1 -l ll
   ```

See Also

`wmargetapp`
wmarsenddata

Enables a user to force an immediate data upload from an endpoint to the database.

Syntax

wmarsenddata [–i] [–d yyyy-mm-dd] [–s hh] [–e hh] endpoint_name

Description

Data is normally uploaded from an endpoint to the gateway according to the schedule and preferences that were specified with the wmarsetstatus command. The wmarsenddata command is used to force an immediate upload of latest collected data from the specified endpoint to the gateway and database. If you use the command without arguments the latest aggregated file is uploaded.

Options

–i
Used to send instance data. If the flag is not used, aggregated data is sent.

–d date
YYYY-MM-DD identifies the data file to be sent to the database. The default is the current day, however; the –d option enables you to specify another set of data if more than one set is still present at the endpoint. If an upload failure occurs, the data remains on the endpoint until the failure has been resolved and is sent with any other aggregated data sets at the next automatic aggregation collection schedule. The wmarsenddata command enables you to send any previous days data, immediately after a problem is resolved and without waiting for the next automatic aggregation.

–s
start time hh:mm for the aggregation period.

–e
end time hh:mm for the aggregation period. The greatest difference allowed between –s and –e is 24. For instance a –s value of 00:00 and –e value of 00:01 specifies an aggregation of 59 sends data.

endpoint_name
Specifies the endpoint from where the transaction data should be retrieved.

Authorization

Senior.

Examples

1. This example sends the current aggregated transaction data file on endpoint LFCNT, to its associated gateway.

   wmarsenddata LFCNT

2. This example sends the aggregated transaction data file for collections during 23rd March 2001 on endpoint dallas, to its associated gateway mid-level manager.

   wmarsenddata -d 2001-03-23 dallas

See Also

wmarsetstatus
**wmarsetstatus**

Sets the status of the engine on the given endpoint or the product mid-level manager on the given managed node.

**Syntax**

```
wmarsetstatus [–g] PROPERTY=value, PROPERTY=value .....host_name
```

**Description**

The *wmarsetstatus* command can be used to modify the behavior of the engine, by enabling you to change properties for log, upload, cleanup and collection. The *wmarsetstatus* command also enables you to modify the settings of a mid-level manager on a managed node that is a gateway.

**Options**

*host_name*

Specifies the endpoint where the engine is located and to which this status information should be sent. If the *–g* option is used, *host_name* refers to the managed node that has the mid-level manager which requires a status change.

*–g*

Used to set the status for a managed node containing a mid-level manager.

**Possible Properties that can be set on an Engine are:**

**LogLevel**

This keyword is used to filter the logging activity. Possible values are:

- **0**  No logging is active.
- **1**  Logging is done to minimum detail level (default).
- **2**  Logging is done to medium detail level.
- **3**  Logging is done to maximum detail level.

**LogComponents**

This keyword is needed to activate the trace on a selected component. Possible values are:

- **NONE**  No trace is active.
- **ARMSERVER**  ARM calls (such as arm_init, arm_getid and others) processing is logged.
- **COMMAND**  Any command such as *wmarsstartcoll, wmargetstatus* is logged.
- **COLLECTION**  Engine collection activity is logged.
- **AGGREGATOR**  Aggregation of the engine data files is logged.
- **ALLENGINE**  Log is active on all engine components (the default).

The engine can be set to log multiple components. In these cases, a full stop or period character "." is used as a separator.
Example: `wmarsetstatus
LogComponents=AGGREGATOR.COLLECTION endpoint_name`

**FileMaxSize** *(range 100-100,000,000)*  
Maximum size that the file `tapm_engine.log` may reach. When this size is reached, a rollover causes the file to be truncated in such a way that the file contains most recent information. Default is 5,000,000 bytes.

**GenerateCorrelator** *(true, false)*  
Possible values are true or false (default is true). If the flag is set to true correlation data is collected by the engine.

**CleanupStartTime** *(hh:mm, where 00:00 <= hh:mm <= 23:59)*  
Starts a daily clean up of ARM IDs. The clean up process occurs either at <hh:mm> after midnight (default value is set to 20:00), or when the engine is stopped.

**CleanupApplicationTimeout** *(hh:mm, where 00:05<=hh:mm<=672:00 (4 weeks))*  
Defines when an application and all its transaction types (ARM IDs) must be removed from the engine memory. The cleanup removes all applications that did not receive an update by either an `arm_start`, `arm_update` or `arm_stop` in the last `CleanupApplicationTimeout` interval. Default value is 168:00 (one week).

**CleanupTransactionTimeout** *(hh:mm, where 00:01<=hh:mm<=24:00)*  
Defines when ARM transaction handles are removed from the engine memory. If a transaction has been assigned an ARM handle at `arm_start` and a time interval has expired without a corresponding `arm_stop` call or `arm_update`, that transaction is removed. That means the `arm_start` is ignored. Default is 01:00

**SummarizationOn** *(true, false)*  
Possible values are true or false (default is true). If the flag is set to true the correlation summarization process is active.

**MaxSummaryDataFiles** *(range 2-30)*  
Sets the maximum number of summary data files that can be stored at the endpoint. Default is 3.

*Note:* Summary files are coupled. For example `mardata20010320.dat` and `marinfo20010320.dat` form a pair and so the effective maximum number is twice that specified.

**AggregationEnabled** *(true, false)*  
Possible values are true or false (default is true). If the flag is set to true the aggregation process is active.

**AggregationTimeStep** *(range 1 - 24)*  
Defines the number of times aggregation can take place in a day. Aggregation is to a one hour interval and can therefore take place up to 24 times.

**AggregationStartTime** *(hh:mm, where 00:00<= hh:mm<=23:59)*  
Defines when the first aggregation process of the day should take place.

**AggregationEndTime** *(hh:mm, where 00:00<= hh:mm<=23:59)*  
Defines when the aggregation processes should close.

**AggregationRandTimeRange** *(hh:mm, where 00:00<= hh:mm<=23:59)*  
Used to randomize the exact time when the aggregation process starts. A random time value (default value is set to 1 hour) is generated in the interval between 0
(zero) and AggregationRandTimeRange. This time value is added to AggregationStartTime to provide the exact time for starting the aggregation process.

InstanceLogging(true, false)
Possible values are true or false (default is false). If the flag is set to true, instances data is collected by the engine.

MaxTransInstanceFiles (range 2-30)
Sets the maximum number of transaction instance files that can be stored on an endpoint. Default is 3.

TransInstanceFilesMaxSize (range 100-100,000,000)
Maximum size that the transaction instance log files marrawdatayyyyymmdd.dat and marrawinfoyyyyymmdd.dat may reach. When this size is reached, a rollover causes the file to be truncated in such a way that the file contains most recent information. Default is 5,000,000 bytes.

InstanceEnabled (true, false)
Possible values are true or false (default is true). If the flag is set to true the instance logging process is active.

InstanceTimeStep (range 1 - 24)
Defines the number of instance logging can take place in a day. Default logging is to a one hour interval and can therefore take place upto 24 times per day.

InstanceStartime (hh:mm, where 00:00<= hh:mm<=23:59)
Defines when the first instance logging process of the day should take place.

InstanceEndTime (hh:mm, where 00:00<= hh:mm<=23:59)
Defines when the instance logging processes should close.

InstanceRandTimeRange (hh:mm, where 00:00<= hh:mm<=23:59)
Used to randomize the exact time when the instance logging process starts. A random time value (default value is set to 1 hour) is generated in the interval between 0 (zero) and InstanceRandTimeRange. This time value is added to InstanceStartTime to provide the exact time for starting instance logging process.

Possible Properties that can be set on the Mid-level Manager are:

LogLevel
This keyword is used to filter the logging activity. Possible values are:

0 No logging is active.
1 Logging is done to minimum detail level (default).
2 Logging is done to medium detail level.
3 Logging is done to maximum detail level.

DBUploadStartTime (hh:mm, where 00:00 <= hh:mm <=23:59)
Specifies the start time after midnight for the period in which the mid-level manager is eligible to send files to the database. Default is 00:00 (midnight).

DBUploadStopTime (hh:mm, where 00:00 <= hh:mm <=23:59)
Specifies the stop time after midnight for the period in which the mid-level manager is eligible to send files to the database. No request to send can be started after this time. Any file currently being sent when this time occurs is allowed to complete. Default is 06:00.
Note: If the stop time is before the start time, for example 22:00 and stop is 03:00, this means the upload to the database is done from 22:00 to 03:00 of the next day, passing through midnight.

Authorization
Senior.

Examples
1. This example sets the endpoint log level to minimum. Components to be logged are COMMANDS and COLLECTION. It also specifies that GW upload should take place at midday.
   wmarsetstatus LogLevel=1,LogComponents=COMMAND.COLLECTION,
   AggregationStartTime=12:00 tivoli_ep
2. This example sets the medium level for logging on a managed node (gateway); it also specifies that the mid-level manager is eligible to send data to the database in the interval from 01:00 to 08:00.
   wmarsetstatus -g DBUploadStartTime=01:00,
   DBUploadStopTime=08:00,LogLevel=2 my_managed_node

See Also
wmargetstatus
wmarstartcoll

Defines a new data collection on the specified application.

Syntax

```
```

Note: When an argument is used to pass values, quotes should always be placed around them to prevent the command shell interpreting any special character in the string as an argument.

Description

The `wmarstartcoll` command starts and defines a new data collection on the specified application. The application must be previously registered to the product application repository using the `wmarregapp` command.

Options

```
-a app_name
  The name of the application.

