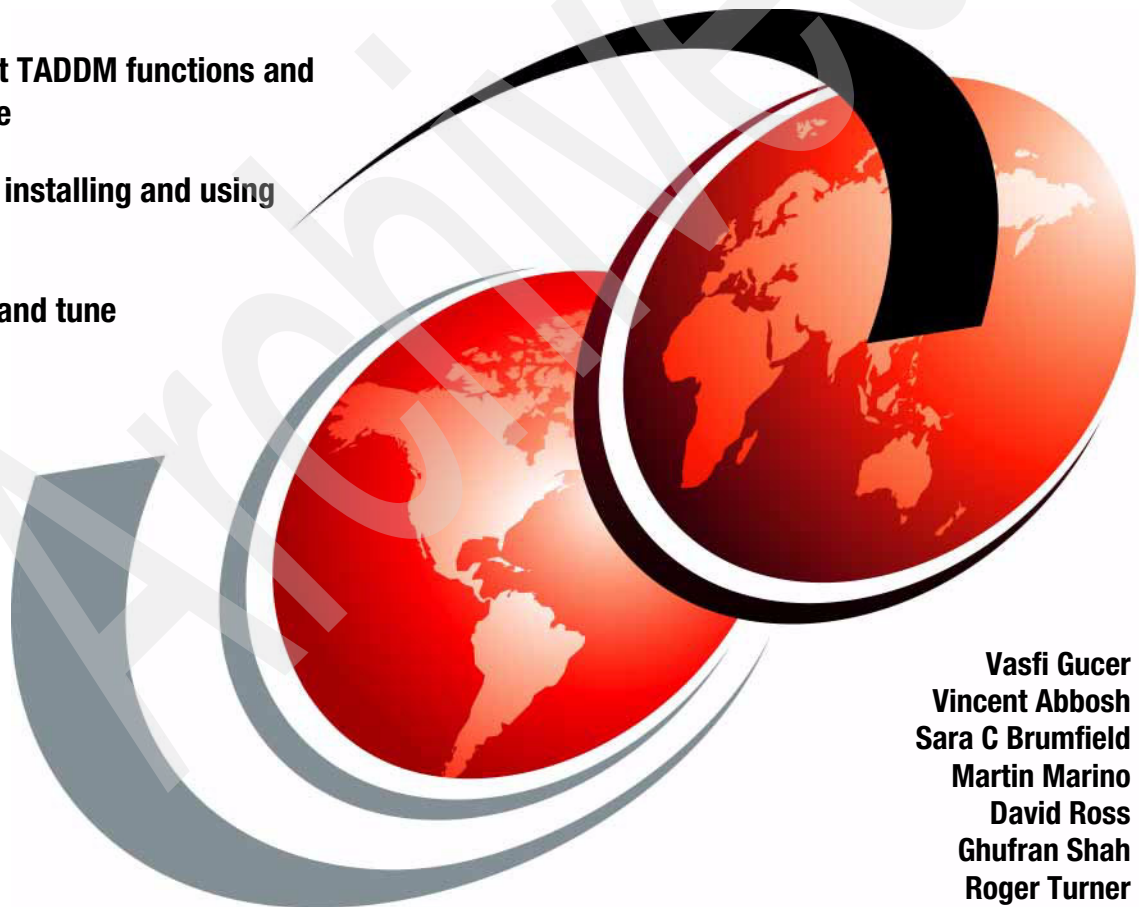


Deployment Guide Series: IBM Tivoli Application Dependency Discovery Manager V7.1

Learn about TADDM functions and
architecture

Get tips for installing and using
TADDM

Customize and tune
TADDM



Vasfi Gucer
Vincent Abbosh
Sara C Brumfield
Martin Marino
David Ross
Ghufran Shah
Roger Turner



International Technical Support Organization

**IBM Tivoli Application Dependency Discovery
Manager V7.1 Deployment Guide**

August 2008

Archived

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

First Edition (August 2008)

This edition applies to Version 7, Release 1, Modification 0 of IBM Tivoli Application Dependency Discovery Manager (product number 5724-N55).

© Copyright International Business Machines Corporation 2008. 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	xi
Tables	xvii
Examples	xix
Notices	xxi
Trademarks	xxii
Preface	xxiii
The team that wrote this book	xxiii
Become a published author	xxv
Comments welcome	xxvi
 Part 1. Tivoli Application Dependency Discovery Manager Introduction and Architectural Overview	 1
 Chapter 1. IBM Service Management overview	 3
1.1 Information Technology Infrastructure Library	4
1.1.1 ITIL Version 3	4
1.1.2 Critical success factors to implement ITIL	4
1.2 IBM and ITIL	6
1.3 IBM Service Management	6
1.3.1 Why businesses need ISM	7
1.3.2 IBM Service Management overview	8
1.3.3 IBM Service Management architecture	10
1.4 TADDM and IBM Service Management	11
1.4.1 Common data model	11
1.4.2 Automatic discovery of components and relationships	12
1.4.3 Automatic topology creation	12
1.4.4 Reconciliation	12
1.5 Summary	14
 Chapter 2. Introduction to Tivoli Application Dependency Discovery Manager	 15
2.1 TADDM overview	16
2.1.1 IT problems addressed by TADDM	16
2.1.2 TADDM capabilities	17
2.1.3 Entities discovered by TADDM	20

2.2 The TADDM discovery process	22
2.2.1 Discovery requirements	22
2.2.2 The discovery process	23
2.2.3 Discovery sensors	23
2.2.4 TADDM and secured environments	24
2.3 TADDM features	25
2.3.1 Auto-discovery	25
2.3.2 Open API and SDK	25
2.3.3 Deep configuration detail	26
2.3.4 Discovery profiles	26
2.3.5 Credential-less discovery	26
2.3.6 Change tracking	26
2.3.7 Secure interface	27
2.3.8 Central viewing console for multifunctional teams	27
2.3.9 Analytics	27
2.3.10 Versioning	30
2.3.11 Summary of TADDM features	31
2.4 Uses of TADDM	31
2.4.1 Configuration management foundation	32
2.4.2 Impact analysis visibility	32
2.4.3 Change management support	33
Chapter 3. Tivoli Application Dependency Discovery Manager architectural design	35
3.1 Introduction	36
3.2 TADDM architecture overview	38
3.2.1 TADDM architectural details	40
3.2.2 Discovery extensibility	44
3.2.3 TADDM APIs	44
3.2.4 Discovery Library technology	45
3.3 TADDM terminology	46
3.3.1 TADDM Server (Domain Manager)	46
3.3.2 TADDM user interface	47
3.3.3 TADDM Database	52
3.3.4 Anchor servers and Windows gateways	52
3.4 eCMDB	54
3.4.1 eCMDB overview	54
3.4.2 eCMDB synchronization	55
3.4.3 eCMDB database	56
3.4.4 eCMDB security	57

Part 2. Tivoli Application Dependency Discovery Manager Planning and Installation . . 59

Chapter 4. Deployment and capacity planning	61
4.1 Sizing your TADDM environment	62
4.1.1 TADDM Server sizing	62
4.1.2 Topology reconciliation is not a linear process	62
4.1.3 Database sizing considerations	63
4.2 Creating a deployment plan	64
4.3 Planning your hardware and software	65
4.3.1 Using Red Hat Enterprise Linux for your TADDM Server	72
4.3.2 Hardware requirements	72
4.4 TADDM deployment checklist	74
4.5 Planning worksheets	76
4.6 Deployment planning case study	80
4.6.1 Client scenario	80
4.6.2 Solution approach	81
4.6.3 Client solution	85
4.6.4 Additional sizing examples	87
 Chapter 5. Tivoli Application Dependency Discovery Manager installation steps	89
5.1 Our lab environment	90
5.2 Installing DB2	92
5.2.1 Install DB2 Enterprise Server	92
5.2.2 Create DB2 database users	106
5.2.3 Create the DB2 instances	107
5.2.4 Run the make_db2_db.sh script	107
5.3 Installing a TADDM Domain Server on Windows	108
5.3.1 Install TADDM 7.1	108
5.3.2 Install interim fix 0007	126
5.4 Installing a TADDM Domain Server on Linux	126
5.4.1 Install TADDM 7.1	127
5.4.2 Install interim fix 0007	143
5.5 Installing a TADDM enterprise server on AIX	144
5.5.1 Install TADDM 7.1	144
5.5.2 Install interim fix 0007	160
5.5.3 Configuring the eCMDB	161
5.6 Configuring LDAP	172
5.7 Deploying anchors and gateways	174
5.7.1 Enabling discoveries across the firewall	175
5.7.2 Defining an anchor host	176
5.7.3 Open ports	182

5.8 Setting up Windows gateways	182
5.8.1 Installing Cygwin SSH	183
5.8.2 Adding or changing a Windows gateway	190
5.9 Troubleshooting	193
5.9.1 Server not started automatically	194
5.9.2 Installation log files	195
Part 3. Discovery and Reporting Case Studies	197
Chapter 6. Discovery scenarios	199
6.1 Discovery sensors	200
6.1.1 Discovery overview	200
6.1.2 Discovery components	200
6.1.3 Discovery process in detail	202
6.1.4 Dependency discovery	207
6.1.5 Understanding sensors	208
6.1.6 Setting up discoveries	209
6.1.7 Discovery profiles	218
6.1.8 Level 2 profile	226
6.2 Customizing and managing discoveries	230
6.2.1 Custom servers	231
6.2.2 Custom server extensions	238
6.2.3 Computer system templates	244
6.2.4 The bulkload program	246
6.3 Reconciliation and prioritization	250
6.3.1 Manually merging discovered configuration items	253
6.3.2 Adding prioritization rules to your configuration items	258
6.4 Discovery Library Adapters	263
6.4.1 Discovery Library Adapter concepts	263
6.4.2 File naming conventions	265
6.4.3 Integration overview	266
6.4.4 Creating a Discovery Library Adapter	266
6.4.5 When to use a Discovery Library Adapter	269
6.5 Understanding the DLA APIs	270
6.5.1 Using the DLA adapter API	271
6.5.2 Managing configuration parameters and discoveries	272
6.5.3 Managing property change listeners	273
6.5.4 Managing Discovery Library Adapter states	274
6.5.5 Using the DLA Book Production API	275
6.5.6 Book properties and methods	275
6.5.7 Managed element properties and methods	278
6.5.8 Attribute properties and methods	280
6.5.9 Relationship properties and methods	281

6.6 Example of Discovery Library Adapter	282
Chapter 7. Reporting scenarios	293
7.1 Introducing BIRT	294
7.2 Deploying BIRT Report Viewer on TADDM	294
7.3 Designing TADDM Reports with BIRT	296
7.3.1 Designing reports with scripted data source	297
7.3.2 Designing reports with TADDM Database Views	317
7.4 Disaster recovery and validation	334
7.4.1 Versions	334
7.5 Root cause analysis with tracking changes	338
Part 4. Performance and Troubleshooting Considerations	341
Chapter 8. Performance considerations	343
8.1 Performance improvements in TADDM V7.1	344
8.2 Discovery tuning	344
8.3 Tuning storage performance	350
8.4 Caching user interface views	351
8.4.1 Understanding caching	351
8.4.2 Configuring caching	353
8.4.3 Maintaining the cache	354
8.5 Database considerations	355
8.5.1 Database indexes	355
8.5.2 Database settings: DB2	356
8.5.3 Initial database statistics on DB2	357
8.5.4 Running statistics	358
8.5.5 Bufferpool	358
8.6 Java Virtual Machine settings	361
8.6.1 Modifying the JVM arguments	361
8.6.2 Java Max memory	362
8.6.3 Java garbage collection	363
8.7 Log settings for production	363
8.8 Maintenance	363
8.8.1 Clearing out unknown servers	363
8.8.2 Finding and applying fixes and updates	364
Chapter 9. Troubleshooting	367
9.1 Log files	368
9.2 Installation logs	369
9.3 Problem determination tools	369
9.3.1 testhang.jy	370
9.3.2 testjdbc.jy	372
9.3.3 testssh.py	372

9.3.4	testos.jy	373
9.3.5	testping.jy	375
9.3.6	testportmap.jy	376
9.3.7	testportscan.jy	377
9.3.8	testprimaryip.jy	380
9.3.9	testsnmp.jy	380
9.3.10	testwmi.jy	381
9.3.11	wmiexec.jy	381
9.4	Log and Trace Analyzer	382
9.5	Specific scenarios	386
9.5.1	Common problems	386
9.5.2	Troubleshooting problems with sensors	387
9.5.3	Storage errors in sensors	389
9.5.4	Application programming interfaces (APIs)	390
9.5.5	Troubleshooting Windows discoveries	391
9.5.6	Troubleshooting SSH	400
Part 5.	Planning for a Client Engagement.	403
Appendix A.	Planning for a client engagement	405
	Services engagement preparation	406
	Implementation skills	406
	Available resources	407
	Solution scope and components	407
	Basic solution definition	409
	Advanced solution definition	410
	Services engagement overview	410
	Executive Assessment	411
	Demonstration system setup	412
	Analyze solution tasks	413
	Creating a contract	415
	Estimating the activities and timings of the engagement	417
	Perform environmental analysis and plan tasks	417
	Plan the solution	419
	Implement the solution	420
	Close the engagement	421
Appendix B.	Sample Statement of Work for Tivoli Application Dependency Discovery Manager	423
	Building an auto-discovery and device dependency solution	424
	Executive summary	424
	Solution description	425
	Assumptions	426
	Business partner responsibilities	426

Client responsibilities	427
Staffing estimates	427
Testing	427
Deliverables	428
Completion criteria	428
Abbreviations and acronyms	429
Related publications	431
IBM Redbooks publications	431
Online resources	431
How to get IBM Redbooks publications	432
Help from IBM	432
Index	435

Figures

1-1	Infrastructure complexity	7
1-2	IBM Service Management	10
2-1	Discover transactional relationships between the components and the applications.	18
2-2	A comparison that shows differences across configuration items.	30
3-1	TADDM architecture	38
3-2	TADDM data model.	42
3-3	Adding extended attributes in TADDM	43
3-4	Multiple Domain Servers	47
3-5	TADDM Java Product Console	48
3-6	TADDM Domain Manager UI	51
3-7	TADDM deployment using eCMDB.	54
5-1	Lab environment	91
5-2	DB2 Installation Welcome panel	93
5-3	Install a Product panel.	94
5-4	Welcome to the DB2 Setup wizard panel	95
5-5	Software License Agreement	96
5-6	Select the Installation type panel	97
5-7	Select installation, response file creation, or both panel	98
5-8	Select the installation directory panel	99
5-9	Set user information for the DB2 Administrator Server	100
5-10	Set up a DB2 instance panel.	101
5-11	Set up notifications panel	102
5-12	Start copying files panel	103
5-13	Installation progress panel	104
5-14	Setup has completed successfully panel	105
5-15	Install DB2 fix pack panel	106
5-16	The setupWin32.exe command	108
5-17	InstallShield Welcome panel	109
5-18	License Agreement	110
5-19	Installation directory	111
5-20	Defining a TADDM user	112
5-21	Choose the installation type	113
5-22	Select the server type	114
5-23	TADDM Server port information	115
5-24	Additional server ports when running in an enterprise environment	116
5-25	Specifying Remote Method invocation (RMI) information.	117
5-26	Optional CCMDB host name and port.	118

5-27	Select the database type.	119
5-28	Database configuration information	120
5-29	Select the user registry option.	121
5-30	Summary information	122
5-31	Installation completion.	123
5-32	The control status command and output.	124
5-33	Tivoli Application Dependency Discovery Manager page.	125
5-34	Product Console	126
5-35	InstallShield Wizard Welcome panel.	127
5-36	License Agreement	128
5-37	Installation directory	129
5-38	Defining a TADDM user	130
5-39	Choose the installation type	131
5-40	Select the server type	132
5-41	TADDM Server port information	133
5-42	Additional server ports when running in an enterprise environment	134
5-43	Specifying RMI information	135
5-44	Optional CCMDB host name and port.	136
5-45	Select the database type.	137
5-46	Database configuration information	138
5-47	Select the user registry	139
5-48	Summary information	140
5-49	The control status command and output.	141
5-50	Tivoli Application Dependency Discovery Manager page.	142
5-51	Product Console	143
5-52	InstallShield Wizard Welcome panel.	145
5-53	License Agreement	146
5-54	Installation directory	147
5-55	Defining a TADDM user	148
5-56	Choose the installation type	149
5-57	Select the server type	150
5-58	TADDM Server port information	151
5-59	Specifying RMI information	152
5-60	Optional CCMDB host name and port.	153
5-61	Select database type.	154
5-62	Database configuration information	155
5-63	Select user registry	156
5-64	Summary information	157
5-65	The control status command and output.	158
5-66	TADDM	159
5-67	Domain Manager.	160
5-68	eCMDB	161
5-69	Domain summary	162

5-70	Add Domains	163
5-71	Add Waco domain	164
5-72	Waco domain added	165
5-73	Add Southend domain	166
5-74	Waco and Southend domains added	167
5-75	Synchronize Domain: Waco	168
5-76	Waco full synchronization complete	169
5-77	Schedule daily synchronization for Waco	170
5-78	Waco daily synchronization scheduled	171
5-79	Domain summary showing scheduled synchronizations	172
5-80	Adding a new Discovery Scope Set	178
5-81	Adding a new target	179
5-82	Entering Access List credentials for Zaire	179
5-83	Entering scope limitations for anchor	180
5-84	Starting a new discovery	181
5-85	Cywin NetRelease Setup Program	184
5-86	Choose A Download Source	185
5-87	Choose Installation Directory	186
5-88	Select Local Package Directory	187
5-89	Select Packages	188
5-90	Installation Status and Create Icons	189
5-91	ssh-host-config utility	190
5-92	Adding a gateway and setting the scope	191
5-93	Running discovery of new Windows gateway	193
6-1	Discovery components	201
6-2	Discovery workflow process	203
6-3	Basic discovery sensor sequence	204
6-4	OS and application discovery	206
6-5	Viewing the loaded scope using the GUI	213
6-6	Viewing the loaded scope using the CLI	214
6-7	Configuring sudo access	220
6-8	Creating a new discovery profile	223
6-9	Deselecting SnmpLightSensor	224
6-10	Executing StackStan discovery	225
6-11	Checking the discovery status	226
6-12	Access Lists	229
6-13	Selecting scopes for Level 2 discovery	230
6-14	Identifying unknown server patterns	233
6-15	Creating a custom server	235
6-16	Selecting config files	236
6-17	Sendmail custom server discovered	237
6-18	Checking updated product version	241
6-19	Checking updated product version using TADDM Product Console	242

6-20	New Config File added	243
6-21	Using TADDM console to check the newly imported CIs	257
6-22	Merging ServerA and ServerB	257
6-23	ServerB after manual merge	258
6-24	Attribute Prioritization window	260
6-25	MSS source LinuxComputerSystem.xls	282
6-26	Viewing loaded Linux computer systems from the IDML Book	292
6-27	Viewing loaded Linux computer system detail from the IDML Book	292
7-1	Introduction panel for BIRT Report Viewer	295
7-2	A test report after it was rendered as HTML	296
7-3	Creating new Report Project	298
7-4	Adding the project name	299
7-5	Creating a new report	300
7-6	New report wizard	301
7-7	New data set for a scripted data source	302
7-8	Column definitions.	303
7-9	Script window for ComputerSystems data set showing available methods	303
7-10	Scripted data source sequence diagram.	304
7-11	Adding Javascript code to the data source	309
7-12	Scripted data set results preview	310
7-13	Edit Text Item dialog with the changes added	311
7-14	New Chart dialog with the Pie type chart selected	312
7-15	Preview of the chart after the data set is bound to it.	313
7-16	Formatting the chart	314
7-17	The final report layout	315
7-18	TADDM Computer Systems report using BIRT.	316
7-19	New JDBC data source.	318
7-20	JDBC connection details for CMDB on DB2	319
7-21	Adding DB2 JDBC JAR files to BIRT-managed drivers	320
7-22	JDBC connection details for Microsoft Excel sheet.	321
7-23	Software_inventory.xls sample content.	321
7-24	Setting up the software component query.	322
7-25	Query definition of Excel data set	323
7-26	Preview results of the Software_Lic data set	324
7-27	Insert Table dialog.	324
7-28	Add grouping feature to the table	325
7-29	The final design of taddm_software_audit.rptdesign.	326
7-30	The final design of excel_software_inventory.rptdesign	327
7-31	Choosing the Hyperlink option	328
7-32	The Hyperlink Options choices	329
7-33	excel_software_audit table filter options	330
7-34	The Highlight options for excel_software_inventory row.	331

7-35 TADDM Software Components Report	332
7-36 Software Inventory report with a highlighted item	333
7-37 Software Inventory report with no highlighted item	333
7-38 Version view	335
7-39 Selecting the Comparison report from the Topology View	336
7-40 Selecting the versions for the comparison report	336
7-41 DB2 Comparison report example	337
7-42 Apache Server Comparison report example	337
7-43 Apache Comparison report details	338
7-44 Change History settings	339
7-45 Change History report for Order Management business application . .	339
8-1 Discovery status	345
8-2 Discovery in progress	346
8-3 Discovery done	347
8-4 In progress status	348
8-5 Script to create the buffer.out file	359
8-6 Sample buffer.out file	360
9-1 Components of a log message	388
9-2 WMI Tester	393
9-3 Namespace	394
9-4 WMI Tester	394
9-5 Connect.	396
9-6 WMI Tester	396
9-7 WMI Tester	397
9-8 TADDM message	401
9-9 Error message.	401
9-10 RSA key added	401
9-11 Portfolio of products	408

Tables

1-1 ComputerSystem naming rules	13
2-1 Entities discovered by TADDM	21
3-1 TADDM Java Product Console uses.	49
4-1 Supported operating systems for TADDM V7.1 components	65
4-2 TADDM deployment checklist	74
4-3 Setting for a typical installation	77
4-4 Additional settings for a custom installation	77
4-5 Additional port values	78
4-6 Settings for Oracle database.	79
4-7 Ports used by the PingSensor and PortSensor to make connections.	79
4-8 Client ABC IT Infrastructure	81
4-9 Sample data showing the number of CIs for an Item	82
4-10 Additional CIs for each database or application server instance	83
4-11 Performance benchmark using various discovery techniques	84
5-1 DB2 database users	107
6-1 Discovery scope information	210
6-2 Access details by components	217
6-3 Discovery profiles by default in TADDM	218
6-4 Custom server information	232
6-5 Directive file format	239
6-6 Reference for the loadidml.sd command	248
6-7 Configuration parameters and discovery methods	272
6-8 Property change listener methods	274
6-9 State methods	274
6-10 Book production properties	276
6-11 Book production methods	276
6-12 Managed element properties	279
6-13 Managed element methods.	279
6-14 Attribute properties	280
6-15 Attribute methods	280
6-16 Relationship properties	281
6-17 Relationship methods	281
8-1 Improvements over TADDM 5.1.3.	344
9-1 Ports and related protocol	377
A-1 Solution tasks	412
A-2 Solution demonstration tasks	413
A-3 Skill adjustment.	415
A-4 Estimated time in hours for identified tasks	417

A-5 Timeline estimates for implementation activities	421
--	-----

Examples

4-1	SmallManufacturer Inc	87
4-2	MediumInsurer Inc	88
4-3	LargeInsurer Inc	88
5-1	Obtaining sslpassphrase and unicastdiscoveryport values.	164
5-2	Adding the relative distinguished name administrator.	173
6-1	Sensor logging	209
6-2	Using the loadscope.jy command	212
6-3	Executing Nmap after install	221
6-4	Preparing csv files to import Level 2 Scopes	227
6-5	Listing scope files to import	227
6-6	Importing the new scopes	228
6-7	Querying sendmail version	239
6-8	Directive file for Sendmail custom server	240
6-9	Querying updated Sendmail version	240
6-10	Editing the screencontent.xml file	241
6-11	Adding the command to create a configuration file	242
6-12	AixUnitaryComputerSystem ServerA	254
6-13	AixUnitaryComputerSystem ServerB	255
6-14	ExcelConnection.java	283
6-15	ExcelConnection.java	287
6-16	LinuxComputerSystem.xml	287
6-17	Validating the IDML Book	290
6-18	Loading the IDML Book	291
7-1	CMDBDriver.java class	304
7-2	Compiling CMDBDriver.java class using the Windows command line	308
7-3	Sample report title in HTML	311
7-4	Sample title text box content for excel_software_inventory report	331
8-1	Performance degradation	349
8-2	Directories	352
8-3	Base recommended view cache settings	353
8-4	The upd_db_cfg.sql script	356
8-5	Javacore	362
9-1	Using testhang.jy	371
9-2	Using testjdbc.jy to connect to tiodb	372
9-3	Using testos.jy	373
9-4	Using testportmap.jy	376
9-5	Using testportscan.jy	378
9-6	testportscan.jy	379

9-7 Using testprimaryip.jy	380
9-8 Using testsnmp.jy	380
9-9 Using testwmi.jy	381
9-10 Using wmiexec.jy	382
9-11 Finding a MQL query in the ApiServer.log	390
9-12 The ERROR message in ApiServer.log	391
9-13 TaddmTool command	398

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

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. These and other IBM trademarked terms are marked on their first occurrence in this information with the appropriate symbol (® or ™), indicating US registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at <http://www.ibm.com/legal/copytrade.shtml>

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

AIX®	Micromuse®	RS/6000®
DB2®	Netcool®	System z®
i5/OS®	PartnerWorld®	Tivoli Enterprise Console®
IBM®	Redbooks®	Tivoli®
Intelligent Device Discovery®	Redbooks (logo)  ®	WebSphere®

The following terms are trademarks of other companies:

ITIL is a registered trademark, and a registered community trademark of the Office of Government Commerce, and is registered in the U.S. Patent and Trademark Office.

IT Infrastructure Library is a registered trademark of the Central Computer and Telecommunications Agency which is now part of the Office of Government Commerce.

SUSE, the Novell logo, and the N logo are registered trademarks of Novell, Inc. in the United States and other countries.

Oracle, JD Edwards, PeopleSoft, Siebel, and TopLink are registered trademarks of Oracle Corporation and/or its affiliates.

SAP, and SAP logos are trademarks or registered trademarks of SAP AG in Germany and in several other countries.

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

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

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

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

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

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

Preface

In this IBM® Redbooks® publication, we describe the capabilities and ways to use the IBM Tivoli® Application Dependency Discovery Manager (TADDM). It is becoming critical for enterprises to track the IT resources in their environment and, more importantly, the dependencies of their business applications on various components. TADDM provides rich capabilities that discover the components of a complex infrastructure and their interdependencies.

In this book, we provide insight into the TADDM V7.1 capabilities and architecture. We include recommended procedures for installing and configuring TADDM and tips and techniques for populating the TADDM Database and customizing its use and performance considerations.

Finally, we describe the sales engagement planning for TADDM V7.1, including a sample statement of work. The primary audience for this section is Business Partners and pre-sales Systems Engineers working in this area.

This book is a major reference for IT Specialists and IT Architects working in TADDM V7.1 projects.

The team that wrote this book

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

Vasfi Gucer is an IBM Certified Consultant IT Specialist at the ITSO Austin Center. He was with IBM Turkey for 10 years and has worked at the ITSO since January 1999. He has more than 13 years of experience in teaching and implementing systems management, networking hardware, and distributed platform software. He has worked on various Tivoli client projects as a Systems Architect and Consultant. Vasfi is also a Certified Tivoli Consultant.

Vincent Abbosh is an Advisory Software Engineer in IBM Tivoli Advanced Technology Group, working at IBM Australia Development Lab on the Gold Coast. He graduated in 1992 from Aleppo University - Syria with BEng degree in Computer Engineering. He has over 15 years of experience in IT and software development in the areas of security and service management. He joined IBM Tivoli Gold Coast Lab in 2003 as software engineer where he worked on developing Global Security Kit (GSKit), IBM JDK Security, Federated Identity Management, and Common Auditing and Reporting Service. His current focus is

on developing Information Technology Infrastructure Library (ITIL®)-based Service Management solutions.

Sara C Brumfield is a Software Engineer at IBM, currently working in the Tivoli Support Center. Her career has included many diverse projects, including developing system management tools for AIX®, starting the hosted services group for a startup, and recruiting development tools vendors for the RS/6000® server platform. Sara holds a Bachelors degree from Rice University, with majors in Computer Science and the Study of Women and Gender.

Martin Marino is an IT Specialist within the IBM Global Technology Services group in IBM Argentina. He has specialized in Tivoli products since 2003. He is currently working with TADDM for Strategic Outsourcing clients in Argentina and Latin America. His areas of expertise include Tivoli Monitoring, Tivoli Configuration Manager, Tivoli Remote Control, and Tivoli Management Framework.

David Ross is a Technical Course Developer with IBM Tivoli in the United States. A former classroom instructor and network administrator, he has been with IBM for over eight years. He holds a degree in Secondary Education with a Mathematics minor from Texas A&M University and another in Computer Science from the University of Texas Permian Basin. He has developed both classroom and Web-based training materials on products involving security, network management, service desk support, and system automation. He has spent the past two years focused on the Information Technology Service Management (ITSM) products, particularly TADDM and CCMDB. In addition to course development, he has taught Tivoli courses and delivered lectures in classrooms and conferences around the world.

Ghufran Shah is a IBM Certified Advanced Deployment Professional in Enterprise, Provisioning, and Business Application Management Solutions. He has ten years of experience in Systems Development and Enterprise Systems Management. He holds a degree in Computer Science from the University of Bradford. His areas of expertise include Tivoli Systems Management Architecture, Implementation, and Tivoli Training, together with Business Process Improvement. He has written extensively about event Management, Monitoring, and Business Systems Management integration and has taught IBM Tivoli courses worldwide.

Roger Turner is a Senior Managing Consultant with the IBM Software Services for Tivoli (ISST) organization. He has been with for IBM for 28 years, the past 12 years as a consultant with the ISST organization. As an ISST consultant, he has been working with clients to implement business system and service management products, including Global Enterprise Manager, Tivoli Business Systems Manager (TBSM) 2.1 and 3.1, and Tivoli Business Service Manager 4.1. He has worked with clients in many focus areas, including banking,

insurance, government, financial services, retail, and health. For the past 2 1/2 years, he has also been working with clients to implement the Tivoli Application Dependency Discovery Manager (TADDM) product. His area of expertise includes the integration of TBSM and TADDM. He holds numerous certifications, including ITIL Service Management - Service Support and Service Delivery, TBSM 3.1 distributed and TBSM 3.1 mainframe, and TADDM 7.1.

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

Arzu Gucer

International Technical Support Organization, Austin Center

Ed Bernal, Byron Gehman, and Mike Mallo
IBM USA

The team would like to express special thanks to Jan Erik Hoel from IBM Norway for his contributions to Chapter 8, “Performance considerations” on page 343 and Chapter 9, “Troubleshooting” on page 367.

Thanks to the authors of the previous editions of this book:

- Authors of *IBM Tivoli Application Dependency Discovery Manager Capabilities and Best Practices*, SG24-7519, which was published in February 2008, were:
 - Bart Jacob
 - Bhavesh Adhia
 - Karim Badr
 - Qing Chun Huang
 - Carol S. Lawrence
 - Martin Marino
 - Petra Unglaub-Lloyd

Become a published author

Join us for a two- to six-week residency program! Help write a book dealing with specific products or solutions, while getting hands-on experience with leading-edge technologies. You will have the opportunity to team with IBM technical professionals, Business Partners, and Clients.

Your efforts will help increase product acceptance and client satisfaction. As a bonus, you will 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 books to be as helpful as possible. Send us your comments about this book or other IBM Redbooks publications in one of the following ways:

- ▶ Use the online **Contact us** review IBM Redbooks publications form found at:

ibm.com/redbooks

- ▶ Send your comments in an e-mail to:

redbooks@us.ibm.com

- ▶ Mail your comments to:

IBM Corporation, International Technical Support Organization
Dept. HYTD Mail Station P099
2455 South Road
Poughkeepsie, NY 12601-5400



Part 1

Tivoli Application Dependency Discovery Manager Introduction and Architectural Overview

In this part, we introduce IBM Service Management portfolio and Tivoli Application Dependency Discovery Manager (TADDM).

IBM Service Management overview

In the fall of 2007, the IBM Systems Journal provided a series of papers focused on the IBM Service Management strategy and related technologies and solutions. This IBM Systems Journal is available at:

<http://www.research.ibm.com/journal/sj46-3.html>

We extracted and paraphrased information from the papers presented in this IBM Systems Journal to create part of the content in this chapter.

In this chapter, we discuss the IBM Service Management initiative. We also describe the role that IBM Tivoli Application Dependency Discovery Manager (TADDM) can play in implementing a comprehensive solution to address the needs of current IT organizations.

In this chapter, we discuss:

- ▶ “Information Technology Infrastructure Library” on page 4
- ▶ “IBM and ITIL” on page 6
- ▶ “IBM Service Management” on page 6
- ▶ “TADDM and IBM Service Management” on page 11
- ▶ “Summary” on page 14

1.1 Information Technology Infrastructure Library

Information Technology Infrastructure Library (ITIL) is an internationally recognized framework that provides comprehensive best practice guidelines for all aspects of end-to-end Service Management. ITIL includes people, processes, products, and the use of partners. It began in the 1980s when the UK Government initiated an exercise to standardize its diverse IT processes.

ITIL has evolved over the years to cover Service Support and Service Delivery, and in 2007, Version 3 was launched, which includes a life cycle management approach in five core volumes: Service Strategy, Service Design, Service Transition, Service Operation, and Continual Service Improvement.

The best practices contained in ITIL are independent of tool, vendor, or industry and can be applied to an organization of any size. ITIL encourages organizations to adapt and adopt its suggestions to meet business needs and improve processes. Though there is a significant amount of detail in the books that make up the library, the books are not themselves the solution to all IT management issues. The processes require significant work to deploy at a level of detail enabling day-to-day use, with dependencies on the three key components (process, people, and tools) of a management system.

Even though there are many references to ITIL as a standard, it is not a standard. Organizations cannot comply with ITIL. It is a set of guidelines that an organization can adopt and adapt to their needs.

1.1.1 ITIL Version 3

ITIL Version 3 focuses on best practices throughout the service life cycle. It focuses essentially on service and solution life cycle management, including five core volumes: Service Strategy, Service Design, Service Transition, Service Operation, and Continual Service Improvement. Further discussion of ITIL Version 3 is outside the scope of this book.

1.1.2 Critical success factors to implement ITIL

Because ITIL is a framework of best practices and not a methodology, it only describes what needs to be done. ITIL does not provide guidance for how to implement the processes, so each company chooses the best way to fit ITIL to its requirements.

A key mind-set when implementing ITIL is “adopt and adapt”. “Adopt” ITIL as a common language and reference point for IT Service Management, and “adapt” ITIL best practices to achieve business objectives.

Generally, IT organizations do not implement all of the ITIL processes, because they do not have sufficient budget, and they determine that they do not need all of the processes. Initially, implementing a subset of all processes can be seen as a way to avoid extra costs. However, depending on the processes that you choose to implement, excluding other processes might result in less benefit from those processes that you implement. For example, choosing to implement Change and Release processes without implementing Configuration might result in an inaccurate impact assessment when approving changes.

You must carefully select the service management processes, taking into consideration the relationship among all processes in addition to the cost perspective and implementation complexity of individual processes.

A successful implementation of IT Service Management must:

- ▶ Be aligned with business needs - business-driven not technology-driven
- ▶ Improve staff awareness about business goals
- ▶ Be adapted to the culture of the organization. This adaptation must be done when defining the roles, responsibilities, tools, processes, procedures, tasks, and so on. After IBM Service Management is implemented, it must be rigorously followed.
- ▶ Have its processes clearly defined, documented, and available
- ▶ Have its main processes integrated with each other
- ▶ Have its inputs measurable and repeatable
- ▶ Have IT processes supported by tools and customized to fit the processes defined
- ▶ Have processes easily changed as necessary
- ▶ Be integrated with external suppliers
- ▶ Include properly training and communicating to all people who will use or provide IT services
- ▶ Have clearly measurable and repeatable key performance indicators

A successful IBM Service Management implementation needs to result in improved IT client satisfaction, better resource utilization, and improved client perception of IT service quality.

1.2 IBM and ITIL

IBM initially contributed to ITIL with its systems management concept “yellow books” and continues to contribute as a developer, reviewer, and user of ITIL.

IBM contributed in many ways to ITIL Version 2, including authoring, quality reviews, project management, and additional support through the IT Service Management Forum. The focus of Version 2 was on process management practices required to enable service management. The ITIL service support and delivery publications contain significant contributions from IBM. The ITIL application management book, co-written by authors from IBM and other companies, is the basis for the life cycle concept in ITIL Version 3. It lays the basic groundwork for how to integrate service management practices throughout the solution life cycle.

IBM supports the development of updates and refreshes to industry-accepted best practices, including supporting the ITIL Advisory Group through quality reviews and other briefings. Thought leaders also serve on the ITIL Advisory Group and other working groups to contribute as the need arises. IBM views ITIL as a valuable set of publications that promote best practices in service management. From a strategic outsourcing perspective, ITIL is requested by many IBM clients all around the globe. Companies that are implementing improvements to their service management capabilities consider ITIL a good place to start.

1.3 IBM Service Management

IBM has developed thought leadership to improve the “state of the art” in service management for the last 25 years and has supported other companies in their efforts as well. In addition to the advancement of management disciplines and technologies, IBM recognized early on that acceptance of common practices and standards is vital to achieving improved value from information technology (IT).

Advances in technologies and management disciplines provide the greatest value when they become part of and extend the body of generally accepted practices and open standards. IBM supports the advancement of practices and open standards, such as ITIL (the IT Infrastructure Library®), COBIT (Control Objectives for Information Technology), ISO IEC 20000, and Carnegie Mellon University’s e-Sourcing Capability Model (e-SCM). The fundamental characteristics of service management require integration and agreement on standards, not only between tools and roles within IT, but also among organizations and even industries.

IT service management is the integrated set of activities required to ensure the cost and quality of IT services valued by the client. It is the management of client-valued IT capabilities through effective processes, organization, information, and technology, including:

- ▶ Aligning IT with business objectives
- ▶ Managing IT services and solutions throughout their life cycles
- ▶ Service management processes, such as those processes described in ISO IEC 20000, ITIL, and the Process Reference Model for IT.

1.3.1 Why businesses need ISM

Today's enterprises face an ever-increasing problem of managing their IT processes to deliver IT services in a manner that is:

- ▶ Efficient
- ▶ Reliable
- ▶ Secure
- ▶ Consistent

At the same time, the complexity of the infrastructure needed to deliver these IT-enabled business services has been increasing rapidly. Figure 1-1 shows a simple example of the complexity of IT environments.

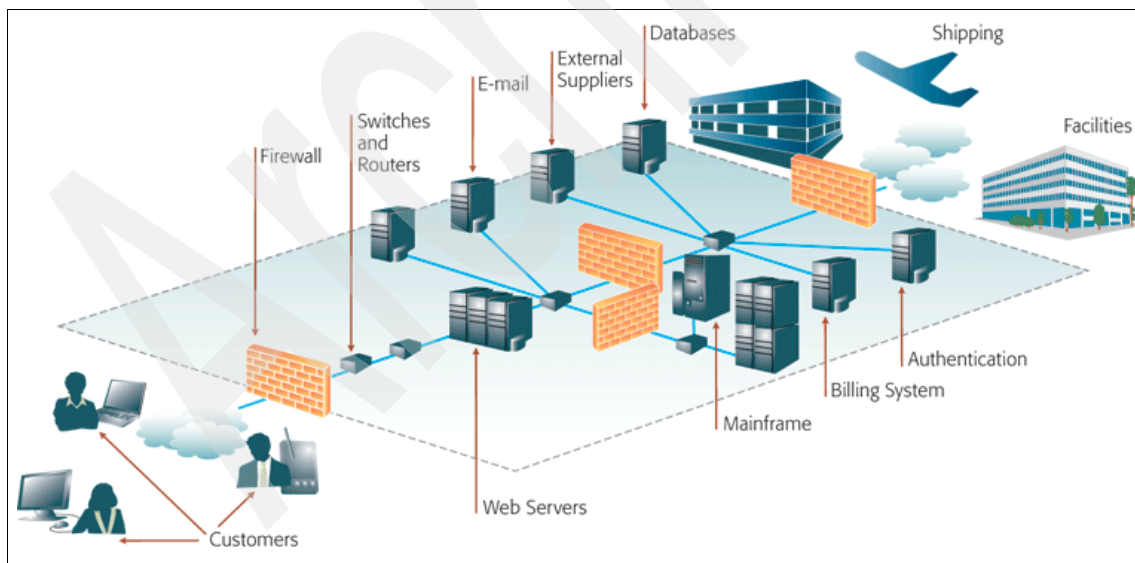


Figure 1-1 Infrastructure complexity

Several of the key challenges faced by businesses include:

- ▶ **Complexity:** The root cause of the problems that IT organizations face lies in the dramatic increase of business complexity due to heterogeneity of environments and the interconnection of applications (composite applications). Architectural and organizational issues, accelerating the proliferation of composite applications and hardware entities, and worldwide operations spanning multiple time zones all contribute to reducing the efficiency and effectiveness of the IT organization.
- ▶ **Change:** Complexity makes for hard-to-manage infrastructures that often break when changed and whose management requires a discipline that few companies achieve without flaws. Increasing workloads, more stringent service-level assurance requirements, staff turnover, and new market opportunities all lead to pressure for change in the IT organization. Change is the leading cause of service or application disruption today, and it often results in visible business impact. In fact, our experience suggests that nearly 80 percent of all critical outages can be traced to faulty change management.
- ▶ **Cost:** Currently, operational IT labor cost constitutes almost 70 percent of the total IT budget of businesses. In the late 1990s, half of the IT labor budget was devoted to new application development, and half of the IT labor budget was devoted to operations. Because IT budgets have been held flat, the chief information officers of IT organizations have faced two unappealing choices: shift resources from new application development or reduce the level of support for current applications. Both options serve to reduce the efficiency and effectiveness of IT.
- ▶ **Governance and compliance:** The introduction of government regulations, such as the Sarbanes-Oxley Act (SOX) and the Health Insurance Portability and Accountability Act (HIPAA) in the United States, have put an additional burden on the IT organization. IT must now support the needs of the business to audit for compliance through the institution of better process controls and the maintenance of audit trails for IT infrastructure changes. This support requires careful consideration because of the penalties of noncompliance, including criminal and civil liabilities and adverse public opinion.

1.3.2 IBM Service Management overview

For many businesses, service excellence is increasingly a competitive differentiator, because organizations need to rapidly adapt to changing conditions in the marketplace and create and deploy new services quickly and efficiently. However, service excellence can only be achieved through effective and efficient service management.

A fundamental goal of IT Service Management is the management of IT services and infrastructure with the same kinds of quality control that enterprises strive to

use for all business processes. When this management is achieved, businesses have the confidence to deploy new and updated services that are critical to their missions.

An effective IT Service Management capability reduces the time needed to deliver a company's IT services according to business policies and reduces the labor cost of the people involved in executing the processes by replacing manual IT process management with autonomic management.

IBM Service Management is an approach designed to automate and simplify the management of business services. It concentrates on four areas of study:

- ▶ Technology integration and standards
- ▶ Improved collaboration among IT personnel spread across organizational silos
- ▶ Best practices-based process modules to enable automated process execution
- ▶ Sharing business critical IT information to improve decision making

In finding workable solutions to these areas, IBM solutions cover four key areas:

- ▶ Process Managers that provide automated ITIL-aligned workflows for key IT processes
- ▶ An open, standards-based IBM IT Service Management platform
- ▶ Integration between process tasks and operational management products to automate the running of those tasks from the process flow
- ▶ Best practices to help pull it all together

These four key areas are pictured in Figure 1-2 on page 10.

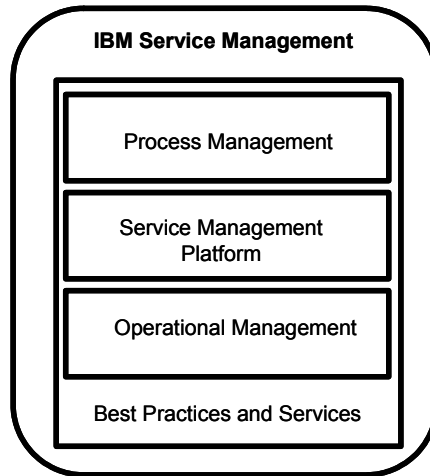


Figure 1-2 IBM Service Management

1.3.3 IBM Service Management architecture

The IBM Service Management architecture is illustrated in Figure 1-2. Note several of the major components:

- ▶ A user interface that provides access to IT personnel across multiple disciplines and areas of concern
- ▶ A process layer to coordinate workflows and process management across the ITIL catalog (service delivery and support, change management, storage management, and so on)
- ▶ An operational management component shown by the operational management products (OMPs) and the integration modules
- ▶ A central information component represented as the Configuration Management Database (CMDB) and the supporting functions

TADDM is a crucial element in the CMDB component. If used with IBM Service Management software, such as the IBM Tivoli Change and Configuration Management Database (CCMDB), TADDM is the primary means of populating the CMDB through discovery.

The CMDB also has requirements, which are addressed by TADDM. Part of the functionality that must be offered or supported by the CMDB includes:

- ▶ Common data model for the entire IT environment (including relationship information):
 - Physical components (computers, devices, and so on)
 - Logical components (business services, applications, and so on)
- ▶ Automatic discovery and change tracking
- ▶ Reconciliation capabilities in order to effectively integrate from multiple sources of data
- ▶ Visual display of relationships and dependencies

1.4 TADDM and IBM Service Management

In this section, we discuss the role that TADDM plays in IBM Service Management in meeting the CMDB requirements that were outlined in the previous section.

1.4.1 Common data model

The common data model (CDM) that is used by TADDM defines the following constructs:

- ▶ *Models*: Models consist of attributes, classes, and relationships.
- ▶ *Classes*: Classes represent IT entities and can participate in relationships. An example is `ComputerSystem`.
- ▶ *Attributes*: Attributes are defined separately from classes and can be reused in any number of class definitions. An example is `GUID`, the globally-unique identifier.
- ▶ *Relationships*: Relationships are strongly typed entities apart from classes. All relationships are binary (having only two objects defined in the relationship). An example is `runsOn`, where one entity, such as `Aix` (the operating system), actually is executing and requires the execution environment provided by another component (`ComputerSystem`). The relationship can be described as `Aix runsOn ComputerSystem`.
- ▶ *Data types*: Data types are similar to the data types in programming languages. An example is `String` as in the attribute `Description` is of type `String`.

The CDM currently has about 1 000 classes, subclasses, and relationships defined.

1.4.2 Automatic discovery of components and relationships

There are many reasons to need to discover IT assets, including deployment, license compliance, and more. There are also many products on the market that are able to discover this information either directly or through the use of agent technology. TADDM is unique in its ability to discover not only the hardware, software, and related information in the IT environment, but also the relationships that exist between these entities.

1.4.3 Automatic topology creation

As part of the discovery process, TADDM also captures open port and listening process information. After all of the given set of systems is discovered, TADDM applies a heuristic-based process to determine the implied relationships between configuration items (CIs) that it has found. For example, a Web server might be communicating through a specific host port, and an application server might be listening to that port on that host. TADDM leverages the implied relationship to build a complete dependencies graph automatically.

1.4.4 Reconciliation

Because multiple products manage the IT environment, each product collects data about the configuration item (CI) for which that product is responsible. Products from multiple vendors, or even the same vendor, often store their collected data in various ways. The information in each product data store might indeed be referencing the same actual CI in the environment, but it might have different information captured as part of the processing. For example, a licensing software application might identify a computer system in the network by collecting the manufacturer, model, and serial number information along with the system board Universally Unique Identifier (UUID) as a 'fingerprint' for the given system. However, another product used for asset management might identify the computer using the Media Access Control (MAC) address of the network interface card (NIC) and the system board UUID. If data from these two OPMs is imported into TADDM (which also might have discovered the same computer system), there is potential for multiple CI entries to be created for the same CI in the database.

This situation is addressed in the CDM through the use of *naming rules* that are defined for each class in the data model. Naming rules have these major components:

- ▶ *Class*: The class used to name instances, such as ComputerSystem
- ▶ *Superior class*: A class larger than the given class (if it exists); as an example, the OperatingSystem class is named as a subordinate to the ComputerSystem class. This example means that the naming rule for an operating system includes the computer system on which it is installed.
- ▶ *Naming attribute*: A string that denotes an attribute, which can be used in naming the CI.
- ▶ *Priority*: A non-negative integer signifying the priority of the naming attributes in identifying a CI

For example, Table 1-1 shows the naming rules for the ComputerSystem class.

Table 1-1 ComputerSystem naming rules

Priority	Attributes
0	Signature
1	Manufacturer, Model, Serial Number
2	systemBoardUUID
3	primaryMACAddress
4	hostSystem, VMID
5	ManagedSystemName
6	VMID, Manufacturer, Model, Serial Number

As an instance of the ComputerSystem class is created, the naming rules are used to generate the Type 3 GUID¹, using the naming rules met and the associated priority. If any existing CI instance matches the GUID, the two CI instances are merged.

In our previous example, the fact that both OPMs are able to read the system board UUID (which is all that is needed to match the naming rule with priority 2) results in merging the CI instances in the CMDB.

¹ Refer to Internet Engineering Task Force (IETF) RFC 4122 for more information about GUID generation.

1.5 Summary

Managing IT environments is a complex and often frustrating proposition. This is particularly true in modern distributed, multi-national and multi-application deployments. In order to gain control of these environments, it is vital that an accurate understanding of the environment is established and that information is made available to all parties involved in managing the environment.

The ITIL library outlines the best practices for managing IT service environments. IBM Service Management leverages the best practices of ITIL and provides a comprehensive and robust solution. IBM Service Management incorporates a broad array of applications and best practices to give IT organizations control of the infrastructure.

TADDM is a vital component in the IBM Service Management initiative. The automatic discovery of applications, software, hosts, and other IT components, along with the dependencies that exist among these components, is critical in understanding the environment. The automatic creation of topology maps that reflect the dependencies and relationships supports robust decision-making when addressing outages, performance issues, upgrades, and change planning. In the next chapter, we explore TADDM in greater detail.

Introduction to Tivoli Application Dependency Discovery Manager

In this chapter, we provide an overview of the Tivoli Application Dependency Discovery Manager (TADDM).

In general, we describe:

- ▶ A description of TADDM
- ▶ A high-level overview of how TADDM discovery works
- ▶ An overview of TADDM features
- ▶ Examples of common uses of TADDM

2.1 TADDM overview

TADDM is an auto-discovery solution that provides automated application dependency mapping and configuration tracking. It provides a high level of visibility into how the computing infrastructure actually delivers the critical applications supporting business operations.

Furthermore, TADDM provides a starting foundation for transforming your IT service management strategy. This transformation can be accomplished by augmenting TADDM application mapping data with other enterprise application data, such as governance, finance, and so on.

2.1.1 IT problems addressed by TADDM

Today's composite applications (for example, Java™ 2 Platform, Enterprise Edition (J2EE), .NET, and so on) consist of large numbers of infrastructure components that have complex runtime dependencies. Without visibility into these applications and their supporting infrastructure, it is difficult to effectively deliver and manage mission critical business services. This problem has become even more acute with the accelerating rate of change in today's application infrastructure.

TADDM helps you answer the following important questions about your enterprise computing environment:

- ▶ How does my infrastructure support the delivery of business applications and services?
- ▶ What are the interdependencies among my business applications, my software applications, and my physical components, such as hosts and network devices?
- ▶ How can I understand the impact of a single configuration change on my business application or service?
- ▶ What changes are taking place in the application environment and where?
- ▶ Am I solving the problems that are introduced by changes quickly and efficiently?

In order to improve their problem detection, isolation, and resolution processes, businesses require complete and accurate cross-tier runtime dependency and configuration tracking. Application visibility is equally important for implementing configuration consistency audits within the enterprise to ensure compliance to technology and regulatory standards.

Additionally, to effectively implement IT Service Management initiatives, enterprises need to incorporate and correlate IT infrastructure information with data from other enterprise applications, such as governance, finance, and service desk. The TADDM Product Console has the ability to supplement application maps with queries into other enterprise data sources.

TADDM is a tool that helps IT operations personnel to ensure and improve application availability in application environments. TADDM provides the operational staff with a top-down view of applications so that the staff can quickly understand the structure, status, configuration, and change history of the business-critical applications. This top-down view enables immediate isolation of issues during performance or availability problems and more effective planning for nondisruptive application change.

By providing top-down, cross-tier views of how the IT infrastructure actually delivers applications, TADDM allows IT organizations to:

- ▶ Understand the structure of interdependent and complex applications
- ▶ Rapidly isolate configuration-related application problems, reducing troubleshooting time dramatically
- ▶ Better understand the impact of component-level events in order to sort issues based on application and service impact
- ▶ More effectively plan change so that application upgrades and deployments can occur without disruptions
- ▶ Create a shared topological definition of applications for use by other management applications, such as service level managers and provisioning tools

2.1.2 TADDM capabilities

In this section, we discuss the capabilities of TADDM.

Native discovery and application mapping

TADDM delivers *native discovery* capabilities that your organization can use to obtain a detailed understanding of its supporting infrastructure. Native discovery includes discovery down to layer 2 network devices, storage devices, cross-tier dependencies, and runtime configuration. Unlike products with limited abilities to visualize infrastructure, TADDM provides detailed maps of business applications and their relationships to one another (Figure 2-1 on page 18).

IT organizations can leverage automated maintenance of these application maps and easily integrate this data with other enterprise information that helps to:

- ▶ Support cost-effective and successful implementation of business service management initiatives using predefined integration with IBM Tivoli Business Systems Manager and other business service management tools
- ▶ Dramatically lower the business risks of service failures and inconsistencies
- ▶ Facilitate efforts to comply with technology and regulatory standards
- ▶ Significantly reduce problem isolation time

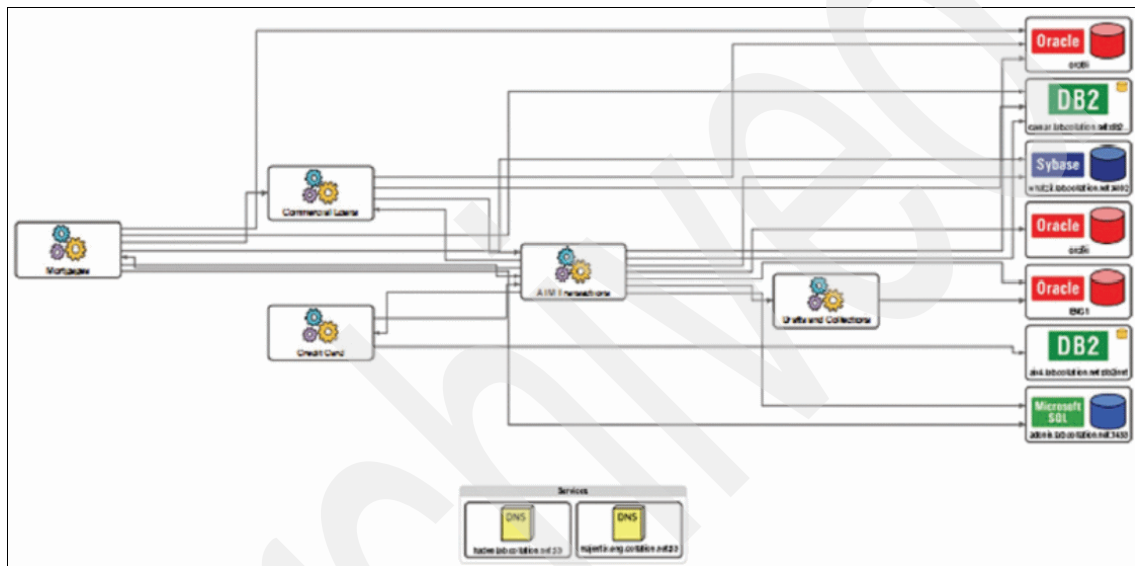


Figure 2-1 Discover transactional relationships between the components and the applications

TADDM provides the breadth and depth of application infrastructure visibility that organizations need to coordinate and help manage configuration changes and processes throughout an enterprise. With more than 200 sensors available at installation, TADDM provides complete visibility into operating systems, custom application platforms, middleware, network routers and switches, and packaged applications.

Within applications, TADDM provides visibility into all of the relevant information that is needed to optimize service delivery and agility, which includes:

- ▶ Changes to deployed application modules
- ▶ Dependencies between individual software processes, whether they are running in Microsoft® Windows®, Linux®, UNIX®, or mixed environments

- ▶ Dependencies on critical network services, such as Lightweight Directory Access Protocol (LDAP), Network File System (NFS), and Domain Name Service (DNS)
- ▶ Software-logic dependencies on the physical network (including layer 2) and storage layers

You can also extend TADDM to meet your specific needs through the use of custom server extensions and templates.

Data integration and federation

With few exceptions, organizations will need to share data from another source and bring it into TADDM. To facilitate this process, IBM developed the concept of a *discovery library*. The discovery library initiative is designed to ease the sharing of data across multiple applications through the use of common specifications and the common data model (CDM). It uses an XML specification called Identification Markup Language (IDML) that enables you to collect data from IBM Operational Management Products (OMPs), independent software vendor (ISV) OMPs, client spreadsheets, and more. IBM provides access to IDML files through Discovery Library Adapters (DLAs) for many IBM and ISV OMPs, including:

- ▶ IBM Tivoli Provisioning Manager
- ▶ IBM Tivoli Configuration Manager
- ▶ IBM Tivoli Monitoring software
- ▶ Tivoli Business Service Manager
- ▶ BMC Remedy
- ▶ HP ServiceCenter
- ▶ Many other Tivoli and third-party operational management products

TADDM maintains the connectivity to (and relevance of) the source data through federation. That is, the database represents a logical aggregation of many real databases. Only certain inventory or asset management application attributes (typically, those attributes that are configurable and belong under change control) are actually populated in TADDM. However, there are many additional attributes that might need to be accessed at any given time. TADDM uses *IBM WebSphere® Information Integration technology* to obtain real-time access from source OMPs. This technology enables an organization to create a single, master view of business objects from disparate sources and augment TADDM with additional rich content.

Reconciliation

Combining multiple data sources into a single, logical view might create duplicate entries of the same configuration items (CIs) if the data was not reconciled. Typically, multiple OMPs manage the same CIs, but each application has its own

local data store and CI representation, which generates the possibility of inconsistencies and errors.

IBM wrote this reconciliation logic into the TADDM common data model for each CI type. Consequently, as the database imports data from management applications, it also compares the data using the reconciliation logic and corrects duplicate instances of the same CI.

Synchronization

One of the main challenges that IT organizations have with their existing approaches to managing change is the inability to manage and monitor configuration drift. TADDM addresses this challenge by allowing you to compare CI configurations with a “golden master” that reflects any approved changes. The comparison identifies where discrepancies from the master exist down to the attribute level.

The database also provides a reporting capability for monitoring where the *configuration drift* (the variance from your desired state configuration) is greatest, such as with applications, networks, or servers. Consequently, the IT organization can quickly understand the level of unauthorized change activity occurring in the environment.

2.1.3 Entities discovered by TADDM

TADDM discovers configurations and interdependencies across the following environment entities:

- ▶ Application components, such as Web servers, application servers, and databases
- ▶ System components, such as hosts, operating systems, load balancers, and database servers
- ▶ Network components, such as routers, switches, and firewalls
- ▶ Services, such as Domain Name System (DNS) and Lightweight Directory Access Protocol (LDAP) services

Table 2-1 Entities discovered by TADDM

Entity	Description
Network tier	<p>TADDM discovers the following devices at the network tier, along with the MIB2 (RFC 1213) parameter values for each device:</p> <ul style="list-style-type: none"> ▶ Routers ▶ Switches ▶ Load balancers ▶ Firewalls ▶ Generic IP devices
System tier	<p>TADDM discovers the following devices at the system tier:</p> <ul style="list-style-type: none"> ▶ Server hosts and disks ▶ Host IP interfaces ▶ Database servers ▶ Load balancers or clusters
Application tier	<p>TADDM discovers the following components at the application tier, along with version, configuration files and properties, host information, and vendor specific extensions for each component (except the generic processes):</p> <ul style="list-style-type: none"> ▶ Custom servers, based on custom templates that you design ▶ J2EE application servers and configurations ▶ J2EE and Java 2 Platform, Standard Edition (J2SE) components and modules ▶ Web server components ▶ Web modules, configuration files, and installation directories ▶ Generic Java virtual machine (JVM) processes ▶ Databases
Infrastructure service components	<p>The system infrastructure services that support the application environment are discovered along with the dependency structure between these service components and the application components. The infrastructure service components are:</p> <ul style="list-style-type: none"> ▶ DNS and Network File System (NFS) services ▶ LDAP

Entity	Description
Relationship structure	<p>In addition to the discovery of components, the physical and logical connectivity at the network, system, and application tiers is discovered at the following level of support in each of the tiers:</p> <ul style="list-style-type: none"> ▶ Layer 3 IP connectivity ▶ Layer 2 connectivity ▶ Application component runtime dependencies ▶ Infrastructure service dependencies

2.2 The TADDM discovery process

TADDM uses agent-free automatic discovery, together with a Data Center Reference Model to produce complete cross-tier dependency maps and topological views.

2.2.1 Discovery requirements

TADDM discovery requires discovery setup information:

- ▶ The discovery scope

Typically a valid IP range, subnet, or a specific address, the discovery scope signifies the span of the discovery process.

- ▶ Access Lists

Access Lists specify the read-only access credentials that are needed to discover and query the components for the appropriate configuration attributes and dependencies. The access mechanism varies based on the type of components discovered, for example:

- Simple Network Management Protocol (SNMP) community strings to discover the network elements
- Secure Shell (SSH) to discover the configuration and dependencies of the UNIX hosts or operating systems
- Windows Management Interface (WMI) to discover the Windows operating system and its applications
- Protocols, such as Java Management Extensions (JMX), SQL, Lightweight Directory Access Protocol (LDAP), and others, which are standard access mechanisms to discover application software

- Schedule

You can execute the TADDM discovery process on demand, as part of a schedule, or driven by events that are triggered externally.

2.2.2 The discovery process

As a high-level overview, when a discovery is initiated, the TADDM discovery engine proceeds through a multi-step, iterative process:

1. The discovery engine uses standard protocols to inspect the defined discovery scope to identify the IP nodes (addresses) of all installed devices.
2. The discovery engine deploys discovery sensors for each valid IP node in the scope. These sensors determine and collect the identity, attributes, and settings of each application, system, and network component.
3. The discovered objects are referenced against the Data Center Reference Model signatures for identification and classification.

This process is iterative—each discovery sensor executed can spawn other discovery sensors (for example, a host discovery triggers the discovery of applications and services that reside on the host) until the entire infrastructure is discovered.

After all sensors have completed processing, TADDM applies a heuristic-based algorithm to analyze the dependencies among the components that it has found. TADDM uses this analysis to create the topology maps.

4. Upon completion of discovery, TADDM processes the discovered component data to populate the Configuration Management Database (CMDB) and generates a graphical representation of the infrastructure topology at multiple levels: physical, application, and business service.

Subsequent discovery runs update the database and topology maps, while maintaining a comprehensive change history of the infrastructure configuration and dependencies.

5. The Product Console provides analytics and topological views of the CMDB.

2.2.3 Discovery sensors

Discovery sensors are small programs that collect data specific to the application for which they are written. These sensors also use protocols that are specific to the resources for which they are written, including:

- JMX
- SNMP

- ▶ Telnet
- ▶ SQL
- ▶ SSH
- ▶ Windows Management Interface (WMI)
- ▶ SAP Computer Center Management System (CCMS)
- ▶ SAP System Landscape Directory (SLD)

During discovery, a secure connection is established between the TADDM Server and the targets to be discovered. The sensor, emulating a user, logs on to the target system and runs commands that are part of the native OS. This data is encapsulated and sent back to the TADDM Server for analysis, and the sensor exits. The TADDM Server processes the returned data and potentially associates it with data from other sensors. This cross-analysis and comparison effectively discover the target resource and its relationships. Discovery is a multi-threaded process and usually occurs on multiple targets at the same time.

The primary job of a sensor is to discover CIs, create model objects, and then persist the model objects to the CMDB. As mentioned earlier, discovery execution is an iterative process. It begins with a *seed*, which is a combination of scope, credentials, and protocol. The process goes on to determine whether the initial target being discovered is a network device or a computer system. After the target identity is established, the discovery engine deploys more sensors to discover further information. These sensors discover with greater specificity the in-scope devices and applications that are running in the environment.

Note: Unlike many other discovery tools, TADDM only discovers *running* applications on a host as opposed to *installed* applications.

For example, if the target happens to be a computer system, the next sensor deployed will attempt to determine the operating system installed on the target. If the sensor determines the operating system, further OS-specific sensors are deployed to determine running applications and OS-level data (such as file systems, network devices, and so on) until there is nothing left to discover. At each step of the way, as more information is learned, the discovery engine often starts more specific sensors to discover these environments.

2.2.4 TADDM and secured environments

TADDM is designed to work in secure environments to enforce authentication and to protect confidential information. Each user must have a valid TADDM user account to use the Product Console to access discovered information about network and infrastructure components. You can use the Domain Manager user interface (UI) to configure user accounts.

When you select the “Establish a secure (SSL) session” option while starting the Product Console and logging in, TADDM encrypts all data (including user names and passwords) before it transmits the data over the network. When attempting to discover components in the environment, the TADDM Server uses SSH to securely communicate with all computer hosts and other devices that support SSH. The TADDM Server supports key-based SSH authentication and login-based, password-based SSH authentication. When login-based, password-based SSH is used, TADDM uses the user names and passwords that you define in the Access List to log in to the computer hosts to be discovered.

2.3 TADDM features

We describe several of the TADDM features in the following sections.

2.3.1 Auto-discovery

Discovery solutions typically use one of two paradigms to discover and collect configuration information. The first paradigm is to install a resident agent on the target system to be managed. The second method is agent-less, in which the management application actively probes the target system without installing agent technology. TADDM agent-free discovery engine manages the overall discovery process:

- ▶ Broad and extensible coverage across L2 through L7 components
- ▶ Deep cross-tier, runtime detail
- ▶ Comprehensive view of runtime dependencies across application, system, storage and network tiers, and supporting virtualized environments
- ▶ Fully automated application discovery

2.3.2 Open API and SDK

TADDM was designed to integrate into the modern data center environment. To facilitate rapid integration, TADDM provides a complete software development kit (SDK) that includes a full set of data, process, and event application programming interfaces (APIs). TADDM also provides predefined Discovery Library Adapter (DLA) integration with other management applications from IBM Tivoli, Micromuse® Netcool®, and other vendors. This integration allows leveraging of application topology data for automating and scaling other application management tasks, such as Service Level Management and IT automation.

2.3.3 Deep configuration detail

TADDM discovers details, such as OS patches, deployed application objects, and hundreds of configuration values. For example, in a typical 100-server application environment, TADDM captures values for over 27 000 individual settings. TADDM allows clients to rapidly isolate and track changes for all relevant information, reducing or eliminating change-related slowdowns and ensuring consistency and compliance.

2.3.4 Discovery profiles

TADDM incorporates the use of discovery profiles. A *profile* provides granular control over the discovery process. For example, certain sensors support multiple modes of discovery, such as shallow or deep discovery. Using a profile, sensors can be disabled or enabled, the discovery mode set, scope elements restricted, and more refinement of the discovery process can be established. Therefore, one profile can disable the DB2® sensors for a particular set of servers, and another profile might have the DB2 sensors enabled for another set of servers.

2.3.5 Credential-less discovery

TADDM also provides the ability to perform credential-less discovery for network-based IT asset discovery and categorization. You can use credential-less discovery to discover active computer systems in the runtime environment. This discovery, known as a level 1 discovery, discovers all computer systems in the scope that you designate in your environment. It also attempts to determine the operating system that is running in each computer system with a certain level of confidence. One use for this discovery information is to provide a starting point for establishing the credential list needed to further discover your environment.

Note: For more information about the levels of discovery and their use, refer to 2.2, “The TADDM discovery process” on page 22.

2.3.6 Change tracking

A single change can impact multiple components and sometimes the entire application environment. TADDM pinpoints the location of change in the infrastructure to expedite troubleshooting. In addition to identifying change, the “before” and “after” parameters or settings are available.

2.3.7 Secure interface

TADDM coexists with security implementations and policies to perform discovery. The Domain Manager UI provides a separate secure HTML interface that isolates TADDM administrative tasks and user access definitions from the main systems and application data in the Product Console. In addition, the Product Console is also accessible using SSH as discussed earlier.

2.3.8 Central viewing console for multifunctional teams

By providing a centralized view of the entire application environment, the Product Console eliminates the manual tasks of assembling and documenting complex configuration and relationship data from disparate functional groups. An overview topology gives users a quick understanding of which components deliver the business service. The infrastructure topology lets team members quickly navigate from the overview down to the component details to help localize where changes occurred. Multidisciplinary teams can share the same information to help with change tracking and troubleshooting.

2.3.9 Analytics

You can generate reports and perform analytics to monitor and troubleshoot configuration changes in your environment, such as:

- ▶ Identify changes in component configurations, perform component comparisons, and identify dormant components.
- ▶ Produce a consolidated summary of changes to a specific business application or throughout your entire environment.

TADDM provides several types of reports that contain detailed information about the components in an infrastructure and about the changes that occurred in components over a specified time period. After you create a report, you can sort the information in the report, edit the column layout, print the report, and save the report to various file formats.

When considering the inherent complexity in the original environment and the need for change to occur in a predictable manner, managing IT environments is a complex process. There is another consideration that often arises, which is the need to quantify what happened (reporting) and, further, to audit these reports. The effort to simply gather the information and certify its correctness is considerable, and it is a generally accepted truism that most data is obsolete as soon as the report is printed. There is, however, an increased awareness in today's IT world that there are significant auditing initiatives that must occur for

certain industries, and the penalties for incorrect data can include jail time. These audit initiatives are conducted for various end goals, such as ensuring:

- ▶ The security of sensitive patient data
- ▶ Access to critical applications is limited
- ▶ Code that is purchased is indeed running in the company's environment

Auditing is a function that needs to be quantified and automated as much as possible to reduce potentially disastrous results.

Inventory Report

The Inventory Report is a comprehensive infrastructure asset report that provides the following information:

- ▶ Hardware
- ▶ Software
- ▶ Entire infrastructure
- ▶ Business application
- ▶ Grouped by type:
 - Software, services, network, system, and other devices
 - Web servers, application servers, and clusters
 - Apache Web servers and Weblogic application servers

Use case

You can use the information in the Inventory Report to:

- ▶ Populate infrastructure into an Asset Management Solution or database
- ▶ Facilitate Sarbanes-Oxley Compliance reports

Change History Report

A Change History Report shows:

- ▶ A changed configuration attribute
- ▶ When the change was detected
- ▶ Old and new value of the attribute
- ▶ Type of change: Created, updated, or deleted

Use Case

You can use the information in the Change History Report to:

- ▶ Troubleshoot an application:
 - Order Management worked yesterday, but not today. What changed?
 - Only show changes to the infrastructure that might be related to the problem.

- ▶ Troubleshoot a component

I was able to ping the gateway from my virtual machine yesterday, but I cannot ping the gateway today. I wonder if there was a configuration change?

- ▶ Audit change in environment

Additional disk space must have been added to that server. Was it?

Component comparison

You can compare components after they are discovered:

- ▶ Comparison options:

- Deep comparison

Include all dependencies of the component in the comparison

- Include system

Include the host on which the software component resides

- Include services

NFS, DNS, LDAP, or AD dependencies

- ▶ Use cases:

- Ensure component consistency in cluster

- Ensure accurate staging to production application provisioning

Figure 2-2 on page 30 shows a sample comparison report where multiple versions of component configurations are compared and differences displayed. Notice that the number of CPUs and memory size for the missouri host differ from the oregon host (the standard).

Component Comparison: Results			
	oregon.lab.company.net:3880	indiana.lab.company.net:3880	missouri.lab.compa
Primary SAP			
Port Number	3880		9090
Config File			
Size	37457		35057
Checksum	gXRmPNd368MIOoICA2MWkA==	BpFYgBQ7Mxc3yqF58sr45A==	fWfCqXNhiAMJsa+
Product Version	Apache/1.3.26 (Unix)		Apache/1.3.27 (Un
Host System			
Num CP Us	1		2
File Systems			
Memory Size	1.5GB		2.0GB
Functions			
Router			
Default Route			
Next Hop			
Dot Notation	10.10.31.1		10.10.10.1
OS Running			
Model	SUNW,UltraAX-i2		SUNW,Sun-Fire-280
CPU Speed	500 MHz		900 MHz
Name	oregon.lab.company.net:3880	indiana.lab.company.net:3880	missouri.lab.compa
Product Name	Apache/1.3.26 (Unix)		Apache/1.3.27 (Un
Process Pools			
Containers			
Config Contents			
Httpd.conf			
Size	37457		35057
Checksum	gXRmPNd368MIOoICA2MWkA==		fWfCqXNhiAMJsa+
Modules			
Name	washington.lab.company.net	california.lab.company.net	illinois.lab.compan

Figure 2-2 A comparison that shows differences across configuration items

The tracking functions of TADDM provide the necessary capabilities to identify unauthorized changes to the infrastructure, which in turn help you streamline your processes and the accountability of the entire IT organization.

2.3.10 Versioning

TADDM provides versioning capability for discovered data.

Versioning discovery data

Versions are read-only views of the entire topology. It is a snapshot of the current infrastructure:

- ▶ Versioning data provides support for:
 - Analytics comparisons between versions
 - Disaster recovery planning
- ▶ Versions differ from change history:
 - Comparison of versions shows only the differences between two components, not all of the intermediate configuration changes.
 - Versions survive until the user deletes them.
 - Change history for a component is deleted along with the component, but version information is retained until the version is deleted.
 - Clearing the topology clears all components and the corresponding change history for those components, but it does not delete versions.

2.3.11 Summary of TADDM features

TADDM's flexible approach to discovery and automatic mapping of dependencies from layer 2 through 7 gives you the visibility and flexibility that are needed to improve service availability at your company. The information that is provided by TADDM supports the alignment of IT with business goals. TADDM's unique approach of combining agent-less, credential-free, and credentialed discovery provides control over where and what you discover and the depth of that discovery. TADDM allows you to continue to leverage your best practices for the business, but it also arms you with powerful data analysis so that you can make the best informed decisions for your organization.

IT organizations can:

- ▶ Ensure cost-effective and successful implementation of their Business Service Management initiatives
- ▶ Dramatically lower the business risks of service failures and inconsistencies
- ▶ Ensure compliance to technology and regulatory standards
- ▶ Reduce the time needed to resolve problems

2.4 Uses of TADDM

The following sections provide an overview of a couple of the common uses of TADDM.

2.4.1 Configuration management foundation

Today's applications are often built using reusable components to reduce turnaround time for developers. Furthermore, application infrastructure components are reused in their entirety (such as databases) to create composite applications. Although this reuse reduces the time to create large business services, at times these composite applications become unwieldy to track. A typical example illustrates the multiple components and computers that are involved in delivering a service:

- ▶ A Web server provides a user interface as the front end of the application.
- ▶ Another application, such as a J2EE server, provides a functional back end.
- ▶ A database provides information and records for use by the other components and stores the results of that work.

This combination of components results in multiple servers to maintain, presenting a problem in itself. You can often patch a small number (fewer than ten) of machines manually, but as the number of systems increases, it becomes desirable to have a repeatable, consistent method. This capability is usually accomplished through either automation through code or rigid oversight through process control. Implementing TADDM can provide the foundation for introducing Information Technology Infrastructure Library (ITIL) practices into an organization by providing accurate and timely infrastructure and application data.

2.4.2 Impact analysis visibility

In addition to the complexity of size and the number of components, there is a server consolidation trend, which means using multiprocessor machines to reduce the overall number of systems to manage and increase the CPU utilization of servers that might not have much load under the typical single-server/single-application architecture. However, server consolidation creates a new dilemma regarding the mappings of functions to a machine or an application. For example, several instances of databases or Web servers can coexist on the same host machine, and each instance is capable of performing separate critical tasks in the environment.

Using virtual machines is also a popular method of recovering CPU and memory that otherwise is wasted, but it can complicate systems management. Because it can be difficult to determine the effect of taking down a virtual machine or a host machine and the associated applications for servicing, it might lead to paralysis in the IT center, which can result in security and performance vulnerability as patches are delayed, applications are not upgraded, and hardware upgrades are not performed in order to meet increased capacity requirements.

TADDM provides visibility into the IT infrastructure relationships and dependencies to greatly enhance the impact analysis processes of change management in any environment.

2.4.3 Change management support

Change management process control is another area that often creates issues. The IBM Service Management products alleviate these issues and TADDM is a central part of this strategy. Change must occur in a production environment for many reasons, including security patches, operating system patching and upgrades, application introduction and retirement, or hardware and network upgrades.

The process for change management is filled with review meetings, back out procedures, risk assessment meetings, determination for which components are worked on directly, and which components are affected by the temporary unavailability of systems, as well as code-level compatibility after the change.

If these proceedings are supplied with accurate and consistent data, they lead to consistency in the change process and reduced disruption caused by change. In addition, clear records are kept to determine cause and effect in troubleshooting a problem or incident management if troubleshooting a problem or incident management become necessary. The application map, which is created by TADDM for business applications and services, highlights changed components. Using the capabilities of TADDM ensures that everyone has access to the same, updated information, which accurately reflects the state of the IT infrastructure.

Tivoli Application Dependency Discovery Manager architectural design

In this chapter, we describe the architecture and components of Tivoli Application Dependency Discovery Manager (TADDM).

In this chapter, we discuss:

- ▶ “Introduction” on page 36
- ▶ “TADDM architecture overview” on page 38
- ▶ “TADDM terminology” on page 46
- ▶ “eCMDB” on page 54

3.1 Introduction

Today's IT organizations are under enormous pressure to deliver high levels of service with agility and efficiency. At the same time, they often find themselves in reactive mode, relying on inadequate tools and manual processes to solve problems that exist in an environment of rapidly increasing complexity and change. The complexity introduced by component architectures, such as Java Two Platform, Enterprise Edition (J2EE) and .NET, increased virtualization of software, OS, and networking layers, as well as ever increasing business demands, means that having a shared understanding of how all of the IT infrastructure components are related and configured to actually deliver business applications is critical to IT success¹.

Leading IT organizations are now evaluating and implementing solutions to this problem using application mapping products, such as TADDM. TADDM delivers a shared understanding of the components, dependencies, and configuration of critical business applications. This visibility is critical in order to:

- ▶ Meet service level commitments
- ▶ Eliminate unanticipated change-related problems
- ▶ Reduce problem resolution times
- ▶ Enforce technology consistency
- ▶ Process conformity and compliance policy
- ▶ Enable agile and responsive change

To achieve the necessary levels of required visibility, a solution must:

- ▶ Provide a fully automated and accurate application-centric view into the runtime structure of business applications, including their software and hardware components, their cross-tier dependencies, and their configurations.
- ▶ Store these application maps and all of their associated data in a well-defined application maps database that is:
 - Comprehensive enough to cover all common runtime data center components, yet flexible enough to meet the specific needs of each implementation
 - Detailed to include deep, runtime configuration and dependency information

¹ Refer To Gartner (Colville) "Organizations Are Paying More Attention to Configuration Management", 31 March 2005; and Enterprise Management Associates, "The ITIL Configuration Management Database: Panacea or Pandora's Box?", December, 2004.

- Easily accessible and understood by operations personnel
- Programmatically sharable with other management and enterprise applications
- ▶ Scalable to enterprise levels
- ▶ Able to provide its value rapidly, securely, and with low overhead
- ▶ Capable of deploying and working securely, consistent with enterprise security policies and infrastructure

The high-level product requirements in the previous list drove the design and architecture of TADDM. In order to meet these requirements, five critical design decisions were made in the architecture of TADDM:

- ▶ Build the product around a predefined, application-centric, standards-based, extensible reference model, which eliminates the need for custom modeling with every implementation.
- ▶ Choose an agent-free approach to eliminating the performance and security risks and the implementation and qualification expenses of deploying a new set of management agents.
- ▶ Publish a well-defined schema with a complete set of documented data and process-level APIs to enable rapid integration into existing management products and processes.
- ▶ Provide the ability to federate and extend the application maps database in order to allow the product to be deployed at an enterprise scale.
- ▶ Utilize existing secure protocols and standards to make implementation secure and straightforward.

The combination of these key design decisions translates into a solution that provides complete visibility into how the infrastructure delivers applications and services, can be securely and rapidly deployed, is easily integrated, scaled, and extended, and whose overall cost of ownership is the lowest possible. Solutions lacking in any of these areas face time and cost obstacles in successfully providing the visibility required to manage and improve service delivery.

In this chapter, we provide a technical evaluator with a detailed description of the TADDM architecture and how the product works to deliver complete infrastructure visibility. It includes:

- ▶ An overview of the TADDM architecture
- ▶ A description of how the TADDM product creates and maintains its application maps
- ▶ An overview of the Enterprise Configuration Management Database (eCMDB) architecture and functions

By deploying TADDM, organizations not only improve application availability and lower cost today, but they lay the groundwork for building a truly scalable IT service management capability. The data and knowledge that TADDM creates is a key building block not only to improving service delivery and management, but to automating and enabling capabilities, such as dynamic application provisioning, measurable and enforceable Service Level Agreements (SLAs), and effective IT Governance. In the remainder of this chapter, we focus on how TADDM is architected and works to deliver this capability.

3.2 TADDM architecture overview

In this section, we describe each element of the TADDM architecture. Figure 3-1 illustrates the TADDM architecture.

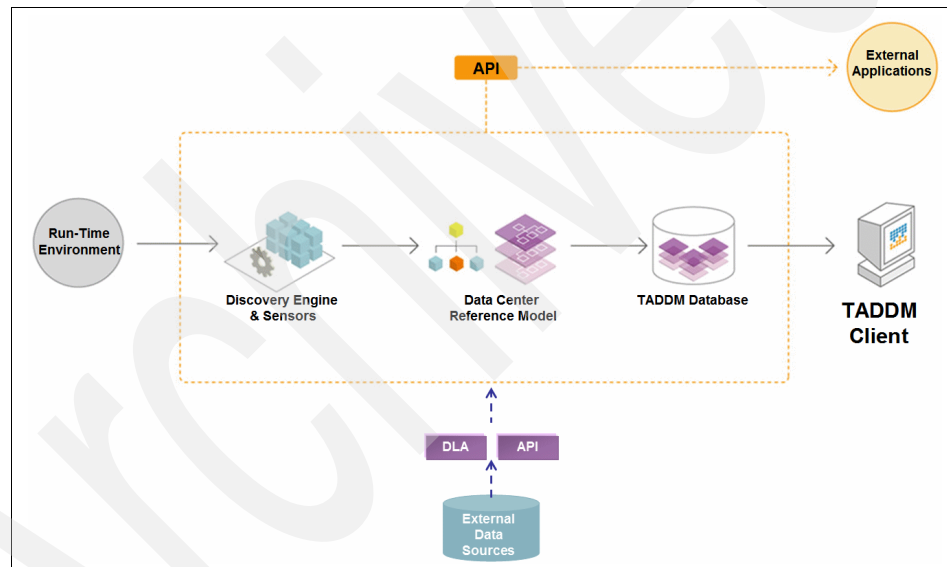


Figure 3-1 TADDM architecture

IBM Common Data Model (CDM)

A *data model* is a conceptual representation of the structure of the data that is stored in the database. In terms defined by the IT Infrastructure Library (ITIL), the data structure consists of the contents of the data objects or Configuration Items (CIs), the associations between data objects, and other possible rules to constrain the operations of the data objects. Without a reference data model, implementation is dependent on expensive, time-consuming, incomplete, and error-prone manual modeling. The TADDM Database data model is based on the

Common Data Model (also referred to as the CDM), which is an information model that can integrate data between Tivoli Management Products. The details of the CDM are documented with Unified Modeling Language (UML). The data model is implemented with Java persistent objects that allow Java programmers to manipulate objects and relationships independently of the physical storage.

Agent-free discovery engine

The discovery engine orchestrates and manages the discovery process. The engine instructs the discovery sensors, which are server-based components that use the knowledge that is built in the reference model to interrogate the data center components and gather the necessary information to build the application maps database.

Topology manager and topology builder

Upon completion of the discovery process, the topology manager consolidates the discovered data and generates a cross-tier topological representation of the application. The application topology includes its underlying infrastructure components (software, systems, and network components) and their associated configurations and cross-tier dependencies.

TADDM Database (CMDB)

The TADDM Database is a cross-tier representation of the application topology, its components, and their configurations. The Configuration Management Database (CMDB) also tracks and documents all configuration changes. The database is optimized for both read and write access and supports extensive query capabilities.

TADDM API

TADDM's open and published API interfaces easily with third-party ecosystem applications. The TADDM API provides authenticated and secure access to the underlying TADDM Database through the Data API and the TADDM process engines through the Control API. TADDM also provides an event API to import and export events to other management applications.

Discovery Library Adapter

The Discovery Library Adapter (DLA) provides an open standards-based interface to integrate data from older data stores into the TADDM Database.

3.2.1 TADDM architectural details

In this section, we provide the details for each of the major architectural components by:

- ▶ Describing both the rationale for design and an overview of implementation.
- ▶ Explaining how the pieces work together to populate the application mapping database and create and maintain application maps.

IBM Common Data Model (CDM)

This model represents the foundation of TADDM and provides the definition for the data center applications and their supporting infrastructure components, cross-tier relationships, and configuration attributes. Without a reference model, implementation is dependent on expensive, time-consuming, incomplete and error-prone manual modeling. TADDM provides definitions for a wide variety of commonly deployed software applications, hosts, network devices, and network services. The extensible reference model includes an event propagation model that provides the underpinnings to interpret infrastructure component events in the context of the applications that they deliver.

The CDM is based on the Distributed Management Task Force (DMTF) Common Information Model (CIM) object model. You can find more information at the following Web site:

<http://www.dmtf.org>

For platform-specific extensions, such as JSR773, access the following Web site:

<http://www.jcp.org/en/jsr/detail?id=77>

The CDM includes a wide variety of object types, including:

- ▶ Software components (Web, application, and database servers)
- ▶ Hosts and operating systems
- ▶ Network elements (routers, switches, load balancers, firewalls, and storage)
- ▶ Network services, such as Lightweight Directory Access Protocol (LDAP), Network File System (NFS), and Domain Name System (DNS)

The CDM is easily extensible based on client-specific needs.

The model representation for each component type includes:

- ▶ Signature

The signature uniquely identifies the component type and its dependencies and configuration template.

► Configurations

Configuration data elements include:

- The static and the dynamic configurations of the component.
The runtime resources that the component uses (for example, the Java Database Connectivity (JDBC) connection pools or the Java Message Service (JMS) topic queue that an application server uses, the patches deployed on an OS, or the IP routing table of a network element)
- The deployed application objects (for example, the Enterprise JavaBeans (EJBs) and JavaServer Pages (JSPs) on an application server that implement the business application and services)

► Dependencies

Dependencies model the runtime relationships among the various components within the data center. TADDM discovers and categorizes several types of cross-tier dependencies, including:

- Transactional dependencies
Transactional dependencies are the logical connections (IP-based) between the components of a distributed application. These connections represent the provider-consumer relationships between the components, for example, an application server is the consumer of a service provided by a database server.
- Containment dependencies
Containment dependencies are the cross-tier hierarchical relationships (for example, an application server is deployed on a host), as well as logical grouping relationships (such as a Web, application, and database server that make up a business application).
- Service dependencies
Service dependencies are the network services upon which most infrastructure components depend (NFS, DNS, and LDAP services).

Figure 3-2 on page 42 is an example of the Common Data Model specification. This example illustrates the *ComputerSystem sub-model* that defines the key object for Tivoli Management Products. In the Common Data Model, a computer system is a combination of hardware (the machine) and software (the operating system) and is flexible enough to allow the representation of various combinations of hardware and software. In the ComputerSystem sub-model, both the computer system and the operating system are considered combinations of interesting entities. As a result, both the Common Data Model specification and the ComputerSystem sub-model are derived from the superclass System. There are also linkages from this model to the Process sub-model and the FileSystem sub-model. Operating system processes are

considered specific parts of an overall business process. One important concept that is included in this model is the representation of the OperatingSystem sub-model as a place where software can run a hosting environment or an interface. This concept is also applied later to other hosting environments, such as WebSphere.

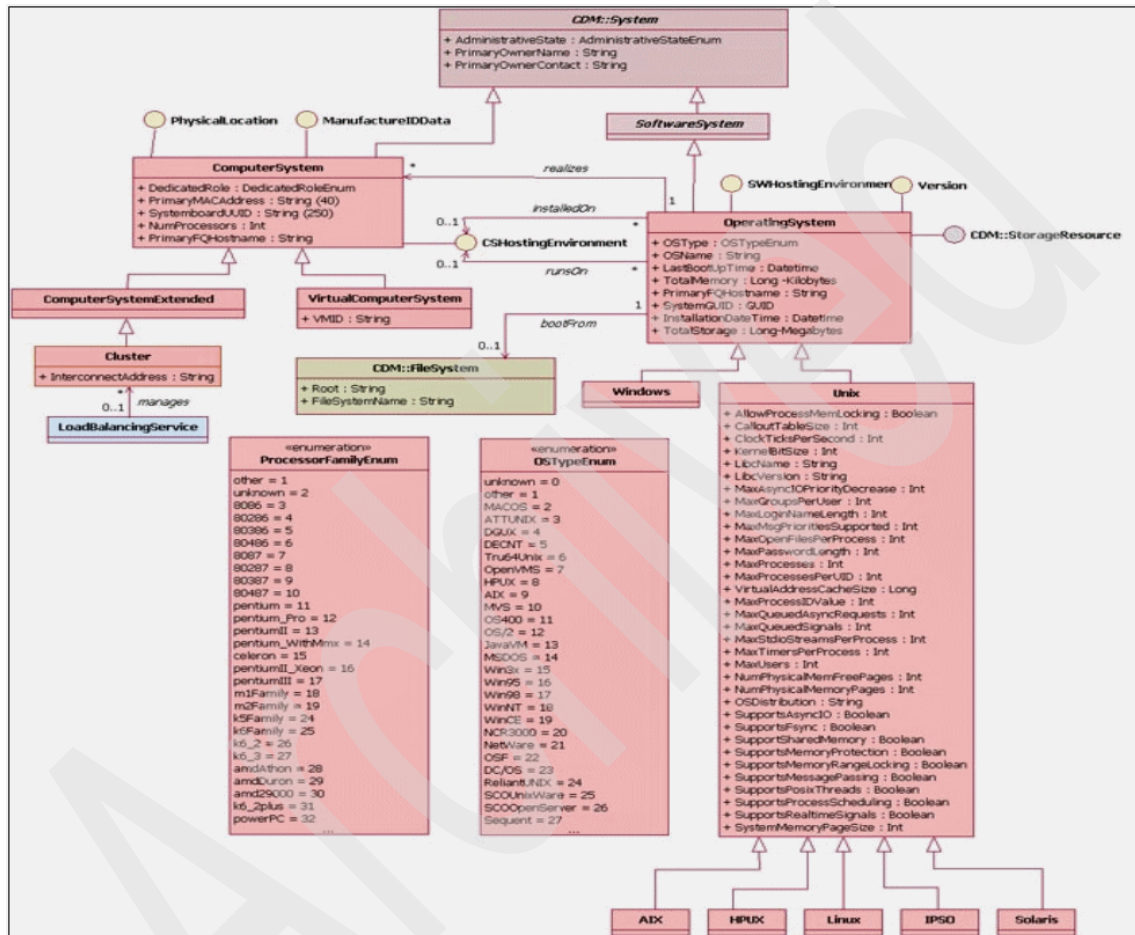


Figure 3-2 TADDM data model

Data model extensibility

The IBM Common Data Model can be easily extended in the field. Through the Java Control Console or the Domain Manager user interface, you can easily add extended attributes to existing object classes. Also, you can either use the GUI or the API to associate values to the extended attributes, and you can view the populated results through the GUI.

Figure 3-3 illustrates a computer system with extended attributes.

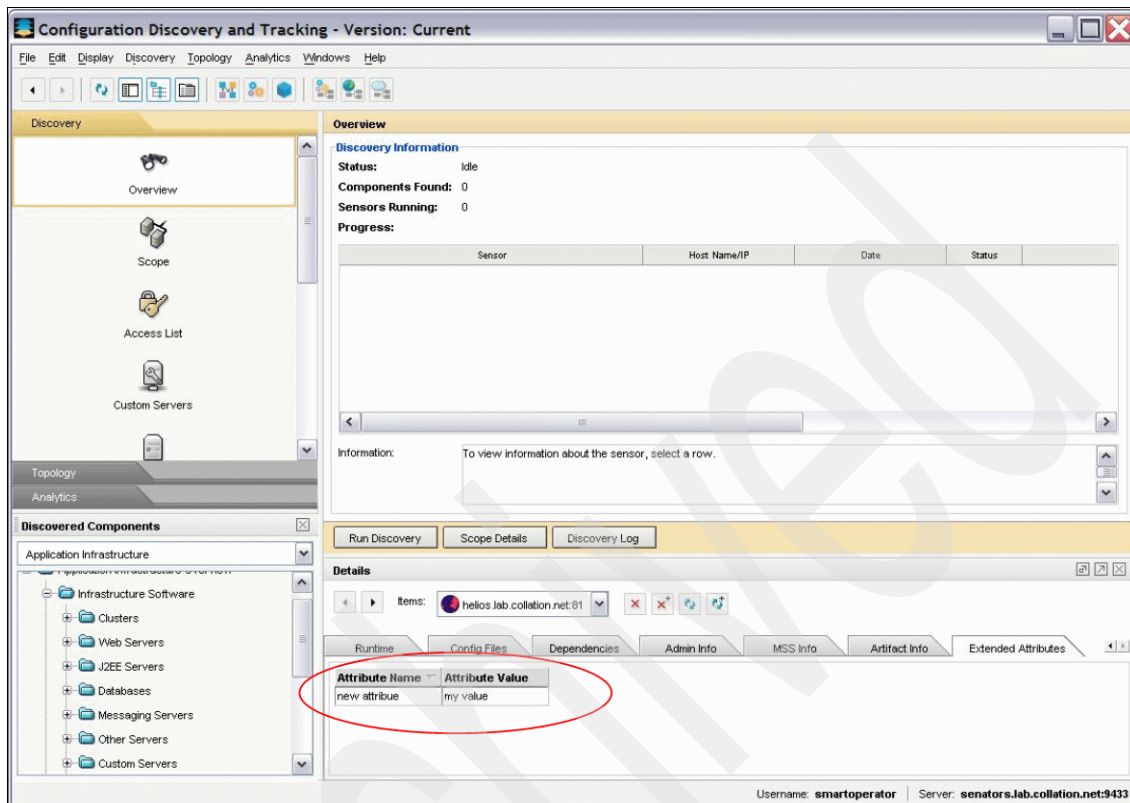


Figure 3-3 Adding extended attributes in TADDM

Agent-free discovery engine and process

The discovery engine orchestrates and manages the discovery process. It also instructs the discovery sensors. TADDM's agent-free discovery engine manages the overall discovery process. The discovery process collects the data that is needed to instantiate the Data Center Reference Model or (Common Data Model) to represent the specific data center infrastructure. At the center of the discovery process are lightweight discovery sensors that build upon the Data Center Reference Model to comprehensively discover the infrastructure components, configurations, and dependencies. Discovery sensors use open and secure protocols and access mechanisms to discover the data center components. Furthermore, unlike persistent and invasive agents, discovery sensors are centrally deployed and managed and consume minimal bandwidth and CPU resources (less than 1% when active) in the target environment. The discovery engine provides a workflow framework to schedule, distribute, coordinate, and manage the various discovery sensors.

For a description of the discovery process, refer to 2.2.2, “The discovery process” on page 23.

3.2.2 Discovery extensibility

IBM provides the broadest available discovery coverage at installation of any solution on the market. However, every data center has unique or new requirements beyond those built into the TADDM application. To meet the specific needs of each implementation, you can extend TADDM's discovery coverage through:

- ▶ Custom server templates

TADDM discovers all running software processes as part of its standard discovery. These discovered processes can subsequently be identified and categorized by their runtime signatures, which allow them to participate in configuration, change tracking, and business service discovery. You can create these custom server templates through the TADDM User Interface and APIs.

- ▶ Sensor extensibility

You can extend the runtime behavior of a sensor. You can also code custom logic (scriptable in Jython), which securely runs within the TADDM Database sensor sandbox architecture and allows sensors to capture additional attributes about the infrastructure. For example, using the scripting capability, you can write a custom script that runs within a sensor to read or parse a custom configuration data file to create new dependencies that the sensor might not ordinarily discover.

3.2.3 TADDM APIs

TADDM provides an open and published API to easily interface with third-party ecosystem applications. The TADDM API provides authenticated and secure access to the underlying TADDM Database (through the Data API) and the TADDM process engines (through the Control API). TADDM also provides an event API to import and export events to other management applications.

The TADDM API provides a secure and modular interface to the TADDM Database. The open and published API provides bindings, such as Java, Web Services or SOAP, and shell scripts. TADDM enforces authenticated API access, and upon authentication, all API clients are assigned access privileges that authorize user actions. API access can also be encrypted through Secure Sockets Layer (SSL) to ensure maximum security.

The API capabilities include:

- ▶ *Data APIs* provide access to the full application topologies, including the components and their detailed configurations, as well as their runtime dependencies. The Data APIs also provide access to the TADDM change analytics and reporting. In addition, the Data APIs allow for the import and storage of additional data that pertains to discovered components, such as asset, financial, and administrative information.
- ▶ *Control APIs* provide asynchronous access to the TADDM discovery processes including discovery setup, as well as the discovery scheduling and control. Using the Control APIs, third-party solutions can control the TADDM Server, including initiating and aborting discovery runs.
- ▶ *Event APIs* allow you to import third-party events into the TADDM application topologies and export change and state events from TADDM to third-party consoles and products.

Using the APIs, IBM provides integration to several leading ecosystem vendors, including Micromuse, Compuware, BMC Remedy, HP Operations Manager Software, and others. The TADDM software development kit (SDK) includes sample integration code and integration tools that allow Java and XML-proficient programmers to easily implement custom integration in the field.

3.2.4 Discovery Library technology

Because the CMDB becomes the central repository for configuration data for your IT components, it is likely that you will want to load data into the CMDB that is already known and maintained by other management solutions. For example, you might want to load inventory data from Tivoli Configuration Manager or monitoring probe data from IBM Tivoli Monitoring. You also might want to extract information from the CMDB to load into operational systems management solutions (such as Tivoli Business Systems Manager or Tivoli Provisioning Manager) to reflect the most up-to-date configuration of the IT environment. For these purposes, TADDM supports the IBM Discovery Library technology that is based on the Identification Markup Language (IDML) industry standard format.

For each subsystem that supplies or consumes data, you can download a Discovery Library Adapter from the IBM Open Process Automation Library (OPAL) at:

<http://catalog.lotus.com/wps/portal/topal>

IBM provides access to the DLAs for many IBM and independent software vendor (ISV) applications.

3.3 TADDM terminology

The following section provides an overview of the TADDM terminology.

3.3.1 TADDM Server (Domain Manager)

A *domain* is a logical subset of a company's infrastructure. It can be based on any convenient boundary criteria, such as geographical or departmental.

Domains can delineate organizational (for example, IT operations or line of business (LOB) operations), functional (for example, finance IT domain or human resources IT domain), or geographical (for example, United States IT domain or Brazil IT domain) boundaries. There is typically one TADDM Server deployed per domain to discover domain applications and their supporting infrastructure, configurations, and dependencies.

The TADDM Server is also referred to as the *Domain Manager*. You might want to use one or multiple TADDM Servers (Domain Managers), depending on how many configuration items exist in your environment. If you have a large environment with thousands of hardware, software, and other configuration items, one Domain Manager might not be enough, and you can expand to multiple Domain Managers. You can have a view of all of the aggregated data at an Enterprise Configuration Management Database (eCMDB) Domain Manager. In an eCMDB environment, the participating Domain Managers perform the discoveries and supply data to the enterprise Domain Manager. Figure 3-4 on page 47 illustrates a setup with multiple TADDM (Domain Manager) servers.

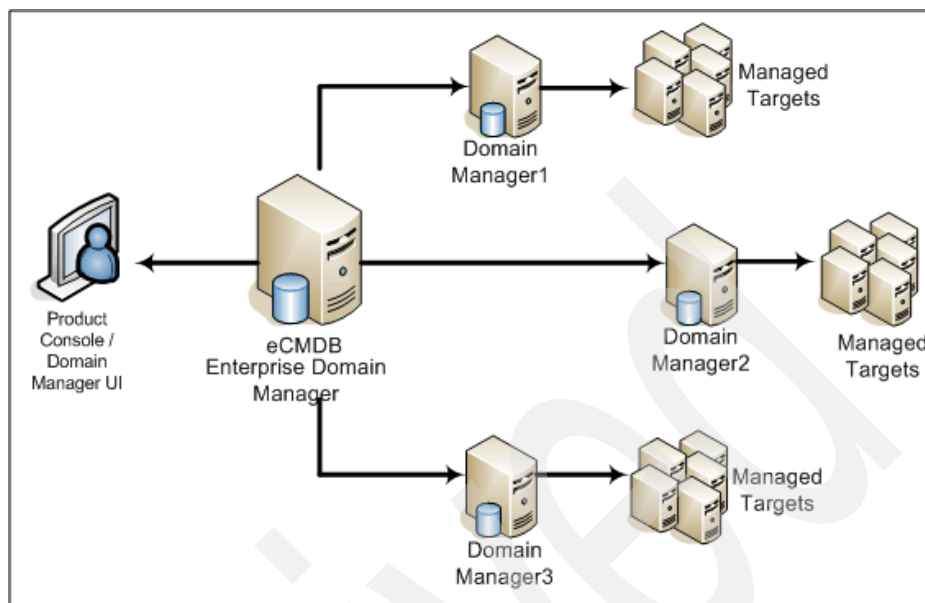


Figure 3-4 Multiple Domain Servers

The enterprise Domain Manager aggregates the CI information from the Domain Servers. This behavior is configurable depending on the needs of an organization. Each Domain Manager provides a Domain Manager UI to manage TADDM users and query TADDM discovered data.

Configuration of the eCMDB is discussed in 3.4, “eCMDB” on page 54.

3.3.2 TADDM user interface

TADDM provides two user interfaces:

- ▶ **Java Product Console:** The TADDM Java Interface provides command and control of the TADDM Server, advanced topology visualization and management, and a set of change and configuration analytics.
- ▶ **Domain Manager UI:** The Domain Manager UI is a Web portal interface into TADDM. This Web-based client provides a sharable interface that you can embed in the TADDM Database.

Java Product Console

Figure 3-5 on page 48 illustrates the TADDM Java Product Console displaying the business applications topology.

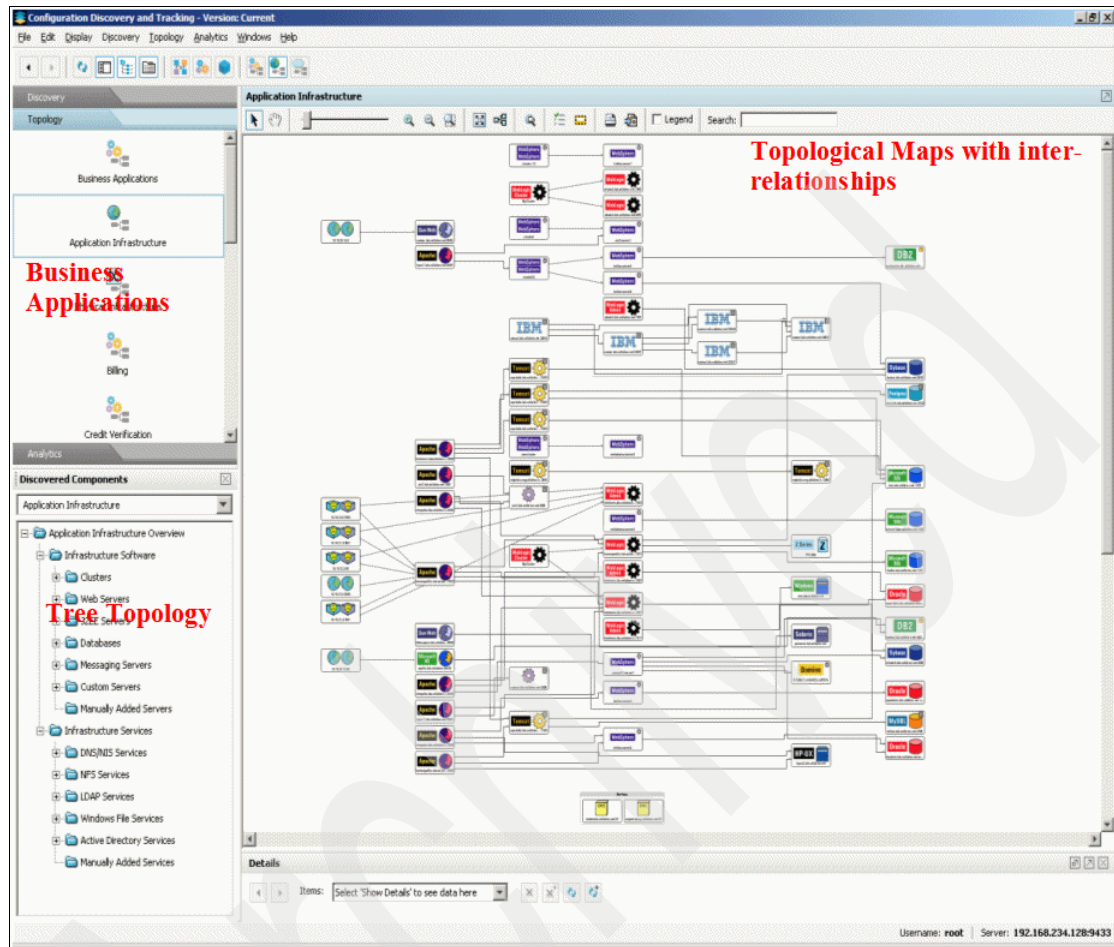


Figure 3-5 TADDM Java Product Console

The Java Product Console offers an advanced, task-oriented, graphical user interface that enables you to manage your computing environment by performing a series of proactive management, problem identification, and troubleshooting tasks.

Table 3-1 outlines the key uses of the TADDM Java Product Console.

Table 3-1 TADDM Java Product Console uses

Use	Description
Deployment planning and management	The TADDM infrastructure comparison feature provides a fast and easy method for comparing components or sets of components between the staging and production environments. You can run a pre-deployment comparison to compare staging with production to understand or verify planned changes, or you can run a post-deployment comparison to verify that staging and production are in sync.
Consistency checking	You can use TADDM comparisons to compare a set of components against a known good set of components.
Troubleshooting to diagnose problems across all tiers	<p>You can quickly diagnose problems by using the topology map to identify and analyze cross-tier relationships among software and hardware components. For example, suppose you become aware that increased user requests to a business application are processed slowly. You can select the impacted business application that you want to troubleshoot, display the topology for the application, and show the dependencies among the application's components.</p> <p>You can also schedule a discovery to run at a specific time and gather runtime configuration data. Following this discovery, you can run a Change History report to see what changed.</p> <p>The TADDM Change History report enables you to quickly detect unauthorized changes in the runtime environment and validate configuration parameters in the production environment before users are affected. You can use this report to verify production configuration parameters.</p>
Proactive change management	After deploying an update to the production environment, run a discovery process on the production environment, and save the topology. You can then open both the production and staging versions and compare the discoveries of the environments.
Verify production configuration parameters	Prior to deploying application updates or changes at any level in the infrastructure, organizations typically test the change in the staging environment. After you verify that the changes in the staging environment are working correctly, you can run a discovery process of the entire network where the staging environment resides.

Use	Description
Schedule discovery to detect unauthorized change parameters	You can schedule a discovery process to run at specified times, which enables you to gather and store the configuration and the topologies of the runtime environment. Later, you can create a Change History report to detect changes in the configuration of the components, such as application servers or database servers and parameter changes in firewall and user applications.
Asset management and data center consolidation	Complex business environments result in constant changes to the application components and their underlying hardware devices. TADDM inventory reports help you to make operation management decisions and answer capacity questions, such as: <ul style="list-style-type: none"> ▶ How many applications are running on a particular server? ▶ How many instances of a Web server or application server are running to service my order entry application? ▶ Do I have a sufficient number of licenses and are they being deployed efficiently? ▶ I want to consolidate my data center. What is the total number of hardware and software components being used in my data center after a merger, acquisition, or reorganization?
Business continuity planning	Mission critical applications are often mirrored in a backup data center to ensure business continuity in the event of a disaster at the primary site. The TADDM comparison feature enables rapid and thorough comparisons of two installations to ensure that the backup site is configured exactly as the primary site is configured.

Domain Manager UI

Figure 3-6 on page 51 shows the TADDM Domain Manager user interface.

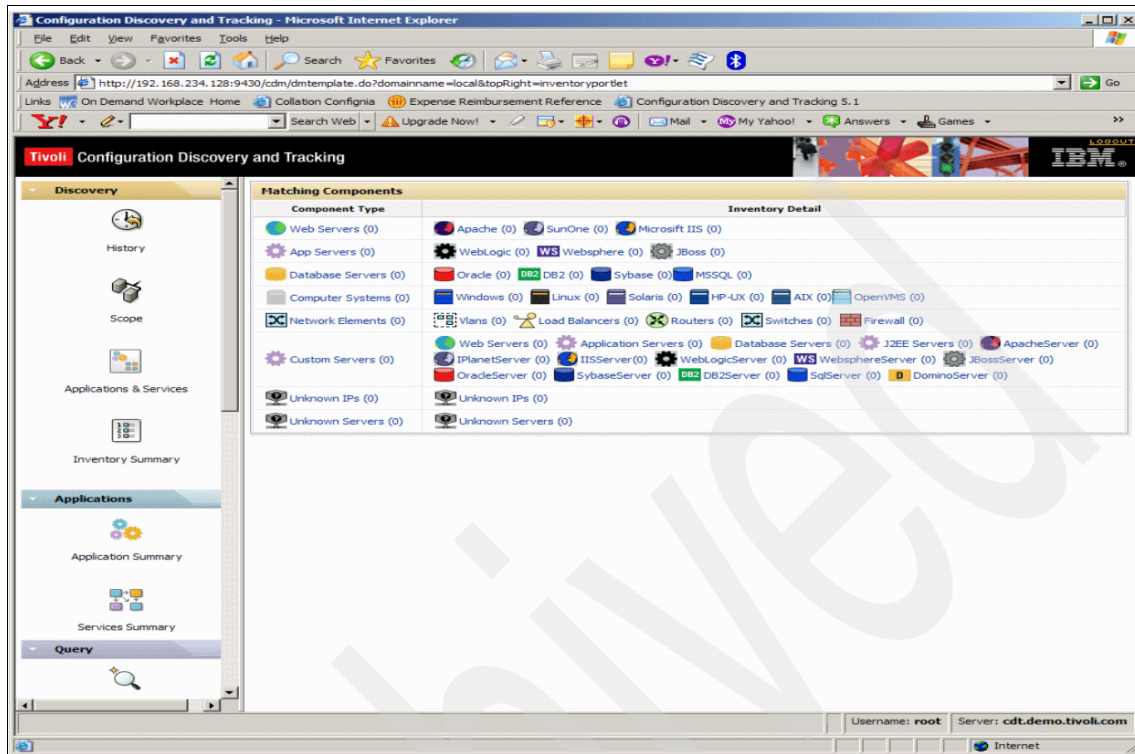


Figure 3-6 TADDM Domain Manager UI

The Domain Manager user interface is a lightweight, Web-based interface. It provides the following features:

- ▶ A view of data for your TADDM Server (if single domain) or an aggregate view of data from multiple TADDM Servers in the case of an Enterprise Domain Manager
- ▶ Extensible Web-based console with factory-supplied enterprise analytics and a framework to easily create and add custom reports
- ▶ Open and proven TADDM APIs leveraged to integrate with external management applications (Launch in Context)
- ▶ Launch in Context provides a view of:
 - Component details
 - Changes to a component

The capabilities and limitations of the Domain Manager UI are that it:

- ▶ Allows read-only access to the aggregate information
- ▶ Can report on individual domains or on a set of domains
- ▶ Does not resolve inter-domain dependencies
- ▶ Cannot initiate discoveries, only provides a view of already discovered data
- ▶ Cannot edit components
- ▶ Cannot change the scope or the Access List
- ▶ Does not show a graphical view of the topology

3.3.3 TADDM Database

You must install the TADDM Database on a separate machine. Each discovery server, whether an enterprise discovery server or a domain discovery server, includes a database that holds the discovered information about configuration items and their relationships.

To install the TADDM Server, the database must be installed and running on the other system. During the installation of TADDM, connectivity to the database is established to populate the database.

3.3.4 Anchor servers and Windows gateways

Depending on your network, you might need to deploy one or more anchor servers and Windows gateways in order to facilitate discovery.

Anchor servers

In order to discover components, each TADDM Server must be able to communicate with other computer hosts and network devices. In cases when a firewall prevents direct access from the discovery server to certain hosts or devices, you can specify a computer system that does have access to the hosts or devices to be an anchor host. An *anchor host* acts as a proxy to assist in the discovery process.

You do not need to configure anchor hosts during the TADDM Server installation process, but you need to include anchor hosts in your installation plan and verify the system requirements for candidate machines.

The TADDM Server, by default, runs a local anchor process. After the TADDM Server installation, you can use the TADDM Product Console to configure which hosts will serve as additional anchor hosts on your network.

Anchor requirements and considerations:

- ▶ You can use any operating system on which the TADDM Server can be installed as an anchor. Refer to “Planning your hardware and software” on page 65 for complete details about supported operating systems and prerequisites.
- ▶ SSH software to communicate to the central TADDM Server. You need to install Bitvise Version 4.06a or later or Cygwin SSH if you plan to use a Windows anchor server.
- ▶ Network connectivity must exist to the remote servers within the discovery scope in the firewall zone.

Windows gateways

If your network contains Windows-based systems and applications that are running on Windows, you must specify a Windows system to serve as a gateway server in order to discover information about the Windows-based systems running in your environment. The gateway server needs to be in the same firewall zone as the Windows hosts to be discovered. The TADDM discovery server must be able to SSH into the Windows gateway.

You do not need to configure Windows gateways during the installation process of the TADDM Server, but you need to include gateways in your installation plan and verify the system requirements for candidate machines. After the installation of the TADDM Server, you can use the TADDM Product Console to configure which hosts will serve as Windows gateways on your network.

Note: Windows Management Interface (WMI) must be available on all Windows hosts that will be discovered. WMI is available by default on Windows 2000 systems and higher.

Gateway requirements and considerations:

- ▶ Windows Server 2003, service pack 2.
- ▶ All Windows gateways must be running either Bitvise WinSSHD 4.06a or later, or Cygwin SSH. If you use Cygwin SSH, take all of the defaults, cygrunsrv from the admin category (Version 1.17-1 or later), and opensshd from the net category (Version 4.6p 1-1 or later).
- ▶ The TADDM Server communicates with the gateway using SSH regardless of the platform that the server is using.

Note: An anchor and a gateway running on the same Windows system are supported.

3.4 eCMDB

In this section, we provide information about the Enterprise Configuration Management Database (eCMDB) and about when you need to consider implementing an eCMDB.

3.4.1 eCMDB overview

TADDM is designed to modularly scale to large data centers. You can also tune TADDM operating characteristics (discovery engine thread counts, discovery sensor time-outs, and so on) and increase the TADDM Server or TADDM Database resources (CPUs, memory, and so on) to achieve increased support for infrastructure discovery and storage.

Figure 3-7 illustrates how you can deploy TADDM in large enterprise deployments.

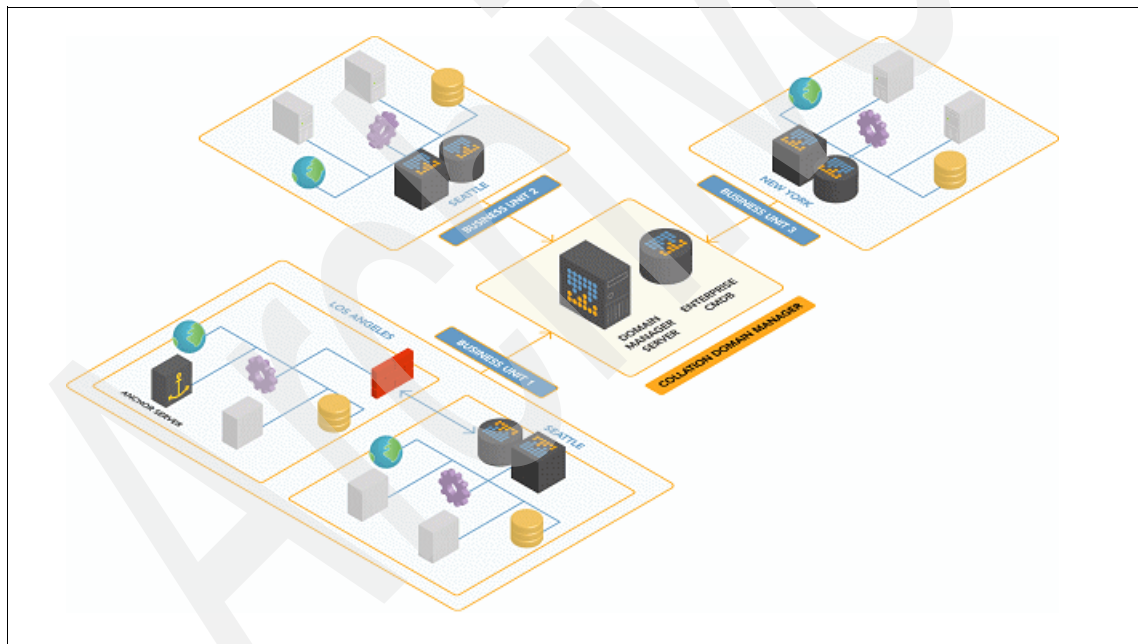


Figure 3-7 TADDM deployment using eCMDB

Although it is possible to scale TADDM Servers to support large enterprise environments, IBM offers a domain-based, best-practice deployment architecture to elegantly scale the solution to support several tens of thousands of infrastructure elements.

Most large enterprise environments are divided into management domains that represent the span of control of a given IT operations team. Domains can be based on organizational, functional, and geographic boundaries or combinations of these boundaries or other criteria. To support the operational needs of the domain, IBM recommends a standalone TADDM Server for each management domain. Each TADDM Server is responsible for its own domain—it discovers and stores all configuration data for its local domain. Users of each management domain use the local TADDM instance to manage the operational aspects of their domain.

However, IT organizations also have the need to have cross-domain views of their IT information, for example, the CIO might want to see an aggregate count of the enterprise-wide Oracle deployments to ensure that the enterprise is compliant with its licensing contracts. To address this capability, IBM provides the TADDM Enterprise Domain Manager. The TADDM Enterprise Domain Manager federates data from multiple local TADDM Server instances to provide a rolled-up single enterprise-wide repository of information.

The Domain Manager provides a Web-based portal to administer the local domain servers and to view and analyze the rolled up enterprise data. It also provides a query and reporting interface that you can customize, which allows data to be easily shared across the enterprise.

The following list overviews what an eCMDB does and what an eCMDB does not do. The eCMDB:

- ▶ Does maintain change history
- ▶ Does provide bulk load and import capability
- ▶ Does provide cross-domain query capability
- ▶ Does provide a common security framework across domains
- ▶ Does provide the same API for access to data across domains
- ▶ Does merge attributes when data is retrieved from the CMDB database with data in eCMDB taking precedence over attribute values currently on domain (to allow for bulk loads)
- ▶ Does *not* perform discovery or build topology
- ▶ Does not allow a domain to belong to more than one eCMDB
- ▶ Does not allow nesting of eCMDBs

3.4.2 eCMDB synchronization

The process of synchronizing data from the domain servers in an eCMDB implementation is analogous to the discovery process at the domain.

When the discovery is completed at a domain server, the eCMDB server will synchronize the data from that domain. As with discoveries, only one synchronization can be running at a time. Also, if discovery is underway at the domain server, synchronization with that domain server will fail. Many organizations run discoveries during production hours (to get a realistic view of the use of the systems) and run synchronization after hours to avoid this conflict.

Note: For further explanation about setting up eCMDB domains and synchronization, refer to Chapter 5, “Tivoli Application Dependency Discovery Manager installation steps” on page 89.

3.4.3 eCMDB database

In this section, we provide an overview of the eCMDB database:

- ▶ The eCMDB database has the same schema as the remote domains.
- ▶ The eCMDB database requires TADDM version compatibility between the installed domain and the eCMDB.
- ▶ You can bulk load or import data into the eCMDB database. There is no write-through of data to the domain servers.
- ▶ Synchronized data is a copy of the remote domain data and additional attributes that include but are not limited to:
 - All housekeeping attributes for each CI, such as lastModifiedTimestamps, version, and so on
 - All select criteria required for reports on the eCMDB portal and for inventory views on the eCMDB portal
 - All the relationship tables
 - Change history information from the domain
 - Authorization information, such as user roles and permission

3.4.4 eCMDB security

The following list contains the security changes that are applicable when you configure an eCMDB in your environment:

- ▶ When a domain is added to an eCMDB, you need to recreate any existing users in the domain (assigned roles and granted access to access collections) in the eCMDB. When you add a domain to the eCMDB, authentication and authorization for the added domain is delegated to the eCMDB:
 - Logins to the domain CMDB are now processed at the eCMDB.
 - Any security manager method calls are now processed by the eCMDB.
- ▶ Active domain sessions (Java console and Domain Manager) require you to log in again.
- ▶ Users, roles, and permissions that are viewable on the Domain Manager are now those users, roles, and permissions from eCMDB. Users that you created using the Domain Manager now appear automatically in the eCMDB's Domain Manager, because they are actually being created there.
- ▶ Roles, permissions, and access collections are stored in the CMDB and are synchronized from the domain CMDBs to the eCMDB just as any other objects; however, their associations are not synchronized.
- ▶ Users are not synchronized to eCMDB.
- ▶ LDAP is the preferred method of authentication for eCMDB setups, so passwords are stored in a single place.
- ▶ Access collections cannot span CMDB domains.
- ▶ When a domain loses connectivity to eCMDB, security falls back to the domain's security manager.
- ▶ This synchronization works one way, from the domain CMDBs to the eCMDB, so objects that are created at the eCMDB do not get propagated to the domain CMDBs.
- ▶ Always create and populate AccessCollections from the domain CMDBs and then synchronize with the eCMDB.
- ▶ Create roles and permissions from the domain CMDBs, and then synchronize with the eCMDB.
- ▶ You can create users at the eCMDB and give access to access collections from multiple domains.



Part 2

Tivoli Application Dependency Discovery Manager Planning and Installation

In this part, we discuss the planning and installation of Tivoli Application Dependency Discovery Manager.

Deployment and capacity planning

In this chapter, we discuss the planning of a working IBM Tivoli Application Dependency Discovery Manager (TADDM) environment that can discover the infrastructure, application system components, and their dependencies.

In this chapter, we discuss:

- ▶ “Sizing your TADDM environment” on page 62
- ▶ “Creating a deployment plan” on page 64
- ▶ “Planning your hardware and software” on page 65
- ▶ “TADDM deployment checklist” on page 74
- ▶ “Planning worksheets” on page 76
- ▶ “Deployment planning case study” on page 80

4.1 Sizing your TADDM environment

In this section, we provide a general rule for the sizing recommendations that are available at the time of writing this book.

4.1.1 TADDM Server sizing

The general rule for TADDM sizing is based on the number of configuration items (CIs) per host. Although there is no common industry agreement on the number of CIs per host, a good rule for CIs for each host is around 100 for each CPU.

The greater the processing capacity of a computer system, the more likely it is that more business application components, or CIs, are hosted by that computer system, which translates into the following numbers:

- ▶ One CPU hosts 100 CIs
- ▶ Two CPUs host 200 CIs
- ▶ Four CPUs host 400 CIs

The general rule for the number of CIs within a single TADDM domain is 2 000 000, which allows timely processing of around:

- ▶ 40 000 hosts if you have 50 CIs per host
- ▶ 20 000 hosts if you have 100 CIs per host
- ▶ 10 000 hosts if you have 200 CIs per host

The TADDM Server itself does not scale linearly beyond four CPUs.

4.1.2 Topology reconciliation is not a linear process

A TADDM discovery run consists of three major phases. In the first phase, sensors are launched, and data is gathered and stored in memory. In the second phase, which is referred to as *topology reconciliation*, newly discovered data and data from previous discovery runs are analyzed, and an updated topology is built. In the third phase, this data is stored in the database.

Increasing the number of CPUs on the system that hosts the TADDM Server appears at first as though it decreases discovery time because of parallelism. However, because topology reconciliation is not linear, the knee in the curve is reached at around four CPUs.

Think of topology reconciliation as an n^* matrix. We do not know in advance how many dependencies exist, and it might keep expanding, as we discover in a tightly meshed environment where there are many dependent application components. The number of dependencies is not directly related to the number of CIs; instead, it might be orders of more magnitude. If this environment is highly meshed, the complexity increases in an exponential manner, which takes a lot of processing time. Adding more CPUs does not reduce that time.

4.1.3 Database sizing considerations

To calculate the configuration management database (CMDB) size, use the general rule: 2 MB per server equivalent, for example, the following numbers of server equivalents work out to the following database sizes:

- ▶ 5 000 server equivalents: 10 GB
- ▶ 10 000 server equivalents: 20 GB

The following numbers were realized from an actual large-scale implementation:

Running weekly discoveries of about 2 500 server equivalents for 14 months uses about 7 GB of space.

The reason that the planning information cannot be better qualified is that the determining factor, beside the number of hosts and devices, is the number of components that are discovered in the environment, the rate of change, and the number of versions kept. These factors can vary tremendously from one environment to another environment. In addition, the size of an individual component varies as well. For example, how many Enterprise JavaBeans (EJBs) are deployed in a WebSphere Application Server, or how many tables exist in a database? Using Custom Server Templates and Computer System Templates to collect and track files can also greatly influence the capacity requirements of the Configuration Management Database (CMDB).

In an enterprise configuration, the TADDM enterprise domain database can operate in the following two modes:

- ▶ *Deep mode*: This is the default mode. The TADDM enterprise domain database synchronizes all of the information contained in the single domains. Thus, when this information is retrieved in an enterprise environment, it is retrieved completely from the TADDM domain databases. There is no runtime access to the single domain to retrieve deep information when the TADDM enterprise domain database operates by default in this mode.

The TADDM enterprise domain database operates by default in this mode, because the \$COLLATION_HOME/etc/domainquery file contains the text

SYNC_ALL_ATTRS. When the domainquery file begins with SYNC_ALL_ATTRS, any remaining content is ignored and deep synchronization is performed.

- *Shallow mode:* The TADDM enterprise domain database stores a small set of top-level information contained in the TADDM domain databases. However, the TADDM enterprise domain database can query the TADDM domain database for more detailed information as needed. This query capability reduces the amount of data that needs to be held at the TADDM enterprise domain database, enabling it to hold more objects and process more transactions than a TADDM domain database. The default top-level information that is stored locally in the TADDM enterprise domain database is specified in the \$COLLATION_HOME/etc/domainquery.shallow file on the TADDM enterprise domain server. You can customize this file.

You need to select either deep or shallow mode one time, which is prior to performing any domain synchronizations. To switch from the deep mode to the shallow mode, make a copy of the domainquery file, and rename the supplied domainquery.shallow file to domainquery.

In deep or shallow mode, you can specify tables to be ignored from synchronization in the \$COLLATION_HOME/etc/sync/importsuppresslist file on the TADDM enterprise server. In deep mode, you can suppress specific attributes from being written to the TADDM enterprise domain database by specifying those attributes in the following manner:
\$COLLATION_HOME/etc/sync/importsuppresslist file:tablename:attr1,attr2.

Note: Refer to 4.6.2, “Solution approach” on page 81 for a discussion of server equivalents.

4.2 Creating a deployment plan

Creating a deployment plan is essential to creating and installing a configuration and tracking environment. This book describes all of the planning considerations and provides scenarios for creating a comprehensive deployment plan. At a minimum, you need to gather the following information before you install any software:

- Base hardware and software requirements for TADDM Version 7.1
- Whether the computer systems in your distributed network can support this new software, whether these systems can be upgraded to meet your business needs, or whether you will need new systems

- ▶ Which TADDM components to install on which computer systems in your distributed network to support your business needs and whether they have additional third-party software requirements

For each system where you plan to install components of TADDM, provide the following information:

- ▶ Host name
- ▶ Operating system
- ▶ Available memory and available disk space
- ▶ Which components of IBM Tivoli Change and Configuration Management Database TADDM to install

4.3 Planning your hardware and software

Table 4-1 summarizes the platforms that TADDM Version 7.1 supports. For each platform, install the latest available service packs.

Table 4-1 Supported operating systems for TADDM V7.1 components

Operating system and supported release	Support details
AIX 5.2 (Release previous to current platform release)	<ul style="list-style-type: none"> ▶ Domain Manager ▶ Product Console ▶ Anchor ▶ TADDM Server <p>Support is provided for the hardware and the operating system running in 64-bit mode; however, the TADDM Server and the anchor run in a 32-bit Java Virtual Machine (JVM).</p>
AIX 5.3 (Current platform release)	<ul style="list-style-type: none"> ▶ Domain Manager ▶ Product Console ▶ Anchor ▶ TADDM Server <p>Support is provided for the hardware and the operating system running in 64-bit mode; however, the TADDM Server and the anchor run in a 32-bit Java Virtual Machine.</p>

Operating system and supported release	Support details
Red Hat Enterprise Linux 4.0 x86 (Release previous to current platform release)	<ul style="list-style-type: none"> ▶ Domain Manager ▶ Product Console ▶ Anchor ▶ TADDM Server
Red Hat Enterprise Linux 4.0 x86_64 (Release previous to current platform release)	<ul style="list-style-type: none"> ▶ Domain Manager ▶ Product Console ▶ Anchor ▶ TADDM Server <p>Support is provided for the hardware and the operating system running in 64-bit mode; however, the TADDM Server and the anchor run in a 32-bit Java Virtual Machine.</p>
Red Hat Enterprise Linux 4.0 for System z® (Release previous to current platform release)	<ul style="list-style-type: none"> ▶ Domain Manager ▶ Product Console ▶ Anchor ▶ TADDM Server <p>Support is provided for the hardware and the operating system running in 64-bit mode only. Both the TADDM Server and the anchor run in a 64-bit Java Virtual Machine on this operating system.</p> <p>Red Hat Update 3 is also required.</p>
Red Hat Enterprise Linux 5.0 x86 (Current platform release)	<ul style="list-style-type: none"> ▶ Domain Manager ▶ Product Console ▶ Anchor ▶ TADDM Server <p>The installation program does not start in GUI mode on Red Hat Enterprise Linux 5.0 systems unless you install the following library file: libXp.so.6. The Red Hat Package Manager (RPM) package libXp-1.0.0-8.i386.rpm must be installed. This package can be found on disk two of the Red Hat Enterprise Linux 5.0 distribution media in the Server directory.</p>

Operating system and supported release	Support details
<p>Red Hat Enterprise Linux 5.0 x86_64 (Current platform release)</p>	<ul style="list-style-type: none"> ▶ Domain Manager ▶ Product Console ▶ Anchor ▶ TADDM Server <p>Support is provided for the hardware and the operating system running in 64-bit mode; however, the TADDM Server and the anchor run in a 32-bit Java Virtual Machine.</p> <p>The installation program does not start in GUI mode on Red Hat Enterprise Linux 5.0 systems unless you install the following library file: libXp.so.6. The RPM package libXp-1.0.0-8.i386.rpm must be installed. This package can be found on disk two of the Red Hat Enterprise Linux 5.0 distribution media in the Server directory.</p>
<p>Red Hat Enterprise Linux 5.0 for System z (Current platform release)</p>	<ul style="list-style-type: none"> ▶ Domain Manager ▶ Product Console ▶ Anchor ▶ TADDM Server <p>Support is provided for the hardware and the operating system running in 64-bit mode only. Both the TADDM Server and the anchor run in a 64-bit Java Virtual Machine on this operating system.</p> <p>The installation program does not start in GUI mode on Red Hat Enterprise Linux 5.0 systems unless you install the following library file: libXp.so.6. The RPM package libXp-1.0.0-8.i386.rpm must be installed. This package can be found on disk two of the Red Hat Enterprise Linux 5.0 distribution media in the Server directory.</p>

Operating system and supported release	Support details
Solaris 9 SPARC (Release previous to current platform release)	<ul style="list-style-type: none"> ▶ Domain Manager ▶ Product Console ▶ Anchor ▶ TADDM Server <p>Support is provided for the hardware and the operating system running in 64-bit mode; however, TADDM Server and anchor run in a 32-bit Java Virtual Machine.</p>
Solaris 10 SPARC (Current platform release)	<ul style="list-style-type: none"> ▶ Domain Manager ▶ Product Console ▶ Anchor ▶ TADDM Server <p>Support is provided for the hardware and the operating system running in 64-bit mode; however, the TADDM Server and the anchor run in a 32-bit Java Virtual Machine.</p>
SUSE Linux Enterprise Server 9.0 x86 (Release previous to current platform release)	<ul style="list-style-type: none"> ▶ Domain Manager ▶ Product Console ▶ Anchor ▶ TADDM Server <p>At the time of release, IBM Software support is investigating intermittent problems with the JVM and SUSE Linux Enterprise Server 9.0 x86. For production environments, use SUSE 10.</p>
SUSE Linux Enterprise Server 9.0 x86_64 (Release previous to current platform release)	<ul style="list-style-type: none"> ▶ Domain Manager ▶ Product Console ▶ Anchor ▶ TADDM Server <p>Support is provided for the hardware and the operating system running in 64-bit mode; however, the TADDM Server and the anchor run in a 32-bit Java Virtual Machine.</p> <p>At the time of release, IBM Software support is investigating intermittent problems with the JVM and SUSE Linux Enterprise Server 9.0 x86_64. For production environments, use SUSE 10.</p>

Operating system and supported release	Support details
<p>SUSE Linux Enterprise Server 9.0 for System z (Release previous to current platform release)</p>	<ul style="list-style-type: none"> ▶ Domain Manager ▶ Product Console ▶ Anchor ▶ TADDM Server <p>Support is provided for the hardware and the operating system running in 64-bit mode; however, the TADDM Server and the anchor run in a 32-bit Java Virtual Machine.</p> <p>At the time of release, IBM Software support is investigating intermittent problems with the JVM and SUSE Linux Enterprise Server 9.0 for System z. For production environments, use SUSE 10.</p> <p>SUSE Patch Level 3 is also required.</p>
<p>SUSE Linux Enterprise Server 10.0 x86 (Current platform release)</p>	<ul style="list-style-type: none"> ▶ Domain Manager ▶ Product Console ▶ Anchor ▶ TADDM Server <p>SUSE Fix Pack 1 is also required.</p>
<p>SUSE Linux Enterprise Server 10.0 x86_64 (Current platform release)</p>	<ul style="list-style-type: none"> ▶ Domain Manager ▶ Product Console ▶ Anchor ▶ TADDM Server <p>Support is provided for the hardware and the operating system running in 64-bit mode; however, the TADDM Server and the anchor run in a 32-bit Java Virtual Machine.</p> <p>SUSE Fix Pack 1 is also required.</p>

Operating system and supported release	Support details
SUSE Linux Enterprise Server 10.0 for System z (Current platform release)	<ul style="list-style-type: none"> ▶ Domain Manager ▶ Product Console ▶ Anchor ▶ TADDM Server <p>Support is provided for the hardware and the operating system running in 64-bit mode only. Both the TADDM Server and the anchor run in a 64-bit Java Virtual Machine on this operating system.</p>
Microsoft Windows Server 2003 Datacenter Edition, Enterprise Edition, and Standard Edition (Current platform release)	<ul style="list-style-type: none"> ▶ Domain Manager ▶ Product Console ▶ Anchor ▶ TADDM Server ▶ Windows Gateway <p>Windows Server 2003 R2 is supported.</p> <p>This environment requires Microsoft Service Pack 2.</p>
Microsoft Windows Server 2003 Datacenter x64 Edition, Enterprise x64 Edition, and Standard x64 Edition (Current platform release)	<ul style="list-style-type: none"> ▶ Domain Manager ▶ Product Console ▶ Anchor ▶ TADDM Server ▶ Windows Gateway <p>Support is provided for the hardware and the operating system running in 64-bit mode; however, the TADDM Server and the anchor run in a 32-bit Java Virtual Machine.</p> <p>Windows Server 2003 R2 is supported.</p> <p>This environment requires Microsoft Service Pack 2.</p>
Microsoft Windows XP Professional (Release previous to current platform release)	<ul style="list-style-type: none"> ▶ Domain Manager ▶ Product Console
Windows Server 2003 DataCenter (Current platform release)	<ul style="list-style-type: none"> ▶ Domain Manager ▶ Product Console
Windows Server 2003 Standard Edition (Current platform release)	<ul style="list-style-type: none"> ▶ Domain Manager ▶ Product Console

Important TADDM Server, Windows OS, and personal firewall software information: If you are installing the TADDM Server on a Windows operating system, and if personal firewall software is installed, messages from the firewall software are displayed during the TADDM installation process. When the messages request that you grant access permission, you need to grant all access. If you do not grant access, the TADDM Server does not start or run properly.

All Windows gateways must be running either the Bitvise WinSSHD 4.06a or later or Cygwin SSH. The TADDM Server communicates with the Windows gateway using SSH, regardless of the platform that the server is using.

You must install the latest patches and updates from the operating system vendor before installing a TADDM Server, anchor, or gateway component. Systems without patches have problems running the latest Java 1.5 Java Runtime Environments (JREs) that are included with TADDM.

For the targets of a discovery, not the servers, in many cases TADDM supports operating systems and applications that are no longer supported by the vendor. While every effort is made to fix issues encountered on these targets of a discovery, you might have to reproduce the problem on a vendor-supported operating system or application and engage the vendor support.

On supported AIX operating systems, you must have an unzip utility available in the /usr/bin or /usr/local/bin directory for both the AIX TADDM Server and the AIX anchor. If you did not install the unzip utility with the supported AIX operating system, you must put an unzip utility into one of those directories before beginning the installation.

To run a 32-bit Java GUI on 64-bit hardware, you need the following 32-bit shared library files for the supported Linux operating systems:

- ▶ libdl.so.2
- ▶ libpthread.so.0
- ▶ libXmu.so.6
- ▶ libXt.so.6
- ▶ libX11.so.6
- ▶ libm.so.6
- ▶ libXtst.so.6
- ▶ libXp.so.6
- ▶ libc.so.6
- ▶ lib/ld.so.1
- ▶ libXext.so.6
- ▶ libSM.so.6
- ▶ libICE.so.6

- ▶ libXau.so.6
- ▶ libXdmcp.so.6

4.3.1 Using Red Hat Enterprise Linux for your TADDM Server

SELinux must be disabled before installing TADDM, or your installation might fail.

When you install Red Hat Enterprise Linux, SELinux is optionally enabled on the operating system:

1. To disable SELinux, turn off SELinux enforcing. Complete the following steps:
 - a. Open the following file: `/etc/sysconfig/selinux`.
 - a. Find the following line: `SELINUX=enforcing`.
 - a. Change it to `SELINUX=disabled`.
2. Restart the server.

4.3.2 Hardware requirements

The following list provides the processor, memory, and disk space requirements for a TADDM Server. The requirements are the same whether the machine is an enterprise TADDM Server or a domain TADDM Server.

Each TADDM Server requires a machine with:

- ▶ 100 GB of available disk space
- ▶ 2-4 CPUs with a minimum process speed of 2 GHz
- ▶ 4-8 GB of memory
- ▶ 4-8 GB of swap space on the disk used by the operating system

You must install the database on another machine. For medium to large environments, use more memory.

Note: Server hardware requirements are directly related to the number of CIs that the domain is going to support.

Database

Before you install the product on the server, you need to have either a local or preferably remote database server installed. You can configure TADDM to use the following external databases to store the information that is collected during the discovery process:

- ▶ IBM DB2 Version 9.1 and Fix Pack 2 for AIX (64-bit), Solaris SPARC (64-bit), Linux on System z (64-bit), Linux x86 (32-bit and 64-bit), and Windows (32-bit and 64-bit) operating systems
- ▶ IBM DB2 Version 8.2 and Fix Pack 10 for AIX (32-bit and 64-bit), Solaris SPARC (32-bit and 64-bit), Linux on System z (32-bit and 64-bit), Linux x86 (32-bit and 64-bit), and Windows (32-bit and 64-bit) operating systems
- ▶ Oracle 9i and 10g

Note: This is the last release of Tivoli Application Dependency Discovery Manager that supports the use of Oracle 9i as a TADDM Domain or Enterprise database.

The option of Oracle Real Application Clusters is not supported. You can install IBM DB2 UDB ESE Version 8.2 with Fix Pack 10 from the product media.

If you use a Windows operating system, you can complete one of the following options for the installation process:

- ▶ Manually install the DB2 server, and create the database.
- ▶ Install the DB2 server, and create the database as a part of the installation process.
- ▶ Manually create the Oracle database users.
- ▶ Create the Oracle database users as a part of the installation process.

If you use a Linux for System z operating system, you must install the DB2 software and create the database, or create an Oracle user on a remote machine before installing the TADDM Server.

Anchor server

Each Anchor server requires a machine with:

- ▶ Processor: Two-Core 2 Ghz minimum, Four-Core 4Ghz each are recommended for large domains. Always choose fewer, more powerful processors. For example, choose two 4 Ghz processors instead of eight 1 Ghz processors.

- ▶ RAM: 2 GB minimum, 4 GB recommended. 2 GB Disk swap space must be enabled.
- ▶ Disk: 2 GB available.

Windows gateway

Each Windows gateway requires a machine with:

- ▶ Processor: Two-Core 3 Ghz minimum, Four-Core 3 Ghz each is recommended.
- ▶ RAM: 1 GB minimum, 2 GB recommended.
- ▶ Disk: Less than 5 MB is needed.

4.4 TADDM deployment checklist

Table 4-2 is a checklist for planning your TADDM deployment.

Table 4-2 TADDM deployment checklist

TADDM deployment checklist	Planned/verified
TADDM components	
TADDM Server hardware/operating system: Refer to Table 4-1 on page 65	
TADDM Service Account: Create a TADDM service account (non-root) that TADDM will run as	
TADDM Windows Discovery Gateway server: <ul style="list-style-type: none"> ▶ Winsshd SSH Server (4.06a or higher) or Cygwin Secure Shell (SSH) must be installed. Host lock-out feature on WinSSHD server disabled ▶ Test Windows Management Interface (WMI) access from the Gateway Server for discovering targets: Log in to the Gateway Server with the account that is set up for the target hosts: cd \windows\system32 cscript prnjobs.vbs -l -s ADONIS (where ADONIS is the Windows target host). <p>The output looks similar to: Number of print jobs enumerated 0 If it prints: "Unable to connect to WMI service Error 0x80070005 Access is denied" or something similar, the account setup is not proper.</p>	

TADDM deployment checklist	Planned/verified
<p>TADDM Database server:</p> <ul style="list-style-type: none"> ▶ Tablespace size of at least 1 GB ▶ Account: Two database accounts with the following privileges: alter, create, delete, drop, select, update, index, and insert ▶ For Oracle DB: A new Oracle instance Create an Oracle user (collation_user), and grant connect and resource roles to collation_user. 	
<p>Anchor servers</p> <ul style="list-style-type: none"> ▶ If you have Linux or UNIX Anchor servers, verify whether you have the following parameter in the sshd.config file: AllowTcpForwarding=yes <p>If you use SSH2, SSH2 adds four TcpForwarding options. These options restrict forwarding for groups and users. These options must be enabled or absent.</p> <ul style="list-style-type: none"> ▶ The StackScan sensor uses Nmap to gather data about the targets for credential-less discovery. Verify that you have Nmap installed on your TADDM Server and all anchor servers. ▶ The StackScan sensor requires sudo access control to collect discovery information. For Windows operating systems, sudo access control is not needed. <p>To configure sudo access, complete the following steps for the TADDM Server and anchor hosts:</p> <ol style="list-style-type: none"> 1. From a command prompt window, use the su command to switch to root authority on the local host. 2. Type the vi sudo command. 3. Type the following line in the /usr/local/etc/sudoers or /etc/sudoers file: <TADDM_USER>ALL=(ALL) NOPASSWD:ALL <TADDM_USER> is the non-root user ID that is used by the TADDM Server. 	
<p>TADDM Web-based Client: You need JRE 1.5.x</p>	
<p>Have you verified host name resolution in all the servers where TADDM components are installed?</p>	
<p>Checklist for discovering targets</p>	

TADDM deployment checklist	Planned/verified
UNIX and Linux targets must have lsof installed: <ul style="list-style-type: none"> ▶ On Linux, lsof must be setuid root or sudo access as root (NOPASSWD) must be granted. ▶ On Solaris, the user credentials given to the TADDM host must be in the local sys group Solaris only. ▶ SUN scup package must be installed, which contains /usr/ucb/ps SSH access from the TADDM host must be enabled (if V2, keys must be distributed prior to the start of the discovery).	
Checklist for network devices	
Do you have Network devices to be discovered? Do you have the read-only community string?	
If you need deep discovery, enable the password for Cisco devices.	
Users and roles	
Have you planned for domain users and roles? If Enterprise Configuration Management Database (eCMDB), create users and roles at the eCMDB Domain Manager.	
Versioning	
Are you going to have multiple versions created? If yes, have you planned how frequently the versions will be created?	
Each version creates a replica of existing tables. Have you planned for additional disk space for the TADDM Database server for multiple versions?	
eCMDB	
Have you planned the number of Domain Managers and their locations? Have you planned the discovery zones that each of them will use?	

4.5 Planning worksheets

These tables list the settings that you need to know when installing TADDM.

The settings in Table 4-3 on page 77 are the basic settings for your TADDM Server.

Table 4-3 Setting for a typical installation

Setting	Default	Your value
Install directory for TADDM	<ul style="list-style-type: none"> ► For Linux, Solaris, AIX, and Linux on System z operating systems, /opt/IBM/cmdb ► For Windows operating systems, c:\ibm\cmdb 	
Non-root user ID	cmdbusr1	
DB2 instance user ID	db2inst1	
DB2 instance password	N/A	
DB2 database server port	50000	
Archive DB2 user ID (must be in the DB2 archuser Administrators group)	archuser	
Archive DB2 user ID password	N/A	

The settings in Table 4-4 are the port numbers and server details that are used by the TADDM enterprise domain server.

Table 4-4 Additional settings for a custom installation

Setting	Default	Your value
Server	TADDM Server	
Security manager port for interactions with the TADDM Enterprise Domain Server	9540	
Topology Manager port for interactions with the TADDM Enterprise Domain Server	9550	
Application programming interface (API) server port for interactions with the TADDM Enterprise Domain Server	9560	
Change manager port for interactions with the TADDM Enterprise Domain Server	9570	
Report server port for interactions with the TADDM Enterprise Domain Server	9580	
IBM Tivoli Change and Configuration Management Database (CCMDB) server host name (for launch-in-context function)		

Setting	Default	Your value
IBM Tivoli CCMDB server port (for launch-in-context function)	9530	
Non-root user ID	cmdbusr1	
Remote Method Invocation (RMI) server host name	default	
Discovery manager server mode (local or distributed)	Local	
Start RMI server after restart (check box)	Yes	
Start RMI server after installation (check box)	Yes	
Database type	DB2	
Database server host name		
Database server port	50000	
Database name	cmdb	
Node name for DB2 client		
Create database during installation (check box)	Yes	

In general, you only change the settings in Table 4-5 if you have already assigned these ports or have standards about port usage. You will need to know these port numbers when you install IBM Tivoli CCMDB.

Table 4-5 Additional port values

Setting	Default	Your Value
Web server port	9430	
Secure Sockets Layer (SSL) Web server port	9431	
GUI server port	9435	
GUI system SSL port	9434	
Java Naming and Directory Interface (JNDI) port	9432	
RMI port	9433	
Topology Manager port	5636	
Topology Builder port	5637	

Setting	Default	Your Value
RMID port	1098	

If you use Oracle as the database for your CMDB, you will need the information in Table 4-6.

Table 4-6 Settings for Oracle database

Setting	Default	Your value
Oracle database system ID	orcl	
Oracle host name		
Oracle database port	1521	
Oracle user ID	cmdbuser	
Oracle password		
Oracle additional (archive) user ID	archuser	
Oracle additional (archive) password		
Oracle system user ID	sys	
Oracle system password		
Oracle connect as role	sysdba	
Oracle home directory		

Table 4-7 contains information about ports that are used by the PingSensor and PortSensor.

Table 4-7 Ports used by the PingSensor and PortSensor to make connections

Port name	Port number
CiscoWorks	1741
DNS	53
LDAP	389
SSH	22
WBEM	5988
WMI	135

4.6 Deployment planning case study

In this section, we provide a sample scenario that can help you plan your TADDM deployment.

For more information about TADDM deployment planning, refer to the *TADDM Best Practices for Deployment Planning* document at:

<http://www.ibm.com/developerworks/wikis/display/tivoliaddm/Best+Practices+for+Deployment+Planning#BestPracticesforDeploymentPlanning->

4.6.1 Client scenario

Table 4-8 on page 81 provides the IT infrastructure for client ABC.

As shown in Table 4-8 on page 81, client ABC has UNIX (102) and Windows (222) servers. These servers are distributed across 16 locations.

Client ABC has a data center that has a total of 482 additional servers. The data center runs applications and databases, which are described in Table 4-8 on page 81. These database and application servers are part of the 482 servers.

The datacenter has four firewall zones and one DMZ for Web servers.

Table 4-8 Client ABC IT Infrastructure

Servers	Quantity
Total number of UNIX/Linux servers	102
AIX	92
Linux	10
Total number of Windows servers	222
Windows servers	222
Total number of discovery targets in data center	482
AIX/Linux	321
Intel® Windows	161
Total number of database servers	92
DB2	10 (5 databases per server)
Oracle	82 (8 databases per server)
Total number of application servers	117
Web server Instances (Apache)	50
Web Application Server (WebSphere)	67 (3 instances per server)

The client plans to deploy the IBM Service Management solution; therefore, you must design the TADDM deployment as Phase I.

4.6.2 Solution approach

Before we derive the solution for the client scenario, we show you how to calculate the approximate number of CIs and the number of Domain Managers required for a solution.

Because enterprise data center environments vary dramatically, IBM defines the concept of a server equivalent (SE) to normalize and present a standard set of performance and scale metrics. A representative unit of IT infrastructure, a server equivalent, is defined as a computer system with standard configurations for the operating system, network interfaces, and storage interfaces and also accounts for installed software, such as a database (DB2), a Web server (Apache or IPlanet), or an application server (WebSphere or WebLogic). An SE also accounts for network, storage, and other sub-systems that provide services

to the optimal functioning of the server. IBM hard-codes 200 CIs to a single SE. As defined by Information Technology Infrastructure Library (ITIL), a Configuration Item (CI) is any component that is under the control of Configuration Management, and therefore subject to formal Change Control. Each CI in the CMDB has a persistent object and change history associated with it. Examples of a CI are a computer system, operating system, L2 interface, and database buffer pool size.

Configuration item counts for discovery of individual targets

It is difficult to provide a standard number for the number of CIs in an SE. The actual number of CIs in an SE varies depending on the complexity of the infrastructure; for example, a complex database server with a large number of instances, databases, and tablespaces has a larger number of CIs per SE, and the number of CIs affects the overall performance. Table 4-9 summarizes the CI counts for a few targets that we tested in our lab.

Table 4-9 Sample data showing the number of CIs for an Item

Description of item	CI	Comments
Overhead	504	Overhead discovery. The server starts with 504 CIs before discovery
Linux System OS	40	Red Hat 4.0 AS U4 server with no applications
Linux and Java	73	Red Hat ES 3.0 with Java
DB2	286	DB2 8.2 and Fix Pack 10 application CI count (excluding OS)
Cisco switch	234	Average for Cisco 3750 and 2950 switches
SUN System and Java	87	SUN server with Java and default OS options
Oracle	235	Net cost for Oracle database
Sybase	35	Net cost for Sybase
AIX System	127	Net cost for AIX system without applications
Windows and Java	421	Net cost for Windows with Java
Apache	59	Net cost for Apache Web server (AIX system)
WebSphere Application Server 6.0	349	Net cost for WebSphere Application Server 6.0

Note: *Net cost* means just the standard base installation with no additional database or application instance.

We performed additional tests for a few other targets, such as DB2, Oracle, and the WebSphere Application Server. Table 4-10 contains the results of the additional tests.

Table 4-10 Additional CIs for each database or application server instance

Platform	Number of CIs introduced by additional database or application server instance
DB2	134
Oracle	147
WebSphere	350

The tests provide an approximate estimate of how large your environment can be. We observed an average of 200 CIs per SE. Because each environment is unique and complex, you need to consider all of the factors during your CI calculation.

Supporting a large scale IT environment

The distributed architecture of TADDM is designed to scale to millions of CIs. In particular, with a scalable relational data store, such as DB2, Tivoli TADDM can accommodate a large number of server equivalents. The major things to consider are:

- ▶ Discovery techniques (Native Discovery, Identification Markup Language (IDML) Book Load, or API)
- ▶ Data loading time
- ▶ Number of servers
- ▶ Complexity of the servers

IBM performed several benchmarks to profile the discovery speeds and integration throughput rates.

Table 4-11 Performance benchmark using various discovery techniques

Discovery technique	Rate of discovery	Comments
Native discovery (Sensor-based discovery)	50 000 CIs in one hour (250 SEs/hour) ¹	Provides deep auto-discovery capability. Can parallel multiple discoveries by adding multiple domains to achieve near linear scaling
StackScanSens or (<i>Shallow discovery</i>)	341 computer systems in one minute, 47 seconds. (1 484 computer systems/hour) ¹	Provides shallow, credential-less discovery
IDML Book Loader	60 840 CIs in one hour (300 SEs/hour)	Useful for integrating products that produce IDML books
API-based loading	120 000 CIs in one hour (600 SEs/hour) ²	Requires development using the APIs and can provide a high rate of data

1. Numbers are based upon lab machine discovery.

2. CIs with computer systems and no array object

Post-processing

After the data loading (discovery, bulkload, or API) is complete, the following post-processing tasks are initiated:

► Topology Builder

It is invoked to compute additional embedded connections between discovered objects to complete the application topology. Additionally, Topology Builder also performs additional data reconciliation to reconcile data that is obtained from multiple data sources.

► View Manager

View Manager builds the in-memory data structures that are required for the GUI to render the topology efficiently.

► Change Manager

Change Manager generates *change events* and updates the change history records.

► State Manager

State Manager also builds the in-memory topology cache for propagating changes in the topology graph.

The average post-processing time per CI is about 20 ms, and the average post-processing time for a server with 200 CIs is four seconds. So, the discovery time needs to take into consideration the time taken to discover (run the sensors) plus the post-processing time.

The following formula can help you to arrive at an estimation for the number of TADDM Server domains for your environment.

Domain_No: Number of TADDM Domains

Discovery_Rate: Rate of discovery (assumed at 50 000 CIs/hour. Refer to Table 4-11 on page 84.)

Approximate Total CI_No: Approximate number of CIs

Duration: Desired amount of time within which discovery must complete (hours)

Domain_No = [Approximate Total CI_No/Discovery_Rate/Duration]

If the Domain_No is greater than one, add an additional eCMDB server to support data aggregation across the TADDM Servers.

4.6.3 Client solution

Let us now derive the solution for our client scenario from 4.6.1, “Client scenario” on page 80, using the formula from the previous section.

As depicted in Table 4-8 on page 81, the client has a primary data center that has four firewall zones and one DMZ to make a total of five firewall zones.

Client ABC has 324 servers (UNIX + Windows) distributed across 16 different locations. However, there are no firewalls between those 16 locations.

We assume that there are Windows machines across each firewall. Hence, we need five anchors and gateways.

As part of the infrastructure details, there is no information currently provided to us about the number of network devices. In our case, we assume 150 Cisco network devices.

Also, we assume that the discovery completes in 12 hours.

As stated earlier:

Domain_No = [Approximate CI_No/Discovery_Rate/Duration]

Number of CI calculations

We need to calculate the approximate total number of CIs first for our client environment.

Note: Refer to Table 4-8 on page 81, Table 4-9 on page 82, and Table 4-10 on page 83 to calculate the number of CIs for each SE type.

The number of CIs per SE type = Number of SE (Table 4-8) x Number of CIs for that SE type (Table 4-9).

The number of CIs for distributed servers across 16 locations is:

- ▶ AIX servers = $92 \times 127 = 11\,684$
- ▶ Linux servers = $10 \times 73 = 730$
- ▶ Windows servers = $222 \times 421 = 93\,462$

CIs for Datacenter servers:

- ▶ AIX/Linux = $321 \times 127 = 40\,767$ (because we do not have a breakdown of the AIX/Linux, we take the CI calculation of AIX on the higher side)
- ▶ Windows servers = $161 \times 421 = 67\,781$

For the CIs for the database servers, we must take the base CIs for a standard database installation, plus the CIs that get added when you create each additional database. According to Table 4-8 on page 81, we have:

- ▶ DB2: 10 (Five databases per server)
- ▶ Oracle: 82 (Eight databases per server)

Therefore:

- ▶ DB2 = $10 \times 286 + 50 \times 134 = 9\,560$ (50 = 10 servers each having five databases)
- ▶ Oracle = $82 \times 235 + 656 \times 147 = 115\,702$ (656 = 82 servers each having eight databases)

CIs for Web servers: Apache = $50 \times 59 = 2\,950$

CIs for WebSphere Application Servers: WebSphere = $67 \times 350 \times 3 = 70\,350$

CIs for network devices: We have assumed 150 Cisco devices, Cisco = $150 \times 234 = 35\,100$

The total number of CIs = $11\,684 + 730 + 93\,462 + 40\,767 + 67\,781 + 9\,560 + 115\,702 + 2\,950 + 70\,350 + 35\,100 = 448\,086$

Number of domain calculations

Now, we calculate the number of domains that we need.

$$\text{Domain_No} = [\text{Approximate Total CI_No/Discovery_Rate/Duration}]$$

$$= 448\,086/50\,000/12 \text{ hours}$$

$$= 0.75$$

$$= 1$$

Hence, we need only one Domain Manager for client ABC.

TADDM Server sizing calculation

Now, we derive the server sizing for our TADDM Server.

If you refer to 4.1.1, “TADDM Server sizing” on page 62, you need to plan for a TADDM Server with at least 4 CPU/4 GB RAM and a database server of at least 2 CPU/4 GB RAM. The database server disk needs to be a minimum of:

$$2 \text{ MB} \times 1457 = \text{Approximately } 3 \text{ GB}$$

Now, 3 GB is a minimum database size. If you plan to perform versioning in your environment, you need equal database capacity for each version. So, if you plan one version every quarter, you need an additional 12 GB, which makes the total 15 GB. Note that the TADDM Database server must always have at least two physical disk drives (more recommended) as a storage array for high disk I/O throughput.

4.6.4 Additional sizing examples

Example 4-1, Example 4-2 on page 88, and Example 4-3 on page 88 are additional deployment examples.

Example 4-1 SmallManufacturer Inc

Example SmallManufacturer Inc:

Discovery Techniques – Native Discovery

Rate of Discovery: 50,000 CIs per hour

Approx. Number of CIs: 1000 (SEs) x 200 CIs per SE

Duration: 8 hours

Number of Domains = $[200000/50000/8] = [0.5] = 1$

Number of CCMDB Servers = 1

Example 4-2 MediumInsurer Inc

Example MediumInsurer Inc:

Discovery Techniques – Native Discovery

Rate of Discovery: 50,000 CIs per hour

Approx. Number of CIs: 3500 (SEs) x 200 CIs per SE

Duration: 24 hours

Number of Domains = $\lceil 700000/50000/24 \rceil = \lceil 0.58 \rceil = 1$

Number of CCMDB Servers = 1

Example 4-3 LargeInsurer Inc

Example LargeInsurer Inc:

Discovery Techniques – Native Discovery

Rate of Discovery: 50,000 CIs per hour

Approx. Number of CIs: 12000 (SEs) * 200 CIs per SE

Duration: 24 hours

Number of Domains = $\lceil 2400000/50000/24 \rceil = \lceil 2.00 \rceil = 2$

Number of CCMDB Servers = 3 (with additional eCMDB server)

You can use the formulas in these examples for the initial deployment planning. However, in large client environments, because the complexity of servers can vary between environments, the process of fine tuning the TADDM deployment is iterative. You need to further verify and refine the deployment using the following mechanisms.

Check the total number of CIs in your environment after the initial discovery. You can obtain the CI number with the following commands (if using DB2):

- ▶ DB2 connect to <CMDB_DB_Name>
- ▶ DB2 select count (*) from PERSOBJ

Check the total number of servers in your environment. The server number can be obtained with the following command (if using DB2):

```
DB2 select count(distinct(contextip_x)) from COMPSYS
```

Divide the total number of CIs by the number of servers. If the result is larger than 200 (as documented in our benchmark results), move several servers to their own domain to help achieve the desired duration.

Tivoli Application Dependency Discovery Manager installation steps

In this chapter, we provide information about installing Tivoli Application Development Discovery Manager (TADDM). Specifically, we provide the steps that we took in successfully installing TADDM V7.1 in our lab environment.

In this chapter, we discuss:

- ▶ “Our lab environment” on page 90
- ▶ “Installing DB2” on page 92
- ▶ “Installing a TADDM Domain Server on Windows” on page 108
- ▶ “Installing a TADDM Domain Server on Linux” on page 126
- ▶ “Installing a TADDM enterprise server on AIX” on page 144
- ▶ “Configuring LDAP” on page 172
- ▶ “Deploying anchors and gateways” on page 174
- ▶ “Setting up Windows gateways” on page 182
- ▶ “Troubleshooting” on page 193

5.1 Our lab environment

We attempted to make our lab environment as realistic as possible. We had two ITSO labs that were separated by firewalls. The first ITSO lab was located in Austin, TX, which is where the TADDM Servers were located. We had two TADDM Domain Servers and one TADDM enterprise server. For variety, each of the TADDM Servers ran a different operating system (OS). The first TADDM domain server, host name Waco, was running on Linux. The second TADDM Domain Server, host name Southend, was running on Windows Server 2003. The TADDM enterprise server, host name Paris, was running on AIX 5.3. All three databases were DB2, and all three databases were located on a single AIX server, host name Copenhagen. In addition to the TADDM Servers, we also had a TADDM Windows gateway, host name Newyork, located in the Austin lab. There were numerous Windows and UNIX servers to discover in the Austin lab.

The other ITSO lab was located in San Jose, CA. There were two firewalls separating the two ITSO labs. In the San Jose lab, we had a TADDM Windows Gateway, host name Wilas, and a TADDM Anchor server, host name Zaire. In addition to these servers, there were other Windows and UNIX servers to be discovered.

Figure 5-1 on page 91 illustrates our lab environment.

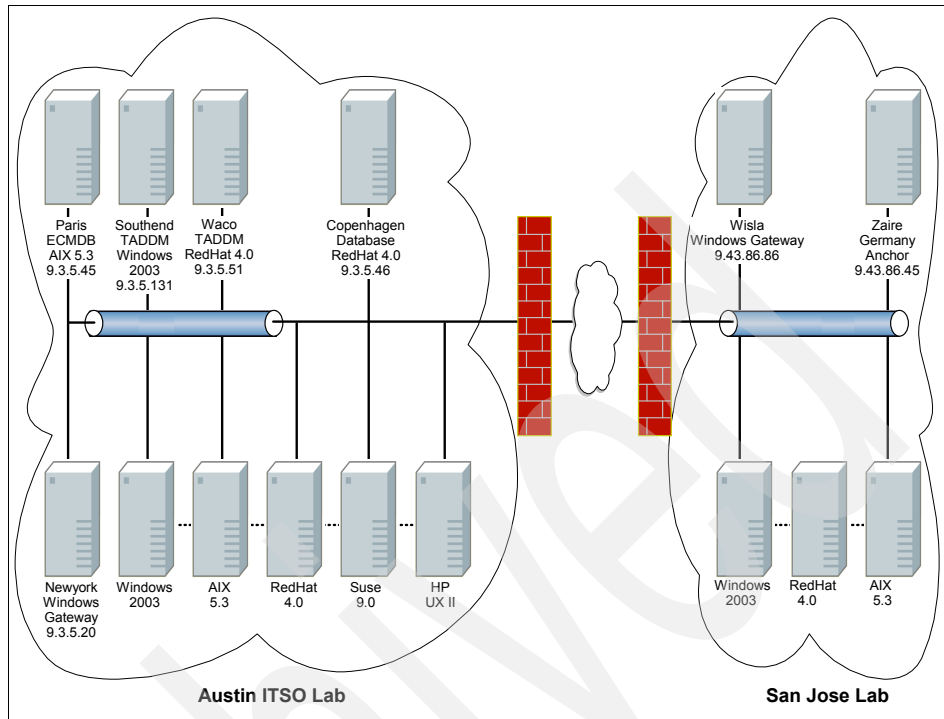


Figure 5-1 Lab environment

The TADDM installation wizard supports several installation scenarios, including:

- ▶ Simple installation with the installation of a DB2 database. This scenario installs and configures a local DB2 database as part of the TADDM installation.
- ▶ Simple installation without the installation of a DB2 database. This scenario is for the case where you already have your local DB2 database installed.
- ▶ Advanced installation with a remote DB2 database. This scenario requires you to install your DB2 database on a remote server prior to the installation of TADDM.
- ▶ Advanced installation with a remote Oracle database. This scenario requires you to install your Oracle database on a remote server prior to the installation of TADDM.

The best practice recommendation for a production environment is for the database to exist on a remote server.

For our implementation, we chose the third option in the previous list of options, which is the advanced installation with a remote DB2 database. This scenario

requires you to install your DB2 database on a remote server prior to the installation of TADDM. We also chose to place all three TADDM Databases on a single AIX server.

Note: Development recommends that you place the database on a remote server for a production environment. The reason for this recommendation is that loads on the TADDM Server and database server are so high that a single machine cannot handle it.

5.2 Installing DB2

For our implementation, we chose to use DB2 for the TADDM Databases. We also chose to install all three databases on a single AIX server.

Here is a high-level overview of the steps involved in creating the TADDM Databases:

1. Install DB2 Enterprise Server, which creates the DB2 Administration Server (DAS). We also installed the DB2 fix pack.
2. Create the DB2 database users.
3. Create the DB2 instances. We created three DB2 instances: one DB2 instance for each of the two TADDM Domain Servers and one DB2 instance for the TADDM enterprise server.
4. Run the `make_db2_db` script to create the new configuration management database (CMDDB) for each DB2 instance.

5.2.1 Install DB2 Enterprise Server

To install DB2 on the AIX operating system, we completed the following steps:

1. Obtain the DB2 installation media and copy it to the server where DB2 will be installed. The DB2 install image ships on disk 3 of the TADDM AIX installation media. We copied this to our AIX server, Copenhagen, and untarred the file into the directory, `/opt/IBM/images/TADDM71/DB2/linux/DB2-ESE_9.1`
2. We logged on to Copenhagen as the root user and changed the directory to the directory where the DB2 install image is located.

The **`db2setup`** command displays the IBM DB2 Setup launchpad panel, which is shown in Figure 5-2 on page 93.

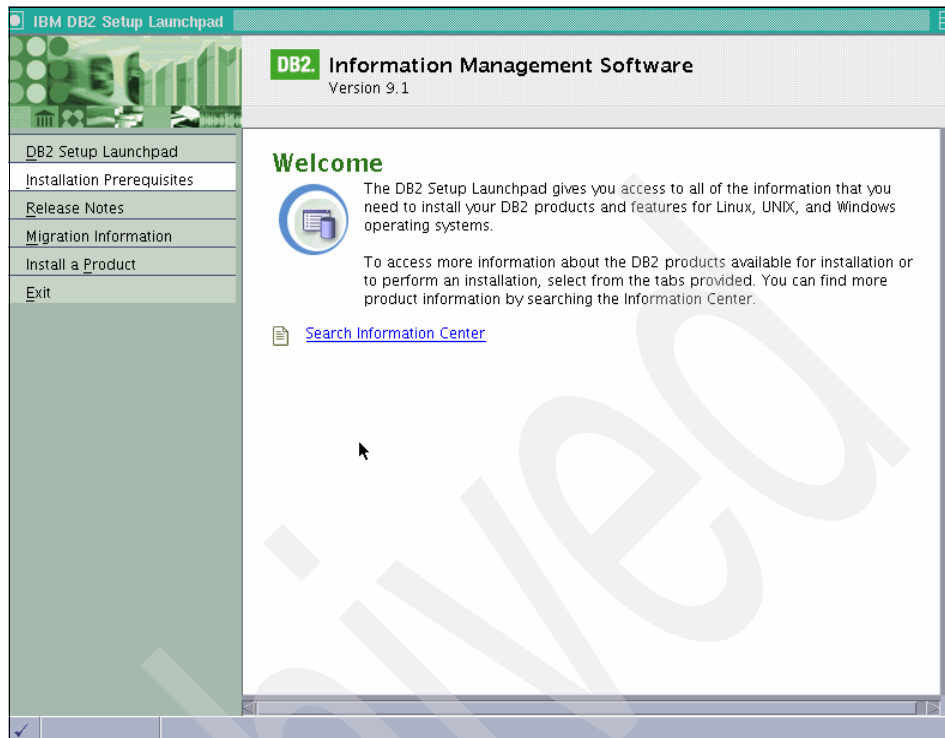


Figure 5-2 DB2 Installation Welcome panel

3. Select **Install a Product** from the left pane. The panel shown in Figure 5-3 on page 94 is displayed.



Figure 5-3 Install a Product panel

4. Click **Install New**, and the Welcome to the DB2 Setup wizard panel, which is shown in Figure 5-4 on page 95, is displayed.

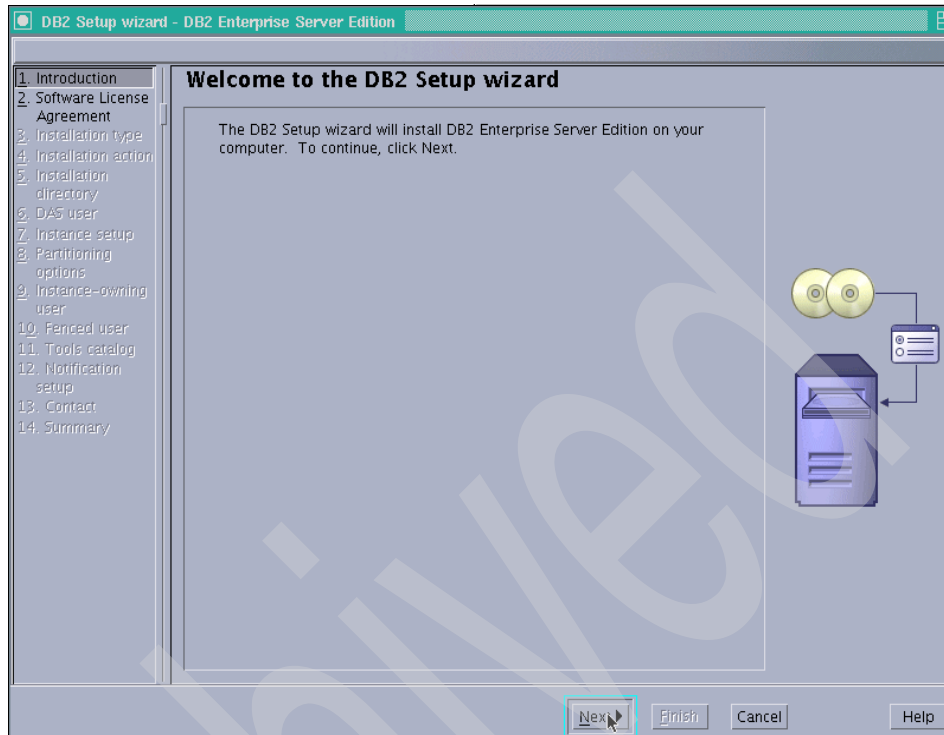


Figure 5-4 Welcome to the DB2 Setup wizard panel

5. The DB2 Setup wizard now leads you through the installation process. Notice the list of 14 installation steps on the left side of the wizard panel. As the setup wizard leads you through the installation of DB2, this list reminds you where you are in the installation process.

Click **Next** and the Software License Agreement panel, which is shown in Figure 5-5 on page 96 appears.

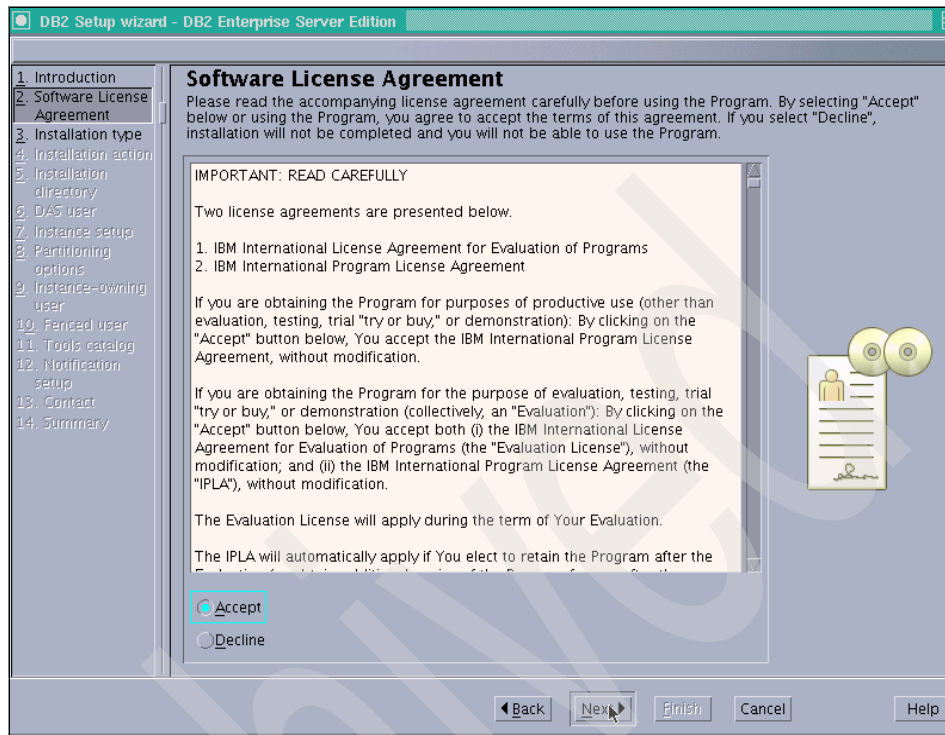


Figure 5-5 Software License Agreement

6. Read through the licensing agreement. If you agree with the terms, click **Accept** and then click **Next**. The Select the installation type panel, which is shown in Figure 5-6 on page 97, appears.

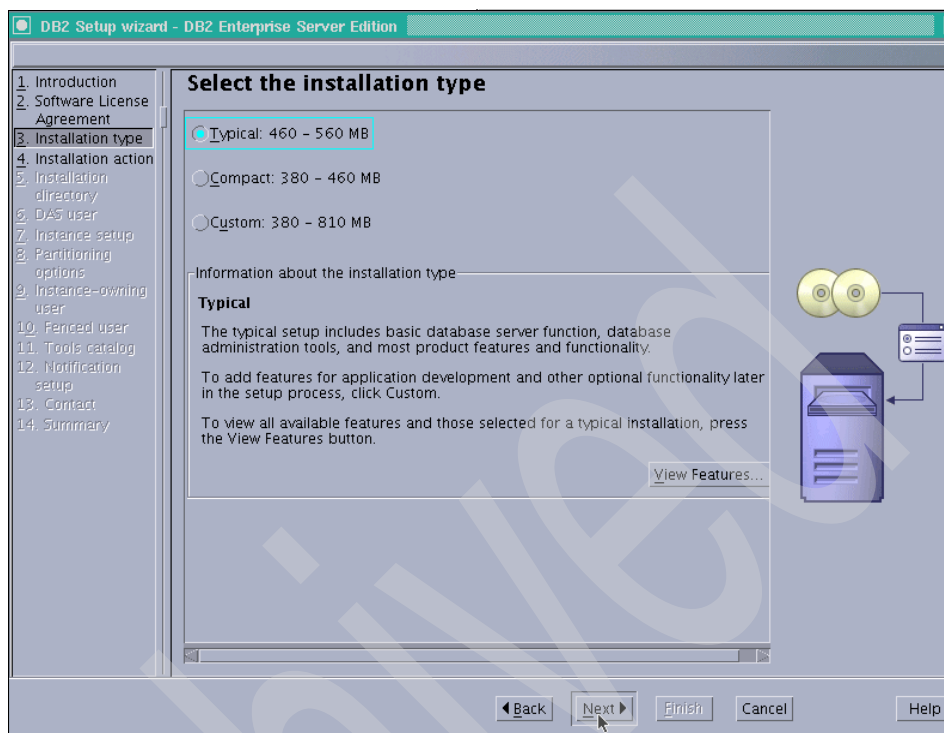


Figure 5-6 Select the Installation type panel

7. We selected **Typical: 460 - 560 MB** and then clicked **Next**. The panel, which is shown in Figure 5-7 on page 98, appears.

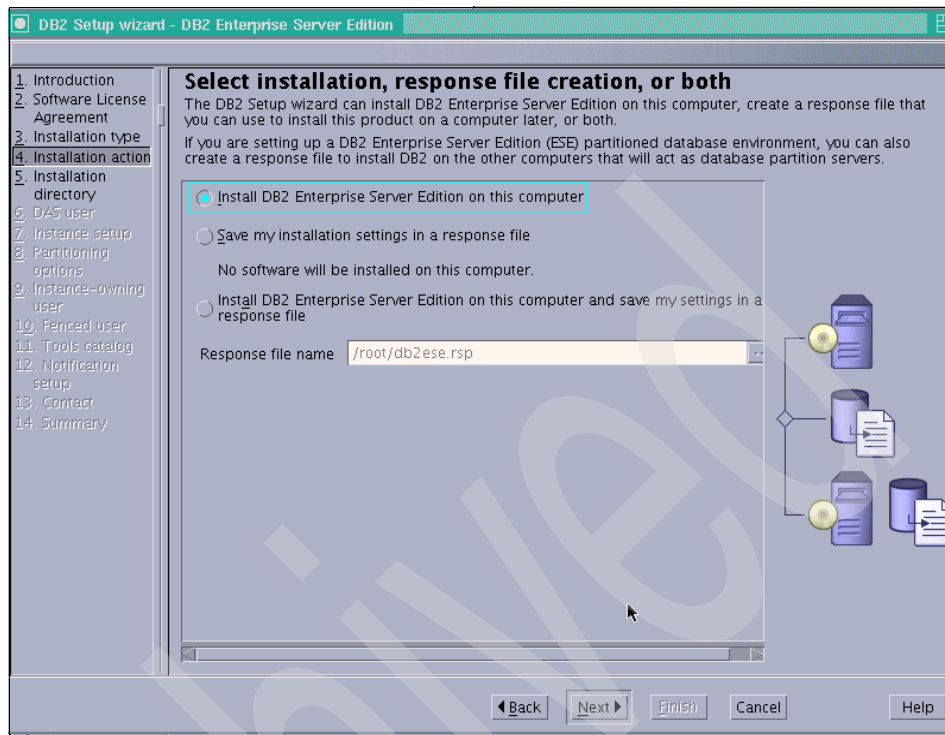


Figure 5-7 Select installation, response file creation, or both panel

8. We selected **Install DB2 Enterprise Server Edition on this computer** and clicked **Next**. The panel, which is shown in Figure 5-8 on page 99, appears.

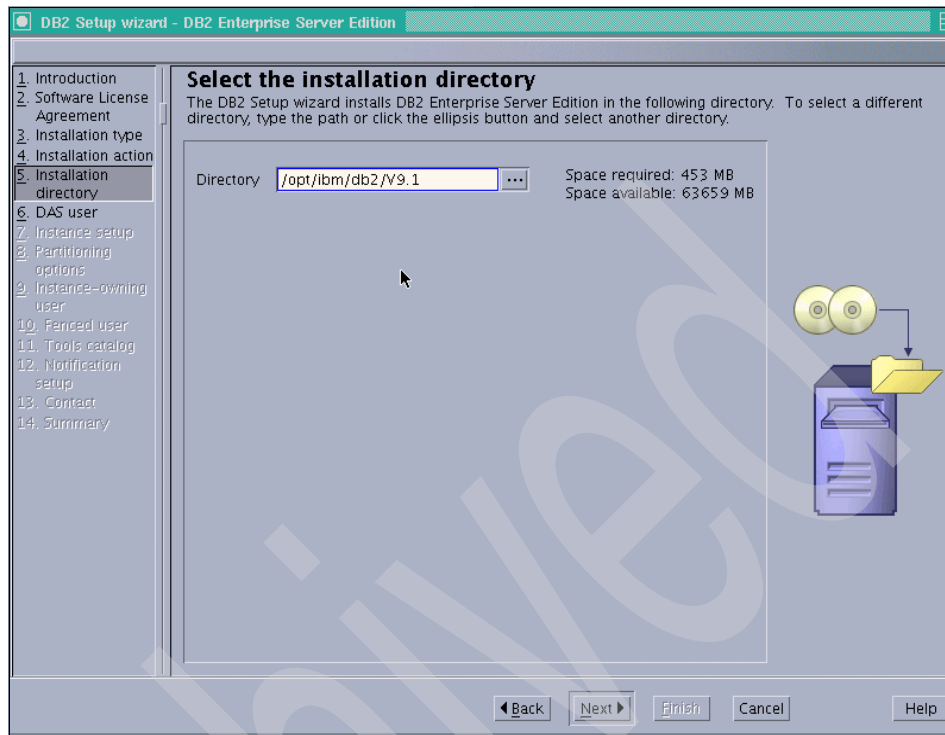


Figure 5-8 Select the installation directory panel

9. Enter the directory where you want to install DB2. We accepted the default value, /opt/ibm/db2/V9.1. Clicking **Next** displayed the panel, which is shown in Figure 5-9 on page 100.

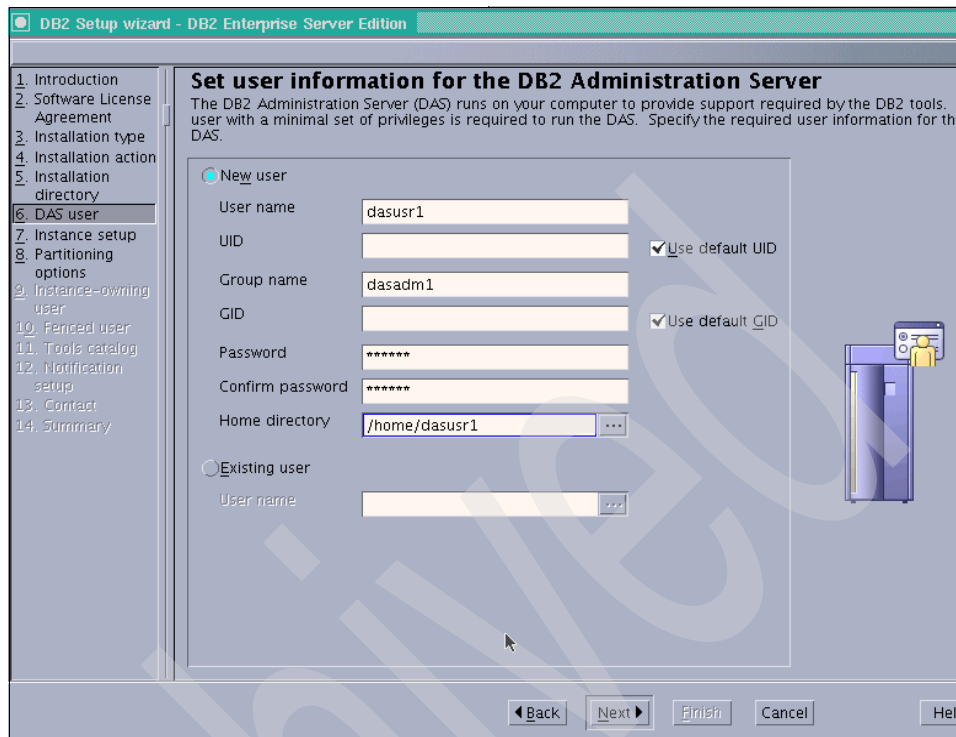


Figure 5-9 Set user information for the DB2 Administrator Server

10. On this panel, enter the information for the DAS user information. We accepted the default values and entered a password for the user. Clicking **Next** displays the panel that is shown in Figure 5-10 on page 101.

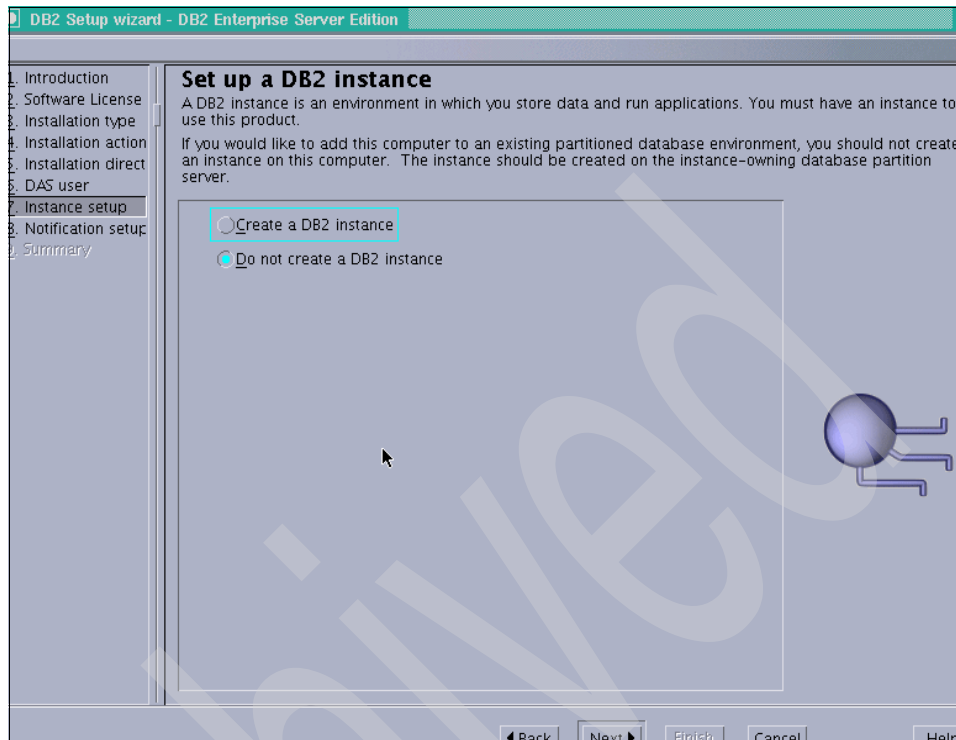


Figure 5-10 Set up a DB2 instance panel

11. This panel lets you create a DB2 instance. In DB2, an *instance* is an independent environment where database objects can be created and applications can be run against them. We are going to create our DB2 instance ourselves, so we selected **Do not create a DB2 instance**. Clicking **Next** displays the panel that is shown in Figure 5-11 on page 102.

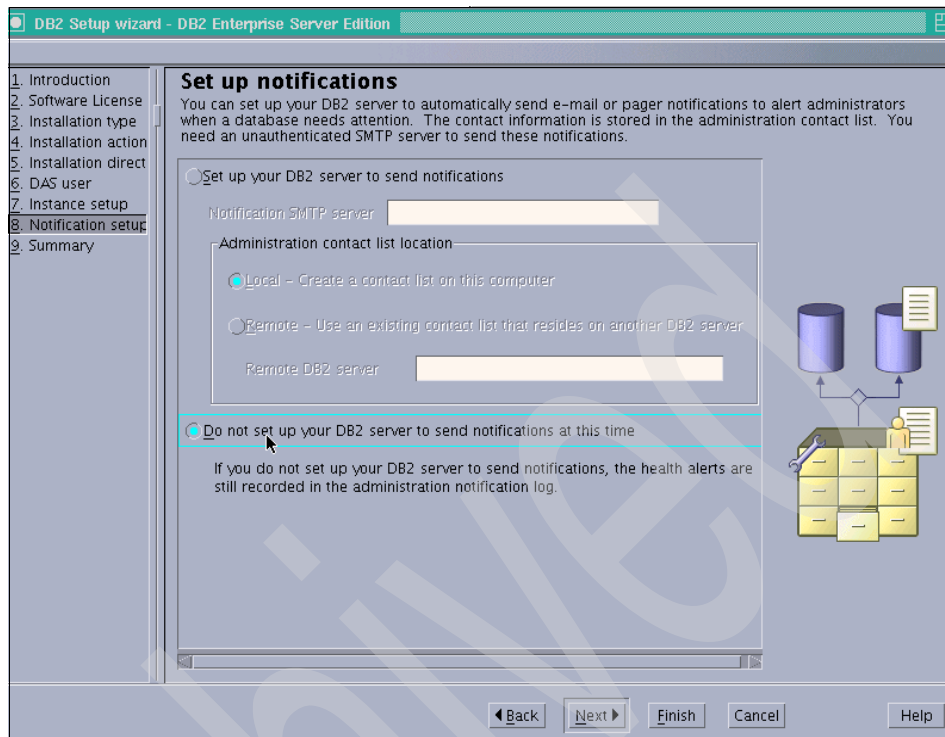


Figure 5-11 Set up notifications panel

12. We selected **Do not set up your DB2 server to send notifications at this time**. Clicking **Next** displays the panel that is shown in Figure 5-12 on page 103.

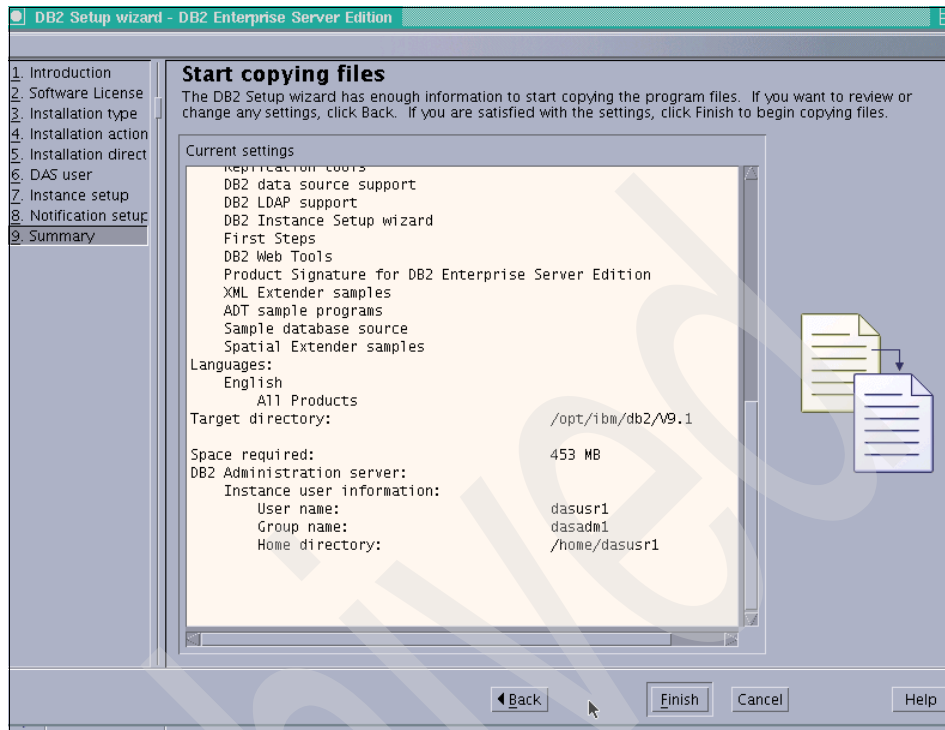


Figure 5-12 Start copying files panel

13. We reviewed the current settings and were satisfied with the settings. When we clicked **Next**, the installation wizard began copying files.

A progress panel, which is similar to Figure 5-13 on page 104, appears while the files are copied.

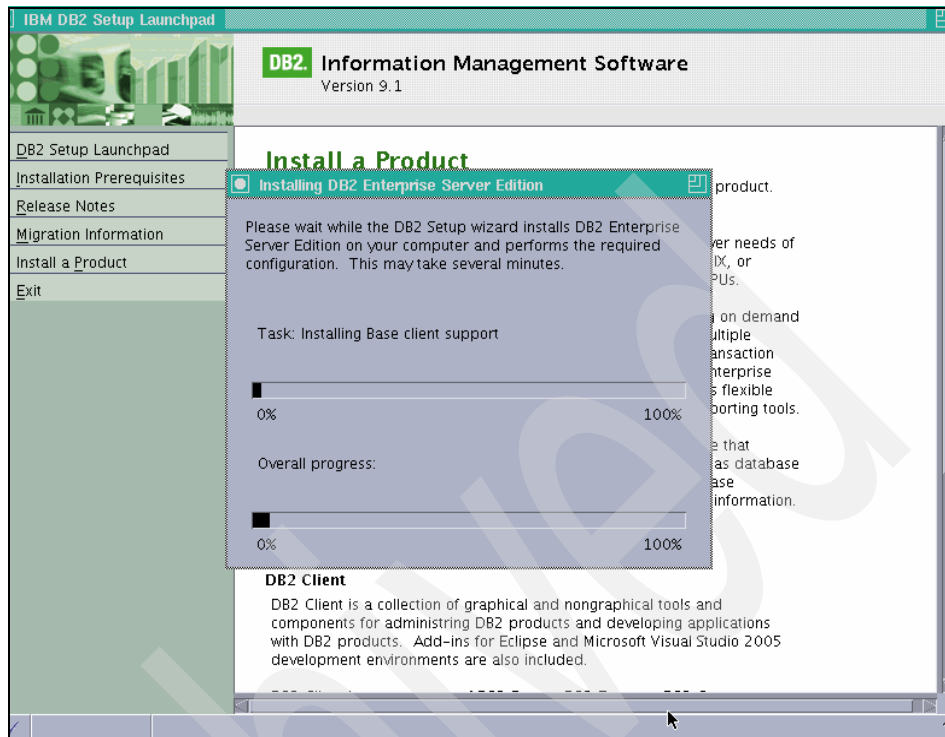


Figure 5-13 Installation progress panel

14. The panel that is shown in Figure 5-14 on page 105 is displayed when all of the files have been copied and DB2 Enterprise Server is successfully installed.

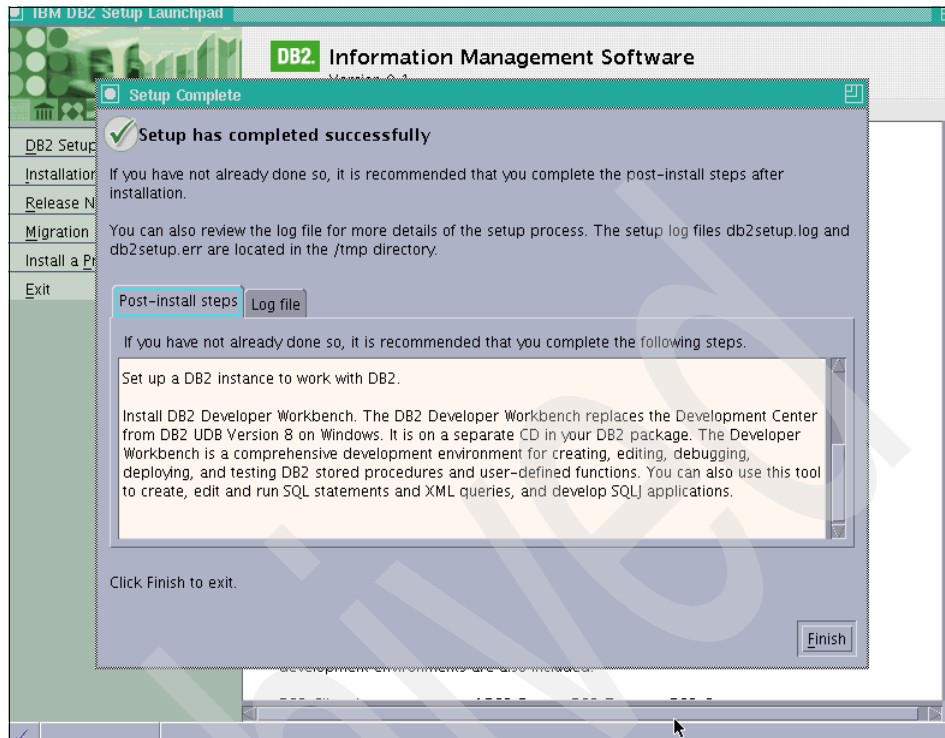


Figure 5-14 Setup has completed successfully panel

15. Before creating the DB2 instances, we installed the DB2 fix pack. Type the command `./installFixPack`.

We were prompted for the full path name of the DB2 installation directory, which is shown in Figure 5-15 on page 106.

```

root@copenhagen:/code/DB2/linux/DB2-ESE_9.1_Fixpacks/ease/disk1

ease/disk1/doc/zh_TW/
ease/disk1/doc/zh_TW/install.txt
ease/disk1/doc/zh_TW/release.txt
ease/disk1/installFixPack
ease/disk1/db2setup
ease/disk1/db2_install
[root@copenhagen DB2-ESE_9.1_Fixpacks]# ls -l
total 336272
drwxr-xr-x  3 root  root      4096 Feb 17  2007 ease
-rwxr--r--  1 90147 90022 343992320 Apr  2  2007 v9fp2_linux_ease.tar
[root@copenhagen DB2-ESE_9.1_Fixpacks]# cd ease
[root@copenhagen ease]# ls -l
ls: 1: No such file or directory
[root@copenhagen ease]# ls -l
total 4
drwxr-xr-x  4 root  root 4096 Feb 17  2007 disk1
[root@copenhagen ease]# cd disk1/
[root@copenhagen disk1]# ls -l
total 32
drwxr-xr-x  6 bin  bin 4096 Feb 17  2007 db2
-r-xr-xr-x  1 bin  bin 5090 Feb 17  2007 db2_install
-r-xr-xr-x  1 bin  bin 5078 Feb 17  2007 db2setup
drwxr-xr-x 15 bin  bin 4096 Feb 17  2007 doc
-r-xr-xr-x  1 bin  bin 5102 Feb 17  2007 installFixPack
[root@copenhagen disk1]# pwd
/code/DB2/linux/DB2-ESE_9.1_Fixpacks/ease/disk1
[root@copenhagen disk1]# ls -l
total 32
drwxr-xr-x  6 bin  bin 4096 Feb 17  2007 db2
-r-xr-xr-x  1 bin  bin 5090 Feb 17  2007 db2_install
-r-xr-xr-x  1 bin  bin 5078 Feb 17  2007 db2setup
drwxr-xr-x 15 bin  bin 4096 Feb 17  2007 doc
-r-xr-xr-x  1 bin  bin 5102 Feb 17  2007 installFixPack
[root@copenhagen disk1]# ./installFixPack
DBI1073E The -b <baseInstallPathOfDB2> is required for the
        installer script installFixPack.

Enter full path name for the install directory -
-----
/opt/ibm/db2/V9.1

```

Figure 5-15 Install DB2 fix pack panel

5.2.2 Create DB2 database users

Now that we have installed DB2 Enterprise Server, we need to create the database instance. However, we first created six user accounts before we created the database instance. These accounts, which are also known as the DB2 user and the DB2 archuser, are the primary and secondary users for the three database instances, as shown in Table 5-1 on page 107.

Table 5-1 DB2 database users

TADDM environment (host name)	Primary/ DB2 user	Secondary/ DB2 Archuser	DB2 instance name
TADDM domain (Southend)	taddmwin	archwin	taddmwin
TADDM domain (Waco)	taddmlin	archlin	taddmwin
TADDM enterprise (Paris)	ecmdb	archecmdb	ecmdb

5.2.3 Create the DB2 instances

Next, we created the three DB2 instances. We changed the directory and created the instances:

```
cd /opt/ibm/db2/V9.1/instance
./db2icrt -a SERVER -p 50010 -s ese -u taddmlin taddmlin
./db2icrt -a SERVER -p 50020 -s ese -u taddmwin taddmwin
./db2icrt -a SERVER -p 50030 -s ese -u ecmdb ecmdb
```

Database users' home directories are where the database is stored, which means that the home directories must rely on a multi-disk partition for high disk I/O throughput.

5.2.4 Run the make_db2_db.sh script

The script named make_db2_db.sh creates the initial cmdb database. This script must be run for each of the three DB2 instances that were just created.

To run the make_db2_db.sh script, first copy the script to the home directory of the instance owner for taddmlin, taddmwin, and ecmdb.

Then, log on (or **su**) as the instance owner, go to the home directory for that user, and run the make_db2_db.sh script with the following command:

```
./make_db2_db.sh cmdb
```

5.3 Installing a TADDM Domain Server on Windows

Perform the following steps to install a TADDM Domain Server on Windows.

5.3.1 Install TADDM 7.1

To complete the installation of TADDM with a remote database, complete the following steps:

1. Log on on to the Windows system with a user account that has Administrator authority.
2. Locate the installation media and copy it to the Windows server. We copied the installation media to the C:\code\TADDM_V710_Windows\TADDM directory. Go to the directory where you placed the installation media and run the setupWin32.exe file (Figure 5-16).

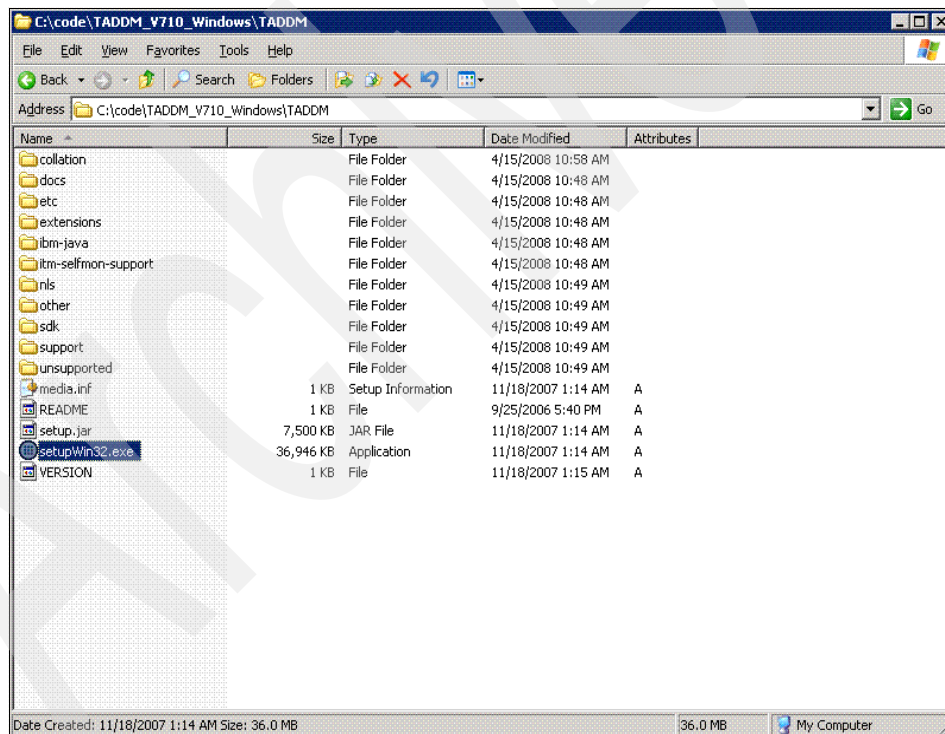


Figure 5-16 The setupWin32.exe command

Issuing the setupWin32.exe command, or double-clicking the command as we did, displays the InstallShield Wizard Welcome panel, which is shown in Figure 5-17.

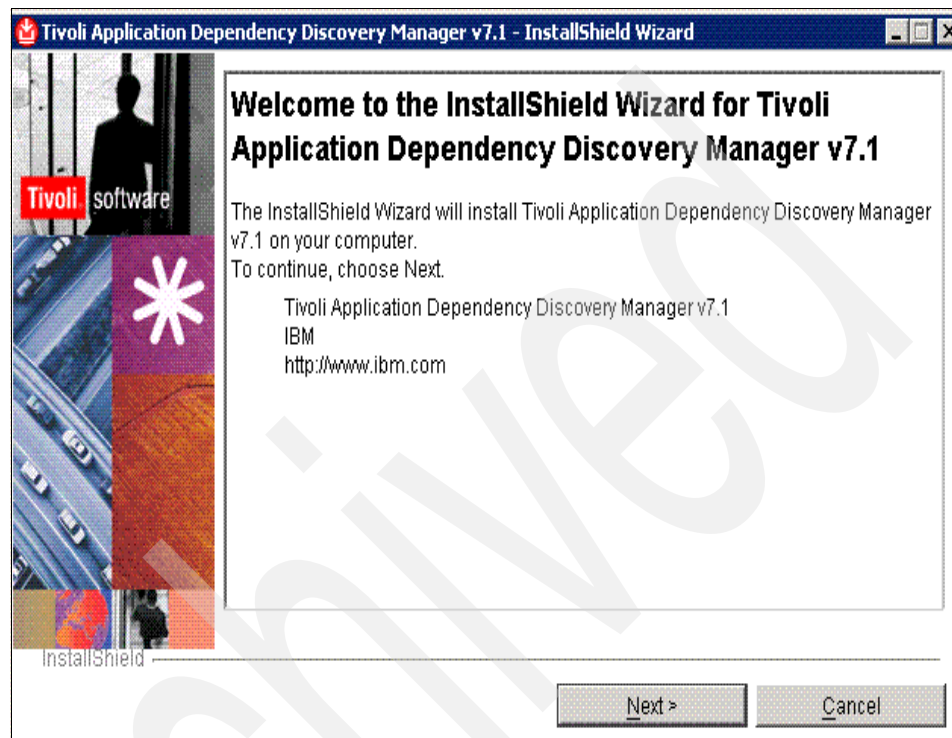


Figure 5-17 InstallShield Welcome panel

3. Click **Next**, and the International Program License Agreement panel, as shown in Figure 5-18 on page 110, is displayed.

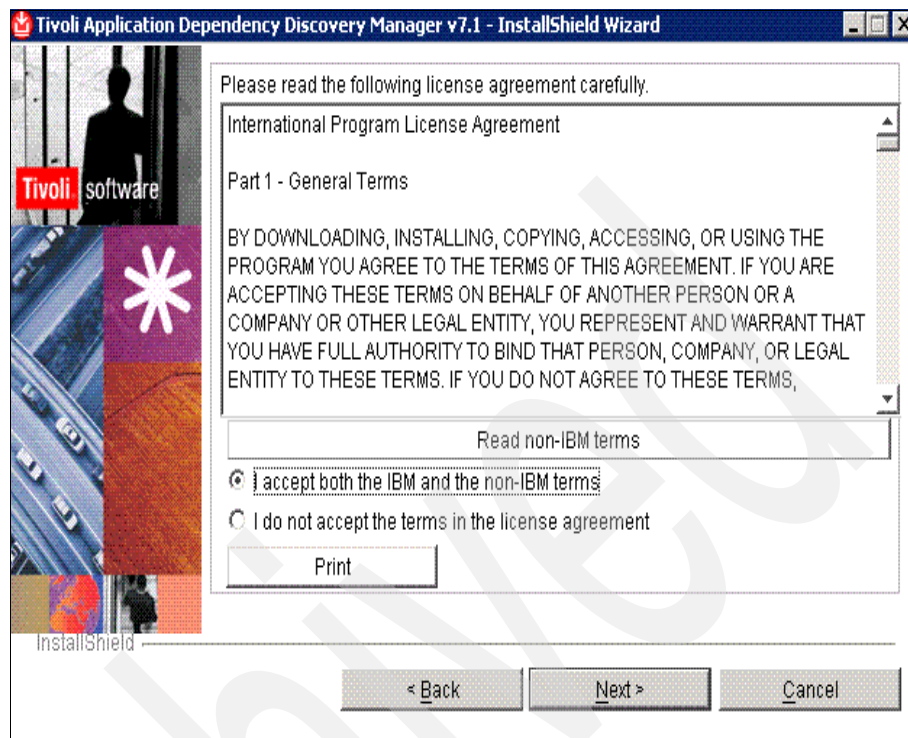


Figure 5-18 License Agreement

4. Read the licensing terms. If you agree to the licensing terms, click the **I accept both the IBM and the non-IBM terms**. You must accept the terms of the licensing agreement to continue the installation. Click **Next**, and the panel that is shown in Figure 5-19 on page 111 is displayed.

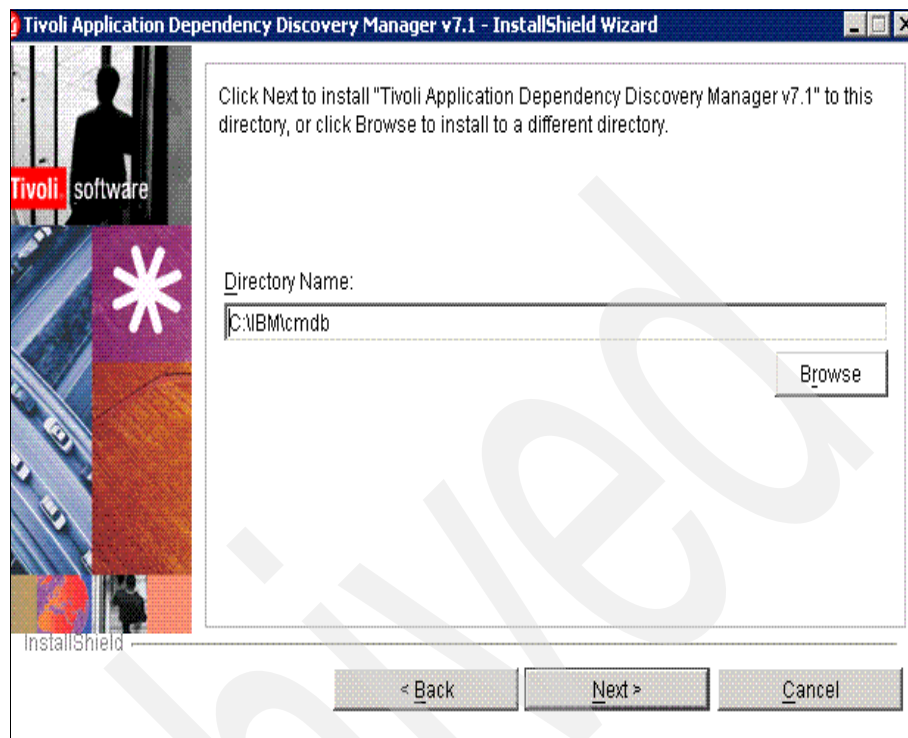


Figure 5-19 Installation directory

5. Enter the name of the directory where you want to install TADDM. Click **Next**, and the panel that is shown in Figure 5-20 on page 112 is displayed.

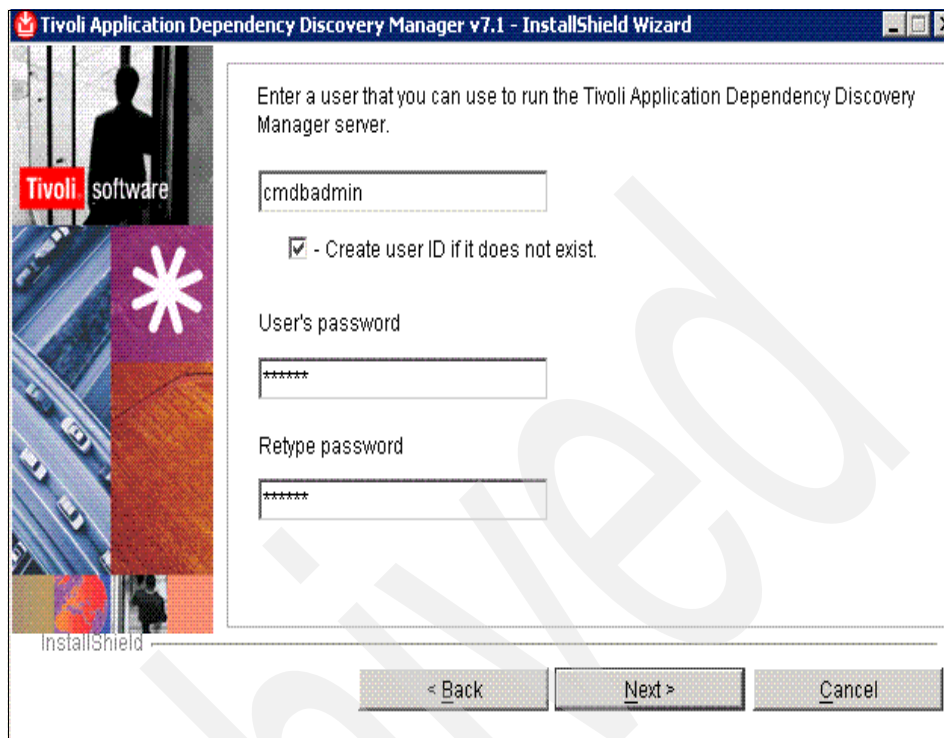


Figure 5-20 Defining a TADDM user

6. Enter the user that will run the TADDM Server. This user must have Administrator authority. If this user does not currently exist, you can have the InstallShield Wizard create the user account for you by checking **Create user ID if it does not exist**. Enter and confirm the password for the user, and click **Next**. The Choose the installation type panel that is shown in Figure 5-21 on page 113 appears.

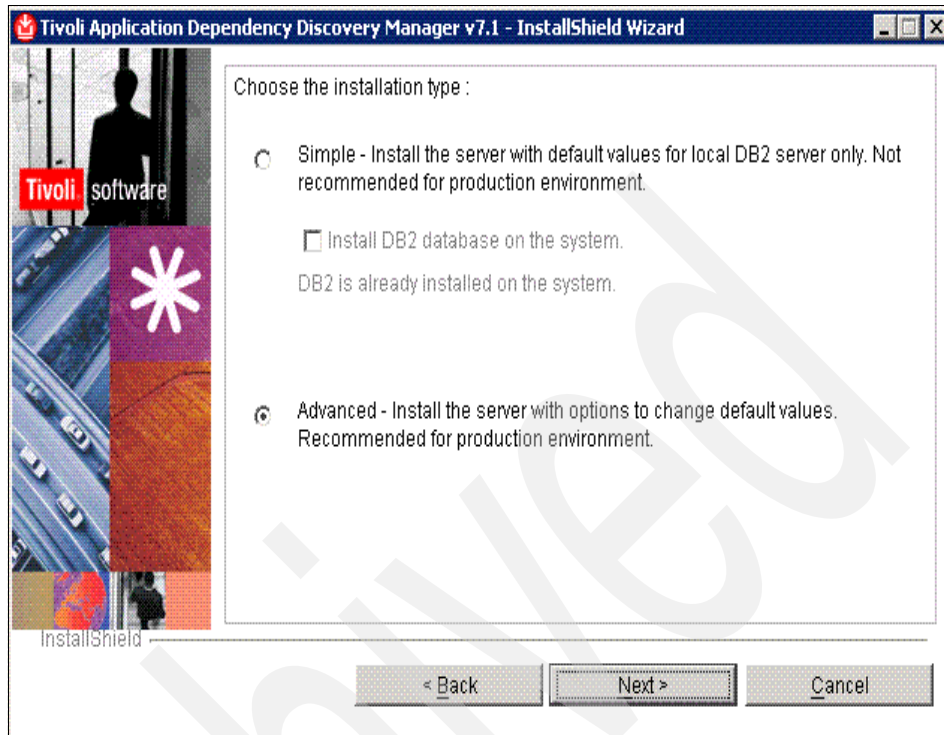


Figure 5-21 Choose the installation type

7. Choose the installation type: simple or advanced. A simple installation uses default values for a local DB2 database. We do not recommend using a simple installation for production environments.

We needed to use the advanced installation type, because we planned to use a remote database. Select **Advanced - Install the server with options to change default values. Recommended for production environment.** Click **Next**. The Select one of the following server types panel that is shown in Figure 5-22 on page 114 appears.

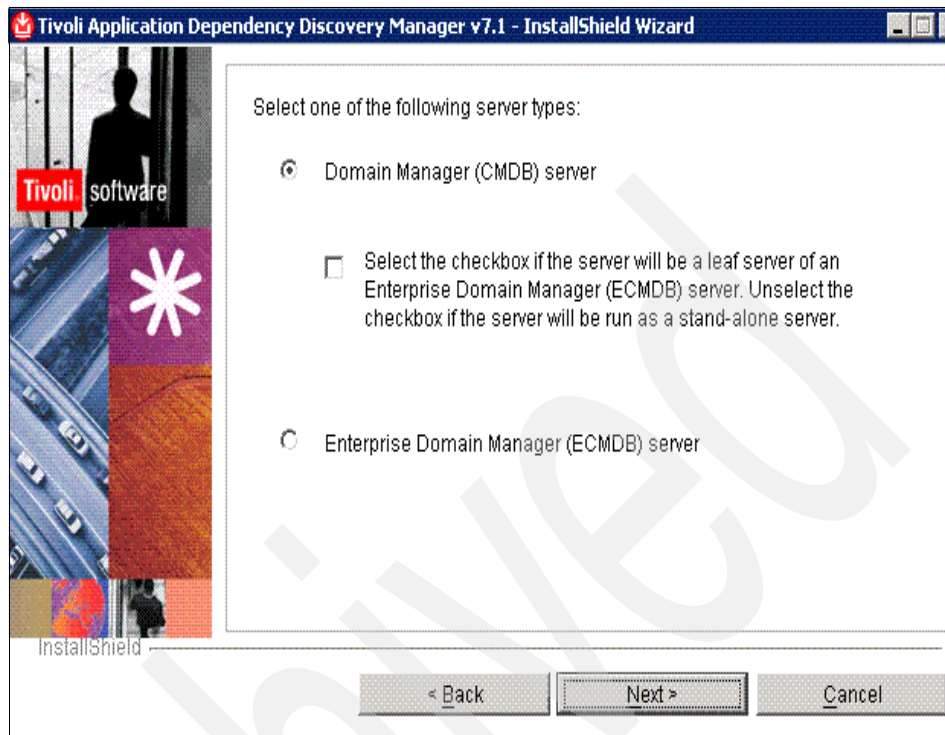


Figure 5-22 Select the server type

8. Select the server type for the TADDM Server that you are installing.

We installed a TADDM domain, so we selected **Domain Manager (CMDB) server**. In our case, this domain will be a leaf node of an enterprise domain manager; however, we chose not to select the check box signifying that this server will be a leaf server of an ECMDB server.

Clicking **Next** displays the panel that is shown in Figure 5-23 on page 115.

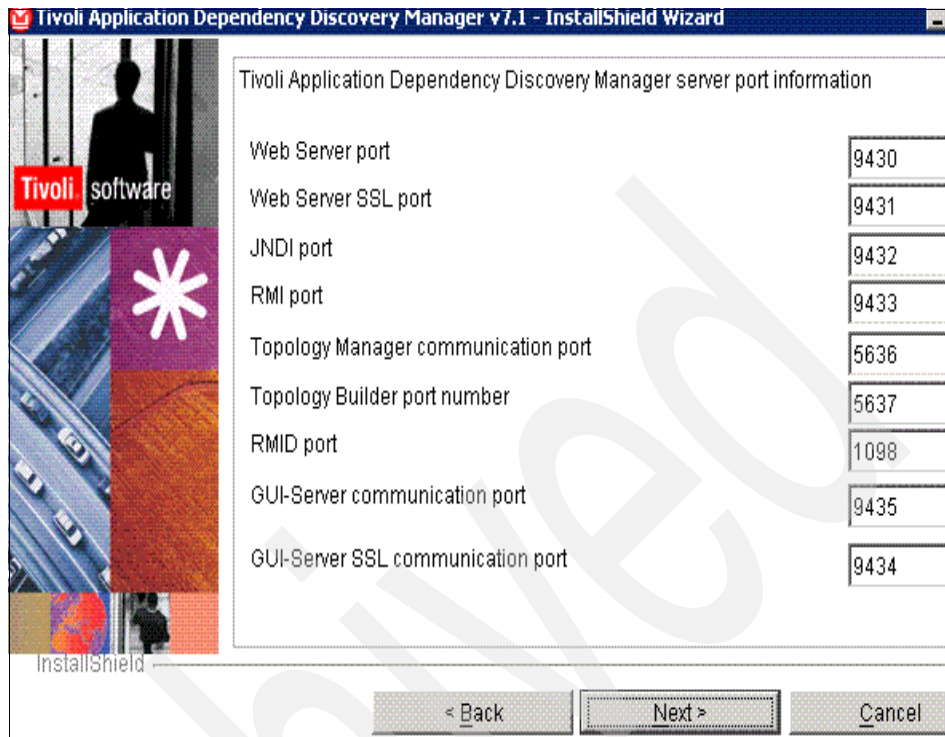


Figure 5-23 TADDM Server port information

9. Review the default port information and change any values that need to be changed. We accepted all of the default port values.

Click **Next**. The panel, which is shown in Figure 5-24 on page 116, is used to gather information about additional ports that will be used only for TADDM domains that are attached to a TADDM enterprise server.

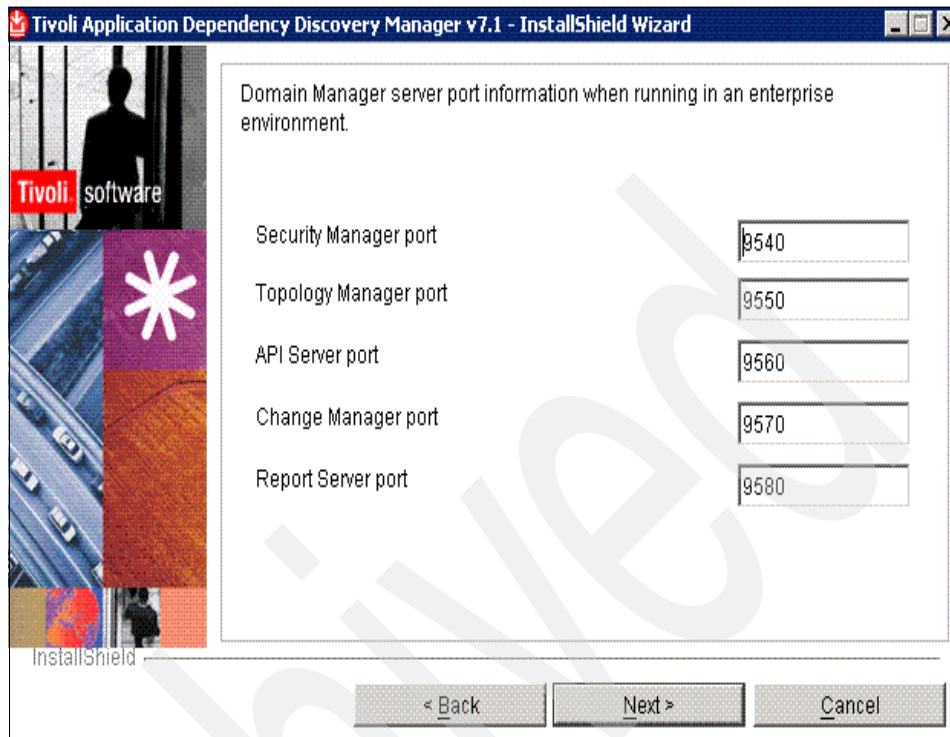


Figure 5-24 Additional server ports when running in an enterprise environment

10. Review the default port information and change any ports that need to be changed. We accepted all of the default ports.

Clicking **Next** displays the panel that is shown in Figure 5-25 on page 117.

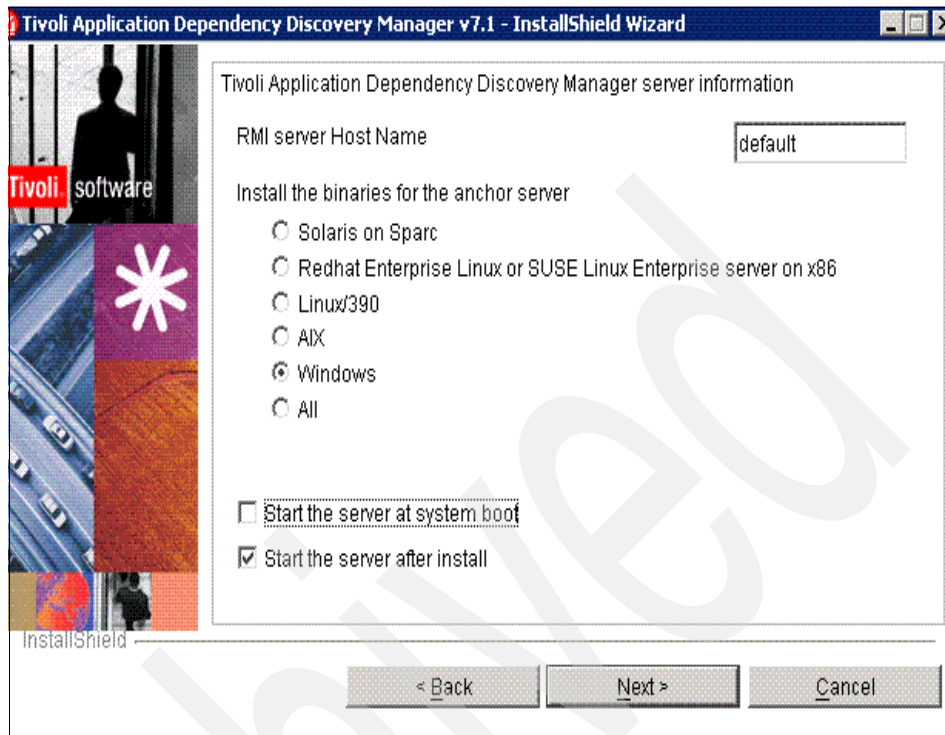


Figure 5-25 Specifying Remote Method invocation (RMI) information

11. You must enter several types of information on this panel. First, enter the host name for the RMI server. The default value is `default`. Use the default value if the RMI server resides on the same system as the Change and Configuration Management Database (CCMDB). If not, enter the IP address (not the host name) of the RMI server.

Next, select the platform binaries that you want to install. These binaries will be copied to Windows gateways and anchor servers. If you know the OSs on which those gateways and anchors will run, choose only the binaries for those OS platforms. If you are unsure of the OSs on which the gateways and anchors will run, select all of the platforms.

To start this TADDM Server when the system is started, select **Start the server at system boot**.

To start the server after the installation of TADDM is complete, select **Start the server after install**.

Click **Next**, and the optional CCMDB host name and port panel, which is shown in Figure 5-26 on page 118, is displayed.

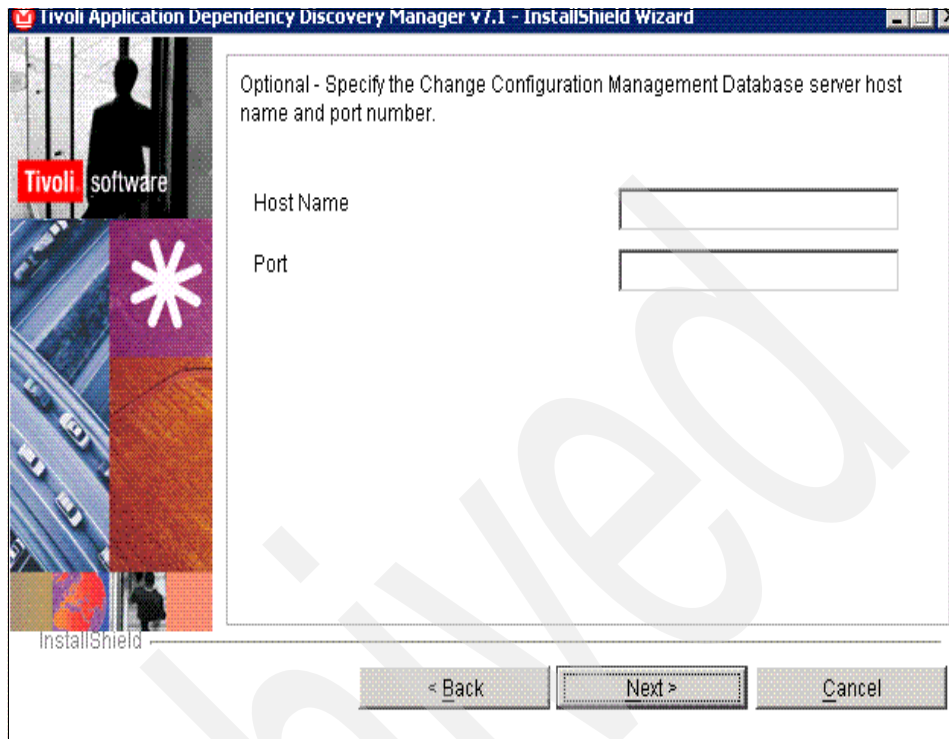


Figure 5-26 Optional CCMDB host name and port

12. This panel asks for the host name and port number of the Change and Configuration Management Database (CCMDB). We left this panel blank, because our implementation did not include a CCMDB.

Click **Next**, and the Select the database type panel, which is shown in Figure 5-27 on page 119, is displayed.

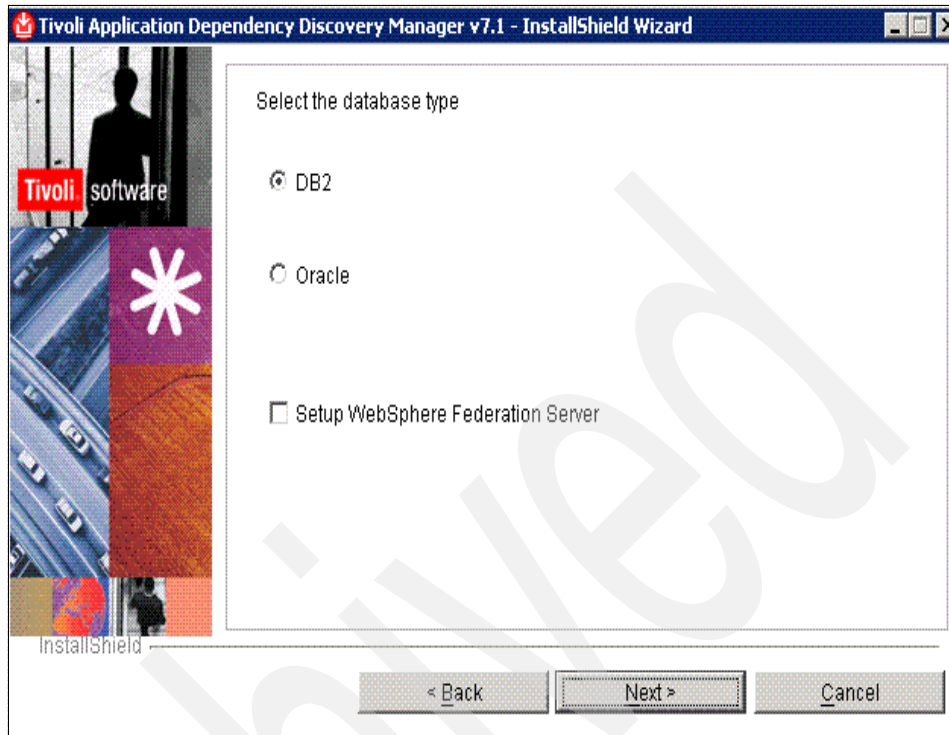


Figure 5-27 Select the database type

13. Select the type of database for your TADDM Database. We are using DB2, so we selected **DB2**.

We chose not to set up the WebSphere Federation Server; therefore, we did not check the Setup WebSphere Federation Server check box.

Click **Next**, and the panel shown in Figure 5-28 on page 120 is displayed.

Tivoli Application Dependency Discovery Manager v7.1 - InstallShield Wizard

DB2 configuration information

Database Host Name: copenhagen

Database Port: 50020

DB2 Instance/node name:

Database Name: cmdb

DB2 Instance User ID: taddmwin

DB2 Instance User Password: *****

Additional User ID for database access: archwin

Additional Password for database access: *****

☐ Create the database during install. Local DB2 only

Help with create database

< Back Next > Cancel

Figure 5-28 Database configuration information

14. This panel is asking for configuration information about the database. We created our database prior to the installing TADDM. Enter the database information here.

Click **Next**, and the Select the User Registry Option panel, which is shown in Figure 5-29 on page 121, is displayed.

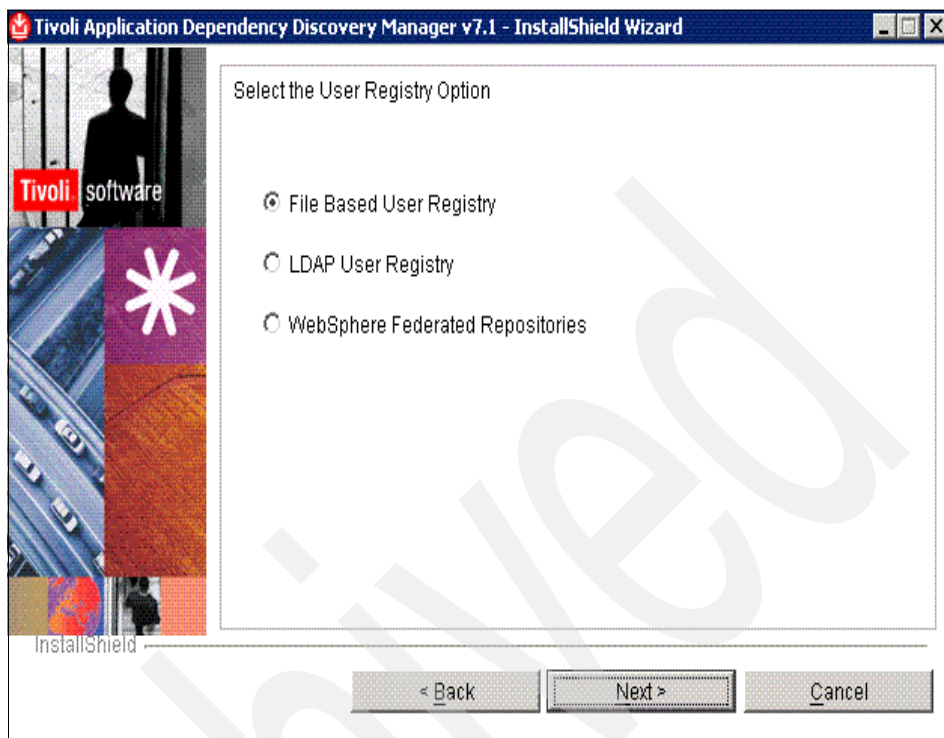


Figure 5-29 Select the user registry option

15. Select the option for the user registry that will be used with TADDM.

For our implementation, we use the file-based user registry at the domain level, but we will use Lightweight Directory Access Protocol (LDAP) at the enterprise level. For normal operations, the domains are connected to the enterprise, and user authentication is done through LDAP. If the connection to the enterprise is down for any reason, user authentication is done through the file-based user registry. Because this is a domain server that we are installing, we selected **File Based User Registry**.

Click **Next**, and the summary panel, which is shown in Figure 5-30 on page 122, appears.

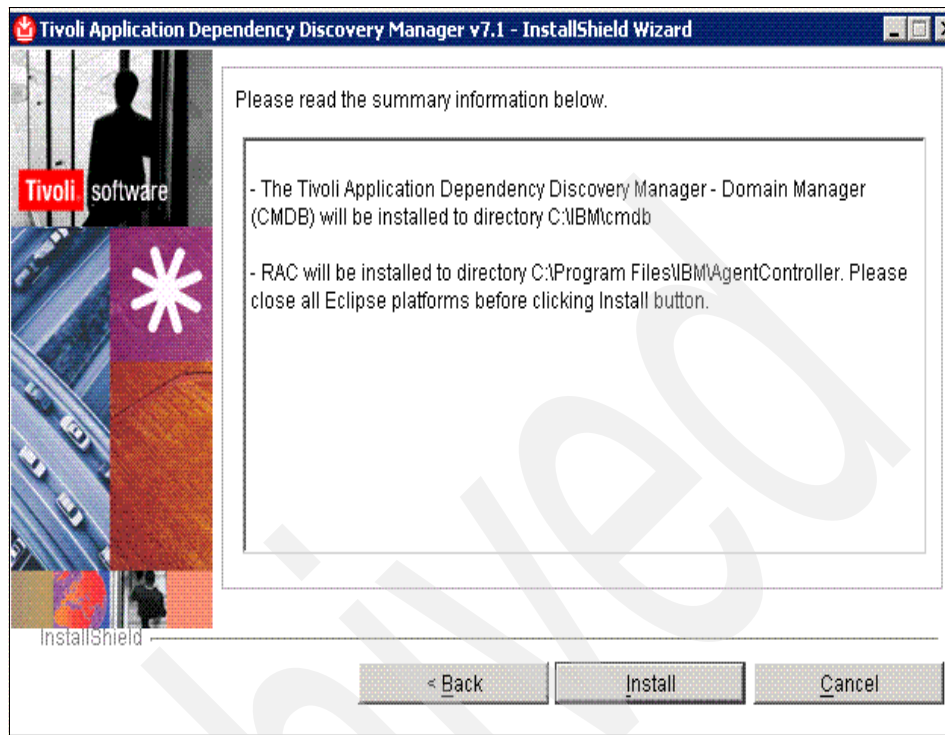


Figure 5-30 Summary information

16. Review the summary information, and if the information is correct, click **Next** to begin the installation.

When the installation completes, the panel that is shown in Figure 5-31 on page 123 is displayed.

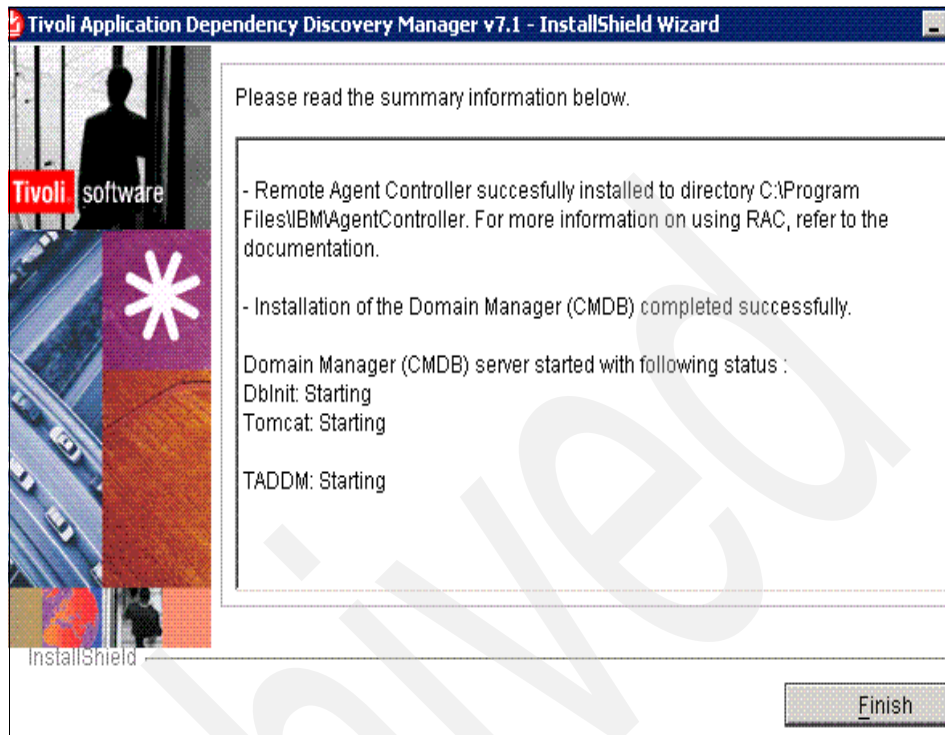


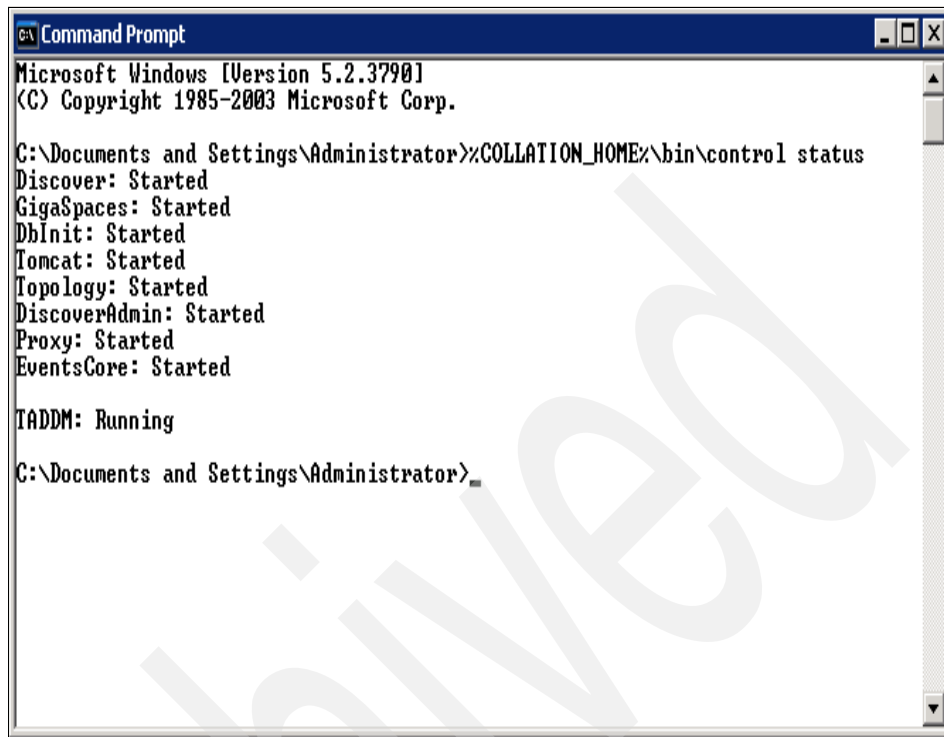
Figure 5-31 Installation completion

17. Review the installation summary information, and click **Finish**. The installation of the TADDM Domain Server on Windows is now complete.
18. Before proceeding, we recommend that you verify that all of the TADDM services are running and that you can log on to the Product Console:

- a. To verify that all of the TADDM services are running, log on to the TADDM Server and issue the following command:

```
%COLLATION_HOME%\bin\control status
```

Figure 5-32 on page 124 shows the command and expected output.



```
Microsoft Windows [Version 5.2.3790]
(C) Copyright 1985-2003 Microsoft Corp.

C:\Documents and Settings\Administrator>%COLLATION_HOME%\bin\control status
Discover: Started
GigaSpaces: Started
DbInit: Started
Tomcat: Started
Topology: Started
DiscoverAdmin: Started
Proxy: Started
EventsCore: Started

TADDM: Running

C:\Documents and Settings\Administrator>
```

Figure 5-32 The control status command and output

- b. Verify that the Product Console will come up successfully by bringing up your browser and entering the URL with the host name of the Windows server where TADDM was installed and port number 9430. For example:

`http://southend:9430`

The Tivoli Application Dependency Discovery Manager page, similar to Figure 5-33 on page 125, is displayed.

The default user name and password combination is administrator for the user name and collation for the password.

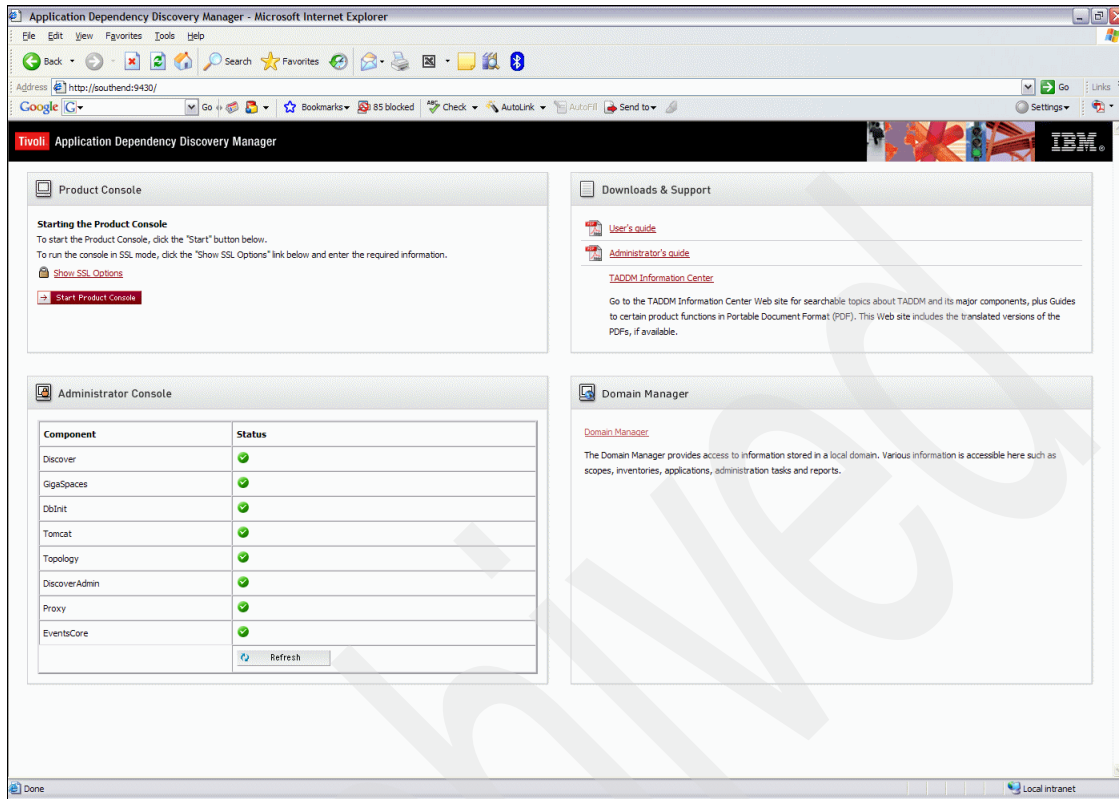


Figure 5-33 Tivoli Application Dependency Discovery Manager page

Notice that the TADDM components and their associated status are displayed in the lower left corner of the panel. Clicking Refresh will refresh the status. All of the statuses need to be green.

Click **Start Product Console**, and when prompted, enter a valid user name and password. The Product Console panel, which is similar to Figure 5-34 on page 126, is displayed.

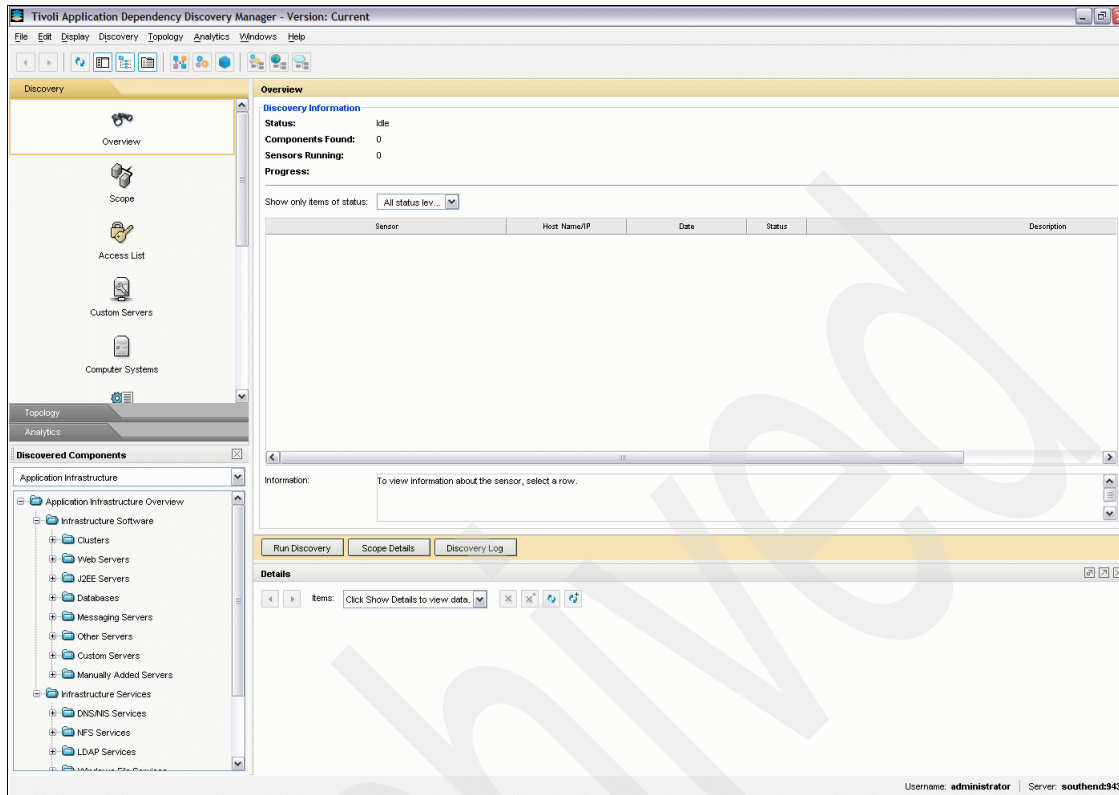


Figure 5-34 Product Console

5.3.2 Install interim fix 0007

At the time of writing this book, interim fix 0007 (IF0007) was the latest interim fix available for TADDM 7.1. To apply IF0007, obtain the image from the IBM support site and load it onto the Windows server. Log on to the Windows server with a user ID that can create and write access %TADDM_HOME%, go to the directory where the IF0007 code has been copied, and type the command along with the directory where TADDM 7.1 is installed. For example:

```
C:\code\7.1.0.0 TIV IF0007\taddm_IF>installIF.bat C:\IBM\cmdb
```

5.4 Installing a TADDM Domain Server on Linux

We performed the following steps to install a TADDM Domain Server on Linux.

5.4.1 Install TADDM 7.1

To complete the installation of TADDM with a remote database, complete the following steps:

1. Log on to the Linux system as the root user.
2. Locate the installation media and copy it to the Linux system. We copied the installation media to the `/root/code/TADDM_V710/TADDM` directory.
3. Go to the directory where you copied the installation media and run the `setupLinux.bin` command, for example:

```
[root@waco]# . setupLinux.bin
```

Issuing the `setupLinux.bin` command displays the InstallShield Wizard Welcome panel, which is shown in Figure 5-35.



Figure 5-35 InstallShield Wizard Welcome panel

4. Click **Next**, and the International Program License Agreement panel, which is shown in Figure 5-36 on page 128, is displayed.

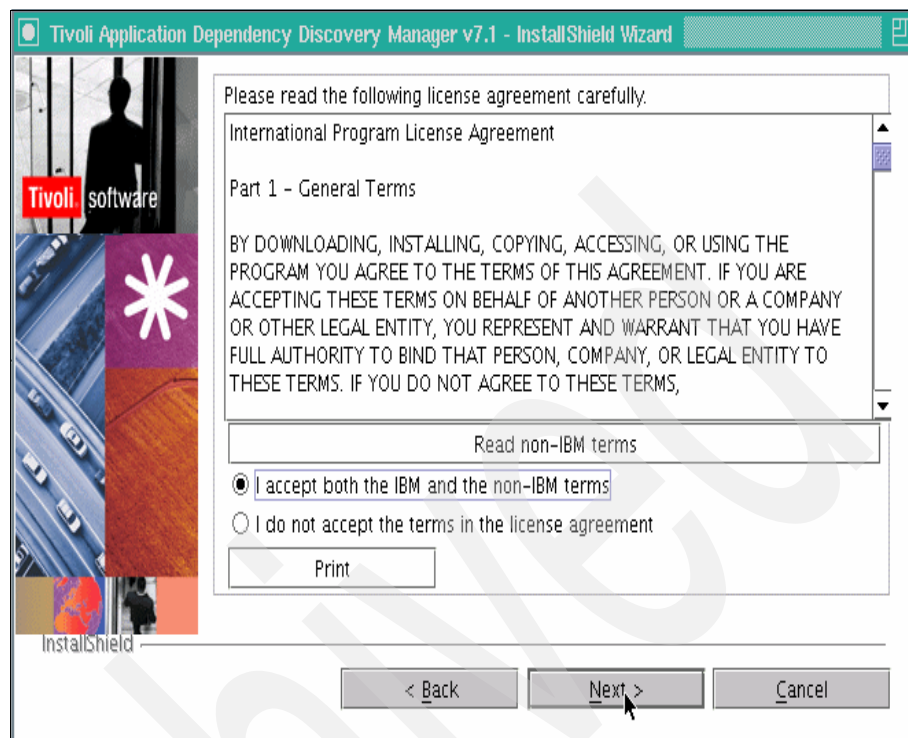


Figure 5-36 License Agreement

5. Read the licensing terms, and if you agree to the licensing terms, select **I accept both the IBM and the non-IBM terms**. You must accept the terms of the licensing agreement in order to continue the installation. Click **Next** and the panel that is shown in Figure 5-37 on page 129 is displayed.

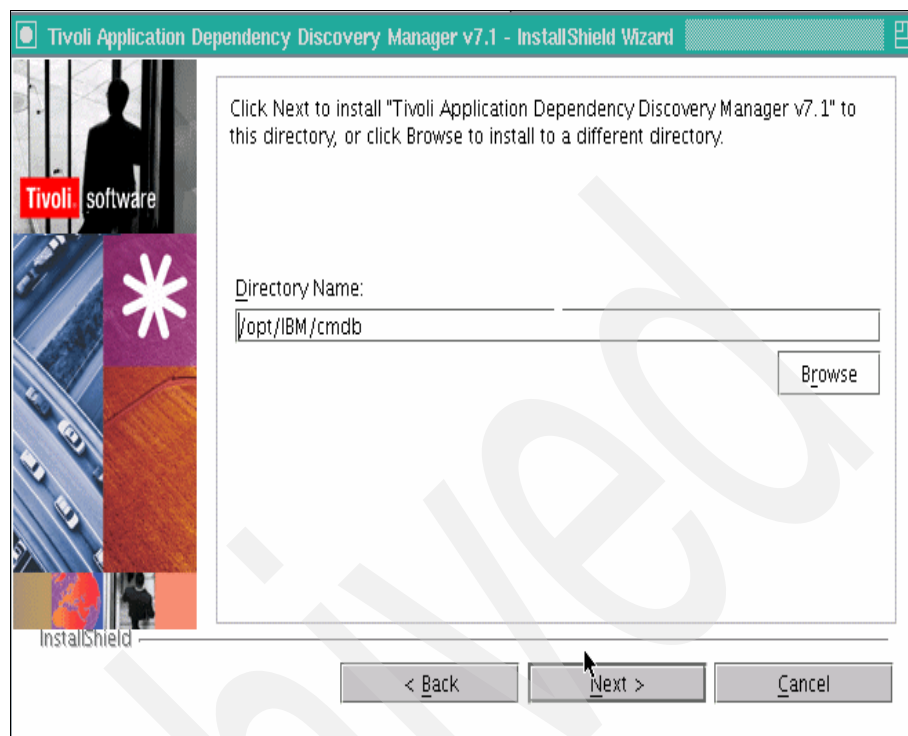


Figure 5-37 Installation directory

6. Enter the name of the directory where you want TADDM to be installed. Click **Next**, and the panel shown in Figure 5-38 on page 130 is displayed.

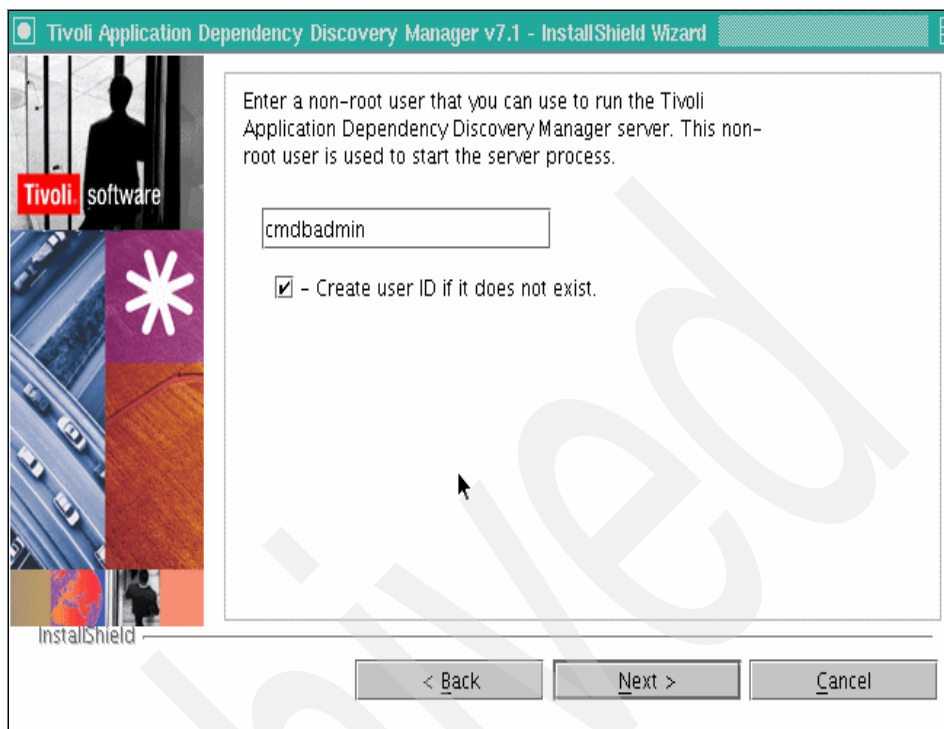


Figure 5-38 Defining a TADDM user

7. Enter the user that you want to start the TADDM Server process. This user must be a non-root user. If this user does not currently exist, you can have the InstallShield Wizard create the user account for you by checking “Create user ID if it does not exist”. Click **Next**. The Choose the installation type panel, which is shown in Figure 5-39 on page 131, is displayed.

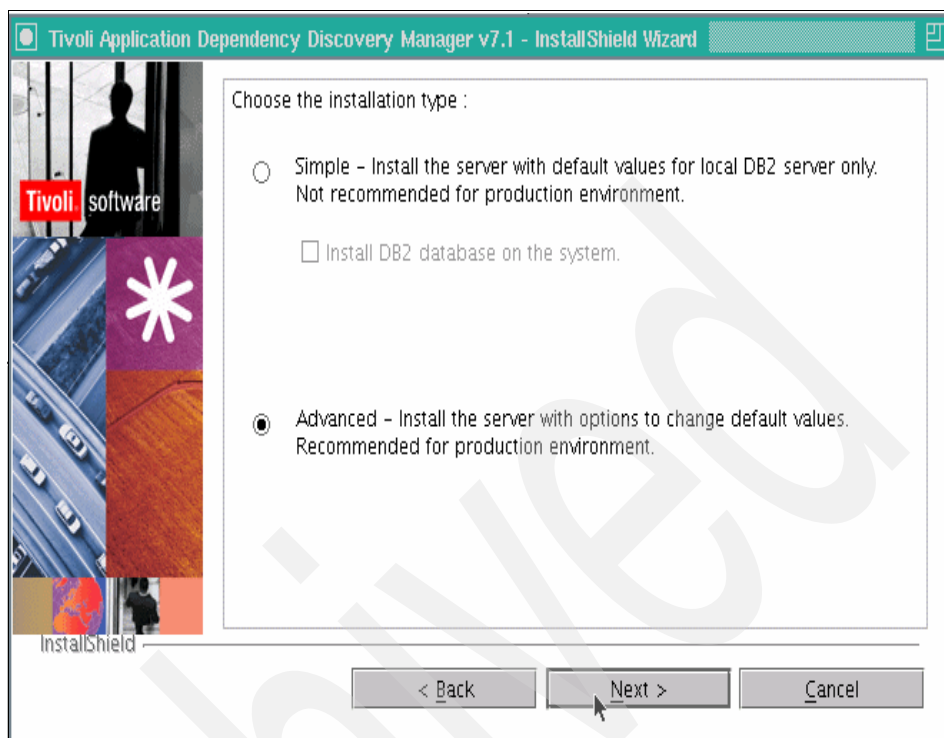


Figure 5-39 Choose the installation type

8. Choose the installation type: simple or advanced. A simple installation uses default values for a local DB2 database. We do not recommend that you use a simple installation for production environments.

We need to use the advanced installation type, because we plan to use a remote database. Select **Advanced - Install the server with options to change default values. Recommended for production environment**, and click **Next**. The Select the server type panel that is shown in Figure 5-40 on page 132 appears.

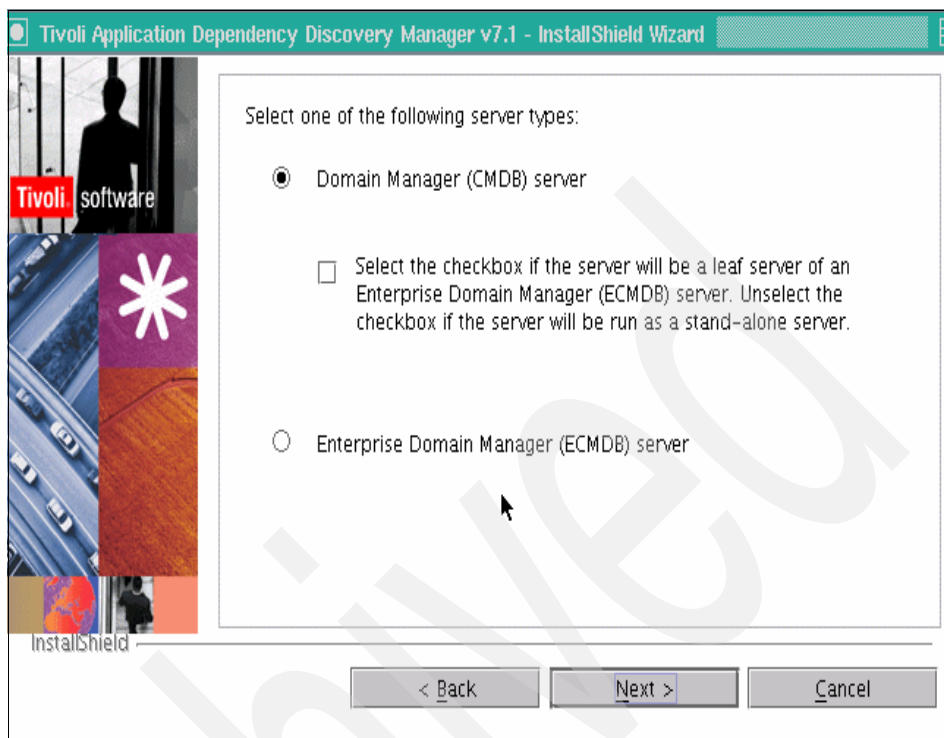


Figure 5-40 Select the server type

9. Select the server type for the TADDM Server that is being installed.

We were installing a TADDM domain, so we selected **Domain Manager (CMDB) server**. This domain will be a leaf node of an enterprise domain manager; however, we chose not to select the check box.

Clicking **Next** will display the panel that is shown in Figure 5-41 on page 133.

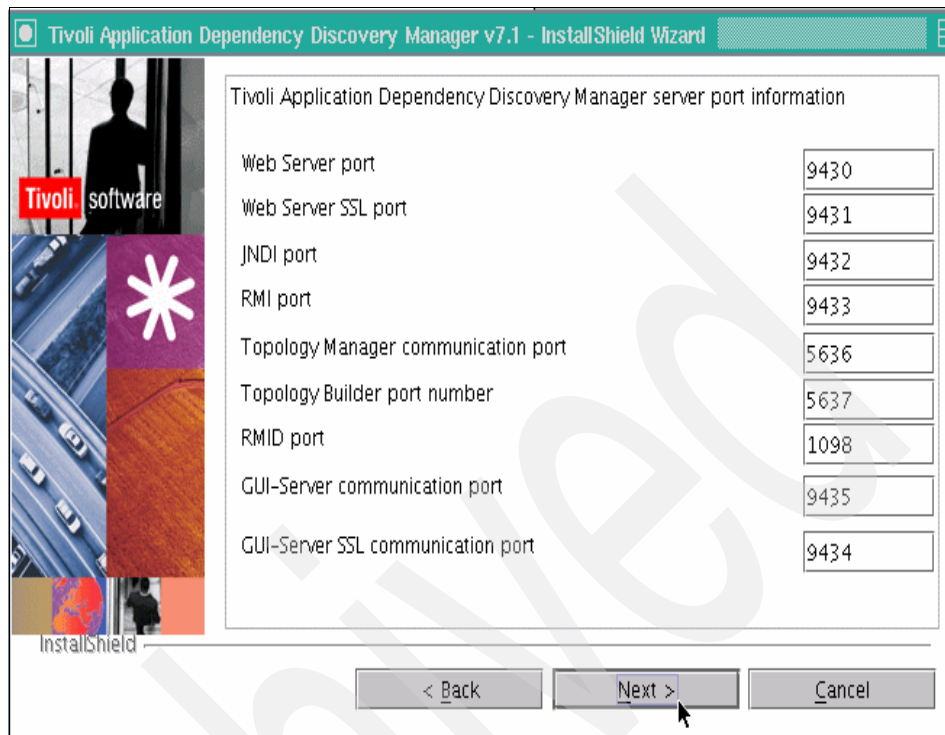


Figure 5-41 TADDM Server port information

10. Review the default port information and change any port numbers that need to be changed. We accepted all of the default port values.

Click **Next**. The panel that is shown in Figure 5-42 on page 134 is displayed to gather information about additional ports that are used only for TADDM domains that will be attached to a TADDM enterprise server.

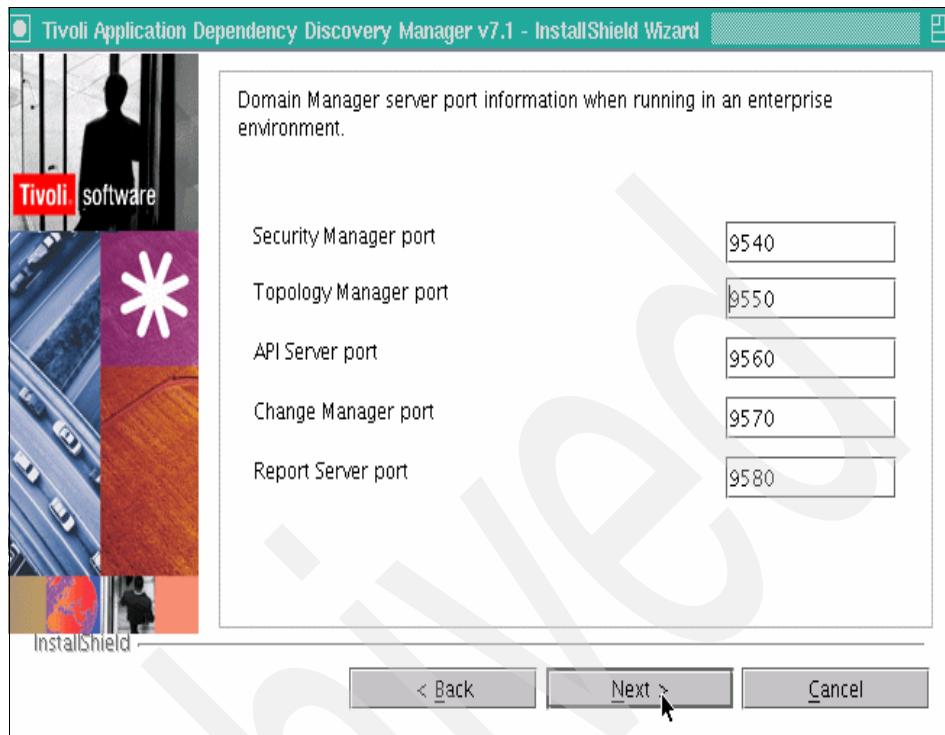


Figure 5-42 Additional server ports when running in an enterprise environment

11. Review the default port information and change any ports that need to be changed. We accepted all the default ports.

Clicking **Next** displays the panel that is shown in Figure 5-43 on page 135.

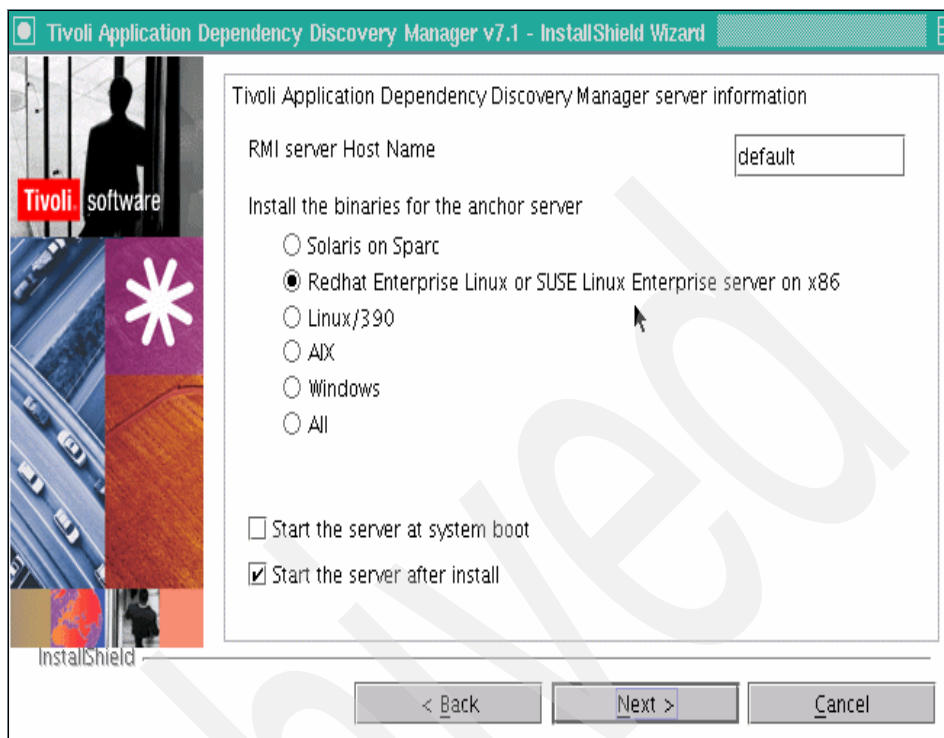


Figure 5-43 Specifying RMI information

12. There are a couple pieces of information that you must enter on this panel. You must enter the host name for the RMI server. The default value is default. Use the default value if the RMI server resides on the same system as the CCMDDB; if not, enter the IP address (not the host name) of the RMI server.

Next, select the platform binaries that you want installed. These binaries will be copied to Windows gateways and anchor servers. If you know the operating systems (OSs) on which those gateways and anchors will run, choose only the binaries for those platforms. However, if you are unsure about the OSs, select all of the platforms.

To start this TADDM Server when the system is started, select **Start the server at system boot**.

To start the server after the installation of TADDM is complete, select **Start the server after install**.

Click **Next**, and the optional CCMDDB host name and port panel that is shown in Figure 5-44 on page 136 is displayed.

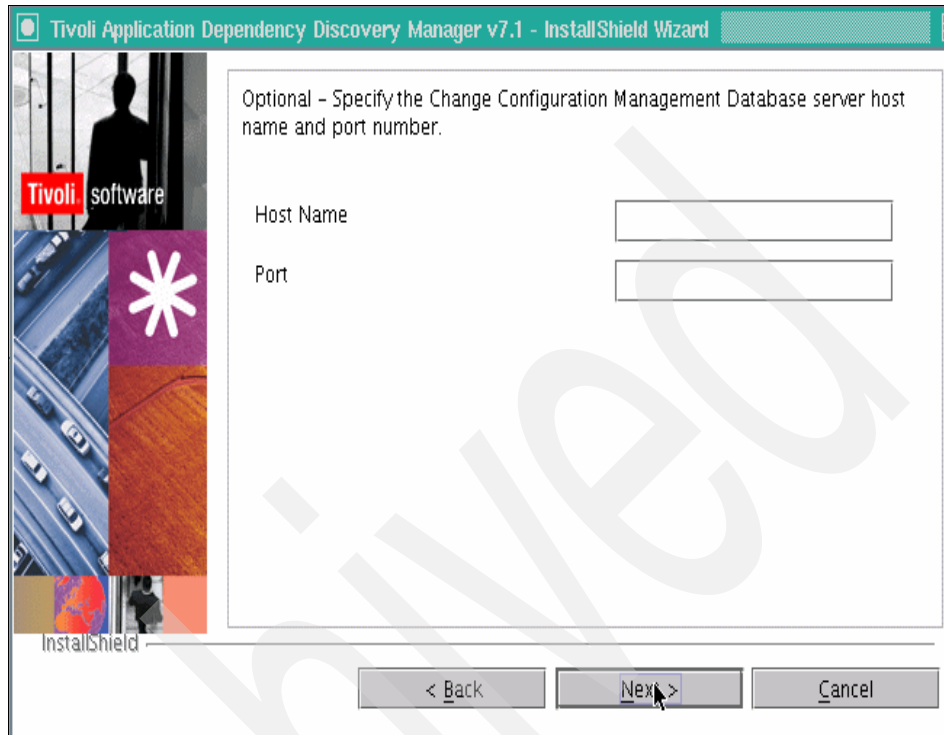


Figure 5-44 Optional CCMDB host name and port

13. This panel asks for the host name and port number of the Change and Configuration Management Database (CCMDB). We left this panel blank, because our implementation did not include a CCMDB .

Click **Next**, and the Select the database type panel, which is shown in Figure 5-45 on page 137, is displayed.

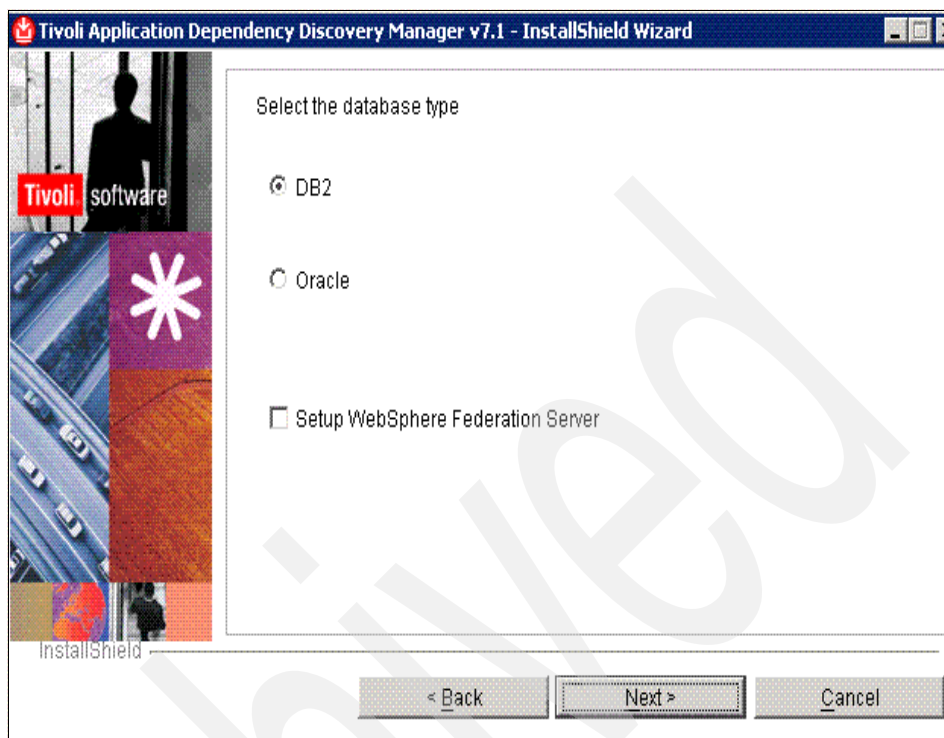


Figure 5-45 Select the database type

14. Select the database type for your TADDM Database. We used DB2; therefore, we selected **DB2**.

We chose not to set up the WebSphere Federation Server; therefore, we did not check the Setup WebSphere Federation Server check box.

Click **Next**, and the panel that is shown in Figure 5-46 on page 138 is displayed.

Tivoli Application Dependency Discovery Manager v7.1 - InstallShield Wizard

DB2 configuration information

Database Host Name: copenhagen.itsc.austin.ibm.com

Database Port: 50010

Database node name (DB2 client only):

Database Name: cmdb

DB2 Instance User ID: taddmlin

DB2 Instance User Password: *****

Additional User ID for database access: archlin

Additional Password for database access: *****

☐ Create the database during install. Local DB2 only.

Help with create database

< Back Next > Cancel

Figure 5-46 Database configuration information

15. This panel asks for information about the database. We created our database prior to the installing TADDM. Enter the database information here.

Click **Next**, and the Select the User Registry Option panel, which is shown in Figure 5-47 on page 139, is displayed.

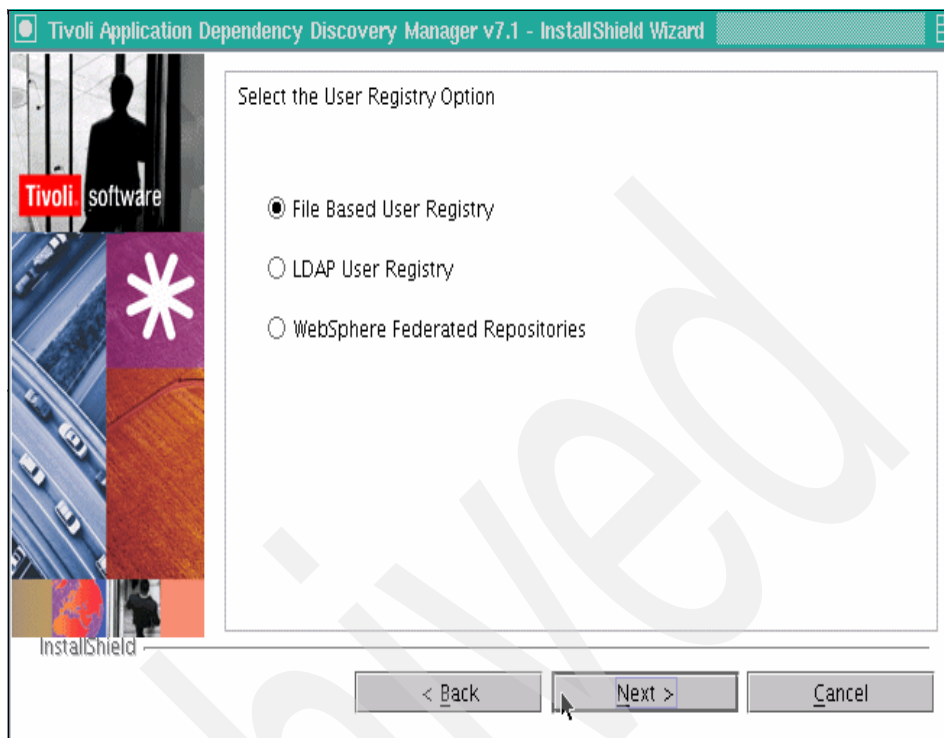


Figure 5-47 Select the user registry

16. Select the user registry option that will be used with TADDM.

For our implementation, we are using the file-based user registry at the domain level, but we will use LDAP at the enterprise level; therefore, for normal operations, the domains are connected to the enterprise, and user authentication is done using LDAP. If the connection to the enterprise is down for any reason, user authentication is done using the file-based user registry. We selected **File Based User Registry**, because we are installing a domain server.

Click **Next**, and the summary panel, which is shown in Figure 5-48 on page 140, is displayed.

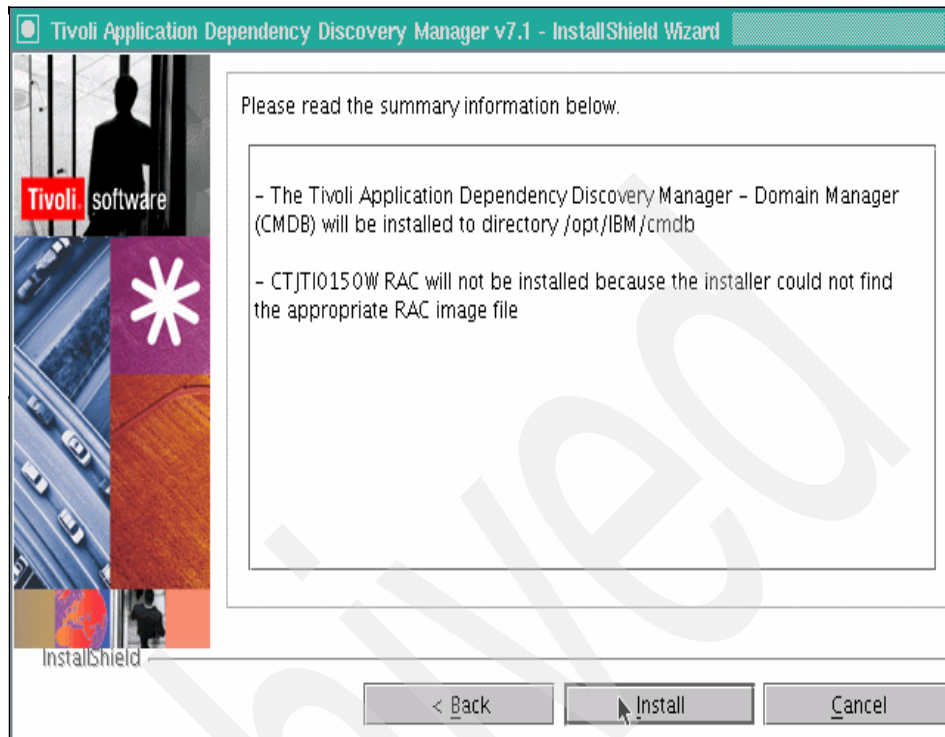


Figure 5-48 Summary information

17. Review the summary information, and if the information is correct, click **Next** to begin the installation.
18. When the installation completes, review the installation summary information, and click **Finish**. The installation of TADDM Domain Server on Windows is now complete.
19. Before proceeding, we recommend that you verify that all of the TADDM services are running and that you can log on to the Product Console:
 - a. To verify that all of the TADDM services are running, log on to the TADDM Server with a non-root user ID, change directories (**cd**) to the `$COLLATION_HOME/bin` directory, and issue the following command:

```
control status
```

Figure 5-49 on page 141 shows the command and the expected output.



```
cmdbadm@waco:/opt/IBM/cmdb/dist/bin
[root@waco bin]# su cmdbadm
[cmdbadm@waco bin]$ pwd
/opt/IBM/cmdb/dist/bin
[cmdbadm@waco bin]$ ./control status
Discover: Started
DbInit: Started
GigaSpaces: Started
Tomcat: Started
Topology: Started
DiscoverAdmin: Started
Proxy: Started
EventsCore: Started

TADDM: Running
[cmdbadm@waco bin]$
```

Figure 5-49 The control status command and output

- b. Verify that the Product Console displays successfully by bringing up your browser and entering a URL with the host name of the Linux server where TADDM was installed and port number 9430. For example:

`http://waco:9430`

The Tivoli Application Dependency Discovery Manager page, which is similar to Figure 5-50 on page 142, is displayed.

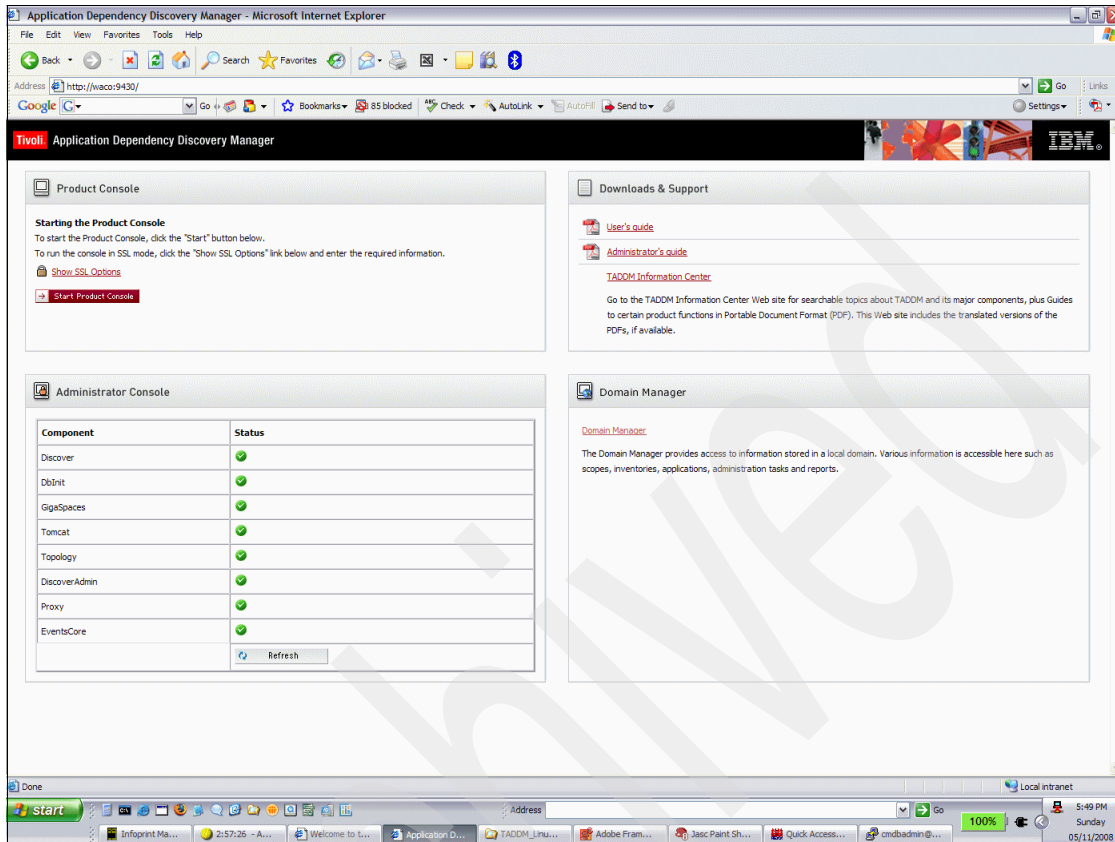


Figure 5-50 Tivoli Application Dependency Discovery Manager page

Notice that the TADDM components and their associated status is displayed in the lower left corner of the panel. Clicking Refresh refreshes the status. All of the statuses need to be green.

20. Click **Start Product Console**, and when prompted, enter a valid user name and password. The Product Console panel, similar to the panel shown in Figure 5-51 on page 143, is displayed.

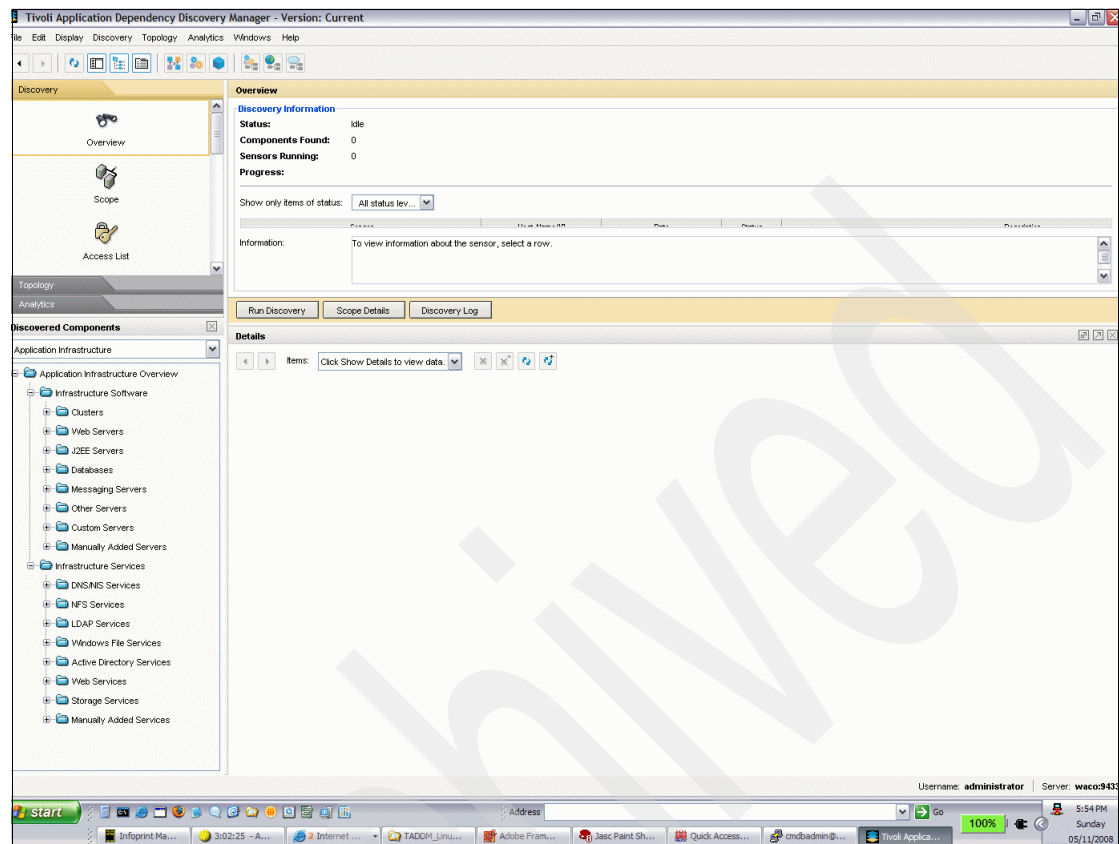


Figure 5-51 Product Console

5.4.2 Install interim fix 0007

At the time of writing this book, interim fix 0007 (IF0007) was the latest interim fix that was available for TADDM 7.1. To apply IF0007, obtain the image from the IBM support site and load it onto the Linux server. Log on to the Linux server with a user ID that has Administrator access, go to the directory where the IF0007 code has been copied, and type the command along with the directory where TADDM 7.1 exists. For example:

```
cd /root/code/ITADDM-IF0007/taddm_IF
installIF.sh /opt/IBM/cmdb
```

5.5 Installing a TADDM enterprise server on AIX

In this section, we guide you through a TADDM enterprise server installation on AIX.

5.5.1 Install TