Unveil Your e-business Transaction Performance with IBM TMTP 5.1

Transaction correlation and breakdown for root cause analysis

Measure, monitor, and report real-time Web, 3270, and SAP

Avoid SLA breaches
Unveil Your e-business Transaction Performance with IBM TMTP 5.1

May 2003
Note: Before using this information and the product it supports, read the information in “Notices” on page xiii.

First Edition (May 2003)

This edition applies to the 5.1. version of IBM Tivoli Monitoring for Transaction Performance (product number: 5724-C02) and to all subsequent releases and modifications until otherwise indicated in new editions.

© Copyright International Business Machines Corporation 2003. All rights reserved.  
Note to U.S. Government Users Restricted Rights -- Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
# Contents

## Figures

<table>
<thead>
<tr>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>vii</td>
</tr>
</tbody>
</table>

## Tables

<table>
<thead>
<tr>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>xi</td>
</tr>
</tbody>
</table>

## Notices

<table>
<thead>
<tr>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>xiii</td>
</tr>
</tbody>
</table>

## Trademarks

<table>
<thead>
<tr>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>xiv</td>
</tr>
</tbody>
</table>

## Preface

<table>
<thead>
<tr>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>xv</td>
</tr>
</tbody>
</table>

The team that wrote this redbook. 

<table>
<thead>
<tr>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>xvi</td>
</tr>
</tbody>
</table>

Become a published author. 

<table>
<thead>
<tr>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>xvii</td>
</tr>
</tbody>
</table>

Comments welcome. 

<table>
<thead>
<tr>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>xvii</td>
</tr>
</tbody>
</table>

### Chapter 1. Business objectives and transaction performance

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1 Why have a Web presence</td>
<td>2</td>
</tr>
<tr>
<td>1.2 Why monitor the performance of transactions</td>
<td>2</td>
</tr>
<tr>
<td>1.3 Business transactions</td>
<td>5</td>
</tr>
<tr>
<td>1.4 e-business applications’ complex layers of services</td>
<td>7</td>
</tr>
<tr>
<td>1.4.1 Managing e-business applications</td>
<td>11</td>
</tr>
<tr>
<td>1.4.2 Architecting e-business application infrastructures</td>
<td>17</td>
</tr>
<tr>
<td>1.4.3 Basic products used to facilitate e-business applications</td>
<td>19</td>
</tr>
<tr>
<td>1.4.4 Managing e-business applications using Tivoli</td>
<td>22</td>
</tr>
<tr>
<td>1.5 Tivoli product structure</td>
<td>24</td>
</tr>
<tr>
<td>1.6 Managing e-business applications</td>
<td>26</td>
</tr>
<tr>
<td>1.6.1 IBM Tivoli Monitoring for Transaction Performance functions</td>
<td>27</td>
</tr>
</tbody>
</table>

### Chapter 2. TMTP Deployment

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1 TMTP architecture</td>
<td>31</td>
</tr>
<tr>
<td>2.1.1 Enterprise Transaction Performance</td>
<td>32</td>
</tr>
<tr>
<td>2.1.2 Web Transaction Performance</td>
<td>34</td>
</tr>
<tr>
<td>2.2 Product installation</td>
<td>36</td>
</tr>
<tr>
<td>2.2.1 Preparing for multiple consoles on one system</td>
<td>38</td>
</tr>
<tr>
<td>2.2.2 Multiple consoles installation outline</td>
<td>41</td>
</tr>
<tr>
<td>2.2.3 Enterprise Transaction Performance installation</td>
<td>43</td>
</tr>
<tr>
<td>2.2.4 Web Transaction Performance installation</td>
<td>47</td>
</tr>
<tr>
<td>2.2.5 Procedures for configuring TMTP console servers</td>
<td>50</td>
</tr>
<tr>
<td>2.2.6 Tivoli Internet Management Server DB2 database creation</td>
<td>58</td>
</tr>
</tbody>
</table>

### Chapter 3. Transaction breakdown: root cause analysis

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1 Transaction breakdown overview</td>
<td>65</td>
</tr>
<tr>
<td>3.1.1 IBM Tivoli Monitoring for Transaction Performance functions</td>
<td>67</td>
</tr>
</tbody>
</table>
3.1.1 Quality of Service tools ........................................... 67
3.1.2 Synthetic transaction availability tools ......................... 68
3.2 Transaction breakdown in detail .................................... 70
  3.2.1 ARM correlation .................................................... 70
  3.2.2 Capturing correlated transaction performance data .......... 72
  3.2.3 How it works: the details ....................................... 73
  3.2.4 Transaction performance data reporting ...................... 78
  3.2.5 How to get reports: the details ............................... 79
  3.2.6 What the various STI reports offer ........................... 84
3.3 Enabling transaction breakdown .................................... 86
  3.3.1 Install the ETP and WTP components ......................... 87
  3.3.2 Install WebSphere Application Server ......................... 88
  3.3.3 Configure WebSphere Application Server for ARM .......... 89
  3.3.4 Distribute ETP to the WebSphere Application Server system . 98
  3.3.5 Use STI while ARM-enabled ................................... 102

Chapter 4. Monitoring 3270 transaction performance ................. 105
4.1 3270 transaction measurement architecture ....................... 106
  4.1.1 Recording .......................................................... 106
  4.1.2 Playback ........................................................... 107
4.2 Installation and configuration ...................................... 108
4.3 Monitoring 3270 transactions ...................................... 109
  4.3.1 Recording .......................................................... 109
  4.3.2 Creating profiles for 3270 transaction monitoring .......... 114
  4.3.3 Executing the 3270 transactions ............................... 115
  4.3.4 Troubleshooting .................................................. 115

Chapter 5. TMTP and Tivoli Enterprise Data Warehouse ............... 117
5.1 Tivoli Enterprise Data Warehouse overview ......................... 118
  5.1.1 Tivoli Enterprise Data Warehouse concepts and components . 119
  5.1.2 The Monitoring process data flow ............................. 122
5.2 Setting up the TMTP Warehouse Enablement packs ............... 124
  5.2.1 Pre-installation steps ............................................. 125
  5.2.2 Installing the TMTP Warehouse Enablement packs ........... 126
  5.2.3 Post-installation steps .......................................... 133
  5.2.4 Activation .......................................................... 141

Chapter 6. Troubleshooting .............................................. 149
6.1 Installation notes ..................................................... 150
  6.1.1 Problems installing the Enterprise Transaction Performance Web
        Console ............................................................. 150
  6.1.2 Installing IBM HTTP Server for the Web Console .......... 151
  6.1.3 Installing and configuring the TAPM RIM database .......... 153
  6.1.4 Different types of distributions ............................... 154
# Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-1</td>
<td>Transaction breakdown</td>
<td>6</td>
</tr>
<tr>
<td>1-2</td>
<td>Growing infrastructural complexity</td>
<td>8</td>
</tr>
<tr>
<td>1-3</td>
<td>Layers of service</td>
<td>10</td>
</tr>
<tr>
<td>1-4</td>
<td>The ITIL Service Management disciplines</td>
<td>13</td>
</tr>
<tr>
<td>1-5</td>
<td>Key relationships between Service Management disciplines</td>
<td>16</td>
</tr>
<tr>
<td>1-6</td>
<td>A typical e-business application infrastructure</td>
<td>17</td>
</tr>
<tr>
<td>1-7</td>
<td>e-business solution-specific service layers</td>
<td>20</td>
</tr>
<tr>
<td>1-8</td>
<td>Logical view of an e-business solution</td>
<td>21</td>
</tr>
<tr>
<td>1-9</td>
<td>Typical Tivoli-managed e-business application infrastructure</td>
<td>23</td>
</tr>
<tr>
<td>1-10</td>
<td>Tivoli Management Solutions</td>
<td>24</td>
</tr>
<tr>
<td>1-11</td>
<td>Tivoli’s Performance and Availability product structure</td>
<td>25</td>
</tr>
<tr>
<td>1-12</td>
<td>e-business transactions</td>
<td>28</td>
</tr>
<tr>
<td>2-1</td>
<td>Enterprise Transaction Performance architecture</td>
<td>33</td>
</tr>
<tr>
<td>2-2</td>
<td>Web Transaction Performance overall architecture</td>
<td>34</td>
</tr>
<tr>
<td>2-3</td>
<td>Web-based consoles: default architecture</td>
<td>38</td>
</tr>
<tr>
<td>2-4</td>
<td>Web-based consoles: single-system architecture</td>
<td>40</td>
</tr>
<tr>
<td>2-5</td>
<td>Enterprise Transaction Performance components</td>
<td>44</td>
</tr>
<tr>
<td>2-6</td>
<td>Enterprise Transaction Performance Web Console installation</td>
<td>45</td>
</tr>
<tr>
<td>2-7</td>
<td>Setting installation parameters for the ETP Web Console</td>
<td>46</td>
</tr>
<tr>
<td>2-8</td>
<td>Setting the port number for the Tivoli Internet Management Server</td>
<td>48</td>
</tr>
<tr>
<td>2-9</td>
<td>TIMS database configuration</td>
<td>48</td>
</tr>
<tr>
<td>2-10</td>
<td>Enabling ARM collection in Tivoli Internet Management Server</td>
<td>49</td>
</tr>
<tr>
<td>2-11</td>
<td>Updating the ImagePath of a service</td>
<td>51</td>
</tr>
<tr>
<td>2-12</td>
<td>Changing ConfigArg values</td>
<td>52</td>
</tr>
<tr>
<td>2-13</td>
<td>Updating the IBM HTTP Server UninstallString information</td>
<td>52</td>
</tr>
<tr>
<td>2-14</td>
<td>Changing service names using the Windows Registry Editor</td>
<td>54</td>
</tr>
<tr>
<td>2-15</td>
<td>Start/Stop IBM HTTP Service shortcut properties</td>
<td>55</td>
</tr>
<tr>
<td>2-16</td>
<td>Creating the TIMS database</td>
<td>59</td>
</tr>
<tr>
<td>2-17</td>
<td>Defining a 32k bufferpool for the TIMS database</td>
<td>59</td>
</tr>
<tr>
<td>2-18</td>
<td>Creating a DB2 table space</td>
<td>60</td>
</tr>
<tr>
<td>2-19</td>
<td>Setting buffer pool usage for a tablespace</td>
<td>61</td>
</tr>
<tr>
<td>2-20</td>
<td>Granting user authorities</td>
<td>62</td>
</tr>
<tr>
<td>2-21</td>
<td>Granting tablespace authorities</td>
<td>63</td>
</tr>
<tr>
<td>3-1</td>
<td>Transaction breakdown architecture overview</td>
<td>72</td>
</tr>
<tr>
<td>3-2</td>
<td>Transaction breakdown reporting</td>
<td>79</td>
</tr>
<tr>
<td>3-3</td>
<td>Sample STI Round-trip time report</td>
<td>80</td>
</tr>
<tr>
<td>3-4</td>
<td>STI Statistics sample report</td>
<td>81</td>
</tr>
<tr>
<td>3-5</td>
<td>ARM Report link in STI Statistics report</td>
<td>81</td>
</tr>
</tbody>
</table>
3-6 Web Console login ................................................................. 82
3-7 WAS Transaction performance data displayed by the Web Console . 83
3-8 Steps for enabling WebSphere Application Server to use ARM ...... 86
3-9 Enabling PMI request metrics with ARM in WAS 5.0 ................. 91
3-10 Setting Generic JVM arguments for PMI ARM calls ................. 93
3-11 Setting up tracing in WebSphere Application Server Version 5.0 . 95
3-12 Enable PMI tracing at startup ............................................ 97
3-13 MarProfile monitoring ARM calls from WebSphere Application Server 100
3-14 Data Storage settings for MarProfile to capture data ............... 101
3-15 Setting the ARM correlator for an STI job ............................ 102
3-16 Average WebSphere Application Server transaction response times 104
4-1 3270 transaction performance measurement architecture .......... 106
4-2 Playback engine architecture ................................................. 108
4-3 3270 Project Family Settings ................................................ 110
4-4 Specifying ARM instrumentation .......................................... 111
4-5 Transaction Explorer view .................................................. 112
4-6 3270 simulation transaction code ......................................... 113
4-7 Saving the recorded transaction ............................................ 114
4-8 Registering a new Playback application .................................. 115
4-9 Debugging the 3270 transaction Visual Basic code .................... 116
5-1 A typical Tivoli Enterprise Data Warehouse environment ........... 119
5-2 Enterprise Transaction Performance data flow .......................... 122
5-3 Installation scenario ......................................................... 123
5-4 Tivoli Enterprise Data Warehouse Installation dialog ................ 128
5-5 Tivoli Enterprise Data Warehouse setup type ......................... 129
5-6 Tivoli Enterprise Data Warehouse host name verification ............ 129
5-7 Tivoli Enterprise Data Warehouse installation: DB2 configuration .. 130
5-8 Installation media path for ITM Generic ETL1 program ............... 130
5-9 Tivoli Enterprise Data Warehouse installation: Additional modules .. 131
5-10 Enterprise Transaction Performance ETL1 program installation ... 132
5-11 Tivoli Enterprise Data Warehouse installation running .............. 132
5-12 Installation Summary window ............................................. 133
5-13 Enterprise Transaction Performance ETL Source ..................... 135
5-14 APF_TAPM_Source user ID information ................................. 136
5-15 Enterprise Transaction Performance ETL target ..................... 136
5-16 APF_TWH_CDW_Target user ID information ............................ 137
5-17 Warehouse source table names immediately after installation .... 138
5-18 Warehouse source table properties ..................................... 139
5-19 Table schema and table name for TMTP Warehouse sources ....... 140
5-20 Warehouse source table names immediately after installation .... 141
5-21 Scheduling an ETL process ................................................ 143
5-22 ETL schedule configuration for one-time execution of a process ... 144
5-23 ETL schedule configuration for periodic process execution .......... 145
5-24 Promoting ETL scheduled processes to Production status .......... 147
5-25 Setting production mode for multiple components ................. 148
A-1 Patterns layered asset model ........................................ 186
A-2 Pattern representation of a Custom design .......................... 188
A-3 Custom design .............................................................. 189
Unveil Your e-business Transaction Performance with IBM TMTP 5.1
Tables

1-1 IBM Tivoli Monitoring for Transaction Performance main components . 30
2-1 Enterprise Transaction Performance installable components . . . . . . 37
2-2 Port usage by ETP Web Console . . . . . . . . . . . . . . . . . . . . . . . . . . . . 39
2-3 Installation settings for IBM HTTP Server . . . . . . . . . . . . . . . . . . . . . . 42
2-4 File changes needed to reassign Tomcat ports . . . . . . . . . . . . . . . . . . . 57
5-1 Measurement codes for TMTP Warehouse Enablement packs . . . . . 127
5-2 Source database names used by the TMTP ETLs . . . . . . . . . . . . . . . . . . 134
5-3 Warehouse processes . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 142
5-4 Warehouse processes and components . . . . . . . . . . . . . . . . . . . . . . . 146
A-1 Business patterns . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 187
A-2 Integration patterns . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 188
A-3 Composite patterns . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 189
Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing, IBM Corporation, North Castle Drive Armonk, NY 10504-1785 U.S.A.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

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

COPYRIGHT LICENSE:
This information contains sample application programs in source language, which illustrates programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. You may copy, modify, and distribute these sample programs in any form without payment to IBM for the purposes of developing, using, marketing, or distributing application programs conforming to IBM's application programming interfaces.

© Copyright IBM Corp. 2003. All rights reserved.
Trademarks

The following terms are trademarks of the International Business Machines Corporation in the United States, other countries, or both:

- AIX®
- CICS®
- Database 2™
- DB2 Universal Database™
- DB2®
- developerWorks®
- eServer™
- @server™
- ibm.com®
- xSeries®
- IBM®
- IMS™
- Lotus®
- NetView®
- Notes®
- Perform™
- Redbooks™
- Redbooks (logo)™
- RMF™
- RS/6000®
- SecureWay®
- Tivoli Enterprise™
- Tivoli Enterprise Console®
- Tivoli Management Environment®
- Tivoli®
- TME®
- VisualAge®
- WebSphere®

The following terms are trademarks of other companies:

- ActionMedia, LANDesk, MMX, Pentium and ProShare are trademarks of Intel Corporation in the United States, other countries, or both.
- Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.
- Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.
- C-bus is a trademark of Corollary, Inc. in the United States, other countries, or both.
- UNIX is a registered trademark of The Open Group in the United States and other countries.
- SET, SET Secure Electronic Transaction, and the SET Logo are trademarks owned by SET Secure Electronic Transaction LLC.

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

This IBM Redbook provides best practices on how to use the new functions for accurately measuring the performance of e-business transactions offered by IBM Tivoli Monitoring for Transaction Performance Version 5.1.

IBM Tivoli Monitoring for Transaction Performance Version 5.1 (sometimes referred to as TMTP) is part of the Tivoli family of products for managing performance and availability. This program offers the functionality of two former Tivoli products, Tivoli Application Performance Management Version 2.1 (TAPM) and Tivoli Web Services Manager Version 1.7, and extends their use by providing the following new key features:

► Breakdown of simulated WebSphere-based e-business application transactions to assist in root-cause analysis and identify potential bottlenecks.
► Performance measurement of simulated 3270 and SAP transactions.
► New Web console for viewing transaction performance metrics on originating systems that are part of the Tivoli Management Environment. This is an extension of the former Tivoli Application Performance Management product that enhances usability and reporting.
► Integration into the IBM Tivoli Monitoring 5.1.1 and related components such as IBM Tivoli Enterprise Console (TEC), IBM Tivoli Business Systems Manager (TBSM), Tivoli Enterprise Data Warehouse (TEDW), and IBM Tivoli Service Level Advisor.

As well, backward compatibility with Tivoli Distributed Monitoring Version 4.1 and Tivoli Decision Support environments is ensured.

As the majority of the functions of IBM Tivoli Monitoring for Transaction Performance Version 5.1 are well-documented in product manuals and previously published redbooks (refer to “Related publications” on page 197 for a list publications related to this topic), this book aims to extend the available information, thus it focuses solely on the new capabilities of the product.

The outline of this book is:

► Chapter 1, “Business objectives and transaction performance” on page 1
► Chapter 2, “TMTP Deployment” on page 31
► Chapter 3, “Transaction breakdown: root cause analysis” on page 65
► Chapter 5, “TMTP and Tivoli Enterprise Data Warehouse” on page 117
► Chapter 4, “Monitoring 3270 transaction performance” on page 105
► Chapter 6, “Troubleshooting” on page 149
The team that wrote this redbook

This redbook was produced by a team of specialists from around the world working at the International Technical Support Organization, Austin Center.

**Morten Moeller** is a Project Leader at the International Technical Support Organization, Austin Center. He applies his extensive field experience as an IBM Certified IT Specialist to his work at the ITSO where he writes extensively on all areas of Systems Management. Before joining the ITSO, Morten worked in the Professional Services Organization of IBM Denmark as a Distributed Systems Management Specialist where he was involved in numerous projects designing and implementing systems management solutions for major customers of IBM Denmark.

**Claudio Cicaterri** is an I/T Specialist in IBM Global Services Italy. He has four years of experience in the system management field and he was involved in several projects implementing Tivoli environments for several customers. He has been published in the IBM ICM Asset Web: “Optimizing Software Distribution through Intelligent Data Analysis” and “Predictive System Monitoring.” Before joining IBM, he worked for two years as Programmer Analyst, mainly developing C, C++, and VisualBasic code. He holds a degree in Physics from the University of Rome.

**Nils Heuer** (himself@nilsheuer.com) is a Senior System Developer for Bohemiens Numerique in Berlin, Germany. Previously he was a technical lead with I-D Media AG, also in Berlin. He has eight years of experience in the IT field, and specializes in end-to-end monitoring, service level management and messaging solutions from Tivoli, Lotus, and Mercury Interactive. He has co-authored the third edition of the IBM Redbook “Tuning IBM eServer xSeries servers for performance.” He is a Cisco Certified Networking Professional and a Principal Certified Lotus Professional for R5 System Administration.

**Randall K. Hill** is a Staff Software Engineer with Tivoli Customer Support in Austin. He holds a Bachelor of Science degree in Computer Science from University of Central Oklahoma. Randall has been with IBM since 1998, and supports several IBM Tivoli Monitoring products, including IBM Tivoli Monitoring for Transaction Performance, and serves as Knowledge Champion for the DM/CTM Technical Support team. Before joining IBM, Randall was Director of Management Information Systems at Seminole State College in Seminole, Oklahoma for 6 years and a Computer Systems Analyst at Tinker Air Force Base, Oklahoma for 12 years.

Thanks to the following people for their contributions to this project:

Betsy Thaggard
International Technical Support Organization, Austin Center
Become a published author

Join us for a two- to six-week residency program! Help write an IBM Redbook dealing with specific products or solutions, while getting hands-on experience with leading-edge technologies. You'll team with IBM technical professionals, Business Partners and/or customers.

Your efforts will help increase product acceptance and customer satisfaction. As a bonus, you'll develop a network of contacts in IBM development labs, and increase your productivity and marketability.

Find out more about the residency program, browse the residency index, and apply online at:

ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us!

We want our Redbooks to be as helpful as possible. Send us your comments about this or other Redbooks in one of the following ways:

- Use the online Contact us review redbook form found at:
  ibm.com/redbooks

- Send your comments in an Internet note to:
  redbook@us.ibm.com

- Mail your comments to:
  IBM Corporation, International Technical Support Organization
  Dept. JN9B Building 003 Internal Zip 2834
  11400 Burnet Road
  Austin, Texas 78758-3493
Business objectives and transaction performance

This chapter provides an overview of the business reasons for looking at transaction performance. In addition, systems management and availability, as well as performance monitoring, will be discussed in broader terms, and the functionality and architecture of the IBM Tivoli Monitoring for Transaction Performance solution will be presented.
1.1 Why have a Web presence

Most organizations—commercial, governmental, and non-profit alike—have established a presence on the World Wide Web. Beyond just being able to say “visit us at www-dot...” the Web presence serves specific purposes, the major one being to provide faster and more cost-effective service than traditional technologies such as telephone, fax, and mail.

The vast majority of Web sites available today provide self-service functions for organizations and individuals to seek information on their own, whenever they like. Information gathering may be regarded as providing value to the individual seeking the information, but the organization making the information and services available may also benefit from gathering statistics about who the users of their services are, when the information is being searched, and from where in the world the information is being requested. General data such as this may be helpful when targeting specific campaigns and promotions.

In addition to information provisioning, many Web site owners have added self-service functions to ease their users’ lives. Many governmental organizations now support submission of applications, petitions, tax declarations, and the like. Non-profit organizations accept donations by providing users the option of supplying credit card information, and financial Web sites allow automatic transfers and withdrawals based on user input. Accordingly, commercial Web sites use these same functions for online shopping.

As the need for telephone service representatives and mail-room personnel has been drastically reduced from the increased use of Web-based self-service functions, businesses and organizations have realized less of need for office space and thereby additional savings. The ability to save even more by using Web technology that enables employees to telecommute from their home offices is improving, and will allow employees a more flexible division between their working and personal lives. The need to be in the office—or in the same geographic region as the office—is slowly decreasing, thereby challenging IT organizations to provide a stable, responsive infrastructure to support Web-based services for employees, wherever they may be.

1.2 Why monitor the performance of transactions

Clearly this evolution toward increasing use of Web services, both to provide a business’ internal systems for information gathering, accounting, and reporting, and externally for marketing and sales purposes, puts more and more emphasis on the IT infrastructure that supports these application systems as well as the applications themselves. To retain and improve productivity and bring down
costs, the tools used by customers and employees need to be adequately responsive in order to be accepted and become the tools of choice rather than the manual processes that they are replacing.

In order to be successful in deploying Web-based tools, an IT department must ensure that both the infrastructure supporting the Web-based applications and the applications themselves are running smoothly. Traditionally, the IT department has focused on monitoring and managing server and network performance and availability and put less emphasis on the business transactions themselves. This has led to situations in which servers and the network performed according to specifications but, due to misdesigned user interfaces, unfortunate application design, or poor database performance, the user's experiences with the new tools were so disappointing that the system failed to gain acceptance and a lot of development effort and money were wasted.

Based on these experiences, more attention has been devoted to monitoring and managing transaction performance. With ensured application responsiveness, users are more likely to accept the new tools and help the business reach its objectives. In this context, transaction performance is regarded as the responsiveness of the enterprise systems providing data and computing power, the time it takes to send data back and forth to the end user, and, perhaps most important, the time it takes the end user's computer to display the results. In the Web-based world the enterprise can fully control the responsiveness of the backend servers. When a server does not respond in a timely fashion, the bottleneck can be identified and more capable hardware may be acquired to resolve the issue. The time needed to transport data back and forth to end users is partly controlled by the enterprise. Both the enterprise and the end user connects to the Internet, and each is responsible for the capacity of that connection. Therefore, the enterprise may at least partly reduce the response time of a Web-based application by providing more powerful communication lines to the Internet. The enterprise as such can provide its employees with the needed bandwidth to the workplace, but it cannot expect customers and external users to acquire extra bandwidth, as well.

Finally, the response time of a Web-based application can rely heavily on the computing power needed to display the data. As was the case with networking resources, businesses should not expect that every potential customer is using state-of-the-art computing hardware, and should therefore try to limit the amount of resources needed to render its Web pages. Over the years, Web programmers have become more and more creative in the use of dynamic icons or pictures that display amusing but useless information while consuming processing power and networking bandwidth for no practical purpose. Programmers usually have very powerful machines and almost unlimited networking resources, which may be one of the reasons that they don't realize the impact that all of these dynamic objects have on the application's response time to the average end user.
Naturally, such animations may have been used deliberately to provide real-time data such as stock quotes, but most often are used for advertisements or no particular reason at all.

This should help emphasize the need and motivation for embracing transaction performance monitoring as a discipline in the enterprise alongside more-traditional systems management disciplines such as performance and availability management. Just as server availability and performance are monitored today, monitoring the response times experienced by end users provides valuable information regarding the business use of Web applications, which in turn supports the business’ decision-making processes. One example of the business use of gathering performance data is that of service-level management. Most service level agreements (SLAs) today include targets for server and network availability, but these metrics do not indicate how well the end users are served. By monitoring transaction performance, a business can get a better indication of how the systems are used and perceived without having to deal with individual server resources such as application and database servers. For both small businesses and large outsourcing companies, transaction performance monitoring provides the means for establishing clearer SLAs that specify targets for transaction availability and performance, and leave the implementation of the technical infrastructure to the outsourcer.

In addition, using transaction performance monitoring techniques during application development may identify hot spots or bottlenecks early on, and this transaction breakdown of back-end network and user response times may help the enterprise develop, revise, and adopt new guidelines for user interface design.

Another obvious use of transaction performance monitoring methodologies is to reveal exactly which resources are used by which transactions and application systems. Web-based application systems of the early 21st century have become so dynamic and complex that it is almost impossible to anticipate the flow of the transaction. An end user interaction with a business Web site might travel through ten or more servers to reach the application server itself. Each of these servers provides specialized functions such as security, portal services, or load balancing. In addition, to ensure availability, many resources—including Web application servers—are duplicated or gathered in grids or clusters, providing even more dynamics and less transparency. By applying transaction performance monitoring techniques and embedding transaction correlation, the complexity and dynamics of a modern Web application may be revealed and provide the organization with the breakdown of all the resources used by the application.

Finally, mastering the transaction performance methodologies will become more and more important when embracing emerging on-demand technologies. In this
kind of environment, data, applications, and transactions will float in the on-demand infrastructure, moving seamlessly from one location to another and from one from host to another based on availability, priorities, and service needs. In a truly dynamic and almost chaotic environment such as this, it will be necessary to track each individual transaction in order to provide accurate account information for billing individual resource usage.

To summarize, the return on investment (ROI) of deploying transaction performance monitoring tools within an organization is likely to be realized in one or more of the following areas:

- Increased customer service through:
  - Easier, more manageable SLAs (for IT service providers)
  - Less application complexity
  - Better, more responsive user interfaces

- Increased IT department productivity through:
  - Better knowledge about resource usage and dependencies
  - Increased proactive management capabilities
  - More-qualified troubleshooting

- More-qualified business decision-making, through:
  - Line of business (LOB) reporting through specific Web sites and pages
  - Better understanding of application usage and integration
  - Clearer, less-complex SLAs

### 1.3 Business transactions

In the Web world, users perceive interacting with an organization or a business through a Web-based interface as a single, continuous interaction or session between the user's machine and the systems of the other party, and that is how it should be. However, the interaction is most likely made up of a large number of individual, interrelated transactions, each one providing its own specific part of the complex set of functions that implement an e-business transaction, perhaps running on systems owned by other organizations or legal entities.

Figure 1-1 on page 6 shows a typical Web-based transaction, the resources used to facilitate the transaction, and the typical components of a transaction breakdown.
In the context of this book, we will differentiate between different types of transactions depending on the location of the machine from which the transaction is initiated:

**Web transaction**
Originating from the Internet, thus we have no predetermined knowledge about the user, the system, and the location of the transaction originator.

**Enterprise transaction**
Initiated from well-known systems, most of which are under our control, and knowledge of the available resources exists. Typically, the systems initiating these types of transactions are managed by our Tivoli Management Environment.

**Application transaction**
Subtransactions that are initiated by the application-provisioning Web transactions to the end users. Application transactions are typically, but not always, also enterprise transactions, but also may initiate from third-party application servers.
A typical application transaction is a database lookup performed from a Web application server, in response to a Web transaction initiated by an end user.

From a management point of view these transaction types should be treated similarly. Responsiveness from the Web application servers to any requester is equally important, and it should not make a difference if the transaction has been initiated from a Web user, an internal user, or a third-party application server. However, business priorities may influence the level of service or importance given to individual requestors.

However, it is important to note that monitoring transaction performance does not in any way obviate the need to perform the more traditional systems management disciplines such as capacity, availability, and performance management. Since the Web applications are comprised of several resources each hosted by a server, these individual server resources must be managed to ensure that they provide the services required by the applications.

With the myriad servers—and exponentially more individual resources and components—involved in an average-sized Web application system, management of all of these resources is more an art than a science. We begin by providing a short description of the challenges of e-business provisioning in order to identify the management needs and issues related to provisioning e-business applications.

### 1.4 e-business applications’ complex layers of services

A modern e-business solution is much more complex than the standard terminal processing—oriented systems of the 1970s and 1980s, as illustrated in Figure 1-2 on page 8. However, despite major revisions, especially during the turn of the last century, legacy systems are still the bread-and-butter of many enterprises, and the e-business solutions in these environments are designed to front-end these mainframe-oriented application complexes.
The complex infrastructure needed to facilitate e-business solutions has been dictated mostly by requirements for standardization of client run-time environments in order to allow any standard browser to access the e-business sites. In addition, application runtime technologies play a major role, as they must ensure platform independence and seamless integration to the legacy back-end systems, either directly to the mainframe or through the server part of the old client-server solution. Further, making the applications accessible from anywhere in the world by any person on the planet raises some security issues (authentication, authorization, and integrity) that did not need addressing in the old client-server systems as all clients were well-known entities in the internal company network.

Because of the central role that the Web and application servers play within a business and the fact that they are supported and typically deployed across a
variety of platforms throughout the enterprise, there are several major challenges to managing the e-business infrastructure, including:

- Managing Web and application servers on multiple platforms in a consistent manner from a central console
- Defining the e-business infrastructure from one central console
- Monitoring Web resources (sites and applications) to know when problems have occurred or are about to occur
- Taking corrective actions when a problem is detected in a platform independent way
- Gathering data across all e-business environments to analyze events, messages, and metrics

The degree of complexity of e-business infrastructure system management is directly proportional to the size of the infrastructure being managed. In its simplest form, an e-business infrastructure is comprised of a single Web server and its resources, but it can grow to hundreds or even thousands of Web and application servers throughout the enterprise.

To add to the complexity, the e-business infrastructure may span many platforms with different network protocols, hardware, operating systems, and applications. Each platform possesses its unique and specific systems management needs and requirements, not to mention a varying level of support for the administrative tools and interfaces.

Every component in the e-business infrastructure is a potential show-stopper, bottleneck or even single point of failure. Each and every one provides specialized services needed to facilitate the e-business application system. The term application systems is used deliberately to enforce the point that no single component by itself provides a total solution: the application is pieced together by a combination of standard off-the-shelf components and home-grown components. The standard components provide general services such as session control, authentication and access control, messaging, and database access, and the home-grown components add the application logic needed to glue all the different bits and pieces together to perform the specific functions for that application system. On an enterprise level, chances are that many of the home-grown components may be promoted to standard status to ensure specific company standards or policies.

At first glance, breaking up the e-business application into many specialized services may be regarded as counterproductive and very expensive to implement. However, specialization enables sharing of common components—such as Web, application, security, and database servers—between more e-business application systems, and it is key to ensuring
availability and performance of the application system as a whole by allowing for duplication and distribution of selected components to meet specific resource requirements or increase the performance of the application systems as a whole. In addition, this itemizing of the total solution allows for almost seamless adoption of new technologies for selected areas without exposing the total system to change.

Whether the components in the e-business system are commercial, standard, or application-specific, each of them will most likely require other general services, such as communication facilities, storage space, and processing power, and the computers on which they run need electrical power, shelter from rain and sun, access security, and perhaps even cooling.

As it turns out, the e-business application relies on several layers of services that may be provided internally or by external companies. This is illustrated in Figure 1-3.

![Figure 1-3 Layers of service](image)

As a matter of fact, it is not exactly the e-business application that relies on the services depicted above. The correct notion is that individual components (such as Web servers, database servers, application servers, lines, routers, hubs, and switches) each rely on underlying services provided by some other component. This can be broken down even further, but that is beyond this discussion. The point is that the e-business solution is exactly as solid, robust, and stable as the weakest link of the chain of services that make up the entire solution, and since the bottom-line results of an enterprise may be affected drastically by the quality of the e-business solutions provided, a worst-case scenario may prove that a power failure in Hong Kong may have an impact on sales figures in Greece and that increased surface activity on the sun may result in satellite-communication problems that prevent car rental in Chattanooga.
While mankind cannot prevent increased activity of the sun and wind (yet), there are a number of technologies available to allow for continuing, centralized monitoring and surveillance of the e-business solution components. These technologies will help manage the IT resources that are part of the e-business solution. Some of these technologies may even be applied to manage the non-IT resources such as power, cooling, and access control.

However, each layer in any component is specialized and requires a different types of management. In addition, from a management point of view, the top layer of any component is the most interesting as it is the layer that provides the unique service that is required by that particular component. For a Web server, the top layer is the HTTP server itself. This is the mission-critical layer, even though it still needs networking, an operating system, hardware, and power to operate. On the other hand, for an e-business application server (although it also may have a Web server installed for communicating with the dedicated Web Server) the mission-critical layer is the application server, and the Web server is considered secondary in this case, just as the operating system, power, and networking are. This said, all the underlying services are needed and must operate flawlessly in order for the top layer to provide its services. It is much like driving a car: You “monitor” the speedometer regularly to avoid penalties by violating changing speed limits, but you check the fuel indicator only from time to time or when the indicator alerts you to perform preventive maintenance by filling up the tank.

1.4.1 Managing e-business applications

Specialized functions require specialized management, and general functions require general management. Therefore, it is obvious that the management of the operating system, hardware layer, and networking layer may be may be general, since they are used by most of the components of the e-business infrastructure. On the other hand, a management tool for Web application servers might not be very well-suited for managing the database server.

Up till now the term “managing” has been widely used, but not yet explained.

Control over and management of the computer system and its vital components are critical to the continuing operation of the system and therefore the timely availability of the services and functions provided by the system. This includes controlling both physical and logical access to the system to prevent unauthorized modifications to the core components, and monitoring the availability of the systems as a whole as well as the performance and capacity usage of the individual resources such as disk space, networking equipment, memory, and processor usage. Of course, these control and monitoring activities have to be performed cost-effectively, so the cost of controlling any resource does not become higher than the cost of the resource itself. It does not make
much business sense to spend $1000 to manage a $200 hard disk, unless the data on that hard disk represents real value to the business in excess of $1000. Planning for recovery of the systems in case of a disaster also needs to be addressed, as being without computer systems for days or weeks may have a huge impact on the ability to conduct business.

There still is one important aspect to be covered for successfully managing and controlling computer systems. We have mentioned various hardware and software components that collectively provide a service, but which components are part of the IT infrastructure, where are they, and how do they relate to one another? A prerequisite for successful management is the detailed knowledge of which components to manage, how the components interrelate, and how these components may be manipulated in order to control their behavior.

In addition, now that IT has become an integral part of doing business, it is equally important from an IT management point of view to know which commitments we have made with respect to availability and performance of the e-business solutions, and what commitments our subcontractors have made to us. And for planning and prioritization purposes it is vital to combine our knowledge about the components in the infrastructure with the commitments we have made in order to assess and manage the impact of component malfunction or resource shortage. In short, in a modern e-business environment, one of the most important management tasks is to control and manage the service catalogue in which all the provisioned services are defined and described, and the SLAs in which the commitments of the IT department are spelled out.

For this discussion, we turn to the widely recognized Information Technology Infrastructure Library (ITIL). The ITIL was developed by the British Government's Central Computer and Telecommunications Agency (CCTA), but has over the past decade or more gained acceptance in the private sector.

One of the reasons behind this acceptance is that most IT organizations, met with requirements to promise or even guarantee performance and availability, agree that there is no point in agreeing to deliver a service at a specific level if the basic tools and processes needed to deploy, manage, monitor, correct, and report the achieved service level have not been established. ITIL groups all of these activities into two major areas, Service Delivery and Service Support, as shown in Figure 1-4 on page 13.
The primary objectives of the Service Delivery discipline are proactive and consist primarily of planning and ensuring that the service is delivered according to the Service Level Agreement. For this to happen, the following tasks have to be accomplished.

**Service Delivery**
Within ITIL, the proactive disciplines are grouped in the Service Delivery area:

**Service Level Management**  Service Level Management involves managing customer expectations and negotiating Service Level Agreements. This involves identifying customer requirements and determining how these can best be met within the agreed-upon budget, as well as working together with all IT disciplines and departments to plan and ensure delivery of services. This involves setting measurable performance targets, monitoring performance, and taking action when targets are not met.

**Cost Management**  Cost Management consists of registering and maintaining cost accounts related to the use of IT services and delivering cost statistics and reports to Service Level Management to assist in obtaining the correct balance between service
cost and delivery. It also means assisting in pricing the services in the service catalog and SLAs.

**Contingency Planning**
Contingency Planning develops and ensures the continuing delivery of minimum outage of the service by reducing the impact of disasters, emergencies, and major incidents. This work is done in close collaboration with the company’s business continuity management, which is responsible for protecting all aspects of the company’s business including IT.

**Capacity Management**
Capacity Management plans and ensures that adequate capacity with the expected performance characteristics is available to support the service delivery. Deliver capacity usage, performance, and workload management statistics as well as trend analysis to Service Level Management.

**Availability Management**
Availability Management means planning and ensuring the overall availability of the services and providing management information in the form of availability statistics, including security violations, to Service Level Management.

Even though not explicitly mentioned in the ITIL definition, for this discussion content management is included in this discipline.

This discipline may also include negotiating underpinning contracts with external suppliers and the definition of maintenance windows and recovery times.

The disciplines in the Service Support group are mainly reactive and are concerned with implementing the plans and providing management information regarding the levels of service achieved.

**Service Support**
The reactive disciplines that are considered part of the Service Support group are:

**Configuration Management**
Configuration Management is responsible for registering all components in the IT service, including customers, contracts,
SLAs, hardware and software components, and maintaining a repository of configurable attributes and relationships between the components.

Help Desk

The Help Desk acts as the main point of contact for users of the service. It registers incidents, allocates severity, and coordinates the efforts of support teams to ensure timely and accurate problem resolution.

Escalation times are noted in the SLA and are agreed on between the customer and the IT department. The Help Desk also provides statistics to Service Level Management to demonstrate the service levels achieved.

Problem Management

Problem Management implements and uses procedures to perform problem diagnosis and identify solutions that correct problems. Register solutions in the configuration repository.

Escalation times should be agreed upon internally with Service Level Management during the SLA negotiation. It also provides problem resolution statistics to support Service Level Management.

Change Management

Change Management plans and ensures that the impact of a change to any component of a service is well known and that the implications regarding service level achievements are minimized. This includes changes to the SLA documents and the Service Catalog as well as organizational changes and changes to hardware and software components.

Software Control and Distribution

It is the responsibility of Software Control and Distribution to manage the master software repository and deploy software components of services. Deploy changes at the request of Change Management, and provide management reports regarding deployment.
The key relationships between the disciplines are shown in Figure 1-5.

**Figure 1-5  Key relationships between Service Management disciplines**

For the remainder of this discussion, we will limit our discussion to capacity and availability management of the e-business solutions. Contrary to the other disciplines that are considered common for all types of services provided by the IT organization, the e-business solutions provide special challenges to management due to their high visibility and importance to the bottom line business results, their level of distribution, and the special security issues that characterize the Internet.
1.4.2 Architecting e-business application infrastructures

In a typical e-business environment, the application infrastructure consists of three separate tiers, and the communication between these is restricted, as Figure 1-6 shows.

![Diagram of e-business application infrastructure](image)

The tiers are typically:

**Demilitarized Zone** The tier accessible by all external users of the applications. This tier functions as the gatekeeper to the entire system, and functions such as access control and intrusion detection are enforced here. The only other part of the intra-company network that the DMZ can talk to is the application tier.

**Application Tier** This is usually implemented as a dedicated part of the network where the application servers reside. End-user requests are routed from the DMZ to the specific servers in this tier, where they are serviced. In case the applications need to use resources from company-wide databases, for example, these are requested from the back-end tier, where all the secured company IT assets reside. As was the case for communication between the DMZ and the...
Application Tier, the communication between the Application Tier and the back-end systems is established through firewalls and using well-known connection ports. This helps ensure that only known transactions from known machines outside the network can communicate with the company databases or legacy transaction systems such as CICS or IMS. Apart from specific application servers, this tier also hosts load-balancing devices and other infrastructural components such as MQ Servers needed to implement a given application architecture.

**Back-end Tier**

This is where all the vital company resources and IT assets reside. External access to these resources is only possible through the DMZ and the Application Tier.

This model architecture is a proven way to provide secure, scalable, high-availability external access to company data with a minimum of exposure to security violations. However the actual components, such as application servers and infrastructural resources, may vary depending upon the nature of the applications, company policies, the requirements to availability and performance, and the capabilities of the technologies used.

If you are in the e-business hosting area or you have to support multiple lines of business that require strict separation, the conceptual architecture shown in Figure 1-6 on page 17 may be even more complicated. In these situations, one or more of the tiers may have to be duplicated to provide the required separation. In addition, the back-end tier might even be established remotely (relative to the application tier). This is very common when the e-business application hosting is outsourced to an external vendor, such as IBM Global Services.

To help design the most appropriate architecture for a specific set of e-business applications, IBM has published a set of e-business patterns that may be used to speed up the process of developing e-business applications and deploying the infrastructure to host them.

The concept behind these e-business patterns is to reuse tested and proven architectures with as little modification as possible. IBM has gathered experiences from more than 20,000 engagements, compiled these into a set of guidelines, and associated them with links. A solution architect can start with a problem and a vision for the solution and then find a pattern that fits that vision. Then, by drilling down using the patterns process, the architect can further define the additional functional pieces that the application will need to succeed. Finally, the architect can build the application using coding techniques outlined in the associated guidelines. Further details on e-business patterns may be found in Appendix A, “Patterns for e-business” on page 183.
1.4.3 Basic products used to facilitate e-business applications

So far, we may conclude that building an e-business solution is like building a vehicle, in the sense that:

- We want to provide the user with a standard, easy-to-use interface that fulfills the needs of the user and has a common look-and-feel to it.
- We want to use as many standard components as possible to keep costs down and be able to interchange them seamlessly.
- We want it to be reliable and available at all times with a minimum of maintenance.
- We want to build in unique features (differentiators) that make the user choose our product over those of the competitors.

The main difference between the vehicle and the e-business solution is that we own and control the solution, but the buyer owns and manages the vehicle. The vehicle owner decides when to have the oil changed and when to fill up the fuel tank or adjust the tire pressure. The vehicle owner also decides when to take the vehicle in for a tune-up, when to add chrome bumpers and alloy wheels to make the vehicle look better, and when to sell it. The user of an e-business site has none of those choices. As owners of the e-business solution, we decide when to rework the user interface to make it look better, when to add resources to increase performance, and ultimately when to retire and replace the solution.

This gives us a few advantages over the car manufacturer, as we can modify the product seamlessly by adding or removing components as needed in order to align the performance with the requirements and adjust the functionality of the product as competition toughens or we engage in new alliances.

No matter whether the e-business solution is the front-end of a legacy system or a new application developed using modern, state-of-the-art development tools, it may be characterized by three specific layers of services that work together to provide the unique functionality necessary to allow the applications to be used in an Internet environment, as shown in Figure 1-7 on page 20.
The presentation layer must be a commonly available tool that is installed on all the machines used by users of the e-business solution. It should support modern development technologies such as XML, JavaScript and HTML pages, and usually is the browser.

The standard communication protocols used to provide connectivity using the Internet are TCP/IP, HTTP, and HTTPS. These protocols must be supported by both client and server machines.

The transformation services are responsible for receiving client requests and transforming them into business transactions that in turn are served by the Solution Server. In addition, it is the responsibility of the transformation service to receive results from the Solution Server and convey them back to the client in a format that can be handled by the browser. In e-business solutions that do not interact with legacy systems, the transformation and Solution Server services may be implemented in the same application, but most likely they are split into two or more dedicated services.

This is a very simple representation of the functions that take place in the transformation service. Among other functions that must be performed are identification, authentication and authorization control, load balancing, and transaction control. Dedicated servers for each of these functions are usually implemented to provide a robust and scalable e-business environment. In addition some of these are placed in a dedicated network segment—the demilitarized zone (DMZ)—which, from the point of view of the e-business owner, is fully controlled, and in which client requests are received by “well-known,” secure systems and passed on to the enterprise network, also known as the intranet. This architecture is used to increase security by avoiding transactions from “unknown” machines to reach the enterprise network, thereby minimizing the exposure of enterprise data and the risk of hacking.
To facilitate secure communication between the DMZ and the intranet, a set of Web servers is usually implemented, and identification, authentication, and authorization are handled by an LDAP Server, typically.

The infrastructure depicted in Figure 1-8 contains all components required to implement a secure e-business solution, allowing anyone from anywhere to access and do business with the enterprise.

Figure 1-8  Logical view of an e-business solution


Tivoli and IBM provide some of the most widely used products to implement the e-business infrastructure. These are:

**IBM HTTP Server**  Communication and transaction control

**Tivoli SecureWay Policy Director**  Identification, authentication, and authorization
IBM WebSphere Application Server | Web application hosting, responsible for the transformation services

IBM WebSphere Edge Server | Web application firewalling, load balancing, Web hosting; responsible for the transformation services

1.4.4 Managing e-business applications using Tivoli

Even though the e-business patterns help in designing e-business applications by breaking them down into functional units that may be implemented in different tiers of the architecture using different hard- and software technologies, the patterns provide only some assistance in managing these applications. Fortunately, this gap is filled by solutions from Tivoli Systems.

When designing the systems management infrastructure that is needed to manage the e-business applications, it must be kept in mind that the determining factor for the application architecture is the nature of the application itself. This determines the application infrastructure and the technologies used. However, it does not do any harm if the solution architect consults with systems management specialists while designing the application.

The systems management solution has to play more or less by the rules set up by the application. Ideally, it will manage the various application resources without any impact on the e-business application, while observing company policies on networking use, security, etc.

Management of e-business applications is therefore best achieved by establishing yet another networking tier, parallel to the application tier, in which all systems management components can be hosted without influencing the applications. Naturally, since the management applications have to communicate with the resources that must be managed, the two meet on the network and on the machines hosting the various e-business application resources.

Using the Tivoli product set, it is recommended that you establish all the central components in the management tier and have a few proxies and agents present in the DMZ and application tiers, as shown in Figure 1-9 on page 23.
Implementing the management infrastructure in this fashion, there is minimal interference between the application and the management systems, and the access to and from the various network segments is manageable as the communication flows between a limited number of nodes using well-known communication ports.

IBM's Tivoli management products have been developed with the total environment in mind. The IBM Tivoli Monitoring product provides the basis for proactive monitoring, analysis, and automated problem resolution.

As we will see, IBM Tivoli Monitoring for Transaction Performance provides an enterprise management solution for both the Web and enterprise transaction environments. The PACS that make up this product provide solutions that are integrated with other Tivoli management products and contribute a key piece to the goal of a consistent, end-to-end management solution for the enterprise.

By using product offerings such as IBM Tivoli Monitoring for Transaction Performance in conjunction with the underlying Tivoli technologies, a comprehensive and fully integrated management solution can be deployed rapidly and provide a very attractive return on investment.
1.5 Tivoli product structure

Let’s take a look at how Tivoli solutions provide comprehensive systems management for the enterprise and how the IBM Tivoli Monitoring for Transaction Performance product fits into the overall architecture.

The Tivoli solutions are organized into the categories shown in Figure 1-10.

<table>
<thead>
<tr>
<th>Business Systems Management</th>
</tr>
</thead>
<tbody>
<tr>
<td>Performance and Availability</td>
</tr>
<tr>
<td>Configuration and Operations</td>
</tr>
<tr>
<td>Storage</td>
</tr>
<tr>
<td>Security</td>
</tr>
</tbody>
</table>

| Common Infrastructure/Services |

Underlying the Tivoli solution set is a group of common services and infrastructure that provide consistency across Tivoli management applications, as well as enabling integration.

Within the Tivoli product family, there are specific solutions that target the same four primary disciplines of systems management:

- Performance and Availability
- Configuration and Operations
- Storage
- Security

Products within each of these areas have been made available over the years and, as they are continually enhanced, have become accepted solutions in enterprises around the world. With these core capabilities in place, IBM has been able to focus on building applications that take advantage of these pillars to provide true business systems management solutions. A typical business application depends not only on hardware and networking, but also on software ranging from the operating system to middleware such as databases, Web servers, and messaging, to the applications themselves. A suite of solutions such as the “IBM Tivoli Monitoring for ...” products, enables an IT department to
provide consistent management of the entire business system from a central site and using an integrated set of tools. By utilizing an end-to-end set of solutions built on a common foundation, enterprises can manage the ever-increasing complexity of their IT infrastructure with reduced staff and increased efficiency.

Within the performance and availability pillar in Figure 1-10 on page 24, three functional areas are used to organize and coordinate the functions provided by Tivoli products. These areas are shown in Figure 1-11.

![Figure 1-11 Tivoli’s Performance and Availability product structure](image)

The lowest level consists of the monitoring products and technologies, such as IBM Tivoli Monitoring and its resource models. At this layer, Tivoli applications monitor the hardware and software and provide automated corrective actions whenever possible.

At the next level is event correlation and automation. As problems occur that cannot be resolved at the monitoring level, event notifications are generated and sent to a correlation engine such as Tivoli Enterprise Console. The correlation engine at this point can analyze problem notifications (events) coming from multiple components and either automate corrective actions or provide the necessary information to operators who can initiate corrective actions.

The third tier in this structure is called Business Impact Management. It is important to know that a component or related set of components has failed as reported by the monitors in the first layer. Likewise, in the second layer it is valuable to understand how a single failure may cause problems in related components. For instance, a router being down could cause database clients to generate errors if they cannot access the database server. Business Impact Management is the most valuable layer as it provides an insight into how a component failure may be affecting the business as a whole. When the router
failure mentioned above occurs, it is important to understand exactly what line of business applications will be affected and how to reduce the impact of that failure on the business. IBM Tivoli Business System Manager provides this capability.

1.6 Managing e-business applications

As we have seen, managing e-business applications requires that basic services such as communications, messaging, database, and application hosting are functional and well-behaved. This should be ensured by careful management of the infrastructural components using Tivoli tools to facilitate monitoring, event forwarding, automation, console services, and business impact visualization.

However, ensuring the availability and performance of the application infrastructure is not always enough. Web-based applications are implemented in order to attract business from customers and business partners who we may or may not know. Depending on the nature of the data provided by the application, company policies for security and access control, as well as access to and use of specific applications, may be restricted to users whose identity can be authenticated. In other instances—for example, online news services—there are user authentication requirements for access to the application.

In either case, the goal of the application is to provide useful information to the user and, of course, attract the user to return later. The service provided to the user, in terms of functionality, ease of use, and responsiveness of the application, is critical to the user's perception of the application's usefulness. If the user finds the application useful, there is a fair chance that the user will return to conduct more business with the application owner.

The usefulness of an application is a very subjective measure, but it seems fair to assume that an individual's perception of an application's usefulness involves, at the very least:

- relevance to current needs
- easy-to-understand organization and navigation
- logical flow and guidance
- the integrity of the information (is it trustworthy?)
- responsiveness of the application

Naturally, the application owner can influence all of these parameters—the application design can be modified, the data can be validated, and so on—but network latency and the capabilities of the user's system are critical factors that may affect the time it takes for the user to receive a response from the
application. To avoid this becoming an issue that scares users away from the application, the application provider can:

- Set the user's expectations by providing sufficient information up front.
- Make sure that the back-end transaction performance is as fast as possible.

Neither of these will guarantee that users will return to the application, but monitoring and measuring the total response time and breaking it down into the various components shown in Figure 1-1 on page 6, will give the application owner an indication of where the bottlenecks might be.

To provide consistently good response times from the backend systems, the application provider may also establish a monitoring system that generates reference transactions on a scheduled basis. This will give early indications about upcoming problems or flux the responsiveness of the applications.

These needs for real-time monitoring and gathering of reference (and historical) data, among others, are addressed by IBM Tivoli Monitoring for Transaction Performance. By providing the tools necessary for understanding the relationships between the various components that make up the total response time of an application, including breakdown of the back-end service times into service times for each subtransaction, IBM Tivoli Monitoring for Transaction Performance is the tool of choice for monitoring and measuring transaction performance.

1.6.1 IBM Tivoli Monitoring for Transaction Performance functions

IBM Tivoli Monitoring for Transaction Performance provides functions to monitor e-business transaction performance in a variety of situations. Focusing on e-business transactions, it should come as no surprise that the product provides functions for transaction performance measurement for various Web-based transaction types originating from external systems (systems situated somewhere on the Internet and not managed by the organization) that provide the e-business transactions or applications that are the target of the performance measurement. These transactions are referred to in the following pages as Web transactions, and they are implemented by the Web Transaction Performance module of IBM Tivoli Monitoring for Transaction Performance.

In addition, a set of functions specifically designed to monitor the performance metrics of transactions invoked from within the corporate network (in the following pages known as enterprise transactions) are provided by the product's Enterprise Transaction Performance module. The main function of Enterprise Transaction Performance is to monitor transaction performance of applications that have transaction performance probes (ARM calls) included. In addition, Enterprise Transaction Performance provides functions to monitor online transactions with mainframe sessions (3270) and SAP systems, non-Web based
response times for transactions with mail and database servers, and Web-based transactions with HTTP servers.

It should be noted that the tools for Web and enterprise transaction performance monitoring complement one another, and that there are no restrictions, if the networking and management infrastructure is in place, for using Enterprise monitors in the Web space or vice versa.

**Web transaction monitoring**

In general, the nature of Web transaction performance measurement is random and generic. There is no way of planning the execution of transactions or the origin of the transaction initiation unless other measures have been taken in order to do so. When the data from the transaction performance measurements are being aggregated, they provide information about the **average** transaction invocation, without affinity to location, geography, workstation hardware, browser version, or other parameters that may affect the experience of the end user. All of these parameters are out of the application provider’s control. Naturally, both the data gathering and reporting may be set up to only handle transaction performance measurements from machines that have specific network
addresses, for example, thus limiting the scope of the monitoring to well-known machines. However, the transactions executed, and the sequence is still random and unplanned.

The monitoring infrastructure used to capture performance metrics of the average transaction may also be used to measure transaction performance for specific, pre-planned transactions initiated from well-known systems accessing the e-business applications through the Internet or intranet. To facilitate this kind of controlled measurements, certain programs must be installed on the systems initiating the transactions, and they will have to be controlled by the organization that wants the measurements. From a transaction monitoring point of view there are no differences between monitoring average or controlled transactions; the same data may be gathered to the same level of granularity. The big difference is that the monitoring organization knows the transaction being executed, as well as the specifics of the initiating systems.

The main functions provided by IBM Tivoli Monitoring for Transaction Performance: Web Transaction Performance are:

- For both unknown and well-known systems:
  - Real-time transaction performance monitoring
  - Transaction breakdown

- For well-known systems with specific programs installed:
  - Transaction simulation based on recording and playback
  - Web site availability monitoring
  - Web site quality investigation

Enterprise transaction monitoring

If the application provider wants to gather transaction performance characteristics from workstations situated within the enterprise network or machines that are part of the managed domain, but initiates transactions through the Internet, a different set of tools is available. These are provided by the Enterprise Transaction Performance module of the IBM Tivoli Monitoring for Transaction Performance product.

The functions provided by Enterprise Transaction Performance are integrated with the Tivoli Management Environment and rely on common services provided by the integration. Therefore, the systems from which transaction performance data is being gathered must be part of the Tivoli Management Environment, and at a minimum have a Tivoli endpoint installed. This will, however, enable centralized management of the systems for additional functions besides the gathering of transaction performance data.
In addition to monitoring transactions initiated through a browser, just like the ones we earlier called Web transactions, Enterprise Transaction Performance provides specialized programs, end-to-end probes, which enable monitoring of the time needed to load a URL and specific transactions related to certain mail and groupware applications. The Enterprise module also provides unique recording and playback functions for transaction simulation of 3270 and SAP applications, and a generic recording/playback solution to be used only on Windows-based systems.

The major functions provided by Enterprise Transaction Performance are:

- Real-time transaction performance monitoring
- Transaction simulation based on recording and playback

Table 1-1 provides an overview of the main components for transaction performance management provided by IBM Tivoli Monitoring for Transaction Performance.

<table>
<thead>
<tr>
<th>Activity</th>
<th>product feature</th>
<th>Env</th>
<th>Gen</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Web site availability</td>
<td>Site Investigator</td>
<td>W</td>
<td>n</td>
<td></td>
</tr>
<tr>
<td></td>
<td>End-to-End Probes</td>
<td>E</td>
<td>n</td>
<td></td>
</tr>
<tr>
<td>Real-time end user Experience</td>
<td>Quality of Service</td>
<td>W</td>
<td>y</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Client Capture</td>
<td>E</td>
<td>n</td>
<td>Browsers only</td>
</tr>
<tr>
<td>Transaction breakdown</td>
<td>Quality of Service</td>
<td>W</td>
<td>y</td>
<td></td>
</tr>
<tr>
<td>Transaction simulation</td>
<td>STI - record-playback</td>
<td>W</td>
<td>n</td>
<td>Any standard browser - no support for JavaScript, MacroMedia Flash etc.</td>
</tr>
<tr>
<td></td>
<td>EPP</td>
<td>E</td>
<td>n</td>
<td>Limited number of probes</td>
</tr>
<tr>
<td>3270/SAP record-playback</td>
<td></td>
<td></td>
<td>n</td>
<td></td>
</tr>
</tbody>
</table>
Chapter 2. TMTP Deployment

This chapter shows how to successfully install both components of IBM Tivoli Monitoring for Transaction Performance Version 5.1—Enterprise Transaction Performance and Web Transaction Performance—including the associated consoles, on a single system.

The following is a quick guide to product installation with no consideration given to performance or availability issues. For full details about all installation and configuration activities, refer to the official product manuals IBM Tivoli Monitoring for Transaction Performance: Enterprise Transaction Performance User's Guide Version 5.1, GC23-4803, and IBM Tivoli Monitoring for Transaction Performance: Web Transaction Performance Installation Guide Version 5.1.0, GC23-4801, in addition to the other available documentation referred to in “Related publications” on page 197.

In section 2.2.1, “Preparing for multiple consoles on one system” on page 38 of this chapter, we demonstrate how to implement both IBM Tivoli Monitoring for Transaction Performance Web interfaces along with IBM Tivoli Monitor Web Health Console on the same system, thereby reducing the number of machines needed to provide the full functionality of the product.
2.1 TMTP architecture

Before describing the procedures for installation of the two major IBM Tivoli Monitoring for Transaction Performance components, let us take a brief look at their architecture.

2.1.1 Enterprise Transaction Performance

The Enterprise Transaction Performance components are used to measure transaction performance from systems that belong to the Tivoli Management Environment. Typically, this implies that the transactions that are monitored take place between systems that are part of the enterprise network, also known as the intranet.

Enterprise Transaction Performance provides four ways of measuring transaction performance:

- ARMed application
- Predefined Enterprise Probes
- Client Capture (browser-based)
- Record and Playback

However, the base technology used in probes, Client Capture, and Record and Playback, is that of ARM, so it would be fair to say that Enterprise Transaction Performance provides the means to capture and manage transaction performance data generated by ARM calls. It also provides a set of ARMed tools to facilitate data gathering and provide transaction performance data from applications that are not ARMed themselves.

Applications that are ARMed issue calls to the Application Response Measurement API to notify the ARM receiver, in this case implemented by Tivoli, about the specifics of the transactions within the application.

The probes are predefined ARMed programs provided by Tivoli that may be used to verify the availability of and the response time to load Web sites, mail servers, Lotus Notes Servers, and more. The specific object to be targeted by a probe is provided as runtime parameters to the probe itself.

Client Capture acts like a probe. When activated, it scans the input buffer of the browser of a monitored system (typically an end user's workstation) for specific patterns defined at the profile level and records the response time of all page loads, which matches the patterns specified.

Finally, two different implementations of transaction recording and playback are provided: Mercury VuGen, which supports a standard browser interface, and the
IBM Recording and Playback Workbench, which provides recording capabilities for 3270 and SAP transactions.

To initiate transaction performance monitoring, a MarProfile, which contains all the specifics of the transactions to be monitored, is defined in the scope of the Tivoli Management Framework and distributed to a Tivoli endpoint for execution. Based on the settings in the MarProfile, data is now collected locally at the endpoint and may be aggregated to provide minimum, maximum, and average values over a preset period of time. Data related to specific runs of the transactions (instance data) and aggregated data may be forwarded to a central database, which may be used as the source for report generation through Tivoli Decision Support, and as data provider for other applications through Tivoli Enterprise Data Warehouse.

Online surveillance is facilitated through a Web-based console, on which current data at the endpoint and historical data from the database may be viewed.

In addition, two sets of monitors, a monitoring collection for Tivoli Distributed Monitoring 3.x and a resource model for IBM Tivoli Monitoring 5.1.1, are provided to enable generation of alerts to TEC and online surveillance through the IBM Tivoli Monitor Web Health Console. Note that both monitors are based on the
aggregated data collected by the ARM receiver running at the endpoints and thus will not react immediately if, for example, a monitored Web site becomes unavailable. The minimum time for reaction is related to the aggregation period and the thresholds specified.

2.1.2 Web Transaction Performance

The IBM Tivoli Monitoring for Transaction Performance: Web Transaction Performance component is similar to the former Tivoli Web Services Manager Version 1.7 product, with enhancements especially in the Synthetic Transaction Investigator and reporting areas. The architecture is shown in Figure 2-2.

![Figure 2-2 Web Transaction Performance overall architecture](image)

The major components of Web Transaction Performance is the central Tivoli Internet Management Server (TIMS) and its database. The Tivoli Internet Management Server governs all activities in the Web Transaction Performance environment and controls the repository in which all objects and data related to Web Transaction Performance activity and use are stored.
Besides the Tivoli Internet Management Server, Web Transaction Performance operates with three types of endpoints, which are used to monitor Web-related activity:

**Synthetic Transaction Investigator** The Synthetic Transaction Investigator (STI) consists of a recorder endpoint and a playback endpoint. Through the recorder endpoint, a Web transaction can be recorded and stored in the TIMS database. This recording can be turned into a job, which may be scheduled to run periodically on the playback endpoint. The metrics (response time, network time, backend time, load time, render time) are stored in the TIMS database and may be viewed from the Web Transaction Performance Console. The record and playback endpoints may be placed anywhere in the network, or even somewhere on the Internet, to measure transaction metrics from remote locations.

For all other Web Transaction Performance components, event generation thresholds may be defined to notify operational staff about pending problems or inavailability. The notification mechanism is facilitated by the Web Services Courier endpoint.

**Quality of Service** As with STI, the Quality of Service (QoS) endpoint records all relevant metrics from a transaction with a Web server. However, QoS acts on real-life transactions, thus intercepting transactions from users accessing monitored Web sites. The technique used is a reverse proxy, which looks like a normal Web server from a user perspective, but in reality passes packages back and forth between the user and the real application server. Meanwhile, QoS records all traffic that passes through and stores data in the TIMS database.
Web Services Courier

All previously mentioned components are usually located in the unsecured part of the network for accessibility reasons. Web Services Courier (WSC) acts as the trusted courier by importing data from the TIMS database on a regular schedule to make the WTP data available in the secure infrastructure. From here, WSC can send events off to TEC on behalf of the other components and store historical data for reporting and trend analysis.

Site Investigator

Finally, Site Investigator (SI) endpoints are used to crawl Web sites to check for availability, broken links, and unwanted content. Verifying the health of a Web site is really not part of measuring and monitoring transaction performance, so SI will not be discussed further in this publication.

For the remainder of this book, we will focus on the Synthetic Transaction Investigator, which is the component that enables integration to Enterprise Transaction Performance in order to facilitate transaction breakdown for WebSphere 5.0 transactions.

More details about the features and functions of Web Transaction Performance may be found in the product documentation:


and in the Redbooks:

- Tivoli Web Services Manager Internet Management Made Easy, SG24-6017
- Tivoli Web Solutions: Managing Web Services and Beyond, SG24-6049

2.2 Product installation

Installation of IBM Tivoli Monitoring for Transaction Performance is performed in two separate steps. Because of the dependency between the Web Transaction Performance console and the Enterprise Transaction Performance Web Console,
the Enterprise Transaction Performance component should be installed before the Web Transaction Performance component.

The Enterprise Transaction Performance component is installed just as any other Tivoli Framework based product: through the Tivoli command line or the Graphical User Interface of the Tivoli Desktop. We briefly describe the GUI method; for more details, consult IBM Tivoli Monitoring for Transaction Performance: Enterprise Transaction Performance User’s Guide Version 5.1, GC23-4803.

The Enterprise Transaction Performance product consists of one required and four optional components, as shown in Table 2-1.

Table 2-1  Enterprise Transaction Performance installable components

<table>
<thead>
<tr>
<th>Component</th>
<th>required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enterprise Transaction Performance</td>
<td>yes</td>
<td>Basic functions, integration transaction performance monitoring in the Tivoli Management Environment. Must be installed on the TMR Server and on all managed nodes that will act as gateways for endpoints where transaction performance will be activated.(^a)</td>
</tr>
<tr>
<td>Web Console</td>
<td>no</td>
<td>Provides a Web-based GUI to view transaction performance online.(^b) This component must be installed on a managed node.</td>
</tr>
<tr>
<td>Record and Playback Component</td>
<td>no</td>
<td>Provides functions for recording and playing back 3270 and SAP transactions</td>
</tr>
<tr>
<td>Support for EPP</td>
<td>no</td>
<td>Provides probe programs to monitor transaction response times with predefined applications such as Web and mail servers.</td>
</tr>
<tr>
<td>Monitors for DM 3.x</td>
<td>no</td>
<td>Provides monitoring collections to be used with Tivoli Distributed Monitoring 3.x (classic DM)</td>
</tr>
</tbody>
</table>

\(^{a}\) You should establish a working RDBMS environment prior to defining the RIM object used by Enterprise Transaction Performance.

\(^{b}\) When installing this component, installation of IBM HTTP Server may be included
2.2.1 Preparing for multiple consoles on one system

This installation scenario demonstrates setting up a TMR Server to host both of the two basic products, as well as all of the Web-based consoles that are relevant to transaction performance monitoring. These are:

- Enterprise Transaction Performance Web Console
- Web Transaction Performance Tivoli Internet Management Server Console
- IBM Tivoli Monitoring Web Health Console

As none of the consoles relies on components installed by others, each console is hosted by a dedicated instance of IBM HTTP Server and, by default, they all listen to the TCP/IP port number 80. Naturally, all three consoles cannot use the same port, so the port numbers must be changed. However, the registration and location of previously installed IBM HTTP Servers may also cause problems during installation, and the same applies to the interaction with the underlying Tomcat servers used by the Tivoli Internet Management Server Console and Web Console.

The default architecture of the consoles is shown in Figure 2-3.

As seen, there are inherent conflicts in the following areas:

- Port numbers
  - for receiving requests (80)
  - for IBM HTTP Server Administration (8008)
  - for Tomcat communication (8007)
Service names

- IBM HTTP Server
- IBM HTTP Administration

In addition, the installation procedure for the Web Console does not recognize existing instances of the IBM HTTP Server and, if no precautions are taken, will overwrite the existing configurations. The IBM HTTP Server used for the IBM Tivoli Monitor Web Health Console is installed in a directory other than the default, but when registering the services uses the default port numbers and service names: IBM HTTP Server (80) and IBM HTTP Administration (8008). It also registers the HTTP Server 1.3.19 product as being installed. Finally, the IBM HTTP Server instance that hosts the Tivoli Internet Management Server Console is more friendly, and is installed by default in a directory different from the IBM HTTP Server default. It is even given a unique service name, TIMS HTTP Server, to allow for hosting multiple instances of IBM HTTP Server on the same physical system.

In our case, we have no conflicts regarding the basic paths of the IBM HTTP Server instances, though this can happen in other implementations. However, for successful installation of the Web Console using its own IBM HTTP Server, the product registration of the Web Health Console HTTP Server must be renamed. We also have to ensure that the installation can create the default services, so the conflict between the Web Console and Web Health Console IBM HTTP Server service names must be addressed before installing the Web Console.

The use of ports can be solved after installation, and we have chosen to apply changes to the last product installed to increase the chance of keeping the existing components running. The ports that must be changed for the Web Console are shown in Table 2-2.

<table>
<thead>
<tr>
<th>Use</th>
<th>Default</th>
<th>Suggested</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basic HTTP request</td>
<td>80</td>
<td>88</td>
</tr>
<tr>
<td>Basic HTTP Administration</td>
<td>8008</td>
<td>8009</td>
</tr>
<tr>
<td>Tomcat communication</td>
<td>8007</td>
<td>8006</td>
</tr>
</tbody>
</table>

Our installation sequence is:

1. IBM Tivoli Monitor Web Health Console
2. Enterprise Transaction Performance Web Console
3. Tivoli Internet Management Server
Thus we have to change the Tomcat port used by the Web Console to enable flawless installation of the Tivoli Internet Management Server. We solve the inherent conflict related to using port 80 on both the Web Console and the Tivoli Internet Management Server by allowing only SSL communication (port 443) when installing the Tivoli Internet Management Server.

The modified architecture is shown in Figure 2-4 on page 40, and Table 2-3 on page 42 summarizes the procedure.

As the installation of the Web Console may overwrite any existing IBM HTTP Server instances using the default configuration parameters, you should move it before starting the Web Console installation. In our case this is not an issue, but if you have an active IBM HTTP Server installed in the default path (C:\IBMHTTPServer), follow the procedure described in “Moving an existing IBM HTTP Server” on page 50 to move it to a different directory.
2.2.2 Multiple consoles installation outline

As mentioned previously, the installation sequence of the consoles is:

1. IBM Tivoli Monitor Web Health Console
2. Enterprise Transaction Performance Web Console
3. Web Transaction Performance Tivoli Internet Management Server

Therefore, the sequence of tasks that must be completed to successfully install all of the components is:

1. Move any IBM HTTP Server implementations using the c:\IBMHTTPSERVER path and the default service names. Rename the HTTP Server product registration. See “Moving an existing IBM HTTP Server” on page 50 for details about how to complete this task.

2. Install IBM Tivoli Monitor Web Health Console according to the standard documentation.
   a. Rename the IBM HTTP Server product registration following the procedure outlined in “Renaming the HTTP Server product registration” on page 53.
   b. Rename the IBM HTTP Server and IBM HTTP Administration services as described in “Renaming IBM HTTP Server service names” on page 53.

3. Install Enterprise Transaction Performance following the directions given in “Enterprise Transaction Performance installation” on page 43.

4. Install Enterprise Transaction Performance Web Console as described in “Enterprise Transaction Performance installation” on page 43
   a. Assign new ports for the Web Console IBM HTTP Server and IBM HTTP Administration services as described in “Changing the ports used by IBM HTTP Server services” on page 55.
   b. Assign new ports for the Web Console Tomcat communication following the procedure in “Changing the ports used by Tomcat” on page 56.

5. Install Tivoli Internet Management Server allowing only secure communications (SSL). See “Web Transaction Performance installation” on page 47.

If you do not have to move any existing IBM HTTP Server instances, or you do not wish to install the Web Health Console, you can skip those steps.

The default and modified settings for HTTP and Web application server setup that accommodate the IBM Web Health Console, the Enterprise Transaction Performance Web Console, and the Web Transaction Performance Tivoli Internet Management Server on a single system are summarized in Table 2-3 on page 42.
Table 2-3  Installation settings for IBM HTTP Server

<table>
<thead>
<tr>
<th>Service</th>
<th>Name</th>
<th>Port</th>
</tr>
</thead>
<tbody>
<tr>
<td>IBM Tivoli Monitor (ITM) Web Health Console</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Base path</td>
<td>C:\ibm\ITMWebHealthConsole\HTTP_Server</td>
<td></td>
</tr>
<tr>
<td>HTTP Service</td>
<td>IBM HTTP Server</td>
<td>80</td>
</tr>
<tr>
<td>Admin Service</td>
<td>IBM HTTP Administration</td>
<td>8008</td>
</tr>
<tr>
<td>Tomcat Service</td>
<td>n/a</td>
<td>8007</td>
</tr>
<tr>
<td>IBM Tivoli Monitoring for Transaction Performance (TMTP) Web Console</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Base path</td>
<td>C:\IBMHTTPServer</td>
<td></td>
</tr>
<tr>
<td>HTTP Service</td>
<td>IBM HTTP Server</td>
<td>80</td>
</tr>
<tr>
<td>Admin Service</td>
<td>IBM HTTP Administration</td>
<td>8008</td>
</tr>
<tr>
<td>Tomcat Service</td>
<td>n/a</td>
<td>8007</td>
</tr>
<tr>
<td>Tivoli Internet Manager Server (TIMS)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Base path</td>
<td>C:\Program Files\Tivoli\Internet\ManagementServer\IBMHTTPSERVER\w32-ix86</td>
<td></td>
</tr>
<tr>
<td>HTTP Service</td>
<td>IBM HTTP Server</td>
<td>80, 443</td>
</tr>
<tr>
<td>Admin Service</td>
<td>n/a</td>
<td>n/a</td>
</tr>
<tr>
<td>Tomcat Service</td>
<td>Tomcat</td>
<td>8007</td>
</tr>
</tbody>
</table>
2.2.3 Enterprise Transaction Performance installation

The following procedure describes the steps for successfully installing Enterprise Transaction Performance components.

**Installing the Enterprise Transaction Performance base**
To install the basic Enterprise Transaction Performance environment.

1. Log on to the Tivoli Desktop as an Administrator.
2. From the Tivoli Desktop choose **Install Product**, and set the media to point to the directory containing the Enterprise Transaction Performance code.

The five product choices are shown in Figure 2-5.

![Figure 2-5 Enterprise Transaction Performance components](image)

3. Install **IBM Tivoli Monitoring for Transaction Performance 5.1 ETP** and accept all the product defaults, including those for RIM definition.


5. Install the **IBM Tivoli Monitoring for Transaction Performance 5.1 Web Gui** (otherwise known as the Web Console).

   This component must be installed on a Windows NT/2000-based Managed Node or the TMR Server. Optionally, an instance of the IBM HTTP Server is installed with the Web Console.

   As shown in Figure 2-6 on page 45, we installed the Web Console on the TMR Server (x220svr), which will also serve as the Web Transaction Performance Tivoli Internet Management Server, so additional steps to
demonstrate how to implement both the Enterprise Transaction Performance and Web Transaction Performance Web components are provided.

Figure 2-6   Enterprise Transaction Performance Web Console installation
Before the installation starts, you will be prompted for additional information related to the IBM HTTP Server installation, as shown in Figure 2-7.

![Figure 2-7 Setting installation parameters for the ETP Web Console](image)

If you want to use an existing instance of the IBM HTTP Server (the service name has to match IBM HTTP Server exactly), disregard the name and password fields on the Install Options dialog, and check the checkbox in the Selectable Installation Options section at the bottom of the dialog.

To let Tivoli install an instance of the IBM HTTP Server as part of the Enterprise Transaction Performance Web Console installation, supply a logon name to be used to start the IBM HTTP Server service. The logon name and password must have administrative privileges on the NT machine and senior role in the TMR.

**Note:** If you have an existing IBM HTTP Server registered (for example, the one that was installed with the IBM Tivoli Monitor Web Health Console) and you select to let Tivoli install a new instance of the IBM HTTP Server as part of the Web Console installation, the installation will fail unless you rename the registry key HKLM\Software\IBM\HTTP Server \1.3.19 to something like HKLM\Software\IBM\WHC-HTTP Server\1.3.19.

You should also be sure that the IBMHTTPServer and IBMHTTPAdministration services do not exist in the Registry and that the target directory C:\IBMHTTPSERVER has been deleted.

6. If you are installing the Web Console on a system that is or will be hosting the IBM Tivoli Monitor Web Health Console or other applications using ports 80
7. If you are installing the Web Console on a system that is or will be hosting the Web Transaction Performance Tivoli Internet Management Server or any other application that uses port 8007 to communicate to a Tomcat server, use the procedure described in “Changing the ports used by Tomcat” on page 56 to assign new ports for the Web Console to use for Tomcat communication.

8. Launch the Enterprise Transaction Performance Web Console through your browser using:

http://<fullyqualifieddomain>:88

9. Installation of the other components is optional and may be accomplished in any order by following the procedures outlined in the IBM Tivoli Monitoring for Transaction Performance: Enterprise Transaction Performance User’s Guide, Version 5.1, GC23-4803.

2.2.4 Web Transaction Performance installation

Before installing the Web Transaction Performance component, you must create a database for the administration server. DB2 or Oracle may be used. In this example DB2 databases have been used, and it is assumed that a DB2 Server is installed on the TMR Server machine where the Tivoli Internet Management Server component will be installed. Follow the procedure outlined in 2.2.6, “Tivoli Internet Management Server DB2 database creation” on page 58 to define and initiate the TIMS database.

Web Transaction Performance component installation

The installation of the Web Transaction Performance component is very straightforward and well documented in IBM Tivoli Monitoring for Transaction Performance: Web Transaction Performance Installation Guide Version 5.1.0, GC23-4801.

1. After the initial screen, accept the license agreement.

2. Accept the default installation directory.

3. For the Internet Management Server Configuration, set the Port Number to 81 as shown in Figure 2-8 on page 48. As already discussed, we will be running several Web servers on the TMR Server and must configure each of them to listen on a different port.
4. As shown in Figure 2-9, enter the Database Configuration for the TIMS database you created previously.
5. In the next dialog, be sure that **Enable ARM Data Retrieval** is checked, as shown in Figure 2-10. This is an integration point with the Enterprise Transaction Performance Component.

![Image of Tivoli Internet Management Server setup](image)

**Figure 2-10  Enabling ARM collection in Tivoli Internet Management Server**

We will install the Enterprise Transaction Performance Web Console on the same machine, but listening on port 88. So enter http://<fullyqualifiedhostname>:88 for the URL.

**Note:** If you need to change the URL of the Enterprise Transaction Performance Web Console after installation, you can do so by editing the transperf.ent.reportconsoleurl entry:

```
transperf.ent.reportconsoleurl=http://x220svr.itsc.austin.ibm.com:88
```

This is found in the server.properties file in the following directory of the Tivoli Internet Management Server system (the TMR in this example):

```
C:\Program Files\Tivoli\Internet\ManagementServer\TIMS\lib\properties
```

6. Click **Next** to allow the installation to complete.
2.2.5 Procedures for configuring TMTP console servers

The following contains detailed descriptions of the procedures used to manipulate service names, and ports for the IBM HTTP Server instances used by the consoles.

Moving an existing IBM HTTP Server

These steps move an existing IBM HTTP Server from the default location:

1. Rename the IBM HTTP Server installation directory.
2. Update the Windows NT/2000 Registry.
3. Update the references from the Windows Desktop.
4. Update the IBM HTTP Server configuration files.
5. Restart the system to activate the changes.

**Important:** Before starting any of these activities, be sure that the IBM HTTP Server and the IBM HTTP Administration services have been stopped and that you have valid backups of the IBM HTTP Server installation directory and the Windows Registry.

1. Rename the directory in which IBM HTTP Server is installed (usually C:\IBMHTTPSERVER) to a name of your own choice. In this example we use C:\IBMHTTPSERVER-OLD and assume that we are updating the service names IBM HTTP Server-WHC and IBM HTTP Administration-WHC.

2. Update registry entries for IBM HTTP Server and IBM HTTP Administration:
   a. Run `regedt32` from the windows command prompt to update key registry parameters for the key values for the IBM HTTP Server service

   HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\IBM HTTP Server-WHC:

   i. Change the ImagePath key to the new path to apache.exe. Our example uses C:\IBMHTTPSERVER-OLD\apache.exe, as shown in Figure 2-11 on page 51.
ii. Change the ConfigArgs parameter of the Parameters key. Double-click the ConfigArgs parameter and update the paths to reflect the changes made to the directory name of the IBM HTTP Server installation directory. In our example, the value C:\IBMHTTPSERVER is changed to C:\IBMHTTPSERVER-OLD in two instances, as shown in Figure 2-12 on page 52.
b. Repeat these steps for the IBM HTTP Administration Server key.

c. To update the uninstall information in the Registry, select the HKLM\Software\Microsoft\windows\CurrentVersion\Uninstall\HTTP Server <version> key and update the UninstallString parameter (the path appears twice) to point to the new base directory of the IBM HTTP Server, as shown in Figure 2-13.
3. Edit the properties of the Start Key Management Utility and the Documentation menu items to reflect the changes to the path name.
   a. Go to the directory:
      \%systemdrive\%\Documents and Settings\ALL Users\Start Menu\Programs\IBM HTTP Server
   b. Edit the properties for Start Key Management Utility and Documentation. Update the path names in the Target value to the new directory C:\IBMHTTPSERVER-OLD.

4. Update the IBM HTTP Server configuration files.
   Update the path information in all of the configuration files governing the operation of the IBM HTTP Server and the IBM HTTP Administration Server.
   a. Use the Windows search facility to find all of the files under the new IBM HTTP Server directory (C:\IBMHTTPSERVER-OLD in our example) that contain the old path, C:/IBMHTTPSERVER. At a minimum you should update httpd.conf and admin.conf.

   **Note:** The path information uses the UNIX notation, forward slashes (/), and not the usual backslashes (\) used in Windows.

   Use an editor to update all instances of the old path name to point to the new name of the IBM HTTP Server base directory.

5. Now restart the system to put your changes into effect.

**Renaming the HTTP Server product registration**
If you have an existing IBM HTTP Server registered, such as the one that was installed with the IBM Tivoli Monitor Web Health Console, and you want to install a new instance of the IBM HTTP Server—for example, as part of the Web Console installation—the installation will fail unless you rename the registry key HKLM\Software\IBM\HTTP Server \1.3.19 to something like HKLM\Software\IBM\WHC-HTTP Server\1.3.19.

**Renaming IBM HTTP Server service names**
The following steps rename existing IBM HTTP Server services and allocate new ports. If you also need to move the server to a new location, you should follow the procedure described in “Enterprise Transaction Performance installation” on page 43 after completing these steps:

1. Update the Windows NT/2000 Registry.
2. Update the references from the Windows Desktop.
3. Update the IBM HTTP Server configuration files.
4. Restart the system to activate the changes.
1. Update registry entries for IBM HTTP Server and IBM HTTP Administration:
   a. Run `regedit` from the Windows command prompt to change the name of the registry entry for IBM HTTP Server. This will avoid a conflict of names when installing IBM HTTP Server for the Web Console.
      i. Select the IBM HTTP Server key under `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services`.
      ii. Make the following changes to the key:
         • Change the name of IBM HTTP Server key to IBM HTTP Server-WHC.
         • Under this key change the DisplayName value by appending “-WHC” to the end, as shown in Figure 2-14.
         • Change the ObjectName key to LocalSystem from “\<someuser>” as shown in Figure 2-14.

   b. Repeat these steps for the IBM HTTP Administration Server key.

2. Edit the Start/Stop Program menu items to reflect the changes to the service names:
   a. Go to the directory:
      %systemdrive%\Documents and Settings\ALL Users\Start Menu\Programs\IBM HTTP Server

**Important:** Before starting any of these activities, be sure to stop the IBM HTTP Server as well as the IBM HTTP Administration services, and have valid backups of the IBM HTTP Server installation directory and the Registry.
b. Edit the properties for Start HTTP Administration, Start HTTP Server, Stop HTTP Administration and Stop HTTP Server to reflect the new service names. In the Properties dialog’s Shortcuts tab, change the Service names from IBM HTTP Server to IBM HTTP Server-WHC and IBM HTTP Administration to IBM HTTP Administration-WHC, as in Figure 2-15.

![Start HTTP Administration Properties](image)

**Figure 2-15  Start/Stop IBM HTTP Service shortcut properties**

**Changing the ports used by IBM HTTP Server services**

These steps are required to allocate new ports to existing IBM HTTP Server services. The procedure outline is:

1. Update the IBM HTTP Server configuration files.
2. Restart the system to activate the changes.
1. Update the IBM HTTP Server configuration files as shown in Example 2-1. To change the port numbers that the IBM HTTP Server and IBM HTTP Administration services listens to, change the port parameter in the configuration files:
   
a. Edit the http.conf file located in the conf subdirectory of the named IBM HTTP Server base directory. Locate the entry Port 80, and change it to Port 88 or any other port that you wish to use.

b. Edit the admin.conf file located in the conf subdirectory of the renamed IBM HTTP Server base directory. Locate the entry Port 8008, and change it to Port 8009 or any other port that you wish to use.

Example 2-1 Updating port information in the httpd.conf IBM HTTP Server control file

```
# LoadModule proxy_module modules/ApacheModuleProxy.dll

# Port: The port the standalone listens to.
# Port 88
AfpaEnable
AfpaCache on
AfpaLogFile "c:/IBMHTTPServer/logs/afpalog" V-ECLF

# HostnameLookups: Log the names of clients or just their IP numbers
#   e.g. www.apache.org (on) or 204.62.129.132 (off)
...
```

2. Now restart the IBM HTTP Server and IBM HTTP Administration Server services to use the newly assigned ports.

### Changing the ports used by Tomcat

To run the Web Transaction Performance component and the Enterprise Transaction Performance component on the same machine, make these changes:

1. Stop the TIMS HTTP Server Service.
2. Modify the Tomcat configuration for the Web Console server: In the following configuration files, the Tomcat port must be changed to an open port other than 8007. In our example we chose port 8006.

All of the relevant Tomcat configuration files for the Web Console can be found in the %BINDIR%\TME\MAR\web\tomcat\conf directory. Apply the changes described in Table 2-4.

Table 2-4 File changes needed to reassign Tomcat ports

<table>
<thead>
<tr>
<th>File</th>
<th>Change:</th>
<th>To:</th>
</tr>
</thead>
<tbody>
<tr>
<td>server.xml</td>
<td>&lt;Parameter name=&quot;port&quot; value=&quot;8007&quot;/&gt;</td>
<td>&lt;Parameter name=&quot;port&quot; value=&quot;8006&quot;/&gt;</td>
</tr>
<tr>
<td>tomcat.conf</td>
<td>port value = 8007</td>
<td>port value = 8006</td>
</tr>
<tr>
<td>tomcat.properties</td>
<td>Apache JServ port value = 8007</td>
<td>Apache JServ port value = 8006</td>
</tr>
<tr>
<td>tomcat-apache.conf</td>
<td>ApJServDefaultPort = 8007</td>
<td>ApJServDefaultPort = 8006</td>
</tr>
<tr>
<td>workers.properties</td>
<td>worker.ajp12port = 8007</td>
<td>worker.ajp12port = 8006</td>
</tr>
<tr>
<td>wrapper.properties</td>
<td>wrapper.shutdown = 8007</td>
<td>wrapper.shutdown = 8006</td>
</tr>
</tbody>
</table>

Example 2-2 Required changes to server.xml to change Tomcat ports

Change:

```xml
<Connector className="org.apache.tomcat.service.SimpleTcpConnector">
  <Parameter name="handler" value="org.apache.tomcat.service.connector.Ajp12ConnectionHandler"/>
  <Parameter name="port" value="8007"/>
</Connector>
```

to:

```xml
<Connector className="org.apache.tomcat.service.SimpleTcpConnector">
  <Parameter name="port" value="8006"/>
</Connector>
```
3. Now restart the IBM HTTP Server service to use the newly assigned ports.

### 2.2.6 Tivoli Internet Management Server DB2 database creation

To create and configure the database for the Web Transaction Performance component using DB2, the following activities have to be completed:

1. Create and catalog the database.
2. Add a 32k bufferpool.
3. Create new regular tablespace.
4. Create new temporary tablespace.
5. Add authorities to use the regular tablespace.

The DB2 Control Center will be used to perform all the tasks involved.

**Create and catalog the database**

To create the Tivoli Internet Management Server control database:

1. Launch and log on to the DB2 Control Center.
2. Expand the tree until you reach the Databases folder under the DB2 instance.
3. Right-click on the Databases folder, and select **Create -> Database Using Wizard**.
4. Enter TIMS for both the database name and the database alias (comments are optional) as shown in Example 2-16 on page 59. The default values under the other tabs will be fine, so click **Finished** to create the database.
Create a 32k bufferpool

Once the database is created, we create a 32K page bufferpool to accommodate the needs of the Tivoli Internet Management Server application.

1. Expand the new TIMS database, and right-click the Bufferpools folder. Select Create to open the Create Buffer Pool dialog, and enter the values shown in Figure 2-17. Click OK to create the bufferpool.
Create a new regular table space

Now create the table spaces in which the data will be stored, and then delete the default tablespace (USERSPACE1), which was created during database creation.

1. Expand the new TIMS database and click the Tablespace folder.
2. Right-click USERSPACE1 and select DROP.
3. Right-click the Tablespace folder and select Create -> Tablespace. This will open the dialog shown in Figure 2-18.
   a. Enter USERSPACE1 as the Tablespace name
   b. Choose Regular and System, as shown.
   c. Click Add to specify where to store the data, and enter c:/db2/tims.dat for the container. (The specified name has to be that of an existing directory.) Click OK to return to the Create Table Space dialog.

![Create Table Space dialog](image)

*Figure 2-18 Creating a DB2 table space*
d. Click **Advanced** to open the window shown in Figure 2-19, and populate the fields as shown. Click **OK** to return to the previous dialog.

![Figure 2-19 Setting buffer pool usage for a tablespace](image)

**Note:** If you get an error message stating that the buffer pool is not yet active, stop and start the database instance (DB2) and retry the table space creation.

### Create a new 32k temporary table space

The Tivoli Internet Management Server application also needs temporary table space with a 32k page size. To create it:

1. Right-click the Tablespace folder, and select **Create -> Tablespace**. Populate the dialog with this information:
   a. Enter TIMS_32TMP for the name.
   b. Select **System temporary** for the type.
   c. Select **System** for the Space management.
   d. Click **Add**. Enter `c:/db2/tims.tmp` for the container. (The name specified must be that of an existing directory.) Click **OK** to close the window.
   e. Click **Advanced**. In the window that opens:
      i. Enter 32 for Page size.
      ii. Enter 64 for Extent size.
      iii. Enter 32 for Prefetch size.
      iv. Select TIMS_32k for the Bufferpool.
v. The other fields should be filled in for you.

vi. Click OK.

f. Click OK to create the tablespace.

**Authorize use of the table spaces**
We must authorize the user for the user ID that will be supplied during installation of Web Transaction Performance to be able to access the newly created table spaces. In our example the user ID we will specify during Web Transaction Performance installation is DB2.

Note: The user ID you plan to use has to be defined in the Windows environment. If it is not, create it using the normal user definition interface: **Computer Management -> Local Users and Groups**.

1. Expand the User and Group Object folder.
2. Right-click the DB User folder and select *Add*.
   a. Select DB2 for the User, as shown in Figure 2-20.
   a. Check all boxes in the Authorities box to grant all authorities.

![Add User](image)

**Figure 2-20** Granting user authorities

62 Unveil Your e-business Transaction Performance with IBM TMTP 5.1
b. Click the Tablespace tab to open the Add User dialog shown in Figure 2-21.

![Figure 2-21  Granting tablespace authorities](image)

- Click **Add Tablespace**.
  - Select **USERSPACE1**.
  - Click **Apply**.
- Select **USERSPACE1**.
- Click **Grant All** and, in the window that opens, select **with Grant option**.
- Click **OK** to close the dialog.
- Click **OK** to complete the table space addition.
Transaction breakdown: root cause analysis

This chapter discusses methods and tools provided by IBM Tivoli Monitoring for Transaction Performance to perform a detailed analysis of transaction performance data and to identify the root causes of performance problems in order to keep high service levels.

“Accomplishing a high level of customer satisfaction requires the IT function of an organization to have a full understanding of and insight into the different aspects of its operation and performance in terms of efficiency and effectiveness.”

This quote from “Introducing IBM Tivoli Service Level Advisor” sums up the business reasons for conducting root cause analysis. IBM Tivoli Monitoring for Transaction Performance is developed to measure the efficiency and effectiveness of the performance of the formerly black-box operations called middleware. Middleware is the term for general-purpose software that serves as a bridge between the business user and the business information that can make a difference in today’s competitive economy. By analyzing the performance and availability behavior of the various middleware components over time, important diagnostic trends are revealed and solutions can be applied during the IT Service Delivery Cycle. More important, the newly available information can be used in planning for future investments that will make a growth difference in the profits and success of the business enterprise.
In the previous chapters, we described the IBM Tivoli Monitoring for Transaction Performance components and architectures that enable measuring and monitoring transaction performance in order to guarantee reasonable service levels and improve the business.

In this chapter, we will provide specific examples of how IBM Tivoli Monitoring for Transaction Performance can be applied to determine the root cause of service delivery failures or to forecast potential problem areas for service delivery prior to actual failure. We will briefly describe some Quality of Service reports that are popular with customers and then focus the main portion of this chapter on the STI tools that are now available to provide ARM data retrieval on WebSphere Application Server Version 5.0 applications.
3.1 Transaction breakdown overview

No matter whether QoS or STI endpoints are being used to capture the transaction performance metrics of a transaction, historically everything that takes place behind the HTTP Server has been hidden from the transaction performance monitors.

An architecture and a first implementation of a solution that unveils the content of the hidden information has been provided with IBM Tivoli Monitoring for Transaction Performance Version 5.1. The STI Playback engine now is capable of generating transaction correlators that may be passed on to the subsequent middleware components such as IBM HTTP Server, WebSphere Application Server, MQ Series, DB2, and many more in order to be used in capturing transaction performance metrics specific to their particular environments. Using the common Application Response Measurement (ARM) interface, response time metrics from multiple environments may now be consolidated to enable true end-to-end transaction breakdown in terms of performance and resource usage. This will become more and more relevant as the industry moves forward into grid and on-demand computing, where tasks do not necessarily use the same resources at all times.

As this book is being written, only WebSphere Application Server Version 5.0 supports this new transaction performance measuring architecture, and the implementation is somewhat limited. However, it provides a valuable first step into the transaction monitoring world and will provide experiences to be used for later versions of both the IBM Tivoli Monitoring for Transaction Performance components, the middleware components, and applications supporting e-business applications.

Restriction: Only STI is capable of generating transaction correlators for Web Transaction Performance.

3.1.1 Quality of Service tools

Quality of Service (QoS) monitors are designed to provide a high-level picture of end-user experience with certain user-specified Web pages, showing page render time, backend service time, and round-trip time for each user access of the specified Web page. Normally, QoS jobs would focus on the Web site’s home page, any frequently accessed Web pages at the site, as well as any heavy-processing Web pages such as a database query page or checkout transaction pages.

Refer to the redbook *Tivoli Web Solutions: Managing Web Services and Beyond*, SG24-6049 for more information on best practices for using QoS monitoring jobs.
QoS provides three real-time graphs for current activity with a moving 60-minute window that is refreshed regularly depending on the frequency of the scheduled QoS Jobs. These graphs show average user experience time, average back-end service time, and average page-render time.

The following reports in the Tivoli Decision Support Guide are also noteworthy:

- **QoS Customer Experience**: Shows the number of requests that violated your company’s round-trip time policy in comparison to the number of requests your site had each day.
- **QoS Trends**: Shows a comparison of round trip time to page render time and backend service time. Web masters can use this report to identify bottlenecks in the system. For example, if round trip time is high and the backend service time is also high, then there is a bottleneck at the Web server.
- **QoS Violations: Backend Service Time**: Shows the number of requests that violated your company’s backend service time policy in comparison to the number of requests your site had each day.
- **QoS Violations: Page Render Time**: Shows the number of requests that violated your company’s page-render time policy in comparison to the number of requests your site had each day.
- **QoS Violations Summary**: Shows the number of requests your site had each day in comparison to the number of requests that violated your company’s round-trip time, page render time, and backend service time policies.

### 3.1.2 Synthetic transaction availability tools

The Web Transaction Performance Synthetic Transaction Investigator (STI) provides the same functionality as QoS with one major difference: STI is based on recording/playback techniques and can be used to invoke the exact same transaction (known as a reference transaction) over and over again from any point in the infrastructure or Internet. An STI job can be used to test the availability of a selected set of Web pages that are used frequently. It will test for an aggregate page size to ensure that the Web page can load quickly. This is especially important for the Web site’s home page. It will test for any error messages in the content as well as test for non-200 HTTP return codes. Finally, it will test for specified response time violations on the Web page availability.

Refer to *Tivoli Web Solutions: Managing Web Services and Beyond*, SG24-6049 for more information about best practices for using SI and STI.

In the previous versions of the product, real-time STI-Event Drill Down reports were customized through the modification of the ReportConfig.properties file, and this option is still available to customers in Web Transaction Performance.
One STI report in the Tivoli Decision Support Guide in particular is notable for management of Web sites using STI.

- **STI Transaction Availability** shows the percentage of transactions attempted by the Synthetic Transaction Investigator that were completed successfully. Web administrators can use this report to quickly see whether the Web site is processing transactions as expected. Increase the time variable in the report to see the trend of successful transactions.

### New synthetic transaction reports in Web Transaction Performance

The following Synthetic Transaction Reports are newly available with Web Transaction Performance Version 5.1:

- Round Trip Graph
- Average Response Time Graph
- Availability Graph
- Transaction Details Table
- Overall Transaction Summary Report
- Page Analyzer Viewer Report

These reports are now available as real-time graphs or as time-range reports. They provide various views for performance success/failure and availability success/failure related to specific STI jobs and endpoints. There is even a view for the five slowest transactions using round-trip time among all the STI jobs.

### Page Analyzer Viewer

The Page Analyzer Viewer (PAV) is a new reporting feature in Web Transaction Performance that, unlike other STI reports that now use JPEG images, still uses Java applets (Jchart). When used for the first time, PAV may ask you to install a Java plug-in for your browser, namely Version 1.3.1_06 to support these applets, before your reports can be displayed.

The Page Analyzer Viewer report provides an interesting transaction breakdown of total time for each of the transactions identified in the transaction detail report. This transaction breakdown occurs for such objects as servlets, JavaScript, DNS, CSS, GIF, JPEG, HTML, and Macromedia Flash, and is colored-coded for SOCKS connect, SSL connect, host name resolution, server response, delivery connect, and total request time. The Web Detailer data displayed by the Page Analyzer Viewer report is not stored in the database in the current version of Web Transaction Performance, so this data is not available to the Tivoli Decision Support Guide reporting.
3.2 Transaction breakdown in detail

Each of the newly available reports mentioned in “New synthetic transaction reports in Web Transaction Performance” on page 69 (except the Page Analyzer Viewer report) now has a supplemental feature: a new type of STI reporting that enables retrieval and reporting of ARM performance data for specific WebSphere Application Server transactions being simulated by STI. This new feature comes in the form of an ARM Report hyperlink on the Synthetic Transaction Investigator statistics report, and its focus is on performance and the measurement of backend service time.

The backend service time provided by Web Transaction Performance Quality of Service monitoring jobs, while good at giving a top-level view of any given Web transaction launched by a specific user, usually does not provide enough detail to determine where the real problems for Web transaction access are being experienced. For this detail, we need the transaction breakdown provided by ARM instrumentation and ARM correlation.

ARM instrumentation refers to the strategic placing of certain ARM API calls, such as arm_init, arm_start, arm_update, and arm_stop, in the source code for a customer's chosen application. With Enterprise Transaction Performance, this source code can be JavaScript code or C++ code. The ARM API calls provide a frame around the specific business transactions that must be monitored in the application for success or failure and for completion response time. With the release of Enterprise Transaction Performance, it is now possible to work with pre-ARM instrumented server software such as WebSphere Application Server Version 5.0, so that it is no longer necessary to modify existing WebSphere applications in order to add the ARM API calls.

3.2.1 ARM correlation

ARM correlation is the method by which parent transactions are mapped to their respective child transactions across multiple processes and multiple servers. ARM correlation is not to be confused with Event Correlation in the Tivoli Enterprise Console provided by Web Transaction Performance or with the correlation of variables in VuGen scripts. When users select enable ARM correlation, they are requesting that pre-defined correlators be generated to trace the path and measure the performance of a specific STI playback job for a specific WebSphere 5.0 application or a specific user application that has been coded with ARM API calls. The latter was required for gathering transaction performance data when using versions of WebSphere Application Server prior to Version 5.0, which were not ARM-instrumented.
We will not discuss the ARMing of user-applications in C++ or JavaScript code as this topic is discussed in other documentation. For more details, refer to the redbook *Tivoli Application Performance Management Version 2.0 and Beyond*, SG24-6048. Another good reference for this topic is the ARMjava.doc file, which describes how to make ARM API calls from Java applications and is available on the IBM Support Web site http://www-1.ibm.com/support/search/index.html. Also available is the ArmJniUnitTest class sample code, an extract from the armjni.jar file, which helps in understanding how to use the libarmjni library.

In this publication, we will focus specifically on STI with ARM correlation data for applications based on WebSphere Application Server Version 5.0.

Measured response times from Enterprise Transaction Performance Transaction Simulation scripts can provide a set of baseline values with which to compare real end-user experience in a production environment. Enterprise Transaction Performance Client Capture can detect when changes occur in a limited product-defined set of applications and, by sensing pattern change in signatures for business transactions, generate ARM API calls to measure response time on those changes. However, the most important tool from the standpoint of root cause analysis is the Enterprise Transaction Performance ARM-instrumentation and correlation now available as a feature with the Synthetic Transaction Investigator feature of Web Transaction Performance.
3.2.2 Capturing correlated transaction performance data

The transaction correlation process is outlined in Figure 3-1.

1. First, a transaction or series of transactions is recorded using the STI recorder. During recording, no considerations or actions are needed to activate transaction correlation support.

![Figure 3-1 Transaction breakdown architecture overview](image)

2. When the transaction has been successfully saved in the TIMS database, the Tivoli Internet Management Server Administrator defines and schedules an STI transaction playback job using the saved transaction generated in step 1.

   **Note:** During the creation or modification of the STI transaction playback, ARM correlation must be activated using the checkbox on the Constraints page of the Transaction Playback page.

3. When the transaction playback job has been defined and scheduled, it will start executing at the selected playback endpoints and start interacting with the Web application that was used during the recording of the transaction.

   During playback all of the original HTTP requests are sent to the target of the transaction just as they were recorded. However, if ARM Correlation is enabled, a unique transaction correlator is appended to the HTTP requests.
This correlator will be used later to combine transaction performance data captured by STI and data captured by WebSphere Application Server.

4. When the HTTP requests, including the appended transaction correlators, reach the WebSphere Application Server hosting the transaction's target application, and this server has been configured to support transaction correlation, WebSphere services captures the information regarding the performance of the specific transaction as seen by the server. WebSphere Application Server then stores this data along with the transaction correlator, which was passed on in the HTTP request.

5. The response from the Web application is received by the STI Recorder, which then stores the transaction response time metrics as seen by STI along with the transaction correlator in the TIMS database.

The overall transaction performance data may be viewed through the Tivoli Internet Management Server console; however, the correlation between the STI data and the WebSphere Application Server data is not available until the WebSphere Application Server performance data has been gathered by the Enterprise Transaction Performance endpoint and transferred to the central Tivoli Application Performance Management database.

6. To be able to combine the gathered data, the WebSphere Application Server's performance data is picked up by the Enterprise Transaction Performance endpoint and sent to the central Tivoli Application Performance Management database. The frequency by which this occurs is determined by the configuration of the Enterprise Transaction Performance endpoint at the WebSphere Application Server.

Using the Enterprise Transaction Performance Web Console it is possible to view the performance data online for the part of the transaction that was executed in the WebSphere Application Server environment.

7. Once the WebSphere Application Server data becomes available in the Tivoli Application Performance Management database, the Enterprise Transaction Performance Web Console may be launched from the STI GUI, revealing to the user of the Tivoli Internet Management Server Console the details captured by WebSphere Application Server.

### 3.2.3 How it works: the details

Using the APIs under the ARM 2.0 standard provides for the linking of subtransactions to parent transactions in order to determine which subtransaction is causing the parent transaction to perform slowly. This process is called correlation and is implemented through what is called an ARM correlator. In previous versions of Tivoli Application Performance Management 2.x as well as in Enterprise Transaction Performance, an ARMed application can request a correlator from the tapmagent through the arm_start API call or assign
a correlator and pass the newly assigned correlator to the arm_start API call. This ARM correlator is what we refer to as the parent transaction correlator or parent correlator. STI now passes this assigned ARM correlator to WebSphere Application Server during transaction simulation so that WebSphere will not assign its own ARM correlator for the ARM API calls but will use this STI-generated ARM correlator instead. Any child correlators assigned during processing will be linked to this parent transaction correlator.

**Example 3-1  Example of ARM_CORRELATOR string in WebSphere trace.log file**

```
ARM_CORRELATOR = 001E01018235063E0015091482AA178DD0A0206A00080903BB8A8B616410
```

The STI-generated ARM correlator will be stored in the Web Transaction Performance TIMS database in the ARMCORRELATOR column of the STMSUBSTATISTICS table where it can be retrieved during later ARM report inquiries and passed to the Enterprise Transaction Performance Web Console along with the start date for the transaction in order to retrieve ARM-related data. Refer to Example 3-2, in which STI provides this ARM correlator to the Enterprise Transaction Performance Web Console to generate the ARM report portion of the STI Statistics Report.

**Example 3-2  Example of the STI ARM Report URL used in invoking Web Console**

```
```

Having received the STI-generated ARM correlator, WebSphere Application Server will use that correlator in its outer blocks to frame its application processing and measure performance of the code within those frames. It then passes transaction information to the local tapmagent along with the STI-generated parent correlator and any other correlators linked to that parent correlator.

There is a delay built into this process in that the tapmagent will not upload the correlator data to the gateway and to the TAPM RIM immediately after WebSphere Application Server provides the information to the tapmagent. So, in some cases, you will want to issue the `wmarsenddata -i <ep>` command to force an immediate upload of instance data to the database tables so that the correlators can be available for the ARM-report inquiry using the hyperlink.
Chapter 3. Transaction breakdown: root cause analysis

Example 3-3 illustrates how the STI ARM Report inquiry is performed against the TAPM RIM database. The SQL `from` clause is interesting in the way it sets up aliases for the CORRELATOR table, namely the CORRELATORS_TABLE and PARENT_CORRELATORS_TABLE. The SQL `where` clause shows that we are matching for TX_INSTANCE.HOST_ID, which is the endpoint name associated with the transaction retrieved by the STI-provided correlator.

Example 3-3   SQL commands generated in tapm.log by STI hyperlink query

```
Dec 16 17:51:24:890 marGetRawDataDB: -------> function entry point <-------
Dec 16 17:51:25:421 buildQuery ( query type = QUERY_SINGLE_INSTANCE):
Dec 16 17:51:25:437 SQLQuery: SELECT APPLICATIONS.APPL_NAME, USERS.USER_NAME,
                          HOSTS.HOST_NAME, HOSTS.TME_OBJECT_LABEL, HOSTS.HOST_TIME_ZONE, HOSTS.DST,
                          TRANSACTIONS.TX_NAME, TRANSACTIONS.TX_DETAIL, CORRELATORS_TABLE.TX_CORRELATOR
                          AS CORRELATOR, PARENT_CORRELATORS_TABLE.TX_CORRELATOR AS PARENT_CORRELATOR,
                          TX_INSTANCE.INSTANCE_ID, TX_INSTANCE.STATUS, TX_INSTANCE.STOP_DATE,
                          TX_INSTANCE.STOP_TIME_S, TX_INSTANCE.IP_ADDRESS, TX_INSTANCE.RT_MS,
                          TX_INSTANCE.RT_NS
Dec 16 17:51:25:437 SQLQuery: from
                          APPLICATIONS,TRANSACTIONS,HOSTS,USERS,UUID_INFO,TX_INSTANCE,
                          CORRELATOR CORRELATORS_TABLE, CORRELATOR PARENT_CORRELATORS_TABLE
Dec 16 17:51:25:453 SQLQuery: where TX_INSTANCE.CORRELATOR_ID =
                          CORRELATORS_TABLE.CORRELATOR_ID and TX_INSTANCE.P_CORRELATOR_ID =
                          PARENT_CORRELATORS_TABLE.CORRELATOR_ID and TRANSACTIONS.TX_ID=UUID_INFO.TX_ID
                          and APPLICATIONS.APPL_ID=UUID_INFO.APPL_ID and
                          TX_INSTANCE.INFO_ID=UUID_INFO.INFO_ID and
                          HOST_ID=TX_INSTANCE.HOST_ID and
                          USERS.USER_ID=TX_INSTANCE.USER_ID and
                          CORRELATORS_TABLE.TX_CORRELATOR='001E01018235063E0015090C82AA178D7BB4665A00080903BB8A992EC78B'
Dec 16 17:51:29:218 buildQuery: building children query......(Parent found)
Dec 16 17:51:29:218 SQLQuery: SELECT APPLICATIONS.APPL_NAME, USERS.USER_NAME,
                          HOSTS.HOST_NAME, HOSTS.TME_OBJECT_LABEL, HOSTS.HOST_TIME_ZONE, HOSTS.DST,
                          TRANSACTIONS.TX_NAME, TRANSACTIONS.TX_DETAIL, B_CORR.TX_CORRELATOR,
                          TX_INSTANCE.INSTANCE_ID, TX_INSTANCE.STATUS, TX_INSTANCE.STOP_DATE,
                          TX_INSTANCE.STOP_TIME_S, TX_INSTANCE.IP_ADDRESS, TX_INSTANCE.RT_MS, TX_INSTANCE.RT_NS,
```

Note: The URL in Example 3-2 does not include an endpoint name or host name. Actually, when doing searches in the interactive Enterprise Transaction Performance Web Console, the endpoint name is not required as a parameter in the SQL inquiry even though it is required input for the interactive session. The endpoint name or host name provided is merely the starting point for the inquiry in Web Console; actually transaction information from all endpoints related to the transaction will be retrieved by the Web Console inquiry and not just transaction information from the specified starting endpoint.
Unlike Version 4.02, WebSphere Application Server Version 5.0 comes pre-instrumented for ARM. If WebSphere Application Server is allowed to assign its own ARM correlator, it would not be possible to trace the correlator and tie it to a specific parent transaction, so it is important that it use the ARM correlator provided by STI in order to link parent correlators to child correlators in TAPM RIM. STI does not request the Enterprise Transaction Performance tapmagent to create ARM correlators during processing, but instead creates its own ARM correlator to pass to WebSphere Application Server.

In order to understand how the correlation is handled internally, we look briefly at some of the details for storing ARM correlators and transaction instance data.

Prior to uploading instance data to the TAPM RIM database, transactions from an endpoint are bundled into transaction classes at the gateway. Each transaction class may have one or several hundred insertion rows. The default setting of 100 for the DBMaxInstanceRowNo variable on the gateway means that when there are more than 100 transactions in one of the transaction classes, special packets are created for the row insertion process. Each transaction class updates the TX_INSTANCE, TX_INT, TX_FLOAT, TX_STR, and CORRELATOR tables with from one to several hundred rows of data for these tables. Transaction classes are important because they help organize the TAPM RIM updates into sets of transactions related to a specific ARM correlator.

The TAPM RIM database has a table named CORRELATOR that stores two items, the CORRELATOR_ID (38 characters) and the TX_CORRELATOR (60 characters). The CORRELATOR_ID is a unique number generated during the row insertion process and calculated by retrieval from the TAPM_ID table, which holds the last generated value for CORRELATOR_ID as well as other ID values for the other tables such as INFO_ID, HOST_ID, TX_ID, and INSTANCE_ID. Any row inserted into CORRELATOR is checked by the PK_CORRELATOR constraint in TAPM RIM to enforce uniqueness. Each generated CORRELATOR_ID is stored in the TX_INSTANCE table (the table holding instance data) in order to provide a pointer to the TX_CORRELATOR value.
stored in the CORRELATOR table. This value is stored in the TX_CORRELATOR column, where it can be retrieved and used to map data from the TAPM RIM database to the WebSphere Application Server database for reporting purposes.

The ARM correlator shown in Example 3-3 on page 75 could just as easily be passed to the new `wmargetrawdbdata` command for use with the `-r` option on that command as follows:

```shell
wmargetrawdbdata -f Data.xml -s 2002-10-13 -r 001E01018235063E0015090C82AA178D7BB4665A00080903BBBA992ECE7B
```

The results of this command will show the details of a single instance filtered only by means of a date and a correlator and will list the endpoint name for the associated transaction on the first output line of the results. Issuing a second command with the `-r` and `-c` options will be necessary in order to retrieve the correlated child transactions associated with the parent correlator. This is also the pattern we see in the SQL commands listed in Example 3-3 on page 75.

```shell
wmargetrawdbdata -f Data.xml -s 2002-10-13 -c -r 001E01018235063E0015090C82AA178D7BB4665A00080903BBBA992ECE7B
```

**Example 3-4  `wmargetrawdbdata` output showing detailed transaction data**

Transaction: URI (/PlantsByWebSphere)
---
Tx Instance ID 1048
Timestamp (YYYY MM DD hh:mm:ss): 2002 11 18 16:01:22
Response Time: 2 (ms), 911450 (ns)
Transaction status: good
IPAddress : 9.3.4.58
Correlator 001E01018235063E00330878000015F000000040080903BBBA992ECE7B
Parent Correlator 001E01018235063E0015090C82AA178D7BB4665A00080903BBBA992ECE7B

Even though the agent’s IP address is included as part of the ARM correlator itself, Enterprise Transaction Performance does not parse the IP address from the correlator to identify the endpoint where the transaction performs, but instead stores the endpoint name and host name in TAPM RIM with each transaction instance in order to make them available for later retrieval. In this way, transaction correlation can occur across multiple machines as well as on one specific machine. That is, subtransactions from multiple machines can be correlated to a single parent transaction. As the goal of root cause analysis is to break down the end-to-end information for transactions that occur across several servers, the ARM correlator is the critical tool for accomplishing this transaction breakdown. If a specific subtransaction on a specific server is holding up all the other transactions, the breakdown will show which server and which transaction are causing the problem. ARM correlation will also give us the total response time for
each subtransaction component of the parent transaction and indicate the specific path that each subtransaction takes in order to accomplish its task.

For more information, please refer to the following redbooks:

- *Introducing Tivoli Application Performance Management*, SG24-5508
- *Tivoli Application Performance Management: Version 2.0 and Beyond*, SG24-6048

### 3.2.4 Transaction performance data reporting

The primary tools used to view and report on the transaction data captured by IBM Tivoli Monitoring for Transaction Performance are the two consoles:

- **Web Console**: the console provided to view data captured by Enterprise Transaction Performance
- **Tivoli Internet Management Server Console**: the interface to administer and view data captured by Web Transaction Performance

When discussing transaction breakdown where data from both IBM Tivoli Monitoring for Transaction Performance environments is combined, the Tivoli Internet Management Server Console is the primary tool. While viewing the transaction performance data gathered by the STI transaction playback, hyperlinks are provided to launch the Web Console in case ARM correlators have been enabled for the specific STI transaction playback job.

*Note:* In this version of IBM Tivoli Monitoring for Transaction Performance, the links are provided based only on the ARM correlator setting for the actual STI transaction playback. This implies that a link to the Web Console will be available, even if no ARM data for the transaction in question has been propagated to the TAPM RIM database.

The reporting mechanisms used to display the transaction performance data are illustrated in Figure 3-2 on page 79.
1. The user logs on to the STI transaction playback report viewer with a browser.
2. A specific report endpoint and transaction performance data are viewed from data held in the TIMS database.
3. In case the ARM Correlator switch was turned on for the specific STI transaction playback, a hyperlink to the relevant ARM data is provided in the reports. When using this link to view the corresponding data collected by WebSphere Application Server, the Web Console is activated and the data relevant to the correlator for the actual transaction instance is collected from the TAPM RIM database and displayed.

3.2.5 How to get reports: the details

This is one method that can be used to produce a simple correlator report for an STI job that has been ARM enabled.

The Synthetic Transaction Investigator statistics report can be displayed using the following actions on the Tivoli Internet Management Server:
1. Log on to the Tivoli Internet Management Server Console.
2. In the left menu pane, select Synthetic Transaction Investigator.
4. From the main report selection list, select Round Trip Time Graph.
5. Select View Report by Job.
6. Select one or more ARM-enabled STI Job(s), and click **Next**.
7. Select one or more endpoint name(s) and click **Select Time Interval**.
8. Select the desired start and end times for the report, and click **View Reports**. The result appears in Figure 3-3.

![Figure 3-3 Sample STI Round-trip time report](image)

9. Double-click any part of the graph that requires further analysis, and the Synthetic Transaction Investigator Statistics Report will be shown.
The ARM correlation report associated with this STI Job can be displayed at this point in the process by clicking on the Report hyperlink located in the ARM column, which will become visible by scrolling far to the right in the Synthetic Transaction Investigator Statistics report.
10. Click on the Report hyperlink and the Web Console will be activated, displaying the transaction performance data collected by WebSphere Application Server.

This action will result in STI accessing the Enterprise Transaction Performance Web Console with a request for retrieval of ARM data from TAPM RIM as shown in Example 3-2 on page 74.

During the retrieval process for the STI job, a specific logon screen for the Enterprise Transaction Performance Web Console will pop up and ask for user name and password as the attempt is made to access TAPM RIM from the Web Console. We will supply tapm and tapmtapm as the username and password for the Web Console at this point. This could just as easily be any other user ID and password combination that you have set up during the installation of the Web Console using the `htpasswd` command. Refer to Chapter 6, “Troubleshooting” on page 149 for more information on `htpasswd`.

As stated earlier, the TAPM RIM stores both the ARM correlator and a parent correlator for each entry of instance data in TX_INSTANCE. In our case, the correlator that STI uses to retrieve detail data is the parent correlator, an example of which is shown in Example 3-5 on page 83.

Once the inquiry is completed, the results will be shown inside the browser, with a vertical line graph representing the URI, EJB, or JDBC transaction being filtered in the WebSphere Application Server processing, as seen in Figure 3-7 on page 83.
Each vertical line graph shown also displays a start time and response time in seconds for the transaction selected from the STI Statistics Report. Double-clicking the vertical line graph produces a drill-down report on the transaction and its status.

Data from the `wmargetrawdatadb` results can also be exported to an XML file. Refer to the setting for PMIRMFilter in the `pmirm.xml` file for WebSphere Application Server as shown in Example 3-5.

**Example 3-5   Example pmirm.xml file showing filters**

```xml
.pmirm:PMIResultMetrics xmi:id="PMIRequestMetrics_1" enable="true" enableARM="true" traceLevel="DEBUG">
  <filters xmi:id="PMIRMFilter_1" type="URI" enable="false">
    <filterValues xmi:id="PMIRMFilterValue_1" value="/servlet/snoop" enable="false"/>
  </filters>
  <filters xmi:id="PMIRMFilter_2" type="EJB" enable="false">
    <filterValues xmi:id="PMIRMFilterValue_2" value="/webapp/examples/HitCount" enable="false"/>
  </filters>
  <filters xmi:id="PMIRMFilter_3" type="SOURCE_IP" enable="false">
    <filterValues xmi:id="PMIRMFilterValue_3" value="255.255.255.255" enable="false"/>
    <filterValues xmi:id="PMIRMFilterValue_4" value="254.254.254.254" enable="false"/>
  </filters>
</pmirm:PMIResultMetrics>
```
3.2.6 What the various STI reports offer

Having looked at the details of how the ARM correlator is generated and stored in the TAPM RIM database, we now look at why the transaction correlators are important for the diagnostic process.

Within any given WebSphere application, such as PlantsByWebSphere, different business transactions—such as Login, Logout, Register, Home, Account, Account Update, and Buy—can be identified, against which a mix of different IT methods (JSP, EJB, CMP Bean Update, etc.) can be tested. A specific business transaction can be correlated with a timing breakdown for backend service time measurement that can be used to measure the service level for a company’s business.

IBM Tivoli Monitoring for Transaction Performance 5.1 has taken a big step toward providing this correlation of business transactions to IT methods, but there are still some areas of backend service time that are hidden in the current version of the software. Future releases promise to provide even more detailed analysis of this backend service time.

**Note:** WebSphere Application Server Version 5.0 supports two types of ARM tracing. The first is tracing resource usage times for requests coming from specific IP addresses (SOURCE_IP filtering). The second is tracing requests matching a URI or EJB method pattern. In our many attempts to trace JDBC types of transactions, we were unsuccessful with WebSphere Application Server Version 5.0 and IBM Tivoli Monitoring for Transaction Performance.
Chapter 3. Transaction breakdown: root cause analysis

As an example, any specific trade session within the PlantsByWebSphere application can access either the Profile CMP, Account CMP, Register CMP, or Buy CMP in order to access the PlantsByWebSphere application database. While we cannot be sure which EJB or URI might be associated with each of these business CMPs, we can map by transaction type where critical processing delays occur in the process. This feature is a major step forward in the diagnosis of Web processing bottlenecks and inefficiencies. Once a significant amount of transaction data has been collected, we can utilize our tools within Tivoli Decision Support (TDS) to report on historical trends and their extrapolation in forecasting future availability and performance of our Web servers.

Available TDS reports for the IBM Tivoli Monitoring for Transaction Performance E/T TDS Guide answer the questions:

- Are my applications meeting the service level target? (Response Times)
- How is my application performing? (Response Time by Application, host name, user, subnet)
- What are my application performance forecasts? (Response Time Trend and Forecast, Transaction Forecast)
- What is my application availability? (Transaction Completions and Failures)
- What is my application throughput? (Execution Load by User, Application)

**Note:** The transaction breakdown provides an overall view of parent transactions across multiple WebSphere Application servers, detailing each WebSphere component of each parent transaction for the application. However, in the current release of Enterprise Transaction Performance, interacting with the current WebSphere PMI is not an *intra*-container breakdown of parent/child transaction relationships, a breakdown of these relationships between containers. WebSphere Application Server Version 5.0 does not ARM-instrument around individual method calls with Performance Monitoring Infrastructure (PMI) Request Metrics, but instead ARM-instruments only the outermost code block to capture the requests time at a specific hop from one process to another or from one server to another.

So what we see in our reports is a parent transaction grouping by EJB or URI, with each grouping providing minimum, maximum, and average response times for all Web container transactions of that transaction type. We do not see a transaction breakdown, for instance, that identifies all of the EJB transactions that belong to the top-level URI transaction. This type of sorting breakdown is not possible with WebSphere Application Server Version 5.0.
3.3 Enabling transaction breakdown

To install, configure, and verify the functionality of the tools needed to perform an STI - WebSphere Application Server Version transaction breakdown you must:

1. Install the Enterprise Transaction Performance and Web Transaction Performance components
2. Install WebSphere Application Server
3. Configure WebSphere Application Server for ARM
4. Distribute Enterprise Transaction Performance onto the server system
5. Use STI while ARM-enabled

These steps are outlined in Figure 3-8.
3.3.1 Install the ETP and WTP components

To deploy all of the tools needed to monitor and break down Web transactions that interact with applications hosted by WebSphere Application Server, both Enterprise Transaction Performance and Web Transaction Performance must be installed and customized. To complete this task:

1. Apply patches to the Tivoli Framework.
2. Install and customize Enterprise Transaction Performance.
3. Install and customize Web Transaction Performance.
4. Deploy a Tivoli endpoint to the system that will be hosting WebSphere Application Server.


Complete these steps to create the basic IBM Tivoli Monitoring for Transaction Performance environment to be used for measuring transaction performance:

1. Prepare the Tivoli Framework by installing TME 3.7.1 and Patch 3.7.1-TMF-0002.
2. Install Enterprise Transaction Performance on your TMR and a Windows-based managed node.
   a. Install Enterprise Transaction Performance Web Console on the Windows-based managed node. This is required to host the Enterprise Transaction Performance Web Console.
   b. Set up the TAPM RIM database. The RDBMS client should be installed on the machine that will be used as the RIM host. To configure the TAPM RIM object, use the icon on the TME Desktop that should appear after installing Enterprise Transaction Performance or use the \texttt{wcrtrim} command. Use the setup script to create the tables and tablespace.
3. Install the required components. The minimum requirements are:
   - Tivoli Internet Management Server with database
   - STI Recorder on a Windows-based system
   - STI Playback endpoint
4. Deploy a Tivoli endpoint to the system that will be hosting WebSphere Application Server.
3.3.2 Install WebSphere Application Server

Follow the directions provided with WebSphere Application Server 5.0 to install and set up a Web application server.

The WebSphere InfoCenter documents can be unzipped from the wasv5aes_infocenter*.zip file on the installation media, and started by opening the index.html file to display the main index in your browser. From this main index, click on Installation Instructions for more details about the installation of WebSphere Application Server Version 5.0.

If the installation is successful but the WebSphere Application Server will not start, look in <installdir>/WebSphere/AppServer/logs/server1/startServer.log for possible startup errors.

**Note:** We encountered the following problem areas when installing the beta version of WebSphere Application Server that was the most recent version available while writing this book.

At the time of publishing, WebSphere Application Server Version 5.0 should be available, and it is unlikely that you will encounter similar problems if your system has adequate resources available.

**Java support**

In some cases, the WebSphere Application Server v5.0 installation procedure may not be able to find the special version of java.exe that it needs. If you receive an error similar to the one shown in Example 3-6, it is likely that the path to java.exe is not fully specified for the installation.

*Example 3-6  Attempting to install without setting the PATH variable*

"A suitable JVM could not be found. Please run the installer again using the option -is:javahome <JAVA HOME DIR>".

To overcome this problem, add the directory path for java.exe to the front of the PATH variable, pointing to the specific directory of the java.exe that is provided with WebSphere Application Server Version 5.0 (i.e. <installdir>/nt/jdk/jre/bin) - as shown in Example 3-7.

*Example 3-7  Example of setting PATH variable for WebSphere 5.0 installation*

```
c:\>cd c:\websphere\nt
C:\websphere\nt>set PATH=C:\websphere\nt\jdk\jre\bin;%PATH%
c:\websphere\nt>java -fullversion
```


**Installation engine**
To initiate the installation of WebSphere Application Server Version 5.0 on a Windows-based system, it is advisable to use `launchpad.bat` rather than `installer.exe`. Our attempts to use `installer -is:javahome <java_home_dir>` failed, but after supplying the path to the correct `java.exe`, the installation succeeded when started through `launchpad.bat`.

**Memory requirements**
The WebSphere Application Server Version 5.0 installation may fail in `IMAGEHLP.dll` if there is not enough virtual memory for the installation to be successful. Make sure that the installation machine has enough memory or virtual memory as required by the server documentation prior to starting the installation.

**Installation directory name**
In other cases, providing an installation path with white spaces in the directory name (e.g. `\Program Files\`) may cause a problem with the service binaries, such as shown in Example 3-4.

**Example 3-8  Failed attempt to start WebSphere Application Server on w32-ix86**

The IBM WebSphere Application Server V5 - server1 service failed to start due to the following error:
IBM WebSphere Application Server V5 - server1 is not a valid Win32 application.

Either place double quotes around the installation directory path for each installation directory specified in the wizard installation dialog (as in the default path “C:\Program Files\WebSphere”) or specify a different directory path without white spaces, such as “C:\WebSphere”.

**3.3.3  Configure WebSphere Application Server for ARM**

To instruct WebSphere Application Server Version 5.0 to report the transaction performance data to the ARM interface and enable the ARM calls to be passed on to the Tivoli implementation, the following steps should be performed:

1. Provide Tivoli armjni classes access.
2. Enable PMI request metrics.
3. Instruct PMI to use Tivoli ARM implementation.
4. Enable tracing for all components.
5. Enable PMI tracing services at startup.
6. Verify the configuration.

The following sections will describe each step of the WebSphere Application Server Version 5.0 configuration needed to enable Enterprise Transaction
Performance to gather transaction performance data related to the WebSphere environment. Most of these steps can be performed either by copying or directly editing selected files or by using the WebSphere Administration GUI. Where relevant, descriptions are provided for both methods.

The WebSphere Administration GUI may be accessed from a Web browser by specifying the following URL:

http://<WAS-Server-name>:9090/admin

where <WAS-Server-name> is either the IP address or the host name of the WebSphere Application Server system.

Note: No matter which method—admin console or file editing—is being used to configure WebSphere Application Server, the changes you make will not take affect until you restart the application server.

Provide Tivoli armjni classes access

In order for the data gathered by WebSphere to be reported to the ARM interface, the ARM API must be made available to the WebSphere Application Server by copying two files from the TMR Server to the WebSphere Application Server. These files, which can be found in the $BINDIR/TME/MAR/armjni directory on the TMR server, are:

armJni.jar Copy to <was_home>\lib
armjni.dll Copy to <was_home>\bin

where <was_home> is the installation directory of the WebSphere Application Server on Windows. This is typically c:\Program Files\WebSphere\AppServer.

Enable PMI request metrics

The following steps relating to Figure 3-9 on page 91 show how to enable PMI request metrics gathering and the use of ARM in WebSphere Application Server:

Admin console method

1. Open the administrative console.
2. Click Troubleshooting -> PMI Request Metrics in the navigation tree.
3. Select the check box in the Enable field to enable request metrics.
4. Select the check box in the Enable ARM field to enable ARM use when reporting request metrics.
5. Select PERF_DEBUG or DEBUG in the Trace Level field.
6. Click Apply or OK. Click Save in the menu bar of the frame.
XML Method
To enable PMI request metrics with ARM by editing, open the file
\texttt{<was\_home>}/config/cells/<BaseApplicationServerCell>/pmirm.xml and change
the value of the enable and enableARM parameters of PMIRequestMetrics clause from \texttt{false} to \texttt{true}. Also, set the traceLevel parameter to PERF\_DEBUG or
DEBUG, as shown in Example 3-9.

Example 3-9 Enabling WebSphere Application Server ARM collection

Change:

\begin{verbatim}
<pmirm:PMIRequestMetrics xmi:id="PMIRequestMetrics_1" enable="false"
enableARM="false" traceLevel="HOPS"/>
\end{verbatim}

To:

\begin{verbatim}
<pmirm:PMIRequestMetrics xmi:id="PMIRequestMetrics_1" enable="true"
enableARM="true" traceLevel="PERF\_DEBUG"/>
\end{verbatim}
Instruct PMI to use Tivoli ARM implementation
Now, combine the two previous steps to instruct the PMI layer of the WebSphere Application Server to use the Tivoli-supplied ARM classes.

Admin console method
1. Open the administrative console.
2. In the console navigation tree, click Servers -> Application Servers.
3. Click server.1.
4. Click the Configuration tab.
5. Scroll down to the Additional Properties section, and open the Process Definition link.
6. Scroll down to the Additional Properties section again, and open the Java Virtual Machine link.
7. Change the value of the Generic JVM arguments field, denoted in Figure 3-10 on page 93, to -Dcom.ibm.websphere.pmi.reqmetrics.ARMIMPL=tivoli
8. Click Apply or OK. Click Save in the menu bar of the frame.
Figure 3-10  Setting Generic JVM arguments for PMI ARM calls

**XML method**

Open the file

```xml
<was_home>/config/\<BaseApplicationServerCell>/nodes/\<nodename>/servers/server1/server.xml.
```

Locate the jvmEntries tag. Change the value of the genericJvmArguments as shown in Example 3-10 on page 94.
Example 3-10  Enable the Tivoli ARM implementation

Change:

```xml
<jvmEntries xmi:id="JavaVirtualMachine_1" classpath="" bootClasspath=""
verboseModeClass="false" verboseModeGarbageCollection="false"
verboseModeJNI="false" initialHeapSize="0" maximumHeapSize="256"
runtime="false" hprofArguments="" debugMode="false"
debugArgs="-Djava.compiler=NONE -Xdebug -Xnoagent
-Xrunjdwp:transport=dt_socket,server=y,suspend=n,address=7777"
genericJvmArguments="">
</jvmEntries>
```

To:

```xml
<jvmEntries xmi:id="JavaVirtualMachine_1" classpath="" bootClasspath=""
verboseModeClass="false" verboseModeGarbageCollection="false"
verboseModeJNI="false" initialHeapSize="0" maximumHeapSize="256"
runtime="false" hprofArguments="" debugMode="false"
debugArgs="-Djava.compiler=NONE -Xdebug -Xnoagent
-Xrunjdwp:transport=dt_socket,server=y,suspend=n,address=7777"
genericJvmArguments="-Dcom.ibm.websphere.pmi.reqmetrics.ARMIMPL=tivoli">
</jvmEntries>
```

Enable tracing for all components

Now activate tracing of the PMI requests within WebSphere Application Server. This step is necessary in order to instruct WebSphere Application Server to write the request metrics to the trace log.

**Admin console method**

Follow these steps to activate PMI tracing in WebSphere Application Server using the admin console, as shown in Figure 3-11 on page 95.

1. Open the administrative console.
2. In the console navigation tree, click *Troubleshooting > Logs and Trace* -> *server1* -> *Diagnostic Trace*.
3. Click on the *Configuration* tab.
4. Change the Trace Specification to `com.ibm.ws.pmi.reqmetrics*=all=enabled`:
5. Click *Apply* or *OK*. Click *Save* in the menu bar of the frame.
XML method

To activate PMI tracing by editing, open the server.xml file at
<was_home>\config\cells\<BaseApplicationServerCell>\nodes\<nodename>\servers\server1\server.xml.

Locate the traceservice:Traceservice tag, and change the value of the
startupTraceSpecification from "*all-disabled" to
"com.ibm.ws.pmi.reqmetrics*=all=enabled" as demonstrated in Example 3-11 on
page 96.
Enable PMI tracing services at startup
Finally, enable PMI tracing to be started when the WebSphere virtual server, which hosts the application used by the transactions we want to monitor, starts.

Admin console method
1. Open the administrative console.
2. In the console navigation tree, click Servers -> Application Servers.
3. Click server 1.
4. Click the Configuration tab.
5. Scroll down, and click Performance Monitoring Service.
6. Select the Startup checkbox.
7. (Optional) Select the PMI modules and levels to set the initial specification level field.
8. Click Apply or OK. Click Save in the menu bar of the frame.
Figure 3-12  Enable PMI tracing at startup

**XML method**
To enable PMI tracing immediately when the virtual server starts, open the server.xml file at

```xml
<was_home>\config\cells\BaseApplicationServerCell\nodes\nodename\servers\server1\server.xml.
```

Locate the pmiservice:PMIService xmi type. Change the value of the enable flag from false to true as shown in Example 3-12 on page 98.
Example 3-12  Enabling PMI tracing at virtual server startup

Change:

```xml
<services xmi:type="pmiservice:PMIService" xmi:id="PMIService_1"
          enable="false" initialSpecLevel="......
```

To:

```xml
<services xmi:type="pmiservice:PMIService" xmi:id="PMIService_1"
          enable="true" initialSpecLevel="......
```

Verify the configuration

Before attempting to verify that the newly added or modified configuration parameters have the desired effect, it is advised to reboot the system hosting WebSphere Application Server.

After rebooting the system, you may test the WebSphere Application Server setup by pointing your browser http://<hostname>/snoop to execute the snoop servlet. If this works properly, there should be trace records in <was_home>/logs/server1/SystemOut.log with an ID of PMRM0003I, which indicates that the ARM calls have been made by WebSphere Application Server and are ready to be picked up by the Tivoli endpoint ARM engine.

3.3.4 Distribute ETP to the WebSphere Application Server system

Before distributing Enterprise Transaction Performance onto the WebSphere Application Server system, verify that the Tivoli endpoint has been installed on the system hosting the WebSphere Application Server.

Create and distribute a MarProfile (the profile type provided by Enterprise Transaction Performance) to the machine where WebSphere Application Server Version 5.0 is or will be installed:

1. From the Tivoli Desktop, add the IBM TMTP notice group by opening the Administrators window, right-clicking the Administrators icon, and choosing Edit Notice Group Subscriptions. Select IBM TMTP and move to current notice groups. Press Change and close.

   **Note:** This step is optional, but as vital messages from the TMTP components are logged in the notice group, its use is recommended.
2. Register the WebSphere Application Server with Enterprise Transaction Performance using:

   wmarregapp -a <application name>

   where the application name is something like
   <nodename>/<nodename>/server1.

   You can determine the name of the application as it is reported to ARM by WebSphere Application Server (or any other application for that matter) by listing the applications that have issued ARM calls at the endpoint. The list can be obtained by using the `wmarlsapp` command, which provides output similar to what is shown in Example 3-13:

```
Example 3-13   Applications that have issued ARM calls at an endpoint

C:\>wmarlsapp ibmtiv8
The following applications match the specified criteria:
---------------------------------------------------------------
Application name:                       UserId:
---------------------------------------------------------------
ibmtiv8/ibmtiv8/server1 Administrator
armtest Administrator
---------------------------------------------------------------
```

   Verify that the application has registered by running `wmargetapp`.

   **Note:** If the WebSphere Application Server does not register after restart of the system, this may be because WebSphere Application Server was started, and issued the `arminit` call, before the Tivoli endpoint was ready to receive and process it.

   If the WebSphere Application Server has not registered, stop it, be sure that the Tivoli Endpoint is active and the IBM Tivoli Monitoring for Transaction Performance engine is running, and then restart the WebSphere Application Server. This should ensure that the endpoint is listening and that it will receive and process the `arminit` call so the WebSphere Application Server application is registered correctly.

3. Add MarProfile as a Managed Resource:

   a. From the Tivoli Desktop, open the policy region window, and choose Properties -> Managed Resources.

   b. Move MarProfile to the Current Resources list.

   c. Choose Set and Close.
4. Create a new profile:
   a. Open the policy region window and then the profile manager you wish to use. If these do not exist, create them.
   b. Select **Create -> Profile**.
   c. Give the new profile a name (such as TAPM-WASProfile), verify that MarProfile is selected, and choose **Create and close**.
   d. Open the new profile, choose **Add Entry**, and select the application you registered in step 2 on page 99, as illustrated in Figure 3-13.

![Figure 3-13 MarProfile monitoring ARM calls from WebSphere Application Server](image)

   e. Click **Data Storage** to open the window shown in Figure 3-14 on page 101. Select **All Data** for both Local Logging and Database Settings. Click **Add & Close** for both windows.
f. Select **Profile -> Distribute**. Be sure the endpoint is selected, and choose **Distribute & Close**.

5. Run the command `wmarlseng <ep_name>` to be sure your profile/collection has started properly. The result should look similar to Example 3-14.

**Example 3-14  wmarlseng output**

c:\>wmarlseng ibmtiv8
Collection active on application : ibmtiv8/ibmtiv8/server1
transaction filter : * 
bucket limits (ms) :  100 500 1000 2000 5000 10000
collection interval (s):     1200
collection active on : always=EveryDay
start date (YYYY MM DD): 2002 11 14
stop date (YYYY MM DD): not specified
Correlator Generation : Enable
Summary Data:
local logging : active
data-base storage : active, detail level: Maximum
Instance Data:
3.3.5 Use STI while ARM-enabled

Schedule an STI job from the Tivoli Internet Management Server using a recorded transaction for something hosted on the WebSphere Application Server. Some examples that can be navigated and recorded are provided at http://<WAS_hostname>/WSsamples/en/index.html.

When creating the STI job, check the ARM correlation box as in Figure 3-15.

Once the job has run for a full cycle, you can view the Synthetic Transaction Investigator Statistics report as demonstrated in 3.2.5, "How to get reports: the
details” on page 79 and click on the ARM Report hyperlink. Log on to the Enterprise Transaction Performance Web Console (most likely the user ID is tapm and the password is tapmtapm).

If the data is not in the TAPM RIM database yet, run the command `wmarsenddata -i <ep_name>` to force the data upload and refresh the report view.

The different STI reports that are available show the transaction response time as recorded by the STI playback system, including network and page render times. The breakdown functions that allow viewing the WebSphere Application Server processing time are available for particular transactions or instances only, if you stick with the STI reports.

For a broader view of these processing times, use the Enterprise Transaction Performance Web Console to see transaction response times data as both aggregated and individual instance data. This shows average, minimum, and maximum transaction response times, as well as the number and status (completed, failed, or aborted) of WebSphere Application Server transactions.

Figure 3-16 on page 104 shows an example of the average response times from the URI part of the STI simulated PlantsByWebSphere transaction.
Figure 3-16  Average WebSphere Application Server transaction response times
Monitoring 3270 transaction performance

This chapter provides information related to features of IBM Tivoli Monitoring for Transaction Performance: Enterprise Transaction Performance for monitoring performance of 3270 and SAP transactions.

The information provided here is to be considered supplemental to what is already available in the standard product documentation:


and the previously released redbook that covers the IBM Tivoli Monitoring for Transaction Performance: Enterprise Transaction Performance Version 5.1 predecessor product Tivoli Application Performance Management:

- **Tivoli Application Performance Management Version 2.0 and Beyond, SG24-6048**
4.1 3270 transaction measurement architecture

The architecture behind SAP and 3270 transaction performance measurement is based on recorded transactions that may be replayed on any capable system.

4.1.1 Recording

To record a 3270 transaction, use the IBM Recording and Playback Workbench, which runs on Windows-based systems. The program is coded in such a way that the specifics of different transaction types are handled by special plug-ins. This enables easy integration of new transaction types, which may be imbedded into the IBM Recording and Playback Workbench in the future.

Upon completion of a recording, the simulation package is sent from the recording system to the Tivoli environment, where it is registered and built into a profile, which then may be distributed and played back on any capable endpoint.

![3270 transaction performance measurement architecture diagram]}

*Figure 4-1  3270 transaction performance measurement architecture*

The major components of the SAP and 3270 recording and playback feature, shown in Figure 4-1, are:

- **Workbench**  The graphical interface through which the user creates and tests the simulation scripts.
**Motherboard**
The main recording infrastructure. Manages the Transaction Families and provides the infrastructure for the workbench to interact with the recording logic.

**Sensors and formatters**
Encapsulates the recording technology and the script generation logic. Each transaction family has one or more sensors and a formatter.

**Playback helpers**
COM objects that enable simulation scripts to perform complex operations during playback. Each transaction Family uses at least one playback helper.

**Playback session**
The package exported by the IBM Recording and Playback Workbench. The recorded simulation is contained in the playback session, ready to be reproduced by the playback engine.

**Playback engine**
Reproduces a recorded script. It is coded as an extended data provider of the ARM engine, and its activities can be scheduled using the standard collection scheduling options.

### 4.1.2 Playback

The architecture of the playback engine for 3270 and SAP transactions, depicted in Figure 4-2 on page 108, works as follows:

- The TMTP agent drives the playback engine by forwarding collection commands to the COM client (rp_data_provider).
- The client handles the list of all active collections and creates a distinct COM object for each one of them.
- The COM server uses SAX to run the script received in the pbs file (created using the Workbench).
- Inside a script you can find COM invocations to either SAP or PCOMM (the two transaction families supported by this Enterprise Transaction Performance release).
- The script must have been properly instrumented to include calls to the Enterprise Transaction Performance ARM interface. Enterprise Transaction Performance can then collect performance data about the application.
- Some information needed by the playback engine to run a simulation is not included in the VB Script file.
- The Workbench saves this information in xml files.
- The xml files are sent to the endpoint together with the VB script (txf and pbs packages).
The xml file contains the following information:

- A description of the used Tx Family name of the transaction family (3270 or SAP)
- A list of runtime parameters with the actual value for this specific execution
- A list of COM dlls and executables that must be registered to the system before the simulation can start

![Diagram](image)

**Figure 4-2  Playback engine architecture**

### 4.2 Installation and configuration

Besides installing and configuring the basic IBM Tivoli Monitoring for Transaction Performance components in the Tivoli environment, at least two specific installation activities are required to start monitoring performance of 3270 applications:

1. The IBM Recording and Playback Workbench should be installed on all systems that will be used to record transactions that will be measured.
2. Either:
   a. IBM Personal Communications Version 5.5 with CSD1 or Version 5.0 with CSD3 must be installed on all systems used for either recording or playback of 3270 transactions.
   b. SAP Version 6.2 must be installed on all systems used for either recording or playback of SAP transactions.
4.3 Monitoring 3270 transactions

As previously noted, the activities required to configure, deploy, and execute 3270 transaction simulations are:

1. Recording
2. Upload, registration, and profile creation
3. Profile distribution and scheduling

4.3.1 Recording

The IBM Recording and Playback Workbench is the Enterprise Transaction Performance component that provides a friendly graphical environment to record, replay, and customize 3270 transaction simulations. It uses the settings collected in the workstation configuration profiles created from the Personal Communication application.

When the 3270 application has been properly configured, the configuration file or workstation profile (*.ws) can be saved and used by the IBM Recording and Playback Workbench. This file is used to determine the host system, keyboard layout, and other parameters of the 3270 emulation session.

To create a new project with the IBM Recording and Playback Workbench, simply check the **New** checkbox from the Start Project dialog to open the New Project window shown in Figure 4-3 on page 110.
Under the General tab, the 3270 checkbox must be checked to specify the 3270 Transaction Family. Under the Family Settings tab, the full path to a .ws file created with the Personal Communication application must be specified.

Figure 4-3  3270 Project Family Settings
When the new project is created, you may specify under the Recording ARM Options tab whether to enable ARM instrumentation—with or without correlation—to measure the response times, as shown in Figure 4-4.

![New Project](image)

**Figure 4-4  Specifying ARM instrumentation**

To start recording a 3270 session using the settings specified in the .ws files and the options set in the project, click the Record (red) button on the Workbench. This action will launch the 3270 terminal, from which an ordinary transaction can be performed.

During the execution of each command, the Workbench will produce the corresponding script code, and the tree view will be upgraded automatically.

When the transaction is stopped, by pressing the Stop button on the Workbench, the VB script code will be available.

Now the recorded transaction can be viewed using one of the following options:

**Transaction Explorer View** Presents a logical organization of all components of the transaction as shown in Figure 4-5 on page 112. The right pane shows all properties of the selected module, and the left pane provides specific details to be viewed or edited.
The whole transaction is translated into VB script code, with pull-down menus to quickly go to objects and procedures contained in the code.

![Transaction Explorer view](image)

If the project was created with the automatic ARM-enabling option, an ARM call is created for every single command issued. In the Transaction Explorer view shown in Figure 4-6 on page 113, an ARM call inserted into the code is denoted by the small watch icon.

If no ARM calls were automatically inserted, this can be done by right-clicking the desired command and choosing **Insert ARM**. If a simple ARM call is inserted, a gray watch icon will superimpose to the selected subtransaction icon. If an ARM call with correlation is inserted, an orange watch icon will appear on the subtransaction icon and on the project icon that represents the parent transaction. Using the Transaction Explorer View, undesired ARM calls can be removed from selected commands by right-clicking the proper command in the left pane and selecting **Remove ARM**.

The code generated by the recorded transaction can be viewed and edited choosing the **Code View** from the menu. ARM calls can be inserted or deleted by editing the VB script code.
Figure 4-6  3270 simulation transaction code

Every recorded transaction can be easily for verification from the Workbench by pressing the **Play** button on the Workbench toolbar.
To perform a scheduled playback session on a Tivoli endpoint for a transaction recorded by the IBM Recording and Playback Workbench, choose **File -> Export Simulation Package**. Save the simulation as a *.pbs file, shown in Figure 4-7.

![Figure 4-7 Saving the recorded transaction](image)

### 4.3.2 Creating profiles for 3270 transaction monitoring

To create and schedule a playback session, a MarProfile must be created and configured to include the exported session file (*.pbs) and other runtime parameters as described in *IBM Tivoli Monitoring for Transaction Performance: Enterprise Transaction Performance User's Guide Version 5.1*, GC23-4803.
The new application that will play back the 3270 transaction can be registered to the TMR using either the wmarregapp command or the GUI shown in Figure 4-8. It must point to the .pbs file copied from the project folder used by the IBM Recording and Playback Workbench.

![Add Application](Image)

Figure 4-8  Registering a new Playback application

Once the MarProfile has been created, it can be distributed to the endpoints from which the performance of the 3270 transaction is to be monitored.

### 4.3.3 Executing the 3270 transactions

To verify that the profile containing the transaction is running on the target endpoint where it has been distributed, issue the usual `wmarlseng` command:

```
wmarseng <endpoint label>
```

The Mar engine running on the endpoint will collect performance data regularly from the 3270 session and write it into the logfiles.

### 4.3.4 Troubleshooting

The visual environment provided by the IBM Recording and Playback Workbench offers a useful debugging facility for the recorded transaction. Upon entering the Code View and pressing the **Play** button, the recorded transaction is played and all of the typical debugging tools and functions are available.

Using the Step Into, Step Over and Step Out functions, you can execute every single statement individually, enter a function or a sub-code procedure code whenever it is called, or exit the local procedure to the calling statement.
Breakpoints can be put into the code, or disabled when not needed, by clicking on the **Toggle Break** button (F9), so that the code can be executed up to the inserted breakpoint.

Using the watch functions shown in Figure 4-9, the runtime value of any variable defined within the script can be viewed. Values are updated dynamically during the execution.

---

**Figure 4-9  Debugging the 3270 transaction Visual Basic code**
Chapter 5. TMTP and Tivoli Enterprise Data Warehouse

An important new feature of IBM Tivoli Monitoring for Transaction Performance Version 5.1 is the integration to the common Tivoli repository for historical data: Tivoli Enterprise Data Warehouse. Both Enterprise Transaction Performance and Web Transaction Performance provide these capabilities by supplying specialized extract, transform, and load (ETL) programs that extract historical data from the enterprise Tivoli Application Performance Management RIM (TAPM RIM) database and the Web Services Courier database, respectively.

The Tivoli Enterprise Data Warehouse (TEDW) collects and manages data from various Tivoli and non-Tivoli system management applications. The data is imported into the TEDW databases through ETL programs from the management application databases, and further processed for historical analysis and evaluation. It is Tivoli’s strategy to have most of its products providing ETLs so that the TEDW databases can be populated with meaningful systems management data. The IBM Tivoli Monitoring for Transaction Performance is one of the many products to use Tivoli Enterprise Data Warehouse.

This chapter provides details about how to set up and configure the IBM Tivoli Monitoring for Transaction Performance Warehouse Enablement packs to enable integration of historical IBM Tivoli Monitoring for Transaction Performance data with other data sources and higher-level systems management solutions such as Tivoli Service Level Advisor.
5.1 Tivoli Enterprise Data Warehouse overview

Having access to historical data regarding the performance and availability of IT resources may prove to be very useful in various ways:

- Tivoli Enterprise Data Warehouse collects historical data from many applications into one central place.

  Tivoli Enterprise Data Warehouse collects the underlying data about the network devices and connections, desktops and servers, applications and software, and the problems and activities that have gone on to manage the infrastructure. This enables the construction of an end-to-end view of their enterprise and a view of the related resource data independent of the specific applications used to monitor and control the resources.

- Tivoli Enterprise Data Warehouse adds value to raw data.

  Tivoli Enterprise Data Warehouse performs data aggregation based on user-specified periods, such as daily or weekly, and enables restriction of the amount of data stored in the central data Tivoli Enterprise Data Warehouse repository. The data is also cleaned and consolidated in order to allow the data model of the central repository to share common dimensions. For example, Tivoli Enterprise Data Warehouse ensures that time, host name, and IP address are the same dimensions across all the applications.

- Tivoli Enterprise Data Warehouse allows for correlation of information from many Tivoli applications.

  Tivoli Enterprise Data Warehouse can also be used to derive added value by correlating data from many Tivoli applications. It allows reports to be written that correlate cross-application data.

- Tivoli Enterprise Data Warehouse uses open, proven interfaces for extracting, storing, and sharing the data.

  Tivoli Enterprise Data Warehouse can extract data from any Tivoli or non-Tivoli application and store it in a common, central database. Tivoli Enterprise Data Warehouse also provides transparent access for third-party Business Intelligence (BI) solutions using the CWM standard, such as IBM DB2 OLAP, Crystal Decisions, Cognos, BusinessObjects, Brio Technology, or Microsoft OLAP Server. CWM stands for Common Warehouse Metadata, an industry standard specification for metadata interchange defined by the Object Management Group (see http://www.omg.org). Tivoli Enterprise Data Warehouse provides a Web-based reporting front end, called the Reporting Interface, but the open architecture provided by the Tivoli Enterprise Data Warehouse allows other BI front ends to be used to access the data in the central warehouse. The value here is flexibility. Customers can use the reporting application of their choice; they are not limited to any specific one.
- Tivoli Enterprise Data Warehouse provides a robust security mechanism.
  Tivoli Enterprise Data Warehouse provides a robust security mechanism by enabling data marts to be built with data from subsets of managed resources. By providing database-level authorization to access those data marts, Tivoli Enterprise Data Warehouse can address most of the security requirements related to limiting access to specific data to those customers or business units with a need to know.

- Tivoli Enterprise Data Warehouse provides a scalable architecture.
  Tivoli Enterprise Data Warehouse depends on the proven, industry-standard RDBMS technology, so it provides a scalable architecture for storing and retrieving the data.

### 5.1.1 Tivoli Enterprise Data Warehouse concepts and components

This section discusses the key concepts and the various components of Tivoli Enterprise Data Warehouse in the logical order that the measurement data flows: from the monitors collecting raw data to the final detailed report. Figure 5-1 depicts a typical Tivoli Enterprise Data Warehouse configuration that will be used throughout this section.

![Figure 5-1 A typical Tivoli Enterprise Data Warehouse environment](image)
It is common for enterprises to have various distributed performance and availability monitoring applications deployed that collect some sort of measurement data and provide some type of threshold management, central event management, and other basic monitoring functions. These applications are referred to as source applications.

The first step in obtaining management data is to enable the source applications. This means providing all tools and filtering needed to import the source operational data into the Tivoli Enterprise Data Warehouse central data warehouse. All components needed for that task are collected in so-called warehouse modules for each source application. In the context of the topic of this publication, IBM Tivoli Monitoring for Web Infrastructure is the source application providing management data for Web server and application server data warehouse modules.

One important part of the warehouse modules is the ETL programs that, generally, process data in three steps:

1. Extract the data from a source application database called the data source.
2. Validate, transform, aggregate, and/or clean the data so that it fits the format and needs of the data target.
3. Load the data into the target database.

Tivoli Enterprise Data Warehouse has two types of ETLs: central data warehouse ETL and data mart ETL.

**Central data warehouse ETL** Pulls the data from the source applications and loads it into the central data warehouse, as shown in Figure 5-1 on page 119. This is also often referred to as the source ETL or ETL1.

**Data mart ETL** Extracts a subset of historical data from the central data warehouse that contains data tailored to and optimized for a specific reporting or analysis task, as shown in Figure 5-1 on page 119. This subset of data is used to populate data marts. The data mart ETL is also known as target ETL or ETL2.

As a generic concept, a data warehouse is a structured, extensible database environment designed for the analysis of consistent data. The data that is inserted in a data warehouse is logically and physically transformed from multiple source applications, updated and maintained for a long time period of time, and summarized for quick analysis. The Tivoli Enterprise Data Warehouse Central Data Warehouse (CDW) is the database that contains all enterprise-wide historical data, with one hour as the lowest granularity. This data store is
optimized for the efficient storage of large amounts of data and has a documented format that makes the data accessible to many analysis solutions. The database is organized in a very flexible way, which lets you store data from new applications without adding or changing tables.

The Tivoli Enterprise Data Warehouse server is an IBM DB2 Universal Database Enterprise Edition server that hosts the TEDW Central Data Warehouse databases. These databases are populated with operational data from Tivoli and/or other third-party applications for historical analyses.

A data mart is a subset of the historical data that satisfies the needs of a specific department, team, or customer. A data mart is optimized for interactive reporting and data analysis. The format of a data mart is specific to the reporting or analysis tool you plan to use. Each application that provides a data mart ETL creates its data marts in the appropriate format.

Tivoli Enterprise Data Warehouse provides a Report Interface (RI) that creates static two-dimensional reports of your data using the data marts. The Report Interface is a role-based Web interface that can be accessed with a simple Web browser without any additional software installed on the client. You can also use other tools to perform OLAP analysis, business intelligence reporting, or data mining.

The TEDW Control Center is the IBM DB2 Universal Database Enterprise Edition server containing the TEDW control database that manages your Tivoli Enterprise Data Warehouse environment. From the TEDW Control Center, you can also manage all source applications databases in your environment. The default internal name for the TEDW control database is TWH_MD. The TEDW Control Center also manages the communication between the various components, such as the TEDW Central Data Warehouse, the data marts, and the Report Interfaces. The TEDW Control Center uses the DB2 Data Warehouse Center utility to define, maintain, schedule, and monitor the ETL processes.

The Tivoli Enterprise Data Warehouse stores raw historical data from all Tivoli and third-party application databases in the TEDW Central Data Warehouse database. The internal name of the TEDW Central Data Warehouse database is TWH_CDW. Once the data has been inserted into the TWH_CDW database, it is available for either the Tivoli Enterprise Data Warehouse ETLs to load to the TEDW Data Mart database (the internal name of the TEDW Data Mart database is TWH_MART) or to any other application-specific ETL to process the data and load the application-specific data mart database.
5.1.2 The Monitoring process data flow

In this section we discuss how the warehouse features of both IBM Tivoli Monitoring for Transaction Performance modules interact with the Tivoli Enterprise Data Warehouse. We also describe the various components that make up the IBM Tivoli Monitoring for Transaction Performance warehouse components. We will demonstrate how the data is collected from the endpoint and how it reaches the data warehouse database as shown in Figure 5-2. The ETLs used by the warehouse components are explained in Table 5-3 on page 142. and Table 5-4 on page 146.

Figure 5-2  Enterprise Transaction Performance data flow

To enable Tivoli Enterprise Data Warehouse to receive data gathered by a IBM Tivoli Monitoring for Transaction Performance profile, you should enable the data collection and upload from the specific endpoints using the `wstartcoll` command. To collect instance data from specific endpoints, use the `wmarssetstatus InstanceLogging=true <endpoint>` command. Note that InstanceLogging is turned off by default. Once data collection and upload have been enabled for an application, the data is logged locally and uploaded periodically to the TAPM Database, based on the settings defined by the `wmarssetstatus` command.

The TMTP Uploader component is responsible for moving data from the endpoint to the TAPM database. The APF ETL1 is then used to collect data from the TAPM
database for any module, transform it, and load it to the staging area tables and dynamic data tables in the central data warehouse (TWH_CDH).

The upload of data from the endpoint to the TAPM database may be forced by using the `wmarsenddata` command.

Before we go into detail about installing and configuring the Tivoli Enterprise Data Warehouse Enablement packs to extract and store data from the IBM Tivoli Monitoring for Transaction Performance components, Figure 5-3 presents the environment used for Tivoli Enterprise Data Warehouse in our lab. This can be used as a starting point for setting up the data-gathering process. We describe the steps of a brand new installation with no pre-existing components used.

![Figure 5-3   Installation scenario](image)

Our Tivoli Enterprise Data Warehouse environment is a small, distributed environment composed of three machines:

1. Tivoli Enterprise Data Warehouse server machine hosts the central warehouse and the Warehouse Data Mart databases.
2. Tivoli Enterprise Data Warehouse Control Center machine hosts the Warehouse metadata database and handles all ETL executions.

3. Tivoli Enterprise Data Warehouse Reporting Interface machine enables end users to obtain reports from data stored in the data marts.

**Note:** The reporting environment is not used with Enterprise Transaction Performance as no target ETL (ETL2) is provided with the product.

The primary tool for reporting Enterprise Transaction Performance activity is Tivoli Decision Support.

### 5.2 Setting up the TMTP Warehouse Enablement packs

The following sections describe the procedures in order to install, configure, and schedule the Warehouse modules for the IBM Tivoli Monitoring for Transaction Performance product. The description of the installation steps is based on our lab environment scenario described in Figure 5-3 on page 123.

It is assumed that the Tivoli Enterprise Data Warehouse Version 1.1 environment is already installed and operational. Details for achieving this can be found in *Introduction to Tivoli Enterprise Data Warehouse*, SG24-6607.

Throughout this section, the Warehouse Enablement Pack for IBM Tivoli Monitoring for Transaction Performance: Enterprise Transaction Performance will be used to demonstrate the tasks that must be performed, and the changes needed to implement the Warehouse Enablement Pack for Web Transaction Performance will be noted at the end.

The installation and configuration of the Warehouse Enablement packs is a four-step process that consists of:

**Pre-installation steps** These ensure that the Tivoli Enterprise Data Warehouse environment is ready to receive the TMTP Warehouse Enablement packs.

**Installation** The actual transferal of code from the installation images to the Tivoli Enterprise Data Warehouse server, and registration of the TMTP ETLs in the TEDW registry.

**Post-installation steps** Provide additional configuration information to ensure the correct function of the TMTP Warehouse Enablement packs.

**Activation** Include scheduling and transfer to production mode of the TMTP-specific ETL tasks.
5.2.1 Pre-installation steps

Prior to the installation of the Warehouse modules, the following tasks must be performed:

1. Upgrade to DB2 UDB Server Version 7.2 Fixpack 6 or higher
2. Apply TEDW Fixpack 1.1-TDW-002 or higher
3. Update the Tivoli Enterprise Data Warehouse environment to 1-1-TDW-FP01a
4. Ensure adequate heap size of the TWH_CDW database

It is only required to perform these steps once, since they apply to the general TWDW environment, and not to any specific ETLs.

Upgrade to DB2 UDB Server Version 7.2 Fixpack 6 or higher
Upgrade IBM DB2 Universal Database Enterprise Edition Version 7.2 to at least Fixpack 6 level on your Tivoli Enterprise Data Warehouse environment

Fixpack6 for IBM DB2 Universal Database Enterprise Edition can be downloaded from the official IBM DB2 technical support Web site:

http://www-3.ibm.com/cgi-bin/db2www/data/db2/udb/winos2unix/support/v7fphist.d2w/report

Apply TEDW Fixpack 1.1-TDW-002 or higher
Apply the Fixpack 1.1-TDW-0002 on every database server in your Tivoli Enterprise Data Warehouse environment.

Fixpack 1.1-TDW-0002 for Tivoli Enterprise Data Warehouse can be downloaded from the IBM Tivoli Software support Web site, under the Tivoli Enterprise Data Warehouse category:


Update the Tivoli Enterprise Data Warehouse environment to 1-1-TDW-FP01a

Fixpack 1-1-TDW-FP01a for Tivoli Enterprise Data Warehouse can be downloaded from the IBM Tivoli Software support Web site, under the Tivoli Enterprise Data Warehouse category:


The documentation that accompanies the fixpacks details the steps for installation.
Ensure adequate heap size of the TWH_CDW database

The applications control heap size on the TWH_CDW database must be set to at least 512, as follows:

1. Log on using the DB2 administrator user ID to your TEDW Server machine, in our case db2admin, and connect to the TWH_CDW database:

```
db2 connect to TWH_CDW user db2admin using <db2pw>
```

where <db2pw> is the database administrator password.

2. In order to determine the actual heap size issue, use this command:

```
db2 get db cfg for TWH_CDW | grep CTL_HEAP
```

Output should be similar to Example 5-1.

```
Example 5-1 Current applications control heap size on the TWH_CDW database

Max appl. control heap size (4KB) (APP_CTL_HEAP_SZ) = 128
```

3. If the heap size is less than 512, run this command:

```
db2 update db cfg for TWH_CDW using APP_CTL_HEAP_SZ 512
```

Output should be similar to Example 5-2.

```
Example 5-2 Output from db2 update db cfg for TWH_CDW

DB20000I The UPDATE DATABASE CONFIGURATION command completed successfully.
DB21026I For most configuration parameters, all applications must disconnect
from this database before the changes become effective.
```

4. Now restart DB2 by issuing the following series of commands:

```
db2 disconnect THW_CDW
db2 force application all
db2 terminate
db2stop
db2admin stop
db2admin start
db2start
```

5.2.2 Installing the TMTP Warehouse Enablement packs

The IBM Tivoli Monitoring for Transaction Performance Warehouse Enablement Pack extracts data from the Enterprise Transaction Performance RIM database (TAPM) and the Web Services Courier database respectively, and loads it into the TEDW Central Data Warehouse database (TWH_CDW). The two modules act as source ETLs.
All Tivoli Enterprise Data Warehouse ETL programs follow a naming convention using a three-letter, application-specific product code, known as measurement source code. Table 5-1 shows the measurement codes used for the IBM Tivoli Monitoring for Transaction Performance Warehouse Enablement packs.

<table>
<thead>
<tr>
<th>Warehouse Module Name</th>
<th>Measurement Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enterprise Transaction Performance</td>
<td>APF</td>
</tr>
<tr>
<td>Web Transaction Performance</td>
<td>BWM</td>
</tr>
</tbody>
</table>

The installation can be performed using the Tivoli Enterprise Data Warehouse command line interface or the GUI-based installation program. Here we describe the process using the GUI method. The following steps should be performed at the TEDW Control Center server, once for each of the IBM Tivoli Monitoring for Transaction Performance Warehouse Enablement packs that are being installed.

1. Insert the Tivoli Enterprise Data Warehouse Installation CD in the CD-ROM drive.
2. Select Start -> Run. Type in D:\setup.exe where D is the CD-ROM drive and click OK to start the installation.
3. When the InstallShield Wizard window for Tivoli Enterprise Data Warehouse Installation appears, as shown in Figure 5-4, click **Next**.

![Tivoli Enterprise Data Warehouse Installation dialog](image)

*Figure 5-4  Tivoli Enterprise Data Warehouse Installation dialog*
4. The dialog specific to the type of installation appears, as shown in Figure 5-5. Select **Application installation only** and specify the directory where the Tivoli Enterprise Data Warehouse components are installed. We used C:\TWH. Click **Next** to continue.

![Figure 5-5 Tivoli Enterprise Data Warehouse setup type](image)

5. The host name dialog appears, as shown in Figure 5-6. Verify that this is the correct host name for the TEDW Control Center server. Click **Next**.

![Figure 5-6 Tivoli Enterprise Data Warehouse host name verification](image)
6. The local system DB2 configuration dialog is displayed next. It should be similar to what is shown in Figure 5-7. The installation process asks for a valid DB2 user ID. Enter the valid DB2 user ID and password that were created during the DB2 installation on your local system. In our case, we used db2admin. Click Next.

![Figure 5-7 Tivoli Enterprise Data Warehouse installation: DB2 configuration](image)

7. The dialog to designate the path to the installation media for the application packages appears next, as shown in Figure 5-8.

![Figure 5-8 Installation media path for ITM Generic ETL1 program](image)

You should provide the location of the appropriate IBM Tivoli Monitoring for Transaction Performance ETL1 program. Replace the TEDW CD in the
CD-ROM drive with the desired installation CD. Specify the path to the installation file named `twh_app_install_list.cfg`.

If you use the Tivoli product CD-ROMs, the paths to the installation files for the ETP and WTP installation files are:

```
TMTP:ETP: <CDROM-drive>:\Warehouse Pack
TMTP:WTP <CDROM-drive>:\tedw_apps_etl
```

Leave the `Now` option checked (prevents typing errors) to verify that the source directory is immediately accessible and that it contains the correct files. Click `Next`.

8. Before starting the installation, do not select to install additional modules when prompted, as shown in Figure 5-9. Click `Next`.

*Figure 5-9  Tivoli Enterprise Data Warehouse installation: Additional modules*
9. The overview of selected features dialog window appears as shown in Figure 5-10. Click **Install** to start the installation.

![Tivoli Enterprise Data Warehouse Installation](image)

*Figure 5-10  Enterprise Transaction Performance ETL1 program installation*

10. During the installation, the window shown in Figure 5-11 will be displayed. Wait for successful completion.

![Tivoli Enterprise Data Warehouse Installation](image)

*Figure 5-11  Tivoli Enterprise Data Warehouse installation running*
11. Once the installation is finished the Installation Summary window appears, as shown in Figure 5-12.

If the installation was not successful, check the TWHApp.log file for any errors. This log file is located at the `<TWH_inst_dir>`\apps\AMX\, where `<TWH_inst_dir>` is the TEDW installation directory.

![Installation Summary window](Image)

**Figure 5-12   Installation Summary window**

5.2.3 Post-installation steps

Some configuration settings will have to be changed:

1. Creating an ODBC connection to the TMTP source databases
2. Defining user authority to the Warehouse sources and targets
3. Modifying the schema information

**Creating an ODBC connection to the TMTP source databases**

The TEDW Control Center server hosts all of the ETLs. This server must have access to the various databases accessed by the SQL scripts imbedded in the ETLs. Tivoli Enterprise Data Warehouse uses ODBC connections to access all databases, so the TTP source databases must be cataloged at the TEDW DB2 server as ODBC system data sources.

The ETL programs provided with the IBM Tivoli Monitoring for Transaction Performance: Enterprise Transaction Performance Warehouse Enablement packs require specific logical names of the data sources to be used. Table 5-2 on page 134 shows the values to be used for each of the data sources.
At the TEDW Control Center server, use a DB2 command line window to issue the following commands (in case your source databases are implemented on DB2 RDBMS systems) for each of the source databases:

```
db2 catalog tcpip node <nodename> remote <hostname> server <db2_port>
db2 catalog database <alias> as <database> at node <nodename> ODBC
db2 catalog system odbc data source <alias>
```

Where:
- `<nodename>` is a logical name you assign to the remote DB2 server.
- `<hostname>` is the TCP/IP host name of the remote DB2 server.
- `<db2_port>` is the TCP/IP port used by DB2 (default is 50000).
- `<alias>` is the logical name assigned to the source database.
- `<database>` is the name of the database as it is known at the DB2 server hosting the database. The values are most likely TAPM for the TAPM RIM database used by Enterprise Transaction Performance and COURIER for the Web Services Courier database used by Web Transaction Performance.

**Tip:** If the source databases are implemented using other RDBMS systems, such as Oracle, the commands vary. Instead of using the DB2 command line interface, you may use the GUI of the DB2 Client Assistant to catalog the appropriate ODBC data sources. This method may also be used for DB2 hosted source databases.

### Defining user authority to the Warehouse sources and targets

You should inform the TEDW Control Center server of user access information for every source and target ETL process installed by the IBM Tivoli Monitoring for Transaction Performance ETL. The following steps should be followed:

1. Start the IBM DB2 Control Center utility by selecting **Start -> Programs -> IBM DB2 -> Control Center.**
2. On the IBM DB2 Control Center utility, start the IBM DB2 Data Warehouse Center utility by selecting \textbf{Tools} -> \textbf{Data Warehouse Center}. The Data Warehouse Center logon window appears.

3. Log on to the IBM DB2 Data Warehouse Center utility using the local DB2 administrator user ID (in our case, db2admin).

4. In the Data Warehouse Center window, expand the Warehouse Sources folder. As shown in Figure 5-13, there are two entries for the IBM Tivoli Monitoring for Transaction Performance ETL programs that must be configured, as follows:
   - APF\_TAPM\_Source
   - BWM\_TWSM\_Source

   Edit the properties of each one of these entries.

![Figure 5-13 Enterprise Transaction Performance ETL Source](image)

To edit the properties of the ETL sources, right-click an object and select \textbf{Properties} from the pop-up menu. Then select the \textbf{Data Source} tab. Fill in the database instance owner user ID information. The values for our environment are shown in Figure 5-14 on page 136, using APF\_TAPM\_Source as an example.
5. For the Enterprise Transaction Performance target ETL shown in Figure 5-15, expand the Warehouse Target folder, right-click the APF_TWH_CDW_Target, and select Change User ID and Password as shown in Figure 5-15.
Fill in the user ID information. The values used in our lab environment are shown in Figure 5-16.

![Change User ID and Password](image)

*Figure 5-16  APF_TWH_CDW_Target user ID information*

**Modifying the schema information**

For the ETLs to successfully access the data within the defined sources, an extra step is needed to make sure that the table names referenced by the ETLs match those found in the source databases.

For all tables used in the two IBM Tivoli Monitoring for Transaction Performance Warehouse sources APF_TAPM_Source and BWM_TWSM_Source, verify that the schema information is filled out and that the table names do not contain creator information. The default situation immediately after installation is shown in Figure 5-17 on page 138. Note that the table names all include the creator information (the part before the period) and the schema field has been left blank.
To provide Tivoli Enterprise Data Warehouse with the correct schema and table information, follow this procedure for every table in each of the IBM Tivoli Monitoring for Transaction Performance ETL sources:

1. On the TEDW Control Center server, using the Data Warehouse Center window, expand **Warehouse Sources**.
2. Select the appropriate source—for example **APF_TAPM_Source**—and expand it to see the subfolders.
3. Open the **Tables** folder.
4. Right-click each table that appears in the right pane of the Data Warehouse Center window, and select **Properties**. The properties dialog appears, as shown in Figure 5-18 on page 139.
Figure 5-18  Warehouse source table properties

Note that Tivoli Enterprise Data Warehouse inserts a default name in the Table schema field, and that Table name contains the fully qualified name of the table enclosed in quotes.
5. Type the name of the table creator or schema to be used in the Table schema field, and remove the creator information (including periods and quotes) from the Table name field.

The values used in our lab are shown in Figure 5-19.

![Figure 5-19 Table schema and table name for TMTP Warehouse sources](image)

These steps should be performed for all of the tables referenced by the two IBM Tivoli Monitoring for Transaction Performance Warehouse sources APF_TAPM_Source and BWM_TWSM_Source. Upon completion, the list of tables displayed in the right pane of the Data Warehouse Center window should look similar to the one shown in Figure 5-20 on page 141, where all the schema information is filled out, and no table names include the creator information.
5.2.4 Activation

Before the newly defined ETLs can start extracting data from the source databases into the Tivoli Enterprise Data Warehouse environment, they must be activated. This implies that a schedule must be defined for each of the main processes of the ETLs. After providing a schedule, it is also necessary to change the operation mode of all of the related ETL components to production in order for Tivoli Enterprise Data Warehouse to start processing the ETLs according to the specified schedule.

Scheduling the ETLs

To extract data periodically from the source database into the data warehouse, a schedule must be specified for all of the periodic processes. This is also the case for one-time processes that have to be run to initiate the data warehouse environment for each application area - such as Tivoli Application Performance Management or Tivoli Internet Management Server.

Table 5-3 on page 142 lists the process that must be scheduled for the IBM Tivoli Monitoring for Transaction Performance ETLs to run.
Use the same steps to schedule a process that has to run once or multiple times. The only difference between one-time and periodically executed processes is the schedule provided. The following provides a brief walk-through using APF_c05_Initialize_Process to describe the required steps:

1. Using the Data Warehouse Center window of the TEDW Control Center server, expand Subject Areas.

2. Select the appropriate Subject Area - for example, APF_TivoliEnterpriseTransactionPerformance_v1.1.0_Subject_Area, and expand it to see the processes. Right-click the process to schedule it; in our example for APF_m05_Initialize_Process, select Schedule, as shown in Figure 5-21 on page 143.
This opens the dialog shown in Figure 5-22 on page 144.
Table 5-22 shows how to use the Interval setting **One time only** to ensure that the process is run only once. This is primarily used to initiate the databases in the data warehouse prior to the first imports, but may also be used to clear out all previously imported historical information.

For most tasks the Interval setting should be different from One time only to accommodate for data import on a regular, periodic basis.

---

**Figure 5-22** ETL schedule configuration for one-time execution of a process
The hourly interval may produce extra load on the source databases and thereby degrade performance of the Tivoli Management Environment. However, in our test environment this aggressive setting did not cause problems.

In general, data import should be scheduled to take place when Management activity is low—for example, every morning at 2am—with a 24-hour interval, or with a very short interval (for example, 15 minutes) to ensure that only small amounts of data have to be processed. The usage pattern (requirements for up-to-date data) of the data in the data warehouse should be used to determine which strategy to follow.

**Note:** As Tivoli Enterprise Data Warehouse does not allow you to change the schedule once the operational mode has been set to production, demote the mode of the processes to Development or Test if you want to change the schedule. Do not forget to promote the mode of the processes back to Production to activate the new schedule.
Changing the ETL status to Production

All IBM Tivoli Monitoring for Transaction Performance ETL processes are run by components that have Development status set as default. In order for them to run, their status must be changed from Development to Production.

Certain steps must be performed for all processes corresponding to your Warehouse Enablement Pack. Table 5-4 provides the complete list.

Table 5-4  Warehouse processes and components

<table>
<thead>
<tr>
<th>Warehouse Enablement Pack</th>
<th>Process</th>
<th>Components</th>
</tr>
</thead>
<tbody>
<tr>
<td>TMTP:ETP</td>
<td>APF_c05.Initialize_Process</td>
<td>APF_c05_s010_init</td>
</tr>
</tbody>
</table>
|                            | APF_c10_CDW_Process          | APF_c10_s010_rim  
|                            |                              | APF_c10_s020_comp 
|                            |                              | APF_c10_s030_compatattr 
|                            |                              | APF_c10_s040_msmt 
|                            |                              | APF_c10_s050_msmt |
| TMTP:WTP                   | BWM_c05.Initialize_Process   | BWM_c05_s010_comp  
|                            |                              | BWM_c05_s020_init_extract_sti  
|                            |                              | BWM_c05_s030_transform_sti 
|                            |                              | BWM_c05_s040_sti_rtt 
|                            |                              | BWM_c05_s050_sti_st 
|                            |                              | BWM_c05_s060_sti_th 
|                            |                              | BWM_c05_s070_init_extract_qos  
|                            |                              | BWM_c05_s080_extract_url 
|                            |                              | BWM_c05_s090_transform_qos 
|                            |                              | BWM_c05_s100_qos_bst 
|                            |                              | BWM_c05_s110_qos_prt 
|                            |                              | BWM_c05_s120_qos_rtt |
|                            | BWM_c10_Load_Warehouse_Process | BWM_c10_s010_comp  
|                            |                              | BWM_c10_s020_extract_sti  
|                            |                              | BWM_c10_s030_transform_sti 
|                            |                              | BWM_c10_s040_sti_rtt 
|                            |                              | BWM_c10_s050_sti_st 
|                            |                              | BWM_c10_s060_sti_th 
|                            |                              | BWM_c10_s070_extract_qos  
|                            |                              | BWM_c10_s080_extract_url 
|                            |                              | BWM_c10_s090_transform_qos 
|                            |                              | BWM_c10_s100_qos_bst 
|                            |                              | BWM_c10_s110_qos_prt 
|                            |                              | BWM_c10_s120_qos_rtt |

We use APF_c05.Initialize_Process as an example to describe the process.
On the TEDW Control Center server using Data Warehouse Center window, right-click each of the processes. Choose **Mode** -> **Production**, as shown in Figure 5-24.

![Figure 5-24  Promoting ETL scheduled processes to Production status](image)
You can also select multiple components and set the desired mode for all of them at the same time as shown in Figure 5-25.

![Figure 5-25 Setting production mode for multiple components](image)

Now all of the processes are ready and scheduled to be run in production mode. When the data collection and ETL processes are performed, historical data from IBM Tivoli Monitoring for Transaction Performance will be available to be used by other applications such as Tivoli Service Level Advisor.
Troubleshooting

This chapter contains information and procedures that have been found useful for recovering from problems when working with the IBM Tivoli Monitoring for Transaction Performance components and related Tivoli products.

The chapter is grouped into the following major areas:

- “Installation notes” on page 150
- “TMTP:ETP configuration parameters” on page 162
- “Increasing performance of data upload to TAPM RIM” on page 165
- “Troubleshooting procedures” on page 168
6.1 Installation notes

In this section you will find useful information to troubleshoot and recover from problems related to installation and/or migration of IBM Tivoli Monitoring for Transaction Performance components and related products.

6.1.1 Problems installing the Enterprise Transaction Performance Web Console

The installation of the Web Console component fails if the installation is performed on a machine on which Norton AntiVirus is installed. A workaround is to uninstall or disable realtime protection by Norton AntiVirus before installing the Web Console, and reactivate virus protection afterward.

The installation notes in the Enterprise Transaction Performance User's Guide are misleading as to the purpose of entering values for Installation Options during the classic install process. Refer to page 24 of the IBM Tivoli Monitoring for Transaction Performance: Enterprise Transaction Performance User's Guide Version 5.1, GC23-4803. The following sentence is incorrect:

“For full installation, type in the userid and password under which the Web GUI server will run.”

It should instead state:

“For full installation, type in the User ID and Password for the user ID that will launch the service for IBM HTTP Server on the installation target machine.”

The Set Install Options dialog for IBM Tivoli Monitoring for Transaction Performance 5.1 Web Console maps values to the Information for Service Setup dialog for the IBM HTTP Server v 1.3.19 as follows:

| User Owner of this service | User ID |
| User Password | Password |
| Verify Password | Enter Password again for verification |

The user ID must have administrative privileges on the machine where the IBM HTTP Server is to be installed along with the Enterprise Transaction Performance Web Console. This user ID also must be defined within the Tivoli software and must have a senior role in the TMR.

If you are installing the IBM HTTP Server separately prior to installing Enterprise Transaction Performance Web Console, then you should leave the Set Install Options dialog empty.
The default user ID and password for logging on to the IBM Tivoli Monitoring for Transaction Performance Web Console are tapm and tapmtapm respectively. This combination is unrelated to the user ID and password used for creating the TAPM RIM database. You may add an additional user ID and password to access the Web Console using the htpasswd.exe utility as shown in Example 6-1.

Example 6-1   How to set additional logon IDs for the Web Console

```
cd c:\IBMHTTPSERVER
htpasswd %BINDIR%\TME\MAR\web\tomcat\conf\tapm.auth Administrator
(Enter Password)
```

Assume the environment variables have been set properly in the example above. This example would add the Administrator user ID to the list of logon IDs for the Enterprise Transaction Performance Web Console.

If you are attempting to upgrade your Tivoli Application Performance Management 2.1 product to IBM Tivoli Monitoring for Transaction Performance Version 5.1, refer to the User's Guide regarding some very important procedures for migration. You should consider any upgrade with this product to be a migration rather than a normal upgrade.

To upgrade the IBM HTTP Server from Version 1.3.12.2 which was delivered with Tivoli Application Performance Management 2.1 to IBM HTTP Server Version 1.3.19, you must run the setup.exe binary and install the new version of IBM HTTP Server to a different directory. This must be done prior to upgrading Web GUI 2.1 to Web Console 5.1, or the following installation error will occur:

Example 6-2   Upgrade failure on IBM HTTP Server 1.3.19

```
+ CHECK_WGETKEY=*** ERROR: Unable to open key,
  RegOpenKey = 2, Line = 385 + set -e
  ERROR: Unable to open key
  ++ echo Unable to open key
  + CHECK_WGETKEY=Unable to open key
  + [ Unable to open key = Unable to open key ]
  + echo IBM HTTP Server 1.3.19 is not installed
```

6.1.2 Installing IBM HTTP Server for the Web Console

You may encounter problems related to the IBM HTTP Server 1.3.19 installation as a separate process instead of using the combined classic install method used with IBM Tivoli Monitoring for Transaction Performance Web Console:

- The error in Example 6-3 on page 152 may occur during the execution of setup.exe for IBM HTTP Server 1.3.19 at about 94% completion of the installation process.
Example 6-3  IBM HTTP Server installation error at 94%

Automatically using MD5 format on Windows...
C:\Program Files\IBM HTTP Server\htpasswd.exe: unable to create temporary file ‘\s13o.1’ fopen Permission denied.

This error will flash by quickly on the screen, so you may not notice it unless you happen to be watching. The error occurs due to the white spaces in the directory path where htpasswd.exe is attempting to create an authority file. Installing IBM HTTP Server in a different directory without white spaces (for example, c:\IBMHTTPSERVER) will fix the problem. In fact, when using the combined installation method with Web Console, IBM HTTP Server is installed to a directory called c:\IBMHTTPSERVER.

After using Start -> Control Panel -> Add/Remove Programs to uninstall IBM HTTP Server, it may be necessary to remove the directory called IBM HTTP SERVER before proceeding to the re-install of the program using setup.exe.

Note that the use of sh wuninst TAPMWEB <managednode> to uninstall the Web Console on the TMR Server also will uninstall any existing versions of IBM HTTP Server on the Managed Node specified in the command. Unlike the previous procedure used to remove the program called IBM HTTP SERVER, this procedure using wuninst will also remove the directory called IBM HTTP SERVER in addition to invoking the uninstall shield.

▶ If you encounter the Internal Server Error shown in Example 6-4 during the Web Console logon, the paging space resource might need to be increased.

Example 6-4  Internal Server Error

Internal Server Error:
The server encountered an internal error or misconfiguration and was unable to complete your request.
Please contact the server administrator and inform them of the time the error occurred, and anything you might have done that may have caused the error.
More information about this error may be available in the server error log.

We fixed this problem by increasing the paging space from 288 to 476 and changing the server profile from Application to Background. Although it is not documented in the Enterprise Transaction Performance 5.1 user’s guide, the unofficial requirement for page space is that it should be twice as large as the amount of memory on the system. When the Web Console is installed on a target system with 512KB RAM, it seems as if this problem does not appear.

▶ Running two Tomcat servers on the same machine also may cause problems, so refer to existing procedures for changing the ports on which the Web
Console listens when the host machine has other applications using IBM HTTP Server:

a. Launch services.

b. Stop the IBM HTTP Server Service.

c. Open the c:\IBMHTTPServer\conf\httpd.conf file and edit the following lines so that the port is 82 instead of 80:

   #Port: The port the standalone listens to.
   Port 80

   Save the file in text format.

d. Restart the IBM HTTP Server Service.

Refer to “Preparing for multiple consoles on one system” on page 38 or the redbook *Tivoli Web Solutions: Managing Web Services and Beyond*, SG24-60-49, for more details about how to change the ports for the Web Transaction Performance Tivoli Internet Management Server, if there appears to be port conflicts between the Web Console and the Tivoli Internet Management Server.

For a discussion about selecting port numbers for installation of Tivoli Enterprise Data Warehouse and IBM Console, refer to the redbook *Introduction to Tivoli Enterprise Data Warehouse*, SG24-6607.

### 6.1.3 Installing and configuring the TAPM RIM database

When using the DB2 Command Line Processor to create a DB2 TAPM RIM database, the following notes may be helpful:

First, note that the designers of the scripts assume that the scripts will be run by a user with the authority to create the database and update the schema, given the policy environments that most customers have related to database creation and maintenance. So, it is very important to seek the help of your site database administrator (DBA) in carrying out the procedures below.

Assuming that the site DBA already has created a DB2 database called TAPM; and that $DB2DIR points to DB2INST1, the parent directory for the DB2 and the bin directories for the DB2 Client; and that the platform is w32-ix86, then the following commands must be considered:

```bash
>cd $DB2DIR\bin
>db2icrt db2inst1 -s client
>set DB2INSTANCE=db2inst1
>db2cmd
>db2 connect to tapm user db2inst1 using <password>
>db2 -td; -f $BINDIR/TME/MAR/SQL/cr_db.db2.sql
```

Chapter 6. Troubleshooting 153
The following DB2 Command Line Processor options can be very helpful in resolving any errors that may arise from the attempts to create the TAPM RIM database:

-**es** SQLCODE is displayed after each SQL statement (used with +o)
-**o** Output (query results and messages) is displayed on stdout
-**s** If an error is encountered during execution of a statement, the CLP stops and exits to the operating system.
-**td$** The statement termination character is set to $. Input lines are concatenated until a $ is encountered, and the result is handled as one statement.
-**v** All statements and commands are echoed to stdout before they are executed.

**Note:** The script referred to above, cr_db.db2.sql, is created by running the script named cr_apm_db.sh provided with the ITMTP 5.1 ETP installation in the $BINDIR/TME/MAR/SQL directory and answering all the dialog questions for the script. The cr_db.db2.sql script will contain commands to create tablespaces and tables and will need to be reviewed by the DBA for consistency prior to being used.

There are two important considerations related to the setup and configuration of DB2 TAPM RIM. First, the DB2 user ID for the RIM object must be the DB2 instance owner, this user ID must match the instance name and must exist on both the DB2 client and server. Second, for w32-ix86–supported platforms, the system variable DB2INSTANCE must be set to the RIM DB2 instance name. (Check the db2cli.ini file for consistency related to the Server ID field.)


### 6.1.4 Different types of distributions

Normally, a MarProfile is distributed to the tapmagent engine running on a TMA endpoint rather than to dm_ep_engine (classic DM 3.7) or to the Java engine (ITM 5.1.1). Here are some example formats for the \texttt{wdistrib} command related to the different types of profiles involved in using IBM Tivoli Monitoring for Transaction Performance.
**Example 6-5  Different types of distributions with wdistrib**

```
wdistrib -m -l over_all_no_merge @SentryProfile:<profilename>
@Endpoint:<epname>

wdistrib -m -l over_all_no_merge @MarProfile:<profilename>
@Endpoint:<epname>

wdistrib -m -l over_all_no_merge @Tmw2kProfile:<profilename>
@Endpoint:<epname>
```

The first distribution example pushes a classic DM 3.7 profile to the dm_ep_engine process running on a TMA endpoint.

The second distribution example pushes a MarProfile to the tapmagent process running on a TMA endpoint.

The third distribution example pushes a resource model to the Java engine for ITM 5.1.1 running on a TMA endpoint. The over_all_no_merge distribution option is only available for the command line and not on the Tivoli Desktop. It prevents any previous distributions from being merged in with the current distribution to the TMA endpoint.

A new command called `wdmdistrib` provided under ITM 5.1.1, which uses mdist2 for distributions, is an entirely new type of distribution. For information, refer to *IBM Tivoli Monitoring Version 5.1: Advanced Resource Monitoring*, SG24-5519.

The syntax of the `wdmdistrib` command is:

```
wdmdistrib -d -p <profilename>
```

Check status with:

```
wmdist -l -v -i <distribution id>
```

Use of the command is shown in Example 6-6.

**Example 6-6  Distribution of the TMTP_AggregatedData Resource Model Profile**

```
$wdmdistrib -d -p TMW2k_TMTP dolphin
AMW0162I - Operation successfully submitted. Distribution ID is 1143638406.1
```

Check `$DBDIR/distmgr.log` for mdist2 distribution success or failure, looking for status=6 to indicate successful distribution on a specific endpoint. If mdist2_gui is installed to the Framework, then click on the Distribution Status icon on your Tivoli Desktop to check the status of the `wdmdistrib` command related to the specific distribution id displayed from the command.
Example 6-7  Example of wmdist command to check status of distribution

```bash
$wmdist -l -v -i 1143638406.1
Distribution ID: 1143638406.1
Name: TMW2k_TMTP(install)
Owner: X220SVR\Administrator@x220svr.itsc.austin.ibm.com
Size: 199
Source Application: Distributed Monitoring
Source Node: 1143638406.1.575
Start Time: 2002.11.20 10:00:08
Finish Time: 2002.11.20 10:00:22
Expire Time: 2002.11.23 10:00:07
Update Time: 2002.11.20 10:00:22
Targets: 1
Complete: 1(100%)
Waiting: 0
Paused: 0
Ready: 0
Unavailable: 0
Receiving: 0
Interrupted: 0
Sending: 0
Successful: 1
Failed: 0
Canceled: 0
Rejected: 0
Expired: 0
Stored: 0
```

During failed distributions, a Profile Manager is created for the failed targets (such as Profile_Manager_For_Failed_Targets) warning of the distribution failure. Check the following logs to troubleshoot the `wmdistrib` results:

<table>
<thead>
<tr>
<th>Component</th>
<th>Log File</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gateway Repeater</td>
<td>$DBDIR/gatelog</td>
</tr>
<tr>
<td>ManagedNode Repeater</td>
<td>$DBDIR/rpt2log</td>
</tr>
<tr>
<td>Distribution Manager</td>
<td>$DBDIR/distmgr.log</td>
</tr>
<tr>
<td>TMR Server</td>
<td>$DBDIR/AMW/logs/msg_profile-name.log</td>
</tr>
<tr>
<td>Gateway</td>
<td>$DBDIR/AMW/logs/trace_tmnt_profile_coren.log</td>
</tr>
<tr>
<td>TMA endpoint</td>
<td>$LCF_DATDIR/lcfd.log</td>
</tr>
</tbody>
</table>

**Note:** To configure the log file for the tmw2k_profile_core process named `trace_tmnt_profile_coren.log`, use the `wdmconfig` command with `core.trace_level=2` and `core.trace_size=2000000 (2MB)`. Defaults are `core.trace_level=1` and `core.trace_size=500000 (0.5MB)`.
For more information about the `wdmdistrib` command and `mdist II`, refer to:

- *Tivoli Software Distribution 4.0*, SG24-6026
- *Tivoli Management Framework Reference Manual*, v 3.7.1
- *Tivoli Planning for Deployment Guide*, v 3.7.1

### 6.1.5 Use of the `/tmp` directory to store semaphore files on AIX

The files shown in Example 6-8 are stored to `/tmp` by Enterprise Transaction Performance 5.1 and should be protected from removal by processes such as skulker on AIX:

**Example 6-8  Semaphore files in `/tmp` that must be protected from cleanup**

```bash
controls_ae.tapm_emp
controls_cl.tapm
controls_um.tapm_mem
controls_ae.tapm_mem
controls_ea.tapm
controls_um.tapm_ovr
controls_ae.tapm_ovr
controls_um.tapm_emp
controls_ext.tapm
```

### 6.1.6 Too many rows in TAPM_VERSION table

If you run the script that creates a new database schema on an existing database schema, a new row is created in the TAPM_VERSION table instead of migrating from the old one. If this happens, remove the new row and run the upgrade script.

There is a TAPM_VERSION database table that should always only have one entry, which is the current version of the database. However, in certain situations an additional row can be added, and this will cause problems with Tivoli Application Performance Management updates in the future from the various gateways. Symptoms of this situation looks like Example 6-9:

**Example 6-9  Symptoms for too many rows in the TAPM_VERSION table**

```bash
Mar 08 19:24:59 TapmDB_Get_DBVersion: *** EXCEPTION *** Too many entries found from the table 'TAPM_VERSION'.

```

The key error is that *too many entries* have been found in the TAPM_VERSION table, not that the new schema has not been applied.
The TAPM_VERSION table in the Tivoli Application Performance Management (TAPM) database schema has two or more rows. This table should have just one row containing the information about the database schema version:

- TAPM 2.0 = 200
- TAPM 2.1 = 210
- TAPM 2.1 + patch 1 = 2101
- TMTP:ETP 5.1 = 210

If the schema upgrade script is run more than one time, it will produce more than one entry in the TAPM_VERSION database table and cause the error in the MLM on the Gateway shown previously.

**Workaround:** Delete the rows in TAPM_VERSION that are not needed. There should be only one row with the DB schema version.

### 6.1.7 Java errors while attempting to uninstall or re-install TEDW 1.1

If the error shown in the Example 6-10 occurs during the re-installation of IBM Tivoli Enterprise Data Warehouse, check the TWH.log file for more information to determine if the error is a Java error for CreateProcess.

**Example 6-10  TEDW 1.1 install error**

- Errors occurred during the installation
- An error occurred and product installation failed. Look at the log file C:\Program Files\TWH\TWH.log for details.
- An error occurred and product uninstallation failed. Look at the log file C:\Program Files\TWH\TWH.log for details.

Check the TWH.log to determine if there are Java errors similar those shown in Example 6-11:

**Example 6-11  TWH.log Java errors**

```plaintext
(Dec 11, 2002 10:13:43 AM), TWH.install,
com.tivoli.pf.install.PsPackageProductAction,
err, An error occurred and product installation failed.
Look at the log file C:\TWH\TWH.log for details.
(Dec 11, 2002, 10:13:43 AM), TWH.instgall,
com.tivoli.pf.install.PsPackageProductAction,
err, ProductException: (error code = 200; message="Java error";
exception = [java.io.IOException: CreateProcess: C:\PS\bin\w32-ix86\mcr.bat
error =3 ])
STACK_TRACE: 14
ProductException: (error code = 200; message="Java error"; exception=
[java.io.IOException: CreateProcess: C:\PS\bin\w32-ix86\mcr.bat error =3 ])
```
If this is the case, you should edit the vpd.properties file located in C:\WINNT or C:\Windows directory to remove the entries for Presentation Services or IBM Tivoli Enterprise Data Warehouse and to leave the intact entries for IBM Tivoli Monitor Web Health Console or other applications that use this file. Entries from a previous installation attempt may still be in this file since the uninstall process does not remove the entries from this file and does not remove the file due to the file being shared among several applications. If there are previous entries in this file for Presentation Services or IBM Tivoli Enterprise Data Warehouse, these entries will prevent the re-installation process from succeeding.

Once the vpd.properties file is checked and amended properly, check again using the net start command to determine whether VWLogger (warehouse logger) and VWKernel (warehouse server) are still running. If not, start these services again before attempting to re-install TEDW 1.1. If you have canceled a TEDW installation for any reason, use DB2 Control Center to drop or remove any of the three databases for TEDW (TWH_MD, TWH_CDW, and TWH_MART) that might have been created during the cancelled installation. If you have experienced a hang of the installation at about 37%, check for a fully qualified host name entry in the hosts file. The installation of TEDW 1.1 will fail unless you specify a full qualified host name (e.g. host.ibm.com) during the installation dialog and provide for resolution of this host name in the hosts file.

Refer to Introduction to Tivoli Enterprise Data Warehouse, SG24-6607 for more information about the installation process.

### 6.1.8 Migration of data on an endpoint?

If you do not stop the engine on the endpoint during the migration process, the engine will be stopped automatically as soon as any of these commands is used:

- wmarstarteng
- wmarstopeng
- wmarcleareng
- wmarlsapp
- wmarlseng
- wmarstartcol
- wmarstopcol
- wmarsetstatus
- wmargetdata
- wmargetrawdata
- wmargetstatus

Therefore, after running any of these commands on the endpoint, the engine must be restarted using the wmarstarteng command.
6.1.9 **Enterprise Transaction Performance ETL limitation**

When the Enterprise Transaction Performance ETL runs, it pulls data from the TAPM RIM database. At the time of data extraction from the TAPM RIM database, the ETL will always pull an entire day of data instead of pulling data for a partial day. For example, if the Enterprise Transaction Performance ETL runs at 7:00 p.m. on Wednesday, then the data for Wednesday will not be copied by the ETL. The data for Wednesday will be copied to the central data warehouse when the Tivoli Application Performance Management ETL runs on Thursday.

In some cases, there is the potential that the ETL will not copy all of the data from the TAPM RIM database. This can happen when there is some delay in getting the data to the database. For example, if Enterprise Transaction Performance is collecting data hourly, then the TAPM RIM database may have data from an endpoint up to Wednesday at 9:00 a.m. If the gateway goes down for three days, the data from that endpoint will not make it to the database for three days. In most production environments, the Enterprise Transaction Performance ETL will run daily. This means that on Thursday the Enterprise Transaction Performance ETL will pull the measurement data up to 9 a.m. for the endpoint. On Saturday, when the gateway comes back up, the data from 9:00 a.m. on Wednesday will arrive along with the data for Thursday and Friday. The late data for Thursday and Friday will be moved into the warehouse, but the late data for Wednesday will never be copied to the Warehouse because the gateway was down.

In short, if all of the following conditions occur, then data will not be copied to the central data warehouse database:

1. The TAPM RIM database has data for a partial day relative to an application name, user name, transaction name, and host name.
2. For some reason, the data associated with the remainder of the day is delayed in arriving to the TAPM RIM database.
3. The data has not arrived to the database when the Enterprise Transaction Performance ETL runs on the next day.

---

**Note:** Under Version 2.1 of Tivoli Application Performance Management, the migration timeout was set to five minutes and, in some cases, this timeout caused problems for migration. Under the Enterprise Transaction Performance Version 5.1, the migration timeout is now greater than five minutes.
6.1.10 WebSphere Application Server Version 5.0 configuration

For ARM data to be sent to the tapmagent:

- ARM must be enabled in PMI request metrics on the WebSphere server using the administrative console.
- Diagnostic tracing must be enabled on the WebSphere server for all components of com.ibm.ws.pmi.reqmetrics.
- ARM implementation must be enabled in the server startup file called server.xml.
- Two files installed by Enterprise Transaction Performance 5.1 (armjni.dll and armjni.jar) have to be copied manually to <was_home>/bin and <was_home>/lib, respectively.

For a detailed discussion of these configurations, refer to 3.3, “Enabling transaction breakdown” on page 86.

In addition to configuring the WebSphere Application Server, it is also important to determine which WebSphere application name should be registered with Enterprise Transaction Performance 5.1 using the wmarregapp command. This application name can vary in different beta versions of WebSphere, so it is important to clarify the version being used and how the application name is constructed. In our case, using build p0236.01 (beta 4) of WebSphere 5.0, we had to register with the wmarregapp -a <cell>/<node>/server1 command, where cell and node are the same in our case, both being the nodename ibmtiv8. (This may not always be the case.) Refer to the WebSphere InfoCenter for more information about cell and node.

We determined the actual application name by looking at the results of the wmarlsapp <ep> command after generating some data from the WebSphere application called PlantsByWebSphere. This is where the <cell>/<node>/server1 application name used in the above wmarregapp command appeared first. Note the forward slashes instead of backslashes and the fact that the first character of the string is not a slash (/) character. You also might use the Add Application button during the creation of the MarProfile to register the application name in the TMR Server repository as an ARM-instrumented application.

Note: When WebSphere starts sending data to the tapmagent, it will record the application name in the source for wmarlsapp first. However, the application name will not be recorded to applications.dat until tapmagent actually collects data from WebSphere, and this cannot happen until a monitoring profile for the application is distributed to the endpoint where the WebSphere server is running.
Once the application name is properly registered, a MarProfile must be created for this registered application and distributed to the Tivoli endpoint on the WebSphere Application Server that is hosting this application. This will install and start the tapmagent that will be interacting with the WebSphere application through ARM calls.

It is highly recommended that a test MarProfile be distributed to the endpoint first and that the armtest binary be employed to test the functionality of the Enterprise Transaction Performance endpoint and especially the tapmagent process.

Note: ARM for PMI Request Metrics is not enabled (true) by default for the WebSphere server and that InstanceLogging is not enabled (true) by default for the TMTP endpoint.

Check the /websphere/appserver/logs/trace.log (or the equivalent customized filename for this log file) for more detail on what should be going into the TAPM RIM database and into the Enterprise Transaction Performance binary files on the endpoint. You will need to configure performance monitoring within Performance Monitoring Infrastructure (PMI) on WebSphere Application Server to get data logged into trace.log as well as sent to TAPM RIM via ARM calls. Another handy debugging tool is the Tivoli Performance Viewer installed with WebSphere Application Server Version 5.0. This tool can be used to view detailed performance records on the WebSphere server application.

6.2 TMTP:ETP configuration parameters

Setting up IBM Tivoli Monitoring for Transaction Performance: Enterprise Transaction Performance may seem complicated. The following explains some of the inner workings of various commands and subcomponents as well as the implications of different configuration parameters.

6.2.1 Scheduling upload of instance data: InstanceRandTimeRange

The InstanceRandTimeRange parameter is used to randomize the exact time when the instance data upload process starts on the endpoint. A random time value (default value is set to 1 hour) is generated in the interval between 0 (zero) and the InstanceRandTimeRange value. This time value is added to the scheduled start time (depending on the InstanceStartTime and the InstanceTimeStep) to provide the exact time for starting the instance data upload process.

This value is calculated when the Tivoli Application Performance Management agent starts on the endpoint, and it remains the same until the agent is restarted.
For example, if these are the settings:

\[
\begin{align*}
\text{InstanceTimeStep} &= 1 \text{ (every 1 hour)} \\
\text{InstanceStartTime} &= 00:00 \\
\text{InstanceStopTime} &= 23:59 \text{ (that means the upload occurs the entire day)} \\
\text{InstanceRandTimeRange} &= 00:30
\end{align*}
\]

then a random value between 0 and 30 minutes is calculated (for example, 12 minutes) so that the upload process will start at the following times:

- 00:12
- 01:12
- 02:12
- 03:12
- ...

and so on for the entire day unless the Tivoli Application Performance Management agent is stopped and restarted and a new value is calculated for that endpoint. If so, the upload process may start at different times. If there is a question here, check the log files to see whether the Tivoli Application Performance Management agent has been recycled.

The InstanceRandTimeRange parameter must be set to a value less than or equal to the InstanceTimeStep. So, for example, if InstanceTimeStep=8, set the InstanceRandTimeRange between 00:00 and 08:00. Setting a value that is out of this range will result in the value being automatically set to the default (01:00).

This is the description from the user's guide:

**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 the instance logging process.

### 6.2.2 The InstanceTimeStep variable

The upload of instance data is performed depending on the InstanceEnabled, InstanceTimeStep, InstanceStartTime, InstanceEndTime, and InstanceRandTimeRange parameters.

- **InstanceTimeStep** is the number of hours between every upload of instance data. For example InstanceTimeStep=1 means that the instance data is uploaded every hour. The default is InstanceTimeStep=24, which means that the data is uploaded only once per day.

- **InstanceStartTime** is the start time of the instance data upload process.
- **InstanceStopTime** is the stop time of the instance data upload process.
  
  For example:
  ```
  InstanceTimeStep=1
  InstanceStartTime=16:00
  InstanceStopTime=19:15
  ```
  
  means that the instance data upload process will occur every hour between 16:00 and 19:15 (that means at 16:00, 17:00, 18:00, 19:00).

- **InstanceRandTimeRange** is used to randomize the exact time when the instance data upload starts.

  In the previous example, InstanceRandTimeRange=10, means that the instance data upload process will occur:
  1. between 16:00 and 16:10
  2. between 17:00 and 17:10
  3. between 18:00 and 18:10
  4. between 19:00 and 19:10

  **Note:** The same explanation is valid for the AggregationTimeStep, AggregationStartTime, AggregationEndTime, and AggregationRandTimeRange parameters related to the summary data.

The marinst<date>.dat file is created and sent to the gateway from the endpoint. Then it is uploaded from the gateway to the database. Also check the DBUploadStartTime and DBUploadStopTime parameters on the gateway using the `wmargetstatus -g` and `wmarsetstatus -g` commands to control when the upload from the gateway to the database will occur.

Here are some sample commands showing how to set the variables for upload process:
  ```
  - wmarsetstatus -g DBUploadStartTime=HH:MM <hostname for gateway>
  - wmarsetstatus -g DBUploadStopTime=HH:MM <hostname for gateway>
  - wmarsetstatus InstanceTimeStep=HH:MM <Endpoint>
  - wmarsetstatus InstanceStartTime=HH:MM <Endpoint>
  - wmarsetstatus InstanceEndTime=HH:MM <Endpoint>
  - wmarsetstatus InstanceRandTimeRange=HH:MM <Endpoint>
  ```
6.2.3 What do the options for wmarsetstatus really mean?

To understand the correct variable to set for the tapmagent related to enabling instance data to be logged and stored to TAPM RIM, remember that:

Setting InstanceLogging = true Instance data will be collected on the TMA endpoint.

Setting InstanceEnabled = true Collected instance data will be uploaded to the TAPM RIM database.

The InstanceEnabled variable is not set by default and we have found that this variable sometimes must be reset after a MarProfile distribution when new monitors have been added to the MarProfile.

This is how to set the variables:

wmarsetstatus InstanceEnabled=true, InstanceLogging=true <ep>

6.3 Increasing performance of data upload to TAPM RIM

The DBMaxInstanceRowNo parameter on the gateway sets the number of instance data rows that are uploaded to the database for each connection to the RIM. The default value is 100. This parameter was introduced in patch 2.1-APM-0001 (Refer to Defects 19836 and 20630) in order to increase performance in uploading 100 rows for each connection to the RIM instead of one row (as the value was originally). This value should be increased carefully, however, with considerable attention as to whether the DB server will need more resources (such as allocated memory) in order to insert data to the database in this way; that is, to upload a large number for rows for each connection to RIM.

Related to performance issues, another parameter that should be considered is InstanceRandTimeRange for the endpoint. If a large number of Endpoints is connected to the same gateway (for example, more than 20), then it is better to set the InstanceRandTimeRange to a higher value (for example, 00:45) in order to avoid all of the Endpoints attempting to upload data at the same time.

6.3.1 Why is the TMTP:ETP gateway not sending data to TAPM RIM?

The gateway for the Enterprise Transaction Performance endpoints may seem to be sitting on data uploaded from the endpoint and not sending the data to the TAPM RIM database, even though the wmargetstatus shows that the window for sending data to TAPM RIM is wide open.
The following example shows that the window for sending the data to TAPM RIM on the gateway is wide open for the entire 24-hour period:

DBUploadStartTime=00:00
DBUploadStopTime=23:59

**Possible causes**

Among the possible causes for this are:

- One strong possibility is that the TMTP gateway in question is having to wait on another TMTP gateway to upload its data. Only one TMTP gateway can upload data to TAPM RIM in any specific time period. During this period, the other TMTP gateways have to wait until the RIM can handle their requests to upload data. The intent of being able to set the upload window on the gateway by setting the parameters DBUploadStartTime and DBUploadStopTime is to provide the customer a way of staging the upload times for each TMTP gateway so that they do not conflict with other TMTP gateways in the uploading of data.

Unfortunately, there are no entries in the gateway's log file, tapm.log, to show the process of checking to see when a gateway can send data to TAPM RIM. This is because the TMTP gateway checks the RIM every minute for an opportunity to upload its data to TAPM RIM. The design decision to exclude this logging sequence from tapm.log was intended to keep the log files smaller and help in the ability to analyze the log files. (The tapm.log is located in %TEMP%\tapm directory....TEMP=`wtemp`;cd $TEMP\tapm ....on the TMTP gateway).

However, there is a process by which to check to see whether another TMTP gateway has locked the TAPM RIM or not. In order to check if a gateway is actually sending data to the TAPM RIM, check the TMR server to see whether a gateway has acquired a DB lock as follows:

a. Check for the existence of the file lockInfo.log at this location:

   C:\Tivoli\db\<TMRSERVER>.db\tmp\apf\cache\lockInfo.log

b. If the file exists on the TMR Server, then the lockInfo.log should contain lines as follows:

   
   [lockOwnerName]1225893409.1.347#TMF_ManagedNode::Managed_Node#
   [timeOutValue]30

   The first line contains the OID of the gateway that locked the TAPM RIM (and is supposed to be sending data to the TAPM RIM).

c. To find out which TMTP gateway is identified by this OID listed in lockInfo.log, issue the command: `wlookup -ar ManagedNode` and identify the TMTP gateway that has the resource locked. For example:

   `wlookup -ar ManagedNode | grep <object-id>`
where <object-id> denotes the actual object number contained in the lockinfo.log file.

The lockinfo.log file is a file that is located on the TMR Server in the %DBDIR%\tmp\tapm\cache directory for w32-ix86. It is used by the MarController to store the actual Object ID of the MLM that is locking the TAPM RIM database. This file is created the first time an upload occurs, and the file should be empty when there are no active uploads.

Another possibility for the TMTP gateway not sending its data files to TAPM RIM is that the files may be locked on the TMTP gateway due to being DEFERRED.

The dbCacheList.log is the list of the files to be sent to the database. There are three types of requests for file upload to TAPM RIM. They are: READY, DEFERRED, and IMMEDIATE.

**READY**  READY requests are processed during the database upload time period. The upload time period is defined by the DBUploadStartTime and DBUploadStopTime attributes on the managed node. Use the `wmarsetstatus -g` command to see these.

If a READY request cannot be sent to the database because of a problem with TAPM RIM or the database is down, for example, the request is marked DEFERRED and left in the DB request list. Once a request is marked DEFERRED, it is ignored from that point on in the process and the MLM will no longer attempt to send it to the database. The request must be changed back to READY status before it will be sent to the database. This will occur automatically when the MLM is rebooted or is restarted by user action with `wmarrestartmlm.sh`.

**IMMEDIATE**  An IMMEDIATE request (started by `wmarsenddata` command) will always be processed immediately and this type of request does not depend on the settings of DBUploadStartTime and DBUploadStopTime.

If an IMMEDIATE request of data cannot be honored and the data is not sent to the TAPM RIM database immediately, then the IMMEDIATE request is discarded. In such a case, the data is never sent to TAPM RIM.

**Note:** IMMEDIATE requests are not cached on the MLM.
DEFERRED  If a file on the TMTP gateway is marked as DEFERRED, it may be due to the inability of the MLM to unmarshall the data from the TAPM endpoint. This may occur if the original marshalling of the data occurred on a previous version of Tivoli Application Performance Management or if the lcfd version on the Tivoli Application Performance Management endpoint is incompatible with the Framework version on the managed node that hosts the TMTP gateway.

If this is the case, then the older data files from the Tivoli Application Performance Management endpoint must be removed from each Tivoli Application Performance Management endpoint so that there will not be inconsistencies in the marshalling and unmarshalling of data in Tivoli Application Performance Management 2.1. Also, ensure that the Framework is installed consistently in the TMR region.

6.4 Troubleshooting procedures

This section contains procedures that have been found useful to recover from problems when working with the TMTP products:

- “Clearing a TAPM endpoint” on page 168
- “Recycling mar_mlm and mar_ctrl processes” on page 169
- “Clearing the Windows NT registry to make ClientCapture work” on page 170
- “Fixing corruption in the TAPM database” on page 172
- “Analyzing ETP log files and trace files” on page 174
- “Analyzing Web Transaction Performance log and trace files” on page 178

6.4.1 Clearing a TAPM endpoint

This procedure will clean up the endpoint. Note that this procedure does not involve clearing the registry entries. The steps are:

1. Stop the following processes:
   - tapmagent
   - tapmupmanager
   - mar_ep

2. Delete the following directories and files:
   - "$LCF_DATDIR/Mar"
   - "$LCF_DATDIR/cache/bin/$INTERP/Tme/Mar"
3. Delete the Tivoli Application Performance Management boot method (Mar_Boot) with the `wep` command:

```
EP_OID=`wlookup -r Endpoint <epname>`
wep boot_method list Mar_Boot $EP_OID
wep boot_method remove Mar_Boot $EP_OID
```

Then, check the removal with:

```
wep boot_method list Mar_Boot $EP_OID
```

4. Re-distribute the MarProfile to the endpoint so that everything will be re-created properly, including the boot_method and all directories listed above. Example:

```
wdistrib -m -l over_all_no_merge @MarProfile:<profilename>@Endpoint:<epname>
```

6.4.2 Recycling mar_mlm and mar_ctrl processes

This procedure is specifically designed for use in troubleshooting problems with the TMTP gateway or TMR Server control processes. It is not the equivalent of recycling these processes by rebooting the gateway or recycling the oserv processes on either the TMR Server or the gateway. This procedure should only be used during troubleshooting and not as a normal operational procedure.

1. Shut down the TAPM MLM process at the Gateway:
   a. Get the object ID of the managed note
      ```
      C:\>wlookup -r ManagedNode -o <GWhostname>
      ```
      The output will be similar to this:
      ```
      1699999991.8.7#TMF_ManagedNode::Managed_Node#
      ```
   b. Then use the `idlcall` command to shut down the mar_mlm:
      ```
      C:\>idlcall 1699999991.8.7 MarMlm::Mlm::shutdownMlm
      ```
      Example:
      ```
      MNODE_OID=`wlookup -r ManagedNode -o <GWhostname>`
      idlcall $MNODE_OID MarMlm::Mlm::shutdownMlm
      ```

2. Shut down the TAPM CONTROLLER process at the TMR Server:
   a. Get the object ID of the MarController
      ```
      C:\>wlookup -r distinguished MarController
      ```
      The output will be similar to this:
      ```
      1699999991.1.1165#TMF_SysAdmin::InstanceManager#
      ```
b. Then use the object ID to shut down the MarController:
   C:\>idlcall 1699999991.1.1165 MarController::shutdownController

   Example:
   MCTL_OID='wlookup -r distinguished MarController'
   idlcall $MCTL_OID MarController::shutdownController

3. Check whether the processes mar_mlm (on the Gateway) and mar_ctrl (on
   the TMR Server) go down or stay up. On UNIX machines, Use the command
   ps -ef and grep for mar_mlm (on a Gateway) or mar_ctrl (on the TMR
   Server), or use ntprocinfo with findstr to find mar_mlm or mar_ctrl on
   w32-ix86 machines.

4. (Optional) Shut down all of the RIM processes at the RIM Host.

5. (Optional) Try the RIM-database connection.
   wrimtest -l tapm

6. Restart the MLM at the Gateway (CONTROLLER starts called by MLM).
   C:\>idlcall 1699999991.8.7 MarMlm::Mlm::bootMlm

   Example:
   MNODE_OID='wlookup -r ManagedNode -o <GWhostname>'
   idlcall $MNODE_OID MarMlm::Mlm::bootMlm

   Note: The idlcall will start the MarController on the TMR Server if the
   Gateway MLM fails to start the process.
   C:\>idlcall 1699999991.1.1165 MarController::startLockManager

   Example:
   MCTL_OID='wlookup -r distinguished MarController'
   idlcall $MCTL_OID MarController::startLockManager

7. (Optional) Change the time window at the Gateway for sending data to DB:
   C:\>wmarsetstatus -g LogLevel=3, DBUploadStartTime=00:00,
       DBUploadStopTime=23:59

6.4.3 Clearing the Windows NT registry to make ClientCapture work

If you encounter the following error during a ClientCapture session, execute
the procedure that follows it.

   APF0468E Data Provider for specified application is unknown to this Tivoli
   Application Performance Management installation on Sample.xml

This procedure may be used to clear the Windows NT (or Windows 2000)
registry and redistribute to a Tivoli Application Performance Management
endpoint in order to restore the registry entries cleanly and correct any problems
so that ClientCapture can work properly. Be sure to shut down all Windows NT windows opened on the desktop (such as file browsing, IE or Netscape windows, etc.) prior to starting this procedure.

1. Stop the ARM engine:
   
   wmarstopeng rhillnt

   The output should be similar to this:

   APF0016I The command 'wmarstopeng' was successfully executed.

   Wait few seconds for execution to complete.

2. Unregister the TAPM bridge and iescanner:

   cd $LCF_DATDIR
   cd Mar
   cd ClientCapture
   tapmbridge.exe /UnRegServer
   tapmiescanner.exe /UnRegServer

3. Unregister TAPM dlls:

   C:\>cd \winnt
   C:\WINNT>cd system32
   C:\WINNT\system32>regsvr32 /u %LCF_DATDIR%\Mar\ClientCapture\TAPMIESensor.dll
   C:\WINNT\system32>regsvr32 /u %LCF_DATDIR%\Mar\ClientCapture\TAPMIEHelper.dll
   C:\WINNT\system32>regsvr32 /u %LCF_DATDIR%\Mar\ClientCapture\TAPMNSSensor.dll

4. Remove the Tivoli Application Performance Management endpoint files:

   cd %$LCF_DATDIR%
   rm -r Mar
   cd cache\binw32-ix86\TME
   rm -r Mar
   cd %LCFROOT%
   bash$ pwd
   C:/Program Files/Tivoli/lcf
   bash$ cd bin
   bash$ cd w32-ix86
   bash$ rm -r tools
   bash$ cd ..
   bash$ cd $LCF_CATDIR
   bash$ cd C
   bash$ rm -f marengine.cat
   bash$ cd /winnt
   bash$ cd system32
   bash$ rm -f tapmextlib.dll
   bash$ rm -f libarm32.dll
   bash$ rm -f tapmextlib.dll
bash$ rm -f libarm32.dll
bash$ wdistrib -m -l over_all_no_merge @MarProfile:TAPM_MarProfile
@Endpoint:rhillnt
bash$ wmarregapp -a IBMsites -f
C:\Tivoli\bin\generic\Mar\ClientCapture\Sample.xml
APF0016I The command 'wmarregapp' was successfully executed.
bash$ wmargetapp -a IBMsites
Application 'IBMsites'
Data Provider: ClientCapture
-------------------------------------------
Dependency '/root/not_shared/IBMsites'
Source Host 'rhillnt'
Source Full Path Name 'C:/Tivoli/bin/generic/Mar/ClientCapture/Sample.xml'
bash$ wmarlseng rhillnt
Collection active on application : IBMsites
transaction filter : *
bucket limits (ms) : 100 500 1000 2000 5000 10000
collection interval (s): 600
collection active on : always=EveryDay
start date (YYYY MM DD): 2002 01 08
stop date  (YYYY MM DD): not specified
Correlator Generation : Enable
Summary Data:
local logging : active
data-base storage : active, detail level: Minimum
Instance Data:
local logging : NO_DATA
data-base storage : NO_DATA
Available Extended Information:
Data Provider Name : ClientCapture
Start Collection Parameters :

6.4.4 Fixing corruption in the TAPM database

To correct corruption in the Tivoli Application Performance Management database without dropping the table and re-creating it, use this procedure to manipulate the basic data in the RMDBS data store used by IBM Tivoli Monitoring for Transaction Performance. Prior to issuing the attached SQL scripts, stop all IBM Tivoli Monitoring for Transaction Performance monitoring, and log on to the database with credentials as the creator of the Tivoli Application Performance Management database tables.
To fix database corruptions, run the SQL commands in Example 6-12 as you would any other SQL command.

**Example 6-12  SQL commands to recover from corruption in the TAPM database**

```sql
select max(correlator_id) from correlator;
{get max value}
select CORRELATOR_ID from TAPM_ID;
{to get the old value}
update TAPM_ID set CORRELATOR_ID=<value from selectmax> where CORRELATOR_ID = <oldvalue>;

select max(HOST_ID) from HOSTS;
{get max value}
select HOST_ID from TAPM_ID;
{get old value}
update TAPM_ID set HOST_ID=<value from selectmax> where HOST_ID = <oldvalue>;

select max(USER_ID) from USERS;
{get max value}
select USER_ID from TAPM_ID;
{get old value}
update TAPM_ID set USER_ID=<value from selectmax> where USER_ID = <oldvalue>;

select max(APPL_ID) from APPLICATIONS;
{get max value}
select APPL_ID from TAPM_ID;
{get old value}
update TAPM_ID set APPL_ID=<value from selectmax> where APPL_ID = <oldvalue>;

select max(TX_ID) from TRANSACTIONS;
{get max value}
select TX_ID from TAPM_ID;
{get old value}
update TAPM_ID set TX_ID=<value from selectmax> where TX_ID = <oldvalue>;

select max(TX_CLASS_ID) from ID_INFO;
{get max value}
select TX_CLASS_ID from TAPM_ID;
{get old value}
update TAPM_ID set TX_CLASS_ID=<value from selectmax> where TX_CLASS_ID = <oldvalue>;

select max(DATA_ID) from METRICDATA;
{get max value}
select DATA_ID from TAPM_ID;
```
{get old value}
update TAPM_ID set DATA_ID=<value from selectmax> where DATA_ID = <oldvalue>;

select max(DEF_ID) from METRIC_DEF;
{get max value}
select DEF_ID from TAPM_ID;
{get old value}
update TAPM_ID set DEF_ID=<value from selectmax> where DEF_ID = <oldvalue>;

select max(METRIC_ID) from METRIC_INFO;
{get max value}
select METRIC_ID from TAPM_ID;
{get old value}
update TAPM_ID set METRIC_ID=<value from selectmax> where METRIC_ID = <oldvalue>;

select max(INFO_ID) from UUID_INFO;
{get max value}
select INFO_ID from TAPM_ID;
{get old value}
update TAPM_ID set INFO_ID=<value from selectmax> where INFO_ID = <oldvalue>;

select max(INSTANCE_ID) from TX_INSTANCE;
{get max value}
select INSTANCE_ID from TAPM_ID;
{get old value}
update TAPM_ID set INSTANCE_ID=<value from selectmax> where INSTANCE_ID = oldvalue>;

6.4.5 Analyzing ETP log files and trace files

This is a summary of all the log files used by Enterprise Transaction Performance Version 5.1 and IBM Tivoli Monitoring for Transaction Performance Web Console:

- **tapm_engine.log**
  - **Location**: $LCF_DATDIR/apf/logs
  - **LogLevel**: Controlled by using wmarsetstatus LogLevel=x <epname> or editing the $LCF_DATDIR/apf/tapm_ep.cfg file under [ENGINE::LOG] section.
  - **Used by**: tapmagent and tapmupmanager processes

- **tapm_ep.log**
  - **Location**: $LCF_DATDIR/apf/logs
LogLevel Controlled by using wmarsetstatus LogLevel=x
<epname> where x=1,2,3 for loglevel, or editing the
$LCF_DATDIR/apf/tapm_ep.cfg file under
[ENDPOINT::LOG] section.

Used by mar_ep, mar_up, and mar_aggregator processes

» tapm_epp_ctrl.log
Location $LCF_DATDIR/apf/logs
LogLevel Controlled by editing the
$LCF_DATDIR/apf/tapm_ep.cfg file under
[TECTRL::LOG] section.

Used by EPP data provider

» tapm_cc.log (formerly TAPMBridge.log)
Location $LCF_DATDIR/apf/logs
LogLevel Can only be modified by special tools available
through Customer Support

Used by TAPMBridge.exe

» tapm.log
Location  'wtemp'/apf/logs
LogLevel Can be modified using wmarsetstatus -g LogLevel=x
<gatewayname> where x=1,2,3 for loglevel, or editing
the 'wtemp'/apf/tapm.cfg file on the gateway machine.

Used by mar_mlm process on the gateway

» tapm_vug.log (formerly TAPMScriptCtrl.log)
Location $LCF_DATDIR/apf/logs
LogLevel Cannot be modified

Used by TAPMScriptController.exe

» trace_playback_driver.log
Location:$LCF_DATDIR/apf/logs
LogLevel Can be modified using wmarsetstatus LogLevel=x
<epname> where x=1,2,3 for loglevel, or editing the
$LCF_DATDIR/apf/tapm_ep.cfg file under
[ENGINE::LOG] section.

Used by RecordPlayback data provider

» trace_playback_session.log
Location $LCF_DATDIR/apf/logs
LogLevel Can be modified using wmarsetstatus LogLevel=x
<epname> where x=1,2,3 for loglevel, or editing the
$LCF_DATDIR/apf/tapm_ep.cfg file under
[ENGINE::LOG] section.

Used by RecordPlayback data provider
- **tapm_web.log**
  
  **Location**
  - `wtemp/apf/logs`
  
  **LogLevel**
  - Can be modified by editing `wtemp/apf/logs/tapm_web.cfg` file under the `[WEBSERVER::LOG]` section.
  
  **Used by**
  - ETP Web Console

- **wrimtrace tapm <ERROR | INFORMATION>**
  
  **Location**
  - Normally, `tmp/rim_db_log`, but can be changed using `odadmin environ set` command to set `$RIM_DB_LOG` variable.
  
  **LogLevel**
  - There are 4 trace levels for RIM (NO_TRACE, ERROR, INFORMATION, and ERROR | INFORMATION). For the LogLevel change to take effect, all RIM_* processes/services must be killed (e.g. RIM_DB2_prog, RIM_DB2_Agent) and allowed to respawn automatically. Error logging is not set by default, so the `wrimtrace tapm <trace leve>` command must be issued to set the appropriate trace level.
  
  **Used by**
  - RIM_<db>_prog and RIM_<db>_Agent processes/services during upload to TAPM RIM

**Notes on the tapm.log file:**

1. If an IMMEDIATE request for upload of data to TAPM RIM is received (those generated by invoking `wmarsenddata` for the endpoint), and the data is not sent to the TAPM RIM database due to some type of error, then the IMMEDIATE request is discarded and is not put into a DEFERRED state.

Example 6-13 indicates entries in the tapm.log related to both an IMMEDIATE request and a normal scheduled request. The IMMEDIATE request generated by `wmarsenddata` will never be placed into a DEFERRED state; only a scheduled request can be placed in a DEFERRED state.

**Example 6-13  Differences between wmarsenddata request and scheduled request**

```plaintext
senddata immediate request:

Mar 07 12:59:49   MarMlm::Mlm::addUploadRequest: Immediate request finished with rc = 1.
Mar 07 16:10:45   MarMlm::Mlm::addUploadRequest: Adding upload request from Endpoint 'ma_test' for file 'C:\PROGRA~1\Tivoli\lcf\dat\1\Mar\Dat\marinst200203070024.dat'
Mar 07 16:10:45   registerImmediateReq: Registering IMMEDIATE request to cache.

normal scheduled upload request:

Mar 07 12:59:49   MarMlm::Mlm::addUploadRequest: Immediate request finished with rc = 1.
Mar 07 16:10:45   MarMlm::Mlm::addUploadRequest: Adding upload request from Endpoint 'ma_test' for file 'C:\PROGRA~1\Tivoli\lcf\dat\1\Mar\Dat\marinst200203070024.dat'
Mar 07 16:10:45   registerImmediateReq: Registering IMMEDIATE request to cache.
```

176  Unveil Your e-business Transaction Performance with IBM TMTP 5.1
Chapter 6. Troubleshooting

After the mar_mlm has been recycled on the Gateway, deferred entries will be uploaded to TAPM RIM first, followed by the current day's upload request entries.

2. During the process of uploading data to TAPM RIM, an SQL retrieval precedes each update in order to determine if an SQL deletion command must be issued prior to the insertion of a row. This process is intended to prevent duplicate uploads of data since `wmarsenddata -i` does not remove any data after it uploads the data to the gateway. Example 6-14 shows the retrieval-deletion-insert sequence.

Example 6-14   Retrieval-deletion-insertion sequence for Instance Data Upload

<table>
<thead>
<tr>
<th>Time</th>
<th>Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sep 18 14:32:15</td>
<td>TapmDB_Check_InstanceData: q1: select INSTANCE_ID from TX_INSTANCE where HOST_ID = 186 AND STOP_TIME_S &gt; 1029124800 AND STOP_TIME_S &lt; 1029211199</td>
</tr>
<tr>
<td>Sep 18 14:32:20</td>
<td>TapmDB_Check_InstanceData: q1: 76 rows found...</td>
</tr>
<tr>
<td>Sep 18 14:32:20</td>
<td>TapmDB_Check_InstanceData: q1: in 1 groups.</td>
</tr>
<tr>
<td>Sep 18 14:32:20</td>
<td>TapmDB_Check_InstanceData: q20: delete from TX_FLOAT where INSTANCE_ID &gt;= 2117420 AND INSTANCE_ID &lt;= 2117495</td>
</tr>
<tr>
<td>Sep 18 14:32:20</td>
<td>TapmDB_Check_InstanceData: q20: delete from TX_INT where INSTANCE_ID &gt;= 2117420 AND INSTANCE_ID &lt;= 2117495</td>
</tr>
<tr>
<td>Sep 18 14:32:20</td>
<td>TapmDB_Check_InstanceData: q20: delete from TX_STR where INSTANCE_ID &gt;= 2117420 AND INSTANCE_ID &lt;= 2117495</td>
</tr>
<tr>
<td>Sep 18 14:32:20</td>
<td>TapmDB_Check_InstanceData: q20: delete from TX_INSTANCE where INSTANCE_ID &gt;= 2117420 AND INSTANCE_ID &lt;= 2117495</td>
</tr>
<tr>
<td>Sep 18 14:32:21</td>
<td>TapmDB_Load_InstanceData: Inserting '24' transaction...</td>
</tr>
<tr>
<td>Sep 18 14:32:21</td>
<td>TapmDB_Load_InstanceData: Inserting data '1' of '24' with 1 rows in TX_INSTANCE, TX_INT, TX_FLOAT, TX_STR, CORRELATOR tables...</td>
</tr>
</tbody>
</table>

3. The mar_mlm process on the gateway does not log any detailed entries for the Framework RIM connection process in the tapm.log file. The design team decided that logging such entries would cause the log file to be too large. So the general error code -58 was used in the tapm.log file to indicate any failures in the RIM connection process, as shown in Example 6-15 on page 177.

Example 6-15   RIM connection failure in tapm.log

<table>
<thead>
<tr>
<th>Time</th>
<th>Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nov 04 11:27:29</td>
<td>handle_log_msg:</td>
</tr>
<tr>
<td>Nov 04 11:27:29</td>
<td>*** EXCEPTION *** Unable to connect to the database.</td>
</tr>
</tbody>
</table>
Nov 04 11:27:29   TapmDB_Load_InstanceData:
*** EXCEPTION *** : FRWTE:0002
11/04/02 11:27:29 (2): operation 'RIM connection' failed
Nov 04 11:27:29   TapmDB_Load_InstanceData:
function return code = -58
Nov 04 11:27:29   sendToDB: *** ERROR *** :
Failure (rc = -58) on
loading INSTANCE data on the Database.

For troubleshooting the RIM connection failures, check the /tmp/rim_db.log file after following the instructions for increasing the log level for RIM. Also, check the results of the following commands:

```
OID='wlookup -ar RIM | grep tapm | awk '{print $2}''
idlcall $OID _get_max_conn
```

If the maximum connections are low, try increasing this value with the following command:

```
idlcall $OID _set_max_conn '"<##>"'
```

By default, IBM DB2 allows a maximum of 40 database connections at one time. Check with your DBA about listing or raising the maximum number of connections for DB2.

### 6.4.6 Analyzing Web Transaction Performance log and trace files

The following list presents the Web Transaction Performance log files and their descriptions. This is intended to make it easier to increase the loglevel and access the log files needed for troubleshooting Web Transaction Performance.

In this list, `installdir` denotes the installation directory under which the TIMS or endpoint directory is created.

**Note:** In some instances, it is necessary to extract a setup file from a JAR file in order to edit the file. These commands show how extract a setup file from a JAR file if it does not already exist:

```
$jar -tf /Program Files/Tivoli/Internet/Endpoint/lib/crawler.jar
$jar -xvf /ProgramFiles/Tivoli/Internet/Endpoint/lib/crawler.jar siteinvestigator.properties
```

These log files are relevant to troubleshooting Web Transaction Performance:

- **timsui.log**
  
<table>
<thead>
<tr>
<th>Location</th>
<th>&lt;installdir&gt;/TIMS/logs</th>
</tr>
</thead>
<tbody>
<tr>
<td>LogLevel</td>
<td>Tracing Level can be modified by editing the uilogging.properties file in the &lt;installdir&gt;/TIMS/lib/properties</td>
</tr>
</tbody>
</table>
directory.
(Logging on is enabled by default for error messaging only.
Options for severityLevel are warning, informational, and
detailed. Options for verboseLevel are user, support, and
development.

Used by Tivoli Internet Management Server

- timsfw.log
  Location <installdir>/TIMS/logs
  LogLevel Tracing Level can be modified by editing the event.properties
  file in the <installdir>/TIMS/lib/properties directory to set
  trace.switches=1-4.1 (default=1-4.0).
  (Logging on is enabled by default for error messaging only.
  Options for severityLevel are warning, informational, and
detailed. Options for verboseLevel are user, support, and
development.

  Used by Tivoli Internet Management Server

- pb.log
  Location <installdir>/Endpoint/logs
  LogLevel Can be modified in the <installdir>/lib/sti-playback.properties
  extracted from <installdir>/Endpoint/lib/properties.jar and
  setting trace.components=sti and trace = 3.
  Used by Endpoint to log STI playback job status information

- STI-error.log:
  Location <installdir>/Management Server/TIMS/STI
  LogLevel Not applicable
  Used by WSI endpoint

- endpoint.log:
  Location <installdir>/Endpoint/logs
  LogLevel Can be modified in endpoint.properties file located in
  <installdir>/Endpoint/configuration directory by setting
  trace.switches=1-9.1 (default=1-9.0). Extended tracing (cswa
  tracing) can be enabled in the cswa.properties file by
  extracting this file from the
  installdir>/Endpoint/cswaproperties.jar file and setting
  trace.switches to 1-13.1 (default=1-13.0). Task tracing can be
  enabled in the task.properties file by extracting
  task.properties from <installdir>/Endpoint/lib/properties.jar
  and setting trace.switches=1-4.1 (default=1-4.0). Parser
  tracing can be enabled by editing the parser.properties file by
  extracting this file from <installdir>/Endpoint/lib/properties.jar
  file and setting the trace.switches (default=). EAAConfig
tracing can be enabled by editing the EAAConfig.properties file and setting trace.switches=1-4.1 (default=1-4.0)

**Note:** The controller.log file is no longer available in Web Transaction Performance Version 5.1. The EAAConfig.properties file now controls logging for EAA Controller, and the trace data is written to Endpoint.log for Java-related processing and to proxy-error.txt for HTTP-related processing.

**Note:** LogLevel: Logging verbosity can also be modified in `<installdir>`/Endpoint/configuration/agentstatic.properties file by setting debugLevel=0,1,2,3 where 0= warnings, 1=informational messages, 2=detail messages in lpwin32.log and endpoint.log, 3=everything. There is also a trace.switches variable in agentstatic.properties that is set to 1-9.1 by default and overrides the debugLevel variable in the same file.

**Note:** Dynamic log level setting is now available in IBM Tivoli Monitoring for Transaction Performance: Web Transaction Performance 5.1, meaning that the endpoint no longer needs to be recycled to change the log level.

- **WSI-error.log**
  - Location: `<installdir>/Management Server/TIMS/WSI`
  - LogLevel: Can be modified in siteinvestigator.properties file by extracting siteinvestigator.properties from `<installdir>/Endpoint/lib/crawler.jar` and setting trace.switches=1-12.1 (default=1-12.0).
  - Used by: Site Investigator

- **QoSM-error.log**
  - Location: `<installdir>/Management Server/TIMS/QoSM`
  - LogLevel: Not applicable
  - Used by: Tivoli Internet Management Server

- **lpwin32.log**
  - Location: `<installdir>/Endpoint/logs`
  - Description: This log file lists the Java classes loaded by the Endpoint process. Refer to the procedure for launching Synthetic Transaction Investigator in visible mode using lpwin32.exe and compare to lphalt.exe command.
  - LogLevel: Can be modified in agentstatic.properties by setting the debug level parameter.
  - Used by: lpwin32.exe
Chapter 6. Troubleshooting

- **twsa_1.log**
  - **Location**: `<installdir>/Endpoint/logs`
  - **Description**: Contains task information on Qos (Task #1), WSI (Task #2), and STI (Task #3) historical tasks.
  - **LogLevel**: Can be modified by extracting cswalogg.properties file from cswaproperties.jar and setting trace.components to ERROR, WARN, or INFO.
  - **Used by**: Historical tasks in Web Transaction Performance 5.1

- **report.log**
  - **Location**: `<installdir>/Endpoint/logs`
  - **LogLevel**: Can be modified by extracting logging.properties file from report.jar and setting the path to the logging.properties file in the CLASSPATH.
  - **Used by**: Process uploading data from Endpoint to Tivoli Internet Management Server.

- **error.log**
  - **Location**: `<installdir>/Management Server/IBMHTTPSERVER/w32-ix86/logs`
  - **LogLevel**: The LogLevel directive in httpd.conf can set loglevel to one of emerg, alert, crit, error, warn, notice, info, or debug. Refer to [http://www.apache.org](http://www.apache.org)
  - **Used by**: IBM HTTP Server

- **access.log**
  - **Location**: `<installdir>/Management Server/IBMHTTPSERVER/w32-ix86/logs`
  - **Description**: Contains accesses that go through IBM HTTP Server.
  - **LogLevel**: Not applicable
  - **Used by**: IBM HTTP Server; QoS Endpoint

- **stip.log**
  - **Location**: `<installdir>/Management Server/TIMS/logs/STI`
  - **LogLevel**: Can be modified in stm.properties file extracted from stm.jar by setting LogLevel to 1,2,3,4,5.
  - **Used by**: QoS Endpoint

- **tec-forwarding log**
  - **Location**: `<installdir>/Endpoint/logs`
  - **LogLevel**: Can be modified in tecforward.properties file extracted from properties.jar by setting trace.switches=1-5.1 (default=1-5.1).
  - **Used by**: TecForwarder
Patterns for e-business

IBM Patterns for e-business is a set of proven architectures that have been compiled from more than 20,000 successful Internet-based engagements. This repository of assets can be used by companies to facilitate the development of Web-based applications. They help an organization understand and analyze complex business problems and break them down into smaller, more manageable functions that can then be implemented using low-level design patterns.
Introduction to Patterns for e-business

As companies compete in the e-business marketplace, they find that they must re-evaluate their business processes and applications so that their technology is not limited by time, space, organizational boundaries, or territorial borders. They must consider the time it takes to implement the solution, as well as the resources (people, money, and time) they have at their disposal to successfully execute the solution. These challenges, coupled with the integration issues of existing legacy systems and the pressure to deliver consistent high-quality service, present a significant undertaking when developing an e-business solution.

In an effort to alleviate the tasks involved in defining an e-business solution, IBM has built a repository of patterns to simplify the effort. In simple terms, a pattern can be defined as a model or plan used as a guide in making things. As such, patterns serve to facilitate the development and production of things. Patterns codify the repeatable experience and knowledge of people who have performed similar tasks before. Patterns not only document solutions to common problems, but also point out pitfalls that should be avoided. IBM Patterns for e-business consists of documented architectural best practices. They define a comprehensive framework of guidelines and techniques that were actually used in creating architectures for customer engagements. The Patterns for e-business bridge the business and IT gap by defining architectural patterns at various levels from Business patterns to Application patterns to Runtime patterns, enabling easy navigation from one level to the next. Each of the patterns (Business, Integration, Application, and Runtime) help companies understand the true scope of their development project and provide the necessary tools to facilitate the application development process, thereby allowing companies to shorten time to market, reduce risk, and most important, realize a more significant return on investment.

The core types of Patterns for e-business are:

- Business Patterns
- Integration Patterns
- Composite Patterns
- Application Patterns
- Runtime Patterns and matching product mappings

When a company takes advantage of these documented assets, they are able to reduce the time and risk involved in completing a project.

For example, a line-of-business (LOB) executive who understands the business aspects and requirements of a solution can use Business patterns to develop a high-level structure for a solution. Business patterns represent common business problems. LOB executives can match their requirements (IT and business...
drivers) to Business patterns that have already been documented. The patterns provide tangible solutions to the most frequently encountered business challenges by identifying common interactions among users, business, and data.

Senior technical executives can use Application patterns to make critical decisions related to the structure and architecture of the proposed solution. Application patterns help refine Business patterns so that they can be implemented as computer-based solutions. Technical executives can use these patterns to identify and describe the high-level logical components that are needed to implement the key functions identified in a Business pattern. Each Application pattern would describe the structure (tiers of the application), placement of the data, and the integration (loosely or tightly coupled) of the systems involved.

Finally, solution architects and systems designers can develop a technical architecture by using Runtime patterns to realize the Application patterns. Runtime patterns describe the logical architecture that is required to implement an Application pattern. Solution architects can match Runtime patterns to existing environment and business needs. The Runtime pattern they implement establishes the components needed to support the chosen Application pattern. It defines the logical middleware nodes, their roles, and the interfaces among these nodes in order to meet business requirements. The Runtime pattern documents what must be in place to complete the application, but does not specify product brands. Determination of actual products is made in the product mapping phase of the patterns.

In summary, Patterns for e-business captures e-business approaches that have been tested and proven. By making these approaches available and classifying them into useful categories, LOB executives, planners, architects, and developers can further refine them into useful, tangible guidelines. The patterns and their associated guidelines enable the individual to start with a problem and a vision, find a conceptual pattern that fits this vision, define the necessary functional pieces that the application will need to succeed, and then actually build the application. Furthermore, the Patterns for e-business provides common terminology from a project's onset and ensures that the application supports business objectives, significantly reducing cost and risk.

The Patterns for e-business layered asset model

The Patterns for e-business approach enables architects to implement successful e-business solutions through the re-use of components and solution elements from proven, successful experiences. The Patterns approach is based on a set of layered assets that can be exploited by any existing development
methodology. These layered assets are structured so that each level of detail builds on the last. These assets include:

- Business patterns that identify the interaction between users, businesses, and data.
- Integration patterns that tie multiple Business patterns together when a solution cannot be provided based on a single Business pattern.
- Composite patterns that represent commonly occurring combinations of Business patterns and Integration patterns.
- Application patterns that provide a conceptual layout describing how the application components and data within a Business pattern or Integration pattern interact.
- Runtime patterns that define the logical middleware structure supporting an Application pattern. Runtime patterns depict the major middleware nodes, their roles, and the interfaces between these nodes.
- Product mappings that identify proven and tested software implementations for each Runtime pattern.
- Best-practice guidelines for design, development, deployment, and management of e-business applications.

These assets and their relation to each other are shown in Figure A-1.
Patterns for e-business Web site

The Patterns Web site provides an easy way of navigating top-down through the layered Patterns’ assets in order to determine the preferred reusable assets for an engagement. For easy reference to Patterns for e-business, refer to the Patterns for e-business Web site at:


How to use the Patterns for e-business

As described in the previous section, the Patterns for e-business are structured so that each level of detail builds on the last. At the highest level are Business patterns that describe the entities involved in the e-business solution. A Business pattern describes the relationship among the users, the business organization or applications, and the data to be accessed.

Composite patterns appear in the hierarchy above the Business patterns. However, Composite patterns are made up of a number of individual Business patterns and at least one Integration pattern. In this section, we discuss how to use the layered structure of the Patterns for e-business assets.

There are four primary Business patterns, as shown in Table A-1.

<table>
<thead>
<tr>
<th>Business patterns</th>
<th>Description</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Self-Service (user-to-business)</td>
<td>Applications where users interact with a business via the Internet</td>
<td>Simple Web site applications</td>
</tr>
<tr>
<td>Information Aggregation (user-to-data)</td>
<td>Applications where users can extract useful information from large volumes of data, text, images, etc.</td>
<td>Business intelligence, knowledge management, Web crawlers</td>
</tr>
<tr>
<td>Collaboration (user-to-user)</td>
<td>Applications where the Internet supports collaborative work between users</td>
<td>E-mail, community, chat, video conferencing, etc.</td>
</tr>
<tr>
<td>Extended Enterprise (business-to-business)</td>
<td>Applications that link two or more business processes across separate enterprises</td>
<td>EDI, supply chain management, etc.</td>
</tr>
</tbody>
</table>
It would be very convenient if all problems fit nicely into these four Business patterns, but in reality things can be more complicated. The patterns assume that all problems, when broken down into their most basic components, will fit more than one of these patterns. When a problem describes multiple objectives that fit into multiple Business patterns, the Patterns for e-business provide the solution in the form of Integration patterns.

Integration patterns enable us to tie together multiple Business patterns to solve a problem. The Integration patterns are shown in Table A-2.

Table A-2  Integration patterns

<table>
<thead>
<tr>
<th>Integration patterns</th>
<th>Description</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Access Integration</td>
<td>Integration of a number of services through a common entry point</td>
<td>Portsals</td>
</tr>
<tr>
<td>Application Integration</td>
<td>Integration of multiple applications and data sources without the user directly invoking them</td>
<td>Message brokers, workflow managers</td>
</tr>
</tbody>
</table>

These Business and Integration patterns can be combined to implement installation-specific business solutions. We call this a Custom design.

We can represent the use of a Custom design to address a business problem through an iconic representation, as shown in Figure A-2.

![Figure A-2  Pattern representation of a Custom design](image)

If any of the Business or Integration patterns are not used in a Custom design, we can show that with lighter blocks. For example, Figure A-3 on page 189 shows a Custom design that does not have a mandatory Collaboration business pattern or an Extended Enterprise business pattern for a business problem.
A Custom design may also be a Composite pattern if it recurs many times across domains with similar business problems. For example, the iconic view of a Custom design in Figure A-3 can also describe a Sell-Side Hub composite pattern.

Several common uses of Business and Integration patterns have been identified and formalized into Composite patterns, which are shown in Table A-3.

**Table A-3  Composite patterns**

<table>
<thead>
<tr>
<th>Composite patterns</th>
<th>Description</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Electronic Commerce</td>
<td>User-to-Online-Buying.</td>
<td>• <a href="http://www.macys.com">www.macys.com</a></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• <a href="http://www.amazon.com">www.amazon.com</a></td>
</tr>
<tr>
<td>Portal</td>
<td>Typically designed to aggregate multiple information sources and applications to provide uniform, seamless, and personalized access for its users.</td>
<td>• Enterprised intranet portal providing self-service functions, such as payroll, benefits, and travel expenses</td>
</tr>
<tr>
<td>Account Access</td>
<td>Provide customers with around-the-clock account access to their account information.</td>
<td>• Online brokerage trading apps</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Telephone company account manager functions</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Bank, credit card, and insurance company online apps</td>
</tr>
</tbody>
</table>
### Composite patterns

<table>
<thead>
<tr>
<th>Description</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Allows buyers and sellers to trade goods and services on a public site.</td>
<td>• Buyer's side - interaction between buyer's procurement system and commerce functions of e-Marketplace&lt;br&gt;• Seller's side - interaction between the procurement functions of the e-Marketplace and its suppliers</td>
</tr>
<tr>
<td>The seller owns the e-Marketplace and uses it as a vehicle to sell goods and services on the Web.</td>
<td>• <a href="http://www.carmax.com">www.carmax.com</a> (car purchase)</td>
</tr>
<tr>
<td>The buyer of the goods owns the e-Marketplace and uses it as a vehicle to leverage the buying or procurement budget in soliciting the best deals for goods and services from prospective sellers across the Web.</td>
<td><a href="http://www.wre.org">www.wre.org</a> (WorldWide Retail Exchange)</td>
</tr>
</tbody>
</table>

The makeup of these patterns is variable in that there will be basic patterns present for each type, but the Composite can easily be extended to meet additional criteria. For more information about Composite patterns, refer to *Patterns for e-business: A Strategy for Reuse* by Jonathan Adams, et al., ISBN 1931182027.

### Selecting Patterns and product mapping

After the appropriate Business pattern is identified, the next step is to define the high-level logical components that make up the solution and how these components interact. This is known as the Application pattern. A Business pattern will usually have multiple Application patterns identified that describe the possible logical components and their interactions. For example, an Application pattern may have logical components that describe a presentation tier for interacting with users, a Web application tier, and a backend application tier.

The Application pattern requires an underpinning of middleware that is expressed as one or more Runtime patterns. Runtime patterns define functional nodes that represent middleware functions that must be performed.

After a Runtime pattern has been identified, the next logical step is to determine the actual product and platform to use for each node. Patterns for e-business
have product mappings that correlate to the Runtime patterns, describing actual products that have been used to build an e-business solution for this situation.

Finally, guidelines assist you in creating the application using best practices that have been identified through experience.

For more information on determining how to select each of the layered assets, refer to the Patterns for e-business Web site at:

Unveil Your e-business Transaction Performance with IBM TMTP 5.1
## Abbreviations and acronyms

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACF</td>
<td>Adapter Configuration Facility</td>
</tr>
<tr>
<td>AIX</td>
<td>Advanced Interactive Executive</td>
</tr>
<tr>
<td>AMI</td>
<td>Application Management Interface</td>
</tr>
<tr>
<td>AMS</td>
<td>Application Management Specifications</td>
</tr>
<tr>
<td>API</td>
<td>Application Programming Interface</td>
</tr>
<tr>
<td>APM</td>
<td>Application Performance Management</td>
</tr>
<tr>
<td>ARM</td>
<td>Application Response Measurement</td>
</tr>
<tr>
<td>ASP</td>
<td>Active Server Pages</td>
</tr>
<tr>
<td>BAROC</td>
<td>Basic Recorder of Objects in C</td>
</tr>
<tr>
<td>BDT</td>
<td>Bulk Data Transfer</td>
</tr>
<tr>
<td>BOC</td>
<td>Business Objects Container</td>
</tr>
<tr>
<td>CA</td>
<td>Certificate Authority</td>
</tr>
<tr>
<td>CGI</td>
<td>Common Gateway Interface</td>
</tr>
<tr>
<td>CICS</td>
<td>Customer Information Control System</td>
</tr>
<tr>
<td>CIM</td>
<td>Common Management Information</td>
</tr>
<tr>
<td>CLI</td>
<td>Command Line Interface</td>
</tr>
<tr>
<td>CMP</td>
<td>Container-Managed Persistence</td>
</tr>
<tr>
<td>CMS</td>
<td>Cryptographic Message Syntax</td>
</tr>
<tr>
<td>CPU</td>
<td>Central Processing Unit</td>
</tr>
<tr>
<td>CTS</td>
<td>Compatibility Testing Standard</td>
</tr>
<tr>
<td>DB2</td>
<td>Database 2</td>
</tr>
<tr>
<td>DBCS</td>
<td>Double-byte Character Set</td>
</tr>
<tr>
<td>DES</td>
<td>Data Encryption Standard</td>
</tr>
<tr>
<td>DLL</td>
<td>Dynamic Link Library</td>
</tr>
<tr>
<td>DM</td>
<td>Tivoli Distributed Monitoring</td>
</tr>
<tr>
<td>DMTF</td>
<td>Distributed Management Task Force</td>
</tr>
<tr>
<td>DNS</td>
<td>Domain Name Service</td>
</tr>
<tr>
<td>DOM</td>
<td>Document Object Model</td>
</tr>
<tr>
<td>DSN</td>
<td>Data Source Name</td>
</tr>
<tr>
<td>DTD</td>
<td>Document Type Definition</td>
</tr>
<tr>
<td>EAA</td>
<td>Ephemeral Availability Agent (now QoS)</td>
</tr>
<tr>
<td>EJB</td>
<td>Enterprise JavaBeans</td>
</tr>
<tr>
<td>EPP</td>
<td>End-to-End Probe platform</td>
</tr>
<tr>
<td>ERP</td>
<td>Enterprise Resource Planning</td>
</tr>
<tr>
<td>ETP</td>
<td>Enterprise Transaction Performance</td>
</tr>
<tr>
<td>GEM</td>
<td>Global Enterprise Manager</td>
</tr>
<tr>
<td>GMT</td>
<td>Greenwich Mean Time</td>
</tr>
<tr>
<td>GSK</td>
<td>Global Security Kit</td>
</tr>
<tr>
<td>GUI</td>
<td>Graphical User Interface</td>
</tr>
<tr>
<td>HTML</td>
<td>Hypertext Markup Language</td>
</tr>
<tr>
<td>HTTP</td>
<td>Hypertext Transfer Protocol</td>
</tr>
<tr>
<td>HTTPS</td>
<td>HTTP Secure</td>
</tr>
<tr>
<td>IBM</td>
<td>International Business Machines Corporation</td>
</tr>
<tr>
<td>IDEA</td>
<td>International Data Encryption Algorithm</td>
</tr>
<tr>
<td>IE</td>
<td>Microsoft Internet Explorer</td>
</tr>
<tr>
<td>IIOP</td>
<td>Internet Inter ORB Protocol</td>
</tr>
<tr>
<td>IIS</td>
<td>Internet Information Server</td>
</tr>
<tr>
<td>IMAP</td>
<td>Internet Message Access Protocol</td>
</tr>
<tr>
<td>IOM</td>
<td>Inter Object Messaging</td>
</tr>
<tr>
<td>ISAPI</td>
<td>Internet Server API</td>
</tr>
<tr>
<td>Acronym</td>
<td>Definition</td>
</tr>
<tr>
<td>---------</td>
<td>------------</td>
</tr>
<tr>
<td>ITM</td>
<td>IBM Tivoli Monitoring</td>
</tr>
<tr>
<td>ITMTP</td>
<td>IBM Tivoli Monitor for Transaction Performance</td>
</tr>
<tr>
<td>ITSO</td>
<td>International Technical Support Organization</td>
</tr>
<tr>
<td>JCP</td>
<td>Java Community Process</td>
</tr>
<tr>
<td>JDBC</td>
<td>Java Database Connectivity</td>
</tr>
<tr>
<td>JNI</td>
<td>Java Native Interface</td>
</tr>
<tr>
<td>JRE</td>
<td>Java Runtime Environment</td>
</tr>
<tr>
<td>JSP</td>
<td>Java Server Page</td>
</tr>
<tr>
<td>JVM</td>
<td>Java Virtual Machine</td>
</tr>
<tr>
<td>LAN</td>
<td>Local Area Network</td>
</tr>
<tr>
<td>LOB</td>
<td>Line of Business</td>
</tr>
<tr>
<td>LR</td>
<td>LoadRunner</td>
</tr>
<tr>
<td>MBean</td>
<td>Management Bean</td>
</tr>
<tr>
<td>MD5</td>
<td>Message Digest 5</td>
</tr>
<tr>
<td>MIME</td>
<td>Multi-purpose Internet Mail Extensions</td>
</tr>
<tr>
<td>MLM</td>
<td>Mid-Level Manager</td>
</tr>
<tr>
<td>ODBC</td>
<td>Open Database Connectivity</td>
</tr>
<tr>
<td>OID</td>
<td>Object Identifier</td>
</tr>
<tr>
<td>OLAP</td>
<td>On-line Analytical Processing</td>
</tr>
<tr>
<td>OMG</td>
<td>Object Management Group</td>
</tr>
<tr>
<td>OOP</td>
<td>Object Oriented Programming</td>
</tr>
<tr>
<td>ORB</td>
<td>Object Request Broker</td>
</tr>
<tr>
<td>OS</td>
<td>Operating Systems</td>
</tr>
<tr>
<td>OSI</td>
<td>Open Systems Interconnection</td>
</tr>
<tr>
<td>PKCS10</td>
<td>Public Key Cryptography Standard #10</td>
</tr>
<tr>
<td>QoS</td>
<td>Quality of Service</td>
</tr>
<tr>
<td>RDBMS</td>
<td>Relational Database Management System</td>
</tr>
<tr>
<td>RIM</td>
<td>RDBMS Interface Module</td>
</tr>
<tr>
<td>RIPEMD</td>
<td>RACE Integrity Primitives Evaluation Message Digest</td>
</tr>
<tr>
<td>RTE</td>
<td>Remote Terminal Emulation</td>
</tr>
<tr>
<td>SAX</td>
<td>Simple API for XML</td>
</tr>
<tr>
<td>SDK</td>
<td>Software Developer’s Kit</td>
</tr>
<tr>
<td>SHA</td>
<td>Secure Hash Algorithm</td>
</tr>
<tr>
<td>SI</td>
<td>Site Investigator</td>
</tr>
<tr>
<td>SID</td>
<td>System ID</td>
</tr>
<tr>
<td>SLA</td>
<td>Service Level Agreement</td>
</tr>
<tr>
<td>SLO</td>
<td>Service-level Objective</td>
</tr>
<tr>
<td>SMTP</td>
<td>Simple Mail Transfer Protocol</td>
</tr>
<tr>
<td>SNMP</td>
<td>Simple Network Management Protocol</td>
</tr>
<tr>
<td>SOAP</td>
<td>Simple Object Access Protocol</td>
</tr>
<tr>
<td>SQL</td>
<td>Structured Query Language</td>
</tr>
<tr>
<td>SSL</td>
<td>Secure Socket Layer</td>
</tr>
<tr>
<td>STI</td>
<td>Synthetic Transaction Investigator</td>
</tr>
<tr>
<td>TAPM</td>
<td>Tivoli Application Performance Management</td>
</tr>
<tr>
<td>TBSM</td>
<td>Tivoli Business Systems Manager</td>
</tr>
<tr>
<td>TCL</td>
<td>Terminal Control Language</td>
</tr>
<tr>
<td>TCP/IP</td>
<td>Transmission Control Protocol/Internet Protocol</td>
</tr>
<tr>
<td>TDS</td>
<td>Tivoli Decision Support</td>
</tr>
<tr>
<td>TEC</td>
<td>Tivoli Enterprise Console</td>
</tr>
<tr>
<td>TEDW</td>
<td>Tivoli Enterprise Data Warehouse</td>
</tr>
<tr>
<td>TIMS</td>
<td>Tivoli Internet Management Server</td>
</tr>
<tr>
<td>TMA</td>
<td>Tivoli Management Agent</td>
</tr>
<tr>
<td>TME</td>
<td>Tivoli Management Environment</td>
</tr>
<tr>
<td>TMR</td>
<td>Tivoli Management Region</td>
</tr>
<tr>
<td>TS</td>
<td>Transaction Simulation</td>
</tr>
<tr>
<td>TMTP</td>
<td>IBM Tivoli Monitor for Transaction Performance</td>
</tr>
<tr>
<td>UDB</td>
<td>Universal Database</td>
</tr>
<tr>
<td>UDP</td>
<td>User Datagram Protocol</td>
</tr>
<tr>
<td>Abbreviation</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>-----------------------------------</td>
</tr>
<tr>
<td>URI</td>
<td>Uniform Resource Identifier</td>
</tr>
<tr>
<td>URL</td>
<td>Uniform Resource Locator</td>
</tr>
<tr>
<td>UUID</td>
<td>Universal Unique Identifier</td>
</tr>
<tr>
<td>VuGen</td>
<td>Virtual User Generator</td>
</tr>
<tr>
<td>VUS</td>
<td>Virtual User Script</td>
</tr>
<tr>
<td>Vuser</td>
<td>Virtual User</td>
</tr>
<tr>
<td>W3C</td>
<td>World Wide Web Consortium</td>
</tr>
<tr>
<td>WAS</td>
<td>WebSphere Application Server</td>
</tr>
<tr>
<td>WSC</td>
<td>Web Services Courier</td>
</tr>
<tr>
<td>WSI</td>
<td>Web Site Investigator</td>
</tr>
<tr>
<td>WTP</td>
<td>Web Transaction Performance</td>
</tr>
<tr>
<td>WWW</td>
<td>World Wide Web</td>
</tr>
<tr>
<td>XML</td>
<td>eXtensible Markup Language</td>
</tr>
</tbody>
</table>
Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this book.

IBM Redbooks

For information about ordering these publications, see "How to get IBM Redbooks" on page 199.

- *Introducing Tivoli Application Performance Management*, SG24-5508
- *Tivoli Application Performance Management Version 2.0 and Beyond*, SG24-6048
- *Tivoli Web Solutions: Managing Web Services and Beyond*, SG24-6049
- *Using Databases with Tivoli Applications and RIM*, SG24-5112
- *Introducing Tivoli Service Level Advisor*, SG24-6611
- *Tivoli Business Systems Manager: An Implementation Case Study*, SG24-6032
- *Tivoli Enterprise Internals and Problem Determination*, SG24-2034
- *Using Tivoli Decision Support Guides*, SG24-5506
- *Deploying a Public Key Infrastructure*, SG24-5512
- *IBM HTTP Server powered by Apache on RS/6000*, SG24-5132
- *Integrated Management Solutions Using NetView Version 5.1*, SG24-5285
- *NetView 6.01 and Friends*, SG24-6019
- *Servlet and JSP Programming with IBM WebSphere Studio and VisualAge for Java*, SG24-5755
- *Tivoli Web Services Manager: Internet Management Made Easy*, SG24-6017
- *IBM Tivoli Monitoring Version 5.1: Advanced Resource Monitoring*, SG24-5519-02
Other resources

These publications are also relevant as further information sources:

- Tivoli Management Framework Installation Guide Version 3.7.1, GC32-0395
- Tivoli Management Framework Reference Manual Version 3.7.1, SC31-8434
- Tivoli Decision Support Administrator Guide Version 2.1.1, GC32-0437
- Tivoli Decision Support Installation Guide Version 2.1.1, GC32-0438
- Tivoli Decision Support for TAPM Release Notes Version 1.1, GI10-9259
- Tivoli Management Framework Installation Guide Version 3.7.1, GC32-0395

The following publications come with their respective products and cannot be obtained separately:

- NetView for NT Programmer's Guide Version 7, SC31-8889
- NetView for NT User's Guide Version 7, SC31-8888
Referenced Web sites

These Web sites are also relevant as further information sources:

- Apache Web site
  http://www.apache.org/
- Computer Measurement Group Web site
  http://www.cmg.org/
- Java Web site for JNI documents
  http://java.sun.com/products/jdk/1.2/docs/guide/jni/
- OpenGroup ARM Web site
  http://www.opengroup.org/management/arm.htm
- IBM support FTP site

How to get IBM Redbooks

You can order hardcopy Redbooks, as well as view, download, or search for Redbooks at the following Web site:

  ibm.com/redbooks

You can also download additional materials (code samples or diskette/CD-ROM images) from that site.

IBM Redbooks collections

Redbooks are also available on CD-ROM. Click the CD-ROMs button on the Redbooks Web site for information about all CD-ROMs offered, as well as updates and formats.
Unveil Your e-business Transaction Performance with IBM TMTP 5.1
Index

Symbols
/tmp 157

Numerics
3270 27
3270 emulation session 109
3270 simulation
  debugging 115
3270 simulations 115
3270 terminal 111
3270 Transaction Family 110
3270 transactions 30
  performance measurement 106
  recording 109

A
access.log 181
action
  corrective 9
admin.conf 53, 56
administrative privileges 150
agents 22
agentstatic.properties 180
aggregate page size 68
aggregated 28
aggregated data 33
aggregation
  data 118
aggregation period 34
AggregationEndTime 164
AggregationRandTimeRange 164
AggregationStartime 164
AggregationTimeStep 164
alerts 33
aliases
  SQL 75
analysis 120
  OLAP 121
  root cause 77
animations 4
API calls
  ARM 70
application 100
  registration 162
tiers 17
application design 26
application development 4
application name 160–161
application patterns 190
application resources 22
application system 9
Application transaction 6
applications
  source 120
applications.dat 161
architecture 24
ARM 27, 67, 112
  Enable 90
ARM API 90
ARM API calls 70
ARM call 112
ARM calls 99
ARM classes
  Tivoli 92
ARM correlation 70, 102
ARM correlator 73–74, 76–78
ARM data 74
ARM data retrieval
  TIMS 49
ARM engine 107
ARM implementation 161
ARM instrumentation 70, 111
ARM Instrumented 161
ARM Interface 107
ARM Report
  hyperlink 70, 103
ARM report 74
ARM tracing 84
arm_init 70
arm_start 70, 73
arm_stop 70
arm_update 70
ARMing 71
ARMjava.doc 71
armjni.dll 90, 161
armJni.jar 90

© Copyright IBM Corp. 2003. All rights reserved.
Unveil Your e-business Transaction Performance with IBM TMTP 5.1

armjni.jar 161
availability 4, 32, 68–69
  Web site 29
Availability Management 14
average response time 103

B
backend application tier 190
backend service time 67, 70, 84
baseline values 71
BI See Business Intelligence
bottlenecks 3, 68, 85
breakdown 4, 85
  transaction 29, 36, 69, 77
breakpoints 116
Brio Technology 118
broken links 36
browser 20, 32
buffer pool 59
Business Intelligence 118
business intelligence reporting 121
Business Objects 118
business system 25
business transaction 3

C
capacity 3
Capacity Management 14
CDW See central data warehouse
central console 9
Central Data Warehouse 120
central data warehouse ETL 120
centralized monitoring 11
Change Management 15
child correlator 74, 76
child transactions 70
class
  transaction 76
Client Capture 32, 71
ClientCapture 170
client-server 8
code
  VB script 111
Cognos 118
collecting data 160
collections 107
COM client 107
COM object 107
combined installation 152
common dimensions 118
Common Warehouse Metadata standard 118
communication protocols 20
components
  service 12
    transaction 111
ConfigArgs 51
configuration
  parameters 162
Configuration Management 14
configuring
  TAPM RIM 87
connection
  RIM 178
console 38, 78
  central 9
content
  unwanted 36
Contingency Planning 14
cost heap size 126
controlled measurements 29
cr_db.db2.sql 154
cr_apm.db.sh 154
crawling 36
Crystal Decisions 118
css 69
cswa tracing 179
cswaloggинг.properties 181
current data 33
customize 87
CWM See Common Warehouse Metadata standard
data

collection 160
correlating 118
extract 120
historical 118, 120–121
instance 76
management 120
measurement 119
performance 70, 73
reference 27
data aggregation 118
data analysis 121
data gathering 28, 123
data mart 119, 121
format 121
data mart ETL 120–121
data mining 121
data source 120
Data Storage 100
data target 120
data upload 162
data warehouse 120, 160
database 153, 158
alias 58
corruptions 173
name 58
TIMS 47, 58
upload to 164, 166, 176
user ID 62
database configuration
TIMS 48
DB2 instance 58
DB2 instance owner 154
DB2 user ID 154
db2cli.ini 154
DB2INSTANCE 154
dbCacheList.log 167
DBMaxInstanceRowNo 76, 165
DBUploadStartTime 164, 166–167
DBUploadStopTime 164, 166–167
debugging 115
decision making 4
default
tablespace 60
DEFERRED requests 168, 176
delivery connect 69
development
application 4
development status 146
diagnostic tracing 161
diagnostics 84
dimensions
common 118
distributing
MarProfile 162
DNS 69
duplication 10
dynamic data table 123

E
EAAConfig.properties 180
ease-of-use 26
e-business patterns 18
Enable ARM 90
enable parameter 91
enableARM parameter 91
endpoint 77, 87
endpoint database 122
endpoint.log 179
endpoint.properties 179
dto-end-end 77
view 118
gine 107
Enterprise transaction 6, 27
error
IBM HTTP Server installation 151
Internal Server 152
Java 158
messages 68
running two Tomcat servers 152
error.log 181
ETL
central data warehouse 120
data mart 120–121
process 121, 134, 146
programs 120
source 120
target 120
ETL process 160
event generation 35
event notifications 25
event.properties 179
events 36
extract
from jar file 178
extract data 120
JavaScript 69
job
  STI 35
    STI transaction playback 72
JPEG 69

K
Key Management Utility 53
keyboard layout 109

L
launchpad 89
layered assets 191
legacy systems 7, 19
load balancing 18
Local Logging 100
lockinfo.log 166
logging.properties 181
loglevel 178
logon
  Web Console 82
lpwin32.log 180

M
mail servers 32
Managed Node 37, 44, 87, 152
management
data 120
gen general 11
  service level 4
specialized 11
management needs 7, 9
managing 11
mar_ctrl 170
mar_mlm 169–170, 177
MarController 167, 169
restart 170
MarProfile 33, 98, 100, 114, 154, 162, 169
distribution 162
marshalling 168
mdist2 155
measurement data 119
measurements 28
controlled 29
metadata interchange 118
metrics 35
  response time 67
middleware 65
migration 151
timeout 160
mission-critical 11
MLM 158
MLM process 169
MLM restart 170
modules
  warehouse 120
monitoring 11
centralized 11
real-time 29
monitoring collection 33, 37
motherboard 107

N
needs
  management 7, 9
Norton AntiVirus 150
Notes Servers 32
ntprocinfo 170

O
Object Management Group 118
OLAP
  analysis 121

P
package
  simulation 106
page analyzer viewer 69
page render time 67
page size
  aggregate 68
paging space 152
parameters
  configuration 162
  runtime 114
parent correlator 74
parent transaction 70, 73, 76–77, 112
Password 150
patterns 18
Patterns for e-business 183
pb.log 179
pbs file 114–115
performance 69
performance data 70, 73
role
  TMR 150
root cause 65
root cause analysis 71, 77
round trip time 67
runtime parameters 114
Runtime patterns 190

S
SAP 27, 108
SAP transaction 30, 106
security
  TEDW 119
self-service 2
sensor 107
separation 18
server profile 152
server response 69
server.properties 49
server.xml 57, 95, 97, 161
service 150
  components 12
  IBM HTTP Server 41, 46, 53
Service Delivery 13
service level 66, 84
Service Level Management 4, 13
service names 39, 50
services 10–11, 39
  specialized 9
servlets 69
session
  playback 107
simulation
  transaction 29
simulation package 106
Site Investigator 36
siteinvestigator.properties 180
slowest transaction 69
SOCKS connect 69
Software Control and Distribution 15
solution 10
source applications 120
source ETL 120
specialized management 11
specialized services 9
SQL 75
  aliases 75
SQL commands 77
SSL communication 40
SSL connect 69
staging area table 123
standardization 8
start date 74
startupTraceSpecification 95
statistics
  development 146
  production 146
STI 66, 68
  job 35, 68
  STI playback 35, 67, 78
  STI recorder 35, 72
  STI report viewer 79
  STI statistics report 79
  STI transaction playback job 72
  STI-error.log 179
  stip.log 181
  sti-playback.properties 179
  stm.properties 181
subtransaction 73, 77, 112
surveillance 11, 33
Synthetic Transaction Investigator 35
system
  recording 106
  systems management
    infrastructure 22

T
table
  dynamic data 123
  staging area 123
  TAPM_VERSION 157
tables 154
tablespace 154
default 60
temporary 61
TAPM bridge 171
TAPM database 158, 172
TAPM dlls 171
TAPM RIM 44, 76, 151, 154, 177
  configuring 87
tapm.log 166, 175–176
tapm_cc.log 175
tapm_engine.log 174
tapm_ep.log 174
tapm_epp_ctrl.log 175
U
uilogging.properties 178
uninstalling 152
UninstallString 52
unmarshalling 168
Unregister 171
unwanted content 36
upgrade 151
upload
data 162
forced 74
to database 164, 166, 176
upload time 166–167
uploading
instance data 76
usefulness 26
user experience time 68
user ID 150
database 62
DB2 154
Web Console default 151

V
values
baseline 71
VB script 107
code 111
verification 113
violations 68
vpd.properties 159
VuGen 32

W
warehouse logger 159
warehouse modules 120
Watch function 116
wctrim 87
wdistrib 154
wdmdistrib 155–156
Web
transaction 27
Web application tier 190
Web Console 39, 44, 78, 87, 103, 150
default user ID 151
logon 82
port 152
Web Detailer 69
Web Health Console 31, 33

Web pages 67
Web Services Courier 35–36
Web site
availability 29
Web transaction 27
Web transactions 6
wep command 169
white spaces 89
wlookup 166
wmarcleareng 159
wmargetapp 99
wmargetdata 159
wmargetrawdata 159
wmargetrawdatadb 83
wmargetrawdbdata 77
wmargetstatus 159, 164–165
wmarlssapp 99, 159, 161
wmarlsseng 101, 159
wmarregapp 99, 115, 161
wmarestartmlm.sh 167
wmarsenddata 74, 103, 167, 176–177
wmarsetstatus 159, 164
wmartartcoll 159
wmarsarteng 159
wmarsstopcoll 159
wmarsstopeng 159, 171
workbench 106
workers.properties 57
workstation profile 109
wrapper.properties 57
wrimtest 170
wrimtrace 176
ws file 110
WSI-error.log 180
wuninst 152

X
xml file 107–108
Unveil Your e-business Transaction Performance with IBM TMTP 5.1

Transaction correlation and breakdown for root cause analysis

Measure, monitor, and report real-time Web, 3270, and SAP transaction times

Avoid SLA breaches

IBM Tivoli Monitoring for Transaction Performance Version 5.1 is part of the Tivoli family of products for managing performance and availability. It combines the functionality of two former Tivoli products, Tivoli Application Performance Management Version 2.1 and Tivoli Internet Services Manager Version 1.7, into one package for transaction performance monitoring. IBM Tivoli Monitoring for Transaction Performance Version 5.1 adds new capabilities for breakdown of e-business transactions that are partly or fully hosted by WebSphere Application Server Version 5.0 servers, as well as functions for simulating transactions run on 3270 and SAP platforms.

In addition to discussing the value and benefits of measuring and monitoring transaction performance, this IBM Redbook discusses the best use of the new capabilities provided by IBM Tivoli Monitoring for Transaction Performance Version 5.1, and it includes valuable hints and tips for troubleshooting the IBM Tivoli Monitoring for Transaction Performance environment.

The information in this book will help Service Level Managers, Network specialists, Web administrators, application owners, and programmers perform their tasks by enabling them to measure, monitor and report end-to-end transaction

For more information: ibm.com/redbooks