-t tx_filter
  Only transactions whose name or specified part of the name (see “Creating a Transaction Filter” on page 83) matches the filter, are collected. The filter can be a string containing “*” or “?” wildcards, where:
  *
    Represents any series of accompanying characters in the string that do not have to match that part of the transaction name.
  ?
    Represents any single character.

-b buckets
  Bucket boundaries. A comma separated list of bucket limits, expressed in milliseconds, specified in ascending order. See “Choosing Bucket Limits” on page 84. At least one bucket limit must be specified if you use this parameter. A maximum number of six boundaries can be specified. If this parameter is not specified, the collection is started with default bucket limits which are: 100, 500, 1000, 2000, 5000 and 10000 milliseconds respectively.

-s start_date
  Represents the date from which the collection is active. The format is yyyy-mm-dd. If this parameter is omitted, the collection starts immediately.

-e end_date
  Represents the date until which, the collection is active. The format is yyyy-mm-dd. If this parameter is omitted, the collection continues until it is explicitly stopped using the `wmarstopcoll` or `wmarcleareng` command.

-c rule_name=cycle_1[start time-end time];rule_name=cycle_2[start time-end time]
  Determines when a collection is active and is a semicolon (;) separated list of schedules for collections. Start time-end time represent the hour interval over which the collection is active during the day in the specified cycle. The format is hh:mm (24h format). If start time and stop time are not specified, the cycle is considered active all the day.
If the –c flag is not specified the collection is considered active throughout the period, between `start_date` and `end_date`.

Possible daily cycles are:

- Monday
- Tuesday
- Wednesday
- Thursday
- Friday
- Saturday
- Sunday

Examples:
- `–c rule1=Monday,Tuesday` represents Monday and Tuesday all the day.
- `–c rule1=Monday[10:30-16:00]` represents Monday from 10:30 to 16:00.
- `–c "rule1=Monday[08:00-13:00];rule2=Saturday [12:15-20:30]"` represents Monday from 8:00 to 13:00 and Saturday from 12:15 to 20:30.

**Note:** The weekday must be typed in the same language as the Tivoli Application Performance Management application version. For instance

```
rule1="y—jú[10:30-16:00] 1ab78028_ep
```

where “y—jú is Monday in Japanese.

- **k** Disables correlation for the specified collection.

- **i collection_interval:**
  Is the interval in minutes (frequency) at which data should be collected. If not specified, its value is defaulted to 10 minutes. The minimum value is one minute, maximum value is 60 minutes.

- **l** Disables summarization for the specified collection. If this flag is present, the `wmarstartcoll` command is used just to drive instance logging and all other parameters related to the summary data are ignored.

- **d detail_level**
  This flag (settings 0|1|2|3) specifies that collected data must be stored on the database. `detail_level` specifies which metrics are to be stored. If –d flag is not specified, data is still sent to the database although `detail_level` is not passed, `detail_level` is defaulted to 1. –d is specified as 1, 2 and 3 corresponding to Minimum, Medium and Maximum. To prevent data being stored in the database, you must specify –d0.

- **r run_interval**
  Accepted only for applications associated to a VUS script. Specifies the interval of time (minutes) between two executions of the virtual user script. If accepted, this flag must be specified. Valid values are greater than one minute and not greater than one day. A script can be considered to have three sections, an initiation part, the run part and the end part. The `run_interval` is the time between commencement of the run part and commencement of the run part for the next script execution.

**Note:** –r is not allowed for ARM instrumented applications.

- **p parameters_list { }**

  The `parameters_list. {value1,value2,value3...}` is a free form string for extended data provider. Parameters are passed to the collector in the order they are inserted. This flag must be specified, if the passed `app_name` is a VUS script that takes parameters. In this case the syntax is –p {name=value, name=value}
Note: –p is not allowed for ARM instrumented applications.

–x file_name
The contents of this file are used in the same way as parameters specified by the –p flag.

Note: –x is only allowed for EPP applications.

–n
Controls data logging; it is a two part option separated by a comma. The first part defines the level of instance data you want logged at the endpoint and the second, optional part, defines the level to be uploaded at aggregation time.

Note: The upload content is dependant on the first setting, for instance if a particular type is not logged at the endpoint, you cannot pass that data on to the database.

Possible values for the flag option are:

- ALL
- CORR (correlators)
- NODATA

The last value NODATA, is only significant when present in the second section of the parameter. The default value is NODATA, NODATA.

endpoint_name
Specifies the endpoint name where the collection should take place.

Authorization
Senior.

Examples

1. This example sets new collection information in the period 5th July 1999 to 6th August 1999 for application vusScript and the filter setting separates out only transaction tx of the application for data collection. Bucket limits are set at 52, 60, 100 and 1500. Data collection is to take place on Mondays between 10am and 4pm and on Thursdays between 9.30am and 6.15pm. Collection interval is every 10 minutes on the transaction which will be run every five minutes. Database detail level is to be maximum. The input parameters for the script is list 1 (one). These new collection settings apply only to endpoint LCF98.

   wmarstartcoll –a vusScript –t tx –b 52,60,100,1500 –s 1999-07-05 –e 1999-08-06 –c "rule1=Monday[10:30-16:00]; rule2=Thursday[09:30-18:15]" –i 10 –r 5 –d 3 –p pl LCF98

2. This command start an the application application_name and all instance data and metrics are logging locally and only correlated data and metrics are send to the database. The possible value for the parameter -n are : ALL, CORR, NODATA (lower or upper case)

   wmarstartcoll -a application_name -n ALL,CORR tivoli_ep

See Also

wmarstopcoll, wmarlseng
**wmarstarteng**

Starts the engine on the specified endpoint.

**Syntax**

```
wmarstarteng endpoint_name.
```

**Description**

The `wmarstarteng` command starts or restarts the engine on the specified endpoint. The configuration, that was saved when the engine was last running, is restored.

**Options**

*endpoint_name*

Specifies the endpoint where you want the engine to be started.

**Authorization**

Senior.

**Examples**

This example starts the engine on endpoint LCFNT.

```
wmarstarteng LCFNT
```

**See Also**

`wmarstopeng`
**wmarstopcoll**

Stops a data collection at given endpoint.

**Syntax**

```
  wmarstopcoll -a app_name endpoint_name
```

**Description**

Stops a data collection at for an application `app_name` at given `endpoint_name`. If the collection is for a simulation, then the simulation is also stopped.

**Options**

- `-a app_name`
  
  The name of the application on which a collection is to stop. If `application_name` has been registered as a virtual application (the `-f` flag has been passed to the `wmarregapp` command during registration) the command also stops execution of the VUS script.

- `endpoint_name`
  
  Specifies the node or endpoint name, where collection is to be stopped.

**Authorization**

Senior.

**Examples**

1. This example stops collections at endpoint LCFNT for ARM instrumented application `armApplication`.
   ```bash
   wmarstopcoll -a armApplication LCFNT
   ```

2. This example stops collections at endpoint LCF98 for application `vusScript`.
   ```bash
   wmarstopcoll -a vusScript LCF98
   ```

**See Also**

`wmarstartcoll`
wmarstopeng

Stops the engine at the given endpoint.

Syntax

wmarstopeng endpoint_name

Description

The wmarstopeng command stops the engine at the given endpoint. The current configuration is saved. Any active collection is restored when the engine is restarted.

Note: If you want to restart the engine with a “clean” configuration, you should issue the wmarcleareng command before using wmarstopeng to stop the engine.

Options

endpoint_name

Specifies the endpoint where the engine is to be stopped.

Authorization

Senior.

Examples

This example stops the engine at endpoint LCF98.

wmarstopeng LCF98

See Also

wmarstarteng
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 Tivoli Application Performance Management 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 are provided with alternative text so that users of the documentation with vision impairment, can better 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 the product. Consult the assistive technology product documentation for specific information about using it to access command line or graphical interfaces.

Keyboard Navigation of the User Interface

Users can access the Tivoli Application Performance Management user interface using facilities provided by the operating systems on which the product is run. For example, in Windows environments every menu can be opened by holding down the Alt key and pressing the key indicated by the underlined character in the menu name.

Information about these facilities is provided in the relevant operating system documentation.

Standard keyboard navigation shortcuts that can be used in the product GUIs or Web browser are as follows:

<table>
<thead>
<tr>
<th>Key Shortcut</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tab</td>
<td>Next field in a form</td>
</tr>
<tr>
<td>Ctrl+Tab</td>
<td>Next task button</td>
</tr>
<tr>
<td>Shift+Tab</td>
<td>Previous field in a form</td>
</tr>
<tr>
<td>Ctrl+Shift+Tab</td>
<td>Previous task button</td>
</tr>
</tbody>
</table>
Magnifying What is Displayed on the Screen

In all components of Tivoli Application Performance Management, 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.
Authorization Roles

This appendix contains tables that list the authorization roles you need to use Tivoli Application Performance Management.

To perform system administration operations from within the Tivoli environment, you must be a Tivoli administrator. Depending on what operations you are required to perform, you can have one or more of the following roles:

- super
- senior
- admin

Setting Up Profiles

The following table lists the roles required to set up profiles:

<table>
<thead>
<tr>
<th>Operation</th>
<th>Required Role</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a profile</td>
<td>senior or super</td>
</tr>
<tr>
<td>Clone a profile</td>
<td>senior or super</td>
</tr>
<tr>
<td>View a profile</td>
<td>senior, or super</td>
</tr>
<tr>
<td>Delete a profile</td>
<td>senior or super</td>
</tr>
<tr>
<td>Set-up the database</td>
<td>senior or super</td>
</tr>
</tbody>
</table>

Note: For more information on setting up the database, see Chapter 12, “Setting up the Database”.

Performing Operations

The following table lists the role required to use Tivoli desktop facilities to distribute a profile and use the product commands:

<table>
<thead>
<tr>
<th>Operation</th>
<th>Required Role</th>
</tr>
</thead>
<tbody>
<tr>
<td>Distribute a profile</td>
<td>senior, or super</td>
</tr>
<tr>
<td>Schedule profile distribution</td>
<td>admin, senior or super</td>
</tr>
</tbody>
</table>

Performing Commands

The following table lists the role required to use the product commands:
<table>
<thead>
<tr>
<th>Command</th>
<th>Required Role</th>
</tr>
</thead>
<tbody>
<tr>
<td>wmarregapp</td>
<td>senior</td>
</tr>
<tr>
<td>wmargetapp</td>
<td>admin or senior</td>
</tr>
<tr>
<td>wmarlsapp</td>
<td>admin or senior</td>
</tr>
<tr>
<td>wmarlseng</td>
<td>senior</td>
</tr>
<tr>
<td>wmarandelapp</td>
<td>senior</td>
</tr>
<tr>
<td>wmarstarteng</td>
<td>senior</td>
</tr>
<tr>
<td>wmarstartcoll</td>
<td>senior</td>
</tr>
<tr>
<td>wmarstopcoll</td>
<td>senior</td>
</tr>
<tr>
<td>wmarstopeng</td>
<td>senior</td>
</tr>
<tr>
<td>wmargetstatus</td>
<td>admin or senior</td>
</tr>
<tr>
<td>wmarcleareng</td>
<td>senior</td>
</tr>
<tr>
<td>wmarsetstatus</td>
<td>senior</td>
</tr>
<tr>
<td>wmarsenddata</td>
<td>senior</td>
</tr>
<tr>
<td>wmargetdata</td>
<td>admin or senior</td>
</tr>
</tbody>
</table>
Glossary

This glossary defines technical terms used in the documentation for Tivoli Application Performance Management.

The following cross-references are used among terms:

**Contrast with:**
This refers the reader to a term that has an opposed or substantively different meaning.

**See:**
This refers the reader to (a) a term that is the expanded form of an abbreviation or acronym, or (b) a synonym or more preferred term.

**See also:**
This refers the reader to a related term.

**Obsolescent term for:**
This indicates that the term should not be used and refers the reader to the preferred term.
access control. In computer security, the process of ensuring that the resources of a computer system can be accessed only by authorized users in authorized ways.

action. (1) An operation on a managed object, the semantics of which are defined as part of the managed object class definition. (2) A defined task that an application performs. An action modifies the properties of an object or manipulates the object in some way.

admin role. See "authorization role"

administrator. See "Tivoli administrator" on page 219

administrator collection. In a Tivoli environment, the collection for administrator objects that is generated by Tivoli Enterprise software. This container is represented by the Administrator icon on the Tivoli desktop; opening the icon provides access to information about each Tivoli administrator.

agent. (1) In Tivoli Application Performance Management, a component that reports to a data file or data repository the data generated by a test. (2) In systems management, a user that, for a particular interaction, has assumed an agent role. (3) An entity that represents one or more managed objects by (a) emitting notifications regarding the objects and (b) handling requests from managers for management operations to modify or query the objects. (4) A system that assumes an agent role.

aggregation. Collect, interpret, and sort data from various locations, such as log files, into a single report file. In most cases, results are averaged during an aggregation process.

API. See application programming interface

application. A collection of software components used to perform specific types of user-oriented work on a computer.

application programming interface (API). A software interface that enables applications to communicate with each other. An API is the set of programming language constructs or statements that can be coded in an application program to obtain the specific functions and services provided by an underlying operating system or service program.

application response measurement (ARM). An application programming interface that was developed by a group of leading technology vendors, including Tivoli Systems Inc., and that can be used to monitor the availability and performance of business transactions within and across diverse applications and systems. The monitoring is done from the perspective of the applications; therefore, it reflects the units of work that are important from the perspective of the business.

ARM. See "Application Response Measurement"

ARM-instrumented application. An application in which the source code is modified by the addition of ARM calls. These calls enable a management system such as Tivoli Application Performance Management to monitor the performance of the (modified) application.

attribute. A characteristic that identifies and describes a managed object. The characteristic can be determined, and possibly changed, through operations on the managed object.

authorization. (1) In computer security, the right granted to a user to communicate with or make use of a computer system. (2) An access right. (3) The process of granting a user either complete or restricted access to an object, resource, or function.

authorization role. In a Tivoli environment, a role assigned to Tivoli administrators to enable them to perform their assigned systems management tasks. A role may be granted over the entire Tivoli management region or over a specific set of resources, such as those contained in a policy region. Examples of authorization roles include: super, senior, admin and user.

background process. (1) A process that does not require operator intervention but can be run by the computer while the workstation is used to do other work. (2) In the AIX operating system, a mode of program execution in which the shell does not wait for program completion before prompting the user for another command.

bash. Bourne-again shell. A portable, command-line interface and script interpreter that is compatible with the UNIX Bourne and Korn shells and includes some features of the UNIX C shell.

browse. To look at records in a file, or to navigate a directory structure looking for, and locating, files.

buffer. (1) A routine or storage used to compensate for a difference in rate of flow of data, or time of occurrence of events, when transferring data from one device to another. (2) To allocate and schedule the use of buffers. (3) A portion of storage used to hold input or output data temporarily.

business system. A group of diverse but interdependent applications and other system resources that interact to accomplish specific business functions.

call. (1) The action of bringing a computer program, a routine, or a subroutine into effect, usually by specifying the entry conditions and jumping to an entry point. (2) To transfer control to a procedure, program, routine, or subroutine.

check box. A square box with associated text that represents a choice. When a user selects the choice, the check box is filled to indicate that the choice is selected. The user can clear the check box by selecting the choice again, thereby deselecting the choice.

checkpoint. (1) Information about the status of a program’s execution or the status of a data transfer that is recorded to enable the program or the data transfer to be restarted if it is ever interrupted. (2) The time at which such information is recorded. (3) To record such information.
class. (1) In object-oriented design or programming, a model or template that can be instantiated to create objects with a common definition and therefore, common properties, operations, and behavior. An object is an instance of a class. (2) In the operating system, pertaining to the I/O characteristics of a device. System devices are classified as block or character devices.

CLI. See "command line interface".

client. A computer system or process that requests a service of another computer system or process that is typically referred to as a server. Multiple clients may share access to a common server.

client daemon. An AIX process that performs the client’s operations.

client/server. The model of interaction in distributed data processing in which a program at one site sends a request to a program at another site and awaits a response. The requesting program is called a client; the answering program is called a server.

cloning. In a Tivoli environment, an operation that enables a Tivoli administrator to replicate profiles. This capability simplifies the task of creating multiple profiles with similar properties.

collection. A container that groups objects on a Tivoli desktop, thus providing the Tivoli administrator with a single view of related resources. Either the Tivoli Management Framework or a Tivoli administrator can create a collection. The contents of a collection are referred to as its members. Examples of collections include the administrator collection and the generic collection; the administrator collection is an example of a collection generated by the Tivoli Management Framework.

command. (1) A request for the performance of an operation or the execution of a particular program. (2) A sequence of characters that is submitted to cause an action. A command contains a verb and an object.

command line interface (CLI). A type of computer interface in which the input command is a string of text characters.

configuration. (1) The manner in which the hardware and software of an information processing system are organized and interconnected. (2) The devices and programs that make up a system, subsystem, or network.

configuration file. A file that specifies the characteristics of a system device or network.

container. A visual user-interface component that holds objects.

control program. A computer program designed to schedule and to supervise the execution of programs of a computer system.

console event. In a Tivoli environment, an event sent to the Tivoli Enterprise Console product.

current directory. The file directory to which the operating system is currently pointing.

D

daemon. A program that runs unattended to perform a standard service. Some daemons are triggered automatically to perform their task; others operate periodically.

database. (1) A collection of data with a given structure for accepting, storing, and providing, on demand, data for multiple users. (2) A collection of interrelated data organized according to a database schema to serve one or more applications. (3) A collection of data fundamental to a system. (4) A collection of data fundamental to an enterprise.

data type. One of the three elements, which also include display type and resource type, that are used to describe the organization of panels. Data types include alerts, events, and statistics.

desktop. See "Tivoli desktop" on page 219.

directory. In a hierarchical file system, a container for files or other directories. See "path" on page 217.

domain. That part of a computer network in which the data processing resources are under common control.

domain name. In the Internet suite of protocols, a name of a host system. A domain name consists of a sequence of subnames separated by a delimiter character. For example, if the fully qualified domain name (FQDN) of a host system is ralvm7.vnet.ibm.com, each of the following is a domain name:

- ralvm7.vnet.ibm.com
- vnet.ibm.com
- ibm.com

drag and drop. To directly manipulate an object by moving it and placing it somewhere else using a pointing device (such as a mouse).

E

e-business. Either (a) the transaction of business over an electronic medium such as the Internet or (b) any organization (for example, commercial, industrial, nonprofit, educational, or governmental) that transacts its business over an electronic medium such as the Internet. An e-business combines the resources of traditional information systems with the vast reach of an electronic medium such as the Internet (including the World Wide Web, intranets, and extranets); it connects critical business constituencies—customers, employees, and suppliers. The key to becoming an e-business is building a transaction-based Web site in which all core business processes (especially all processes that require a dynamic and interactive flow of information) are put online to improve service, cut costs, and sell products.
e-commerce. The subset of e-business that involves the exchange of money for goods or services purchased over an electronic medium such as the Internet.

endpoint. (1) In a Tivoli environment, a Tivoli client that is the ultimate recipient for any type of Tivoli operation. (2) A Tivoli service that runs on multiple operating systems and performs Tivoli operations on those systems, thereby enabling the Tivoli Management Framework to manage the systems as Tivoli clients.

endpoint list. In a Tivoli environment, a list of all endpoint clients in the Tivoli Management Region with their assigned gateways.

endpoint manager. In a Tivoli environment, a service that runs on the Tivoli server, assigns endpoint clients to gateways, and maintains the endpoint list.

End-to-end probe platform (EPP). In Tivoli Application Performance Management, a software application that simulates transactions being monitored.

gateway. (1) A functional unit that interconnects two computer networks with different network architectures. A gateway connects networks or systems of different architectures. A bridge interconnects networks or systems with the same or similar architectures. (2) A functional unit that connects two networks or subnetworks having different characteristics, such as different protocols or different policies concerning security or transmission priority. (3) In a Tivoli environment, software running on a managed node that provides all communication services between a group of endpoints and the rest of the Tivoli environment. This gateway includes the multiplexed distribution (MDist) function, enabling it to act as the fan out point for distributions to many endpoints.

graphical user interface (GUI). A type of computer interface consisting of a visual metaphor of a real-world scene, often of a desktop. Within that scene are icons, representing actual objects, that the user can access and manipulate with a pointing device.

G

GUI. See “graphical user interface”

H

heartbeat. In software products, a signal that one entity sends to another to convey that it is still active.

hook. A location in a computer program where an instruction is inserted for invoking a particular function.

host. (1) A computer that is connected to a network (such as the Internet or an SNA network) and provides an access point to that network. Also, depending on the environment, the host may provide centralized control of the network. The host can be a client, a server, or both a client and a server simultaneously.

field. (1) An identifiable area in a window. Examples of fields are: an entry field, into which a user can type or place text, and a field of radio button choices, from which a user can select one choice. (2) The smallest identifiable part of a record. (3) A field is characterized by a field name. A field can contain data only when it is associated with an object.

file name substitution. The process in which the shell substitutes an alphabetically sorted list of file names in the place of a pattern. The shell recognizes a pattern (as opposed to a file name) by the occurrence of a word (character string) with either of the following characteristics:
- The word contains any of these characters: *, ?, [ ], or {.
- The word begins with this character: ~.

file package. In Tivoli Software Distribution, a profile. The file package describes which files and directories to distribute and how to distribute them.

filter. (1) A device or program that separates data, signals, or material in accordance with specified criteria. (2) A function that limits the data recorded in the database or displayed at the terminal.

full path name. The complete path name from root to the end directory. In contrast a part path name only contains the last few related directories which may be childs of a completely different directory structure.
host name. In the Internet suite of protocols, the name given to a machine. Sometimes, “host name” is used to mean identity of a controlling machine. Other times, it is used to mean the most specific subname of a fully qualified domain name. For example, if ralvm7.vnet.ibm.com is the fully qualified domain name, either of the following may be considered the host name:

- ralvm7.vnet.ibm.com
- ralvm7

host processor. (1) A processor that controls all or part of a user application network. (2) In a network, the processing unit in which the data communication access method resides.

indicator. In Tivoli Distributed Monitoring, an icon on the Tivoli desktop that graphically displays the status of a monitor that has been associated with it. The icon resembles a thermometer, which the Tivoli administrator can read to determine the status of the monitor.

instrument. In application or system software, to use monitoring functions to provide performance and other information to a management system.

instrumentation. In application or system software, (a) monitoring functions that provide performance and other information to a management system (b) the use of monitoring functions to provide performance and other information to a management system.

Internet Protocol (IP). In the Internet suite of protocols, a connection less protocol that routes data through a network or interconnected networks. IP acts as an intermediary between the higher protocol layers and the physical network. However, this protocol does not provide error recovery and flow control and does not guarantee the reliability of the physical network.

intranet. A private network that integrates Internet standards and applications (such as Web browsers) with an organization's existing computer networking infrastructure.

IT. Information technology.

job. (1) A unit of work defined by a user that is to be accomplished by a computer. Loosely, the term job is sometimes used to refer to a representation of a job. This representation may include a set of computer programs, files, and control statements to the operating system. (2) A Printing Systems Manager (PSM) object that represents a request to print one or more documents in a single printing session. (3) In a Tivoli environment, a resource consisting of a task and its pre configured execution parameters. Among other things, the execution parameters specify the set of hosts on which the job is to execute.

LoadRunner. A Mercury toolkit module that resides on an endpoint and enables VuGen (See page 220) scripts to run. Loadrunner is designed to work with client-server systems. It can incrementally increase the load on a network to identify possible network and server bottlenecks.

local area network (LAN). (1) A computer network located on a user’s premises within a limited geographical area. Communication within a local area network is not subject to external regulations; however, communication across the LAN boundary may be subject to some form of regulation. (2) A network in which a set of devices are connected to one another for communication and that can be connected to a larger network.

local distribution. In a Tivoli environment, a distribution to target machines in the same Tivoli Management Region as the source machine.

lock. The means by which integrity of data is ensured by preventing more than one user from accessing or changing the same data or object at the same time.

logged-on operator. An operator station task that requires a terminal and a logged-on user.

managed node. (1) In a Tivoli environment, any managed resource on which the Tivoli Management Framework is installed. (2) In Internet communications, a workstation, server, or router that contains a network management agent. In the Internet Protocol (IP), the managed node usually contains a Simple Network Management Protocol (SNMP) agent.

managed object. (1) A component of a system that can be managed by a management application. (2) The systems management view of a resource that can be managed through the use of systems management protocols.

managed resource. In a Tivoli environment, any hardware or software entity (machine, service, system, or facility) that is represented by a database object and an icon on the Tivoli desktop. Managed resources must be a supported resource type in a policy region and are subject to a set of rules. Managed resources include, but are not limited to, managed nodes, task libraries, monitors, profiles, and bulletin boards.

management region. In Tivoli NetView®, the set of managed objects on a particular map that defines the extent of the network that is being actively managed. The management region may vary across maps.

manager. (1) In systems management, a user that, for a particular interaction, has assumed a manager role. (2) An entity that monitors or controls one or more managed objects by (a) receiving notifications regarding the objects and (b) requesting management operations to modify or query the objects. (3) A system that assumes a manager role.
monitoring collection. In Tivoli Distributed Monitoring, a collection of predefined monitors. Several monitoring collections are packaged with Tivoli Distributed Monitoring, but Tivoli administrators can use custom-developed and third-party monitoring collections as well.

monitor. (1) A device that observes and records selected activities within a data processing system for analysis. Possible uses are to indicate significant departure from the norm, or to determine levels of utilization of particular functional units. (2) Software or hardware that observes, supervises, controls, or verifies operations of a system. (3) Software that monitors specific applications or the systems on which the applications rely. Monitors typically monitor information such as available disk space or application errors and compare the information to defined thresholds. When thresholds are exceeded, either system or network administrators can be notified, or an automated response can be performed.

monitoring collection. In Tivoli Distributed Monitoring, a collection of predefined monitors. Several monitoring collections are packaged with Tivoli Distributed Monitoring, but Tivoli administrators can use custom-developed and third-party monitoring collections as well.

man page. In UNIX systems, one page of online documentation. “Man page” is an abbreviation for “manual page.” Each UNIX command, utility, and library function has an associated man page that can be viewed by entering this command: man command name.

map. In Tivoli NetView, a database represented by a set of related submaps that provide a graphical and hierarchical presentation of a network and its systems.

menu bar. (1) The area near the top of a window, below the title bar and above the rest of the window, that contains choices that provide access to other menus. (2) In the AIX operating system, a rectangular area at the top of the client area of a window that contains the titles of the standard pull-down menus for that application.

mid-level manager (MLM). In Tivoli Application Performance Management, the component on the gateway that performs certain systems and network management tasks (for example, polling, status monitoring, data upload) between the region manager and endpoints, and between endpoints and the product database.

monitor. (1) A device that observes and records selected activities within a data processing system for analysis. Possible uses are to indicate significant departure from the norm, or to determine levels of utilization of particular functional units. (2) Software or hardware that observes, supervises, controls, or verifies operations of a system. (3) Software that monitors specific applications or the systems on which the applications rely. Monitors typically monitor information such as available disk space or application errors and compare the information to defined thresholds. When thresholds are exceeded, either system or network administrators can be notified, or an automated response can be performed.

monitoring collection. In Tivoli Distributed Monitoring, a collection of predefined monitors. Several monitoring collections are packaged with Tivoli Distributed Monitoring, but Tivoli administrators can use custom-developed and third-party monitoring collections as well.

N

navigate. (1) In directory structure, using the mouse to move through the directory tree structure to locate a required file or application. See “path” on page 217. (2) In the NetView Graphic Monitor Facility, to move between levels in the view hierarchy.

network class. In Tivoli NetView, an object class used for symbols that represent compound objects that may contain objects such as hosts and network devices.

network computing. The use of a scalable distributed computing infrastructure that encompasses the key elements of today’s networking technologies, such as systems and network management; the Internet and intranets; clients and servers; application programs; databases; transaction processing; and various operating systems and communication protocols.

network file system (NFS). A protocol developed by Sun Microsystems, Incorporated, that allows any host in a network to mount another host’s file directories. Once mounted, the file directory appears to reside on the local host.

notice. In a Tivoli environment, a message generated by a systems management operation that contains information about an event or the status of an application. Notices are stored in notice groups. See “notice group” below.

notice group. In a Tivoli environment, an application- or operation-specific container that stores and displays notices pertaining to specific Tivoli functions. The Tivoli bulletin board is comprised of notice groups. A Tivoli administrator can subscribe to one or more notice groups; the administrator’s bulletin board contains only the notices that reside in a notice group to which the administrator is subscribed.

notification. (1) An unscheduled, spontaneously generated report of an event that has occurred. (2) In systems management, information emitted by a managed object relating to an event that has occurred within the managed object, such as a threshold violation or a change in configuration status.

O

object. (1) In object-oriented design or programming, a concrete realization of a class that consists of data and the operations associated with that data. (2) An item that a user can manipulate as a single unit to perform a task. An object can appear as text, an icon, or both.

open database connectivity (ODBC). A standard application programming interface (API) for accessing data in both relational and non relational database management systems. Using this API, database applications can access data stored in database management systems on a variety of computers even if each database management system uses a different data storage format and programming interface. ODBC is based on the call level interface (CLI) specification of the X/Open SQL Access Group and was developed by Digital Equipment Corporation (DEC), Lotus, Microsoft, and Sybase.

operation. In object-oriented design or programming, a service that can be requested at the boundary of an object. Operations include modifying an object or disclosing information about an object.

operations and administration. The Tivoli management discipline that addresses the automation of activities that ensure the operational integrity and reliability of a network computing system.

operator. A person or a program that manages activities that are controlled by a specific computer program.

operator profile. A specification of the resources and activities over which a network operator has control. The profile is stored in a file that is activated when the operator logs on.
oserv. The name of the object request broker used by the Tivoli environment. Oserv runs on the Tivoli management region server and each region client.

P

parameter. (1) A variable that is given a constant value for a specified application and that may denote the application. (2) In Common User Access® (CUA®) architecture, a variable used in conjunction with a command to affect its result. (3) An item in a menu for which the user specifies a value or for which the system provides a value when the menu is interpreted. (4) Data passed to a program or procedure by a user or another program, namely as an operand in a language statement, as an item in a menu, or as a shared data structure.

patch. A code change that is sent to the owners of a software product license after the release of a product. The licensees can then apply this code change to correct a reported problem.

path. (1) A list of one or more directory names and an object name (such as the name of a file) that are separated by an operating system-specific character, such as the slash (/) in UNIX operating systems, the backslash (\) in Windows operating systems, and the semicolon (;) in OS/2® operating systems. The directory names detail the path to follow, in left-to-right order, to locate the object within the file system. This concept of path is also known as the “pathname.” (2) A list of directory names, usually separated by a colon (:), that are to be searched (in left-to-right order) to locate an object. This concept of path is also known as the “search path.” (3)

pathname. The list of connected directories that lead to a specific file location.

PC managed node. In a Tivoli environment, an object that represents a client PC. The Tivoli Management Framework can communicate with the client PC only if the PC agent is installed on the PC. Client PCs are most often referred to as PC managed nodes.

PDF. See "Portable Document Format".

platform. An ambiguous term that may refer to the hardware, the operating system, or a combination of the hardware and the operating system on which software programs run.

policy. In a Tivoli environment, a set of rules that are applied to managed resources. A specific rule in a policy is referred to as a “policy method.”

policy region. In a Tivoli environment, a group of managed resources that share one or more common policies. Tivoli administrators use policy regions to model the management and organizational structure of a network computing environment. The administrators can group similar resources, define access to and control the resources, and associate rules for governing the resources. The policy region contains resource types and the list of resources to be managed. A policy region is represented on the Tivoli desktop by an icon that resembles a capitol building (dome icon). When a Tivoli Management Region is created, a policy region with the same name is also created. In this case, the managed node has only one policy region. However, in most cases, a Tivoli administrator creates other policy regions and subregions to represent the organization of the managed region. A managed region addresses the physical connectivity of resources whereas a policy region addresses the logical organization of resources.

policy subregion. In a Tivoli environment, a policy region created or residing in another policy region. When a policy subregion is created, it initially uses the resource and policy properties of the parent policy region. The Tivoli administrator can later change or customize these properties to reflect the specific needs and differences of the subregion.

polling. (1) On a multipoint connection or a point-to-point connection, the process whereby data stations are invited, one at a time, to transmit. (2) Interrogation of devices for such purposes as to avoid contention, to determine operational status, or to determine readiness to send or receive data. (3) In network management, the process by which a manager interrogates one or more managed nodes at regular intervals.

populate. In a Tivoli environment, to fill a profile with information that is to be distributed to the subscribing managed resources.

portable document format (PDF). A standard specified by Adobe Systems, Incorporated, for the electronic distribution of documents. PDF files are compact; can be distributed globally via e-mail, the Web, intranets, or CD-ROM; and can be viewed with the Acrobat Reader, which is software from Adobe Systems that can be downloaded at no cost from the Adobe Systems home page.

profile. In a Tivoli environment, a container for application-specific information about a particular type of resource. A Tivoli application specifies the template for its profiles; the template includes information about the resources that can be managed by that Tivoli application. A profile is created in the context of a profile manager; the profile manager links a profile to the Tivoli resource (for example, a managed node) that uses the information contained in the profile. A profile does not have any direct subscribers.

profile manager. In a Tivoli environment, a container for profiles that links the profiles to a set of resources, called “subscribers.” A profile manager can contain (a) profiles of multiple types or (b) multiple profiles of the same type. Tivoli administrators use profile managers to organize and distribute profiles. A profile manager is created in the context of a policy region and is a managed resource in a policy region. See "policy region".

prototype profile. In a Tivoli environment, a model profile from which a Tivoli administrator can create other profiles, often by cloning the prototype profile.

push. A network operation that sends information to resources.
query. In a Tivoli environment, a combination of statements that are used to search the configuration repository for systems that meet certain criteria.

RDBMS. See “relational database management system”.

RDBMS interface module (RIM). In the Tivoli Management Framework, the module in the distributed object database that contains information about the installation of the relational database management system (RDBMS).

registered name. In a Tivoli environment, the name by which a particular resource is registered with the name registry when it is created.

relation. (1) In a relational database, a set of entity occurrences that have the same attributes. (2) The comparison of two expressions to see if the value of one is equal to, less than, or greater than the value of the other. (3) In a relational database, a table that identifies entities and their attributes.

relational database. A database in which the data are organized and accessed according to relations.

relational database management system (RDBMS). A collection of hardware and software that organizes and provides access to a relational database.

relative path. A path that begins with the working directory.

remote distribution. In a Tivoli environment, a distribution to target machines in a connected Tivoli Management Region.

remote host. Any host on a network except the host at which a particular operator is working.

resource. Any facility of a computing system or operating system required by a job or task, and including main storage, input/output devices, the processing unit, data sets, and control or processing programs.

resource role. In a Tivoli environment, the role an administrator has over specific resources in the local Tivoli management region (region) and any connected region (for example, policy regions or the Administrator collection).

response time. (1) The elapsed time between the end of an inquiry or demand on a computer system and the beginning of the response; for example, the length of time between an indication of the end of an inquiry and the display of the first character of the response at a user terminal. (2) For response time monitoring, the time from the activation of a transaction until a response is received, according to the response time definition coded in the performance class.

RIM. See “RDBMS Interface Module”.

RIM repository. The database storage set aside for RIM objects. See “RIM”.

root directory. The highest level directory in a hierarchical file system.

root user. In the UNIX operating system, a user who has superuser authority.

rule. In the Tivoli Enterprise Console, a set of one or more logical statements that enable the event server to recognize relationships among events (event correlation) and to execute automated responses accordingly.

rule base. In the Tivoli Enterprise Console, a set of rules and the event class definitions for which the rules are written. The Tivoli Enterprise Console uses the rule base in managing events. An organization can create many rule bases, with each rule base fulfilling a different set of needs for network computing management.

S

scheduler. A computer program designed to perform functions such as scheduling, initiation, and termination of jobs.

schema. The set of statements, expressed in a data definition language, that completely describe the structure of a database.

script. A computer program that is interpreted.

senior role. See “authorization role” on page 212.

server. A functional unit that provides services to one or more clients over a network. Examples include a file server, a print server, and a mail server.

server workstation. In the NetView Graphic Monitor Facility, a workstation with the graphic data server. This workstation uses the graphic monitor and the view administrator for administrative functions. The server workstation sends status information to client workstations.

service level reporter (SLR). A licensed program that generates management reports from data sets such as System Management Facility (SMF) files.

shell. A software interface between a user and the operating system of a computer. Shell programs interpret commands and user interactions on devices such as keyboards, pointing devices, and touch-sensitive screens and communicate them to the operating system. Shells simplify user interactions by eliminating the user’s concern with operating system requirements. A computer may have several layers of shells for various levels of user interaction.

shell prompt. In the UNIX operating system, the character string on the command line indicating that the system can accept a command (typically the $ character).
shell script. In the UNIX operating system, a series of commands, combined in a file, that carry out a particular function when the file is run or when the file is specified as a value to the SH command.

software installation service (SIS). A Tivoli product that provides an easy-to-use, efficient interface for installing Tivoli Enterprise software. SIS uses Tivoli’s MDist technology and provides automated checking for prerequisite software, a reusable repository of installation images, and both graphical and command line interfaces for deploying Tivoli products to a large number of computers.

source host. In Tivoli Software Distribution, the managed node on which the files and directories referenced in a file package reside.

structured query language (SQL). A programming language that is used to define and manipulate data in a relational database.

subagent. In the Simple Network Management Protocol (SNMP), something that provides an extension to the utility provided by the SNMP agent.

subscriber. In a Tivoli environment, a Tivoli client, a profile manager, or any endpoint type (for example, a PC managed node or a proxy endpoint) that is subscribed to a profile manager. Although profiles are distributed to a subscriber, the subscriber may or may not be the final destination of the profile distribution.

subscription. In a Tivoli environment, the process of identifying the subscribers to which profiles will be distributed.

subscription list. In a Tivoli environment, a list that identifies the subscribers to a profile manager. Including a profile manager on a subscription list (in effect, a list within a list) is a way of subscribing several resources simultaneously rather than adding each one individually. In Tivoli Plus modules, a profile manager functions as a subscription list.

super role. See “authorization role” on page 212.

symbol. In Tivoli NetView, a picture or an icon on a submap that represents an object (a network resource or an application). Each symbol belongs to a class, represented by the symbol’s shape, and to a subclass, represented by the design within the shape. The symbol reflects characteristics of the object it represents, such as its status; it also has characteristics of its own, such as behavior.

system configuration. A process that specifies the devices and programs that form a particular data processing system.

task. (1) In a multiprogramming or multiprocessor environment, one or more sequences of instructions treated by a control program as an element of work to be accomplished by a computer. (2) In a Tivoli environment, the definition of an action that must be routinely performed on various managed nodes throughout the network. A task defines the executables to be run when the task is executed, the authorization role required to execute the task, and the user or group name under which the task will execute.

task library. In a Tivoli environment, a container in which a Tivoli administrator can create and store tasks and jobs.


threshold. In software products, a value that defines a limit for a monitored condition. The monitored condition, the significance of the limit, and the particular software product’s response when the monitored condition reaches the specified threshold vary widely according to product.

Tivoli administrator. In a Tivoli environment, a system administrator who has been authorized to perform systems management tasks and manage policy regions in one or more networks. Each Tivoli administrator is represented by an icon on the Tivoli desktop.

Tivoli application performance management. A Tivoli product that gathers comparative data in the form of the response times of client-server applications, while they are carrying out defined tasks. Analysis of the accumulated data enables you to measure the service level of any application, and thus enable you to tune application and network management more efficiently.

Tivoli Decision Support. A Tivoli product that consolidates, transforms, and presents IT data in many different views, enabling an enterprise to gain insight into patterns and relationships among the data and to make critical business decisions based on this data.

Tivoli desktop. In a Tivoli environment, the desktop that system administrators use to manage their network computing environment.

Tivoli Distributed Monitoring. A Tivoli product that monitors system resources, initiates any necessary corrective actions, and informs system administrators of potential problems. Tivoli Distributed Monitoring consists of a group of monitors that are installed on each managed node that is to be monitored. It resolves some events on its own and may send others to the Tivoli Enterprise Console.

Tivoli enterprise console. A Tivoli product that collects, processes, and automatically initiates corrective actions for system, application, network, and database events; it is the central control point for events from all sources. The Tivoli Enterprise Console provides a centralized, global view of the network computing environment; it uses distributed event monitors to collect information, a central event server to process information, and distributed event consoles to present information to system administrators.

Tivoli environment. The Tivoli applications, based upon the Tivoli Management Framework, that are installed at a specific customer location and that address network computing management issues across many platforms. In a Tivoli environment, a system administrator can distribute software,
manage user configurations, change access privileges, automate operations, monitor resources, and schedule jobs.

**Tivoli management framework.** The Tivoli base software that is required to run the applications in the Tivoli product suite. This software infrastructure enables the integration of systems management applications from Tivoli Systems Inc. and the Tivoli Partners. The Tivoli Management Framework includes the following: object request broker (oserv), distributed object database, basic administration functions, basic application services, and basic desktop services (such as the graphical user interface). In a Tivoli environment, the Tivoli Management Framework is installed on every client and server; however, the Tivoli management region server is the only server that holds the full object database.

**Tivoli management gateway.** In a Tivoli environment, a system that enables bidirectional communication with Tivoli management agents.

**Tivoli management region (region).** In a Tivoli environment, a Tivoli server and the set of clients that it serves. An organization can have more than one region. A region addresses the physical connectivity of resources whereas a policy region addresses the logical organization of resources.

**Tivoli manager.** Tivoli management software that manages specific vendor systems, networks, applications, or databases.

**Tivoli management region role.** In a Tivoli environment, the role an administrator has in the local Tivoli management region (region) and any connected region. The region role propagates the assigned authorization level to all resources in the region. For example, if a Tivoli administrator has a senior role in a region, then the administrator has the senior role over every resource in that region.

**Tivoli security management.** Tivoli Enterprise software that enables the consistent definition, implementation, and enforcement of security policy in a network computing environment.

**Tivoli server.** The server that holds or references the complete set of Tivoli software, including the full object database. Also referred to as the region server.

**toggle button.** A graphical object that simulates a toggle switch; it switches sequentially from one optional state to another.

**trace.** A record of the execution of a computer program. It exhibits the sequences in which the instructions were executed.

**transaction.** A specific set of input data that triggers execution of a specific process or job; a message destined for an application program.

**Transmission Control Protocol/Internet Protocol (TCP/IP).** A set of communications protocols that support peer-to-peer connectivity functions for both local and wide area networks.

**trap.** In the Simple Network Management Protocol (SNMP), a message sent by a managed node (agent function) to a management station to report an exception condition.

**U**

**upcall.** In a Tivoli environment, a method invocation from an endpoint “up” to the gateway.

**user login.** A single user login name with a user account on a specified operating system.

**user profile.** In Tivoli User Administration, a profile that is used to manage user accounts, including account information, home directories, startup files, and group membership.

**user role.** See "authorization role" on page 212.

**V**

**validation.** The checking of data for correctness or for compliance with applicable standards, rules, and conventions.

**validation policy.** In a Tivoli environment, policy that ensures that all resources in a policy region comply with the region’s established policy. Validation policy prevents Tivoli administrators from creating or modifying resources that do not conform to the policy of the policy region in which the resources were created.

**variable.** (1) In programming languages, a language object that may take different values, one at a time. The values of a variable are usually restricted to a certain data type. (2) A quantity that can assume any of a given set of values. (3) A name used to represent a data item whose value can be changed while the program is running. (4) In the Simple Network Management Protocol (SNMP), a match of an object instance name with an associated value. (5) In the NetView command list language, a character string beginning with “&” that is coded in a command list and is assigned a value during execution of the command list.

**virtual user script.** Simulated applications or partial applications that enable you to monitor the performance of your distributed applications, by emulating real end-users performing typical business processes. The actions that a Vuser performs are described in a Vuser script.

**VuGen.** The primary tool for creating and developing Vuser scripts. It simulates the actions of real end users by performing typical business processes or transactions.

**Vuser.** See "virtual user script".

**W**

**wildcard character.** A character that is substituted in a search string when that character or series of characters is unknown and therefore the search must be for all instances of combinations accompanying known characters within the search string wildcards. where:
* represents any series of accompanying characters that do not match.

? represents a single character that need not match.

**wizard.** A dialog within an application that uses step-by-step instructions to guide a user through a specific task.

**working directory.** The directory that is currently in use by an operating system or application. If no path is specified, this is the directory to which data is written, from which data is deleted, or in which data is searched.

**Z**

**zoom.** In a user interface, to progressively increase or decrease the size of a part of an image on a screen or in a window.
## Index

### A
- accessibility 207
- aggregation 11, 91, 134, 136
- aggregation, forcing 138
- alternative text 207
- API 61
  - application
    - registration 194
    - registry commands 182
    - response measurement 59
    - simulation 9
- ARM
  - API calls 8, 61
  - applications 9
  - monitors 135
- ARM applications, instrumenting 61
- arm_end command 63
- arm_getid command 62
- arm_init command 61
- arm_start command 62
- arm_stop command 62
- arm_update command 62
- ARMing functions, understanding 59
- authority, system 103
- authorization 13, 103
- availability, transaction 44
- average response time 84
- cleanup, file 134
- CLI 2, 3, 181
- client capture 4, 7, 47
- client/server transactions 65
- collection
  - changing details 94
  - period 99
- command line interface 2, 3, 181
- command line syntax 183
- command reference 181
- commands 181
  - arm_end 63
  - arm_getid 62
  - arm_init 61
  - arm_start 62
  - arm_stop 62
  - arm_update 62
  - engine 182
  - getting help 183
  - reference 181
  - registry 182
  - the product 181
  - wchkdb 141
  - wcrttrb 136
  - wcrttrim 106
  - wgetrim 108
  - wininstall 19, 24, 25, 26, 30, 32
  - wmarcleareng 184
  - wmardelapp 185
  - wmargetapp 39, 73, 74, 147, 186
  - wmargetdata 187
  - wmargetstatus 189
  - wmarlsappl 191
  - wmarlseng 193
  - wmarregapp 72, 73, 74, 194
  - wmarsetdata 196
  - wmarsetstatus 137, 138, 197
  - wmarstartcoll 83, 201
  - wmarstarteng 204
  - wmarstopcoll 205
  - wmarstopeng 34, 206
  - wsetrim 108
  - wsetrimpw 107
  - wtemp 145
  - wuninst 34
- commands, help for 183
- configuration files 7, 48
- console, Tivoli event 135
- correlated data 92
- correlation 65
- correlator data 11
- correlator records 67
- creating
  - a RIM object 105
  - filter 83
- bash shell 109
- books
  - feedback xii
  - online xii
  - ordering xii
- boundaries, bucket 201
- browser, web 97
- bucket boundaries 201
- bucket limits 84
- buckets 135, 187, 201
- building a profile 10
- button, refresh 83, 100

### C
- call, remote procedure 43
- capture, client 4, 7, 47
- changing collection details 94
- checking the product notice group 141
- child transaction 65
- cleanup, file 134
- CLI 2, 3, 181
- client capture 4, 7, 47
- client/server transactions 65
- collection
  - changing details 94
  - period 99
- command line interface 2, 3, 181
- command line syntax 183
- command reference 181
- commands 181
- command_end 63
- command_getid 62
- command_init 61
- command_start 62
- command_stop 62
- command_update 62
- command engine 182
- getting help 183
- reference 181
- registry 182
- the product 181
- command wchkdb 141
- command wcrttrb 136
- command wcrttrim 106
- command wgetrim 108
- command wininstall 19, 24, 25, 26, 30, 32
- command wmarcleareng 184
- command wmardelapp 185
- command wmargetapp 39, 73, 74, 147, 186
- command wmargetdata 187
- command wmargetstatus 189
- command wmarlsappl 191
- command wmarlseng 193
- command wmarregapp 72, 73, 74, 194
- command wmarsetdata 196
- command wmarsetstatus 137, 138, 197
- command wmarstartcoll 83, 201
- command wmarstarteng 204
- command wmarstopcoll 205
- command wmarstopeng 34, 206
- command wsetrim 108
- command wsetrimpw 107
- command wtemp 145
- command wuninst 34
- commands, help for 183
- configuration files 7, 48
- console, Tivoli event 135
- correlated data 92
- correlation 65
- correlator data 11
- correlator records 67
- creating
  - a RIM object 105
  - filter 83
creating (continued)
  profile 78
  profile manager 39
  cubes 133
  customer support xiv
  cycles 201

D
data
  analysis 138
  collection 91, 133, 138, 201, 205
  collection schedule 81
  correlator 11, 92
  instance 11, 68, 91, 133
  synchronizing collection 91
  database 5
    installation 103
    settings 91
  dataless endpoint mode 40
  decision support 11, 12, 138
  details, changing collection 94
  directory names, notation xv
  disabling and enabling an entry 95
  distributed monitoring 14, 133, 135
  distributed monitors 26
  distributing profiles 96

E
e-mail contact xiv
  editing a profile 95
  editor, rule 89
  empty profile 78
  enabling and disabling an entry 95
  endpoint 96, 99
  endpoint mode, dataless 40
  engine 2, 4, 96, 204, 206
    commands 182
  engine, the 2, 4, 96
  enterprise products, Tivoli 4
  entry, enabling and disabling an 95
  environment variables, notation xv
  EPP 4
  EPP extended parameters 92
  error messages 147
  event console, Tivoli 135

F
feedback about publications xiv
  file cleanup 134
  files, configuration 7, 48
  files, signature 7

G
gateway 2, 3
  graphical user interface 2, 3, 7
  group, notice 37, 141
  GUI 2, 3, 7
  GUI, Web 12, 21, 139

H
help for commands 183
  HTML format 208

I
identifying a message 147
  improving performance 44
  information messages 147
  installation 13
    CD-ROM 15
    disk 13
    disk content 15
    location 15
    of the database 103
    paths 16
    requirements 13
    using SIS 20, 23, 24, 26, 27, 31, 33
    using the CLI 19, 22, 23, 25, 26, 32
    using the GUI 16, 23, 26
    Web Browser GUI 20
    Web GUI 20
  installing the product 16
  instance data 11, 68, 91, 133
  instrumenting ARM applications 61
  instrumenting virtual user scripts 60
  interface, command line 2, 3, 181
  interface, graphical user 2, 3, 7
  interval, simulation 93
L
levels, service 12, 45
library, the product 4
limits, bucket 84
line interface, command 2, 3, 181
line syntax, command 183
load, work 45
LoadRunner 9
log files 142
log files, using 142
long response time 44
lr functions 59
lr functions, understanding 59
lr_end_transaction function 60
lr_start_transaction function 59, 60

M
macros 49
managed resource 40
Management Region, Tivoli 78
manager, mid-level 3
manuals
  feedback xii
  online xii
  ordering xii
MAR profiles 78
message type 147
message, identifying a 147
messages 147
  error 147
  information 147
  notation 147
  system 147
  user 147
  warning 147
mid-level manager 3, 137
migration
  using the CLI 30
migration requirements 27
MLM 3, 137
mode, dataless endpoint 40
monitoring
  distributed 14
  methods 45
  performance 6
monitoring, distributed 133, 135
monitors 2, 187
  description 135
monitors, distributed 26
multi-tier transactions 67

N
new rule 88

notation
  environment variables xv
  messages 147
  path names xv
  typeface xv
notes, release 103
notice group 37, 141
notice group, checking the 141
notices 141

O
online publications xiii
ordering publications xiv

P
parent transaction 65
passing correlators 67
path names, notation xv
performance
  analysis and reporting 12
  data 12
  improvement 44
  monitoring 1, 6
  period, collection 99
  planning 10
  preparing scripts 10
  prerequisites, installation 13
  procedure call, remote 43
product
  architecture 3
  components 3
  installation 13, 16
  interface 4
  library 4
  server 3
  utilities 9
products, Tivoli enterprise 4
profile
  building 10
  creation 78
  delete entry 96
  distribution 11, 41, 96
  editing 95
  key points 78
  manager 78
  names 80
profile, editing a 95
profile, empty 78
profiles 77, 78
profiles, distributing 96
profiles, MAR 78
publications
  feedback xii
  online xii
  ordering xii
R
reference, command 181
refresh button 83, 100
Region, Tivoli Management 78
registering
  applications 72, 73, 194
  scripts 72, 74
registration 39
release notes 103
remote procedure call 43
repository 71
requirements, software 14
resource, managed 40
response time 1, 8, 44, 63, 65, 187
response time, average 84
response time, long 44
RIM 104
  agent 104
  creation 105
  daemon 104
  host 104
role, authorization 13
rule editor 89
rule, new 88
rules, schedule 86
rules, timing 88

S
schedule 86
schedule rules 86
screen-reader 207
script 9
  parameters 93
scripts settings, virtual user 93
scripts, instrumenting virtual user 60
scripts, preparing 10
scripts, virtual user 60
sensors 47
server, the product 3
service delivery 43
service levels 12, 43, 45
setting
  engine properties 137
  mid-level manager properties 138
  up monitors 41
settings, virtual user scripts 93
signature files 7
simulation 9
  interval 93
  lr functions 59
  script 9
SIS 16, 20, 24, 26, 27, 31, 33
SLA 12, 43
slow transaction 43
software requirements 14
software requirements, upgrade patch 27
speech synthesizer 207
sub transaction 65
summary data logging 134
support for EPP 23
support, decision 12, 138
support, Tivoli decision 11
synchronizing data collection 91
syntax, command line 183
system authority 103
system messages 147

T
TDS 97
the engine 2, 4, 96
throughput 44
time, average response 84
time, long response 44
time, response 1, 8, 44, 63, 65, 187
timing rules 88
timings 77
Tivoli Application Performance Management
  configurations 2
  profiles 77
Tivoli Customer Support xiv
Tivoli decision support 11
  interface 138
Tivoli enterprise products 4
Tivoli event console 135
Tivoli Management Region 78
TMR server
  region server 104
toolkits 9
traces 142
tracing 142
transaction
  availability 44
  client/server 65
  filter 83
  parent 65
  simulation 43
  sub 65
transaction availability 44
transaction, slow 44
transactions 44, 135
troubleshooting, fault finding 141
type, message 147

U
understanding lr functions 59
uninstallation 33
uninstalling
  EPP support 35
  monitors for distributed monitoring 35
  the product 34
  Web browser GUI 35
upcalls 137
upgrade patch software requirements 27
upgrading from Version 2.0.1 27
upload 136
URLs 49
user interface, graphical 3, 7
user messages 147
user scripts settings, virtual 93
user scripts, virtual 60
user, virtual 9
using log files 142
utilities 9

V
variables, notation for xv
Version 1.1, upgrading from 27
virtual user 9
virtual user scripts 60
virtual user scripts settings 93
virtual user scripts, instrumenting 60
VuGen 10

W
warning messages 147
wchkd command 141
wcrdb command 136
wcrtrim command 106
web browser 97
Web GUI 2, 12, 21, 139
Web GUI installation 20
wgettrim command 108
winstall command 19, 24, 25, 26, 30, 32
wmarcleareng command 184
wmargetapp command 39, 73, 74, 147, 186
wmargetdata command 187
wmargetstatus command 189
wmarlsappl command 191
wmarlseng command 193
wmarregapp command 73, 194
wmarregapp command 72, 74
wmarsenddata command 196
wmarsetstatus command 137, 138, 197
wmarstartcoll command 83, 201
wmarstarteng command 204
wmarstopcoll command 205
wmarstopeng 34
wmarstopeng command 206
work load 45
wrimtrace 145
wrimtrace command 145
wsetrim command 108
wsetrimpw command 107
wtemp command 145
wuninst command 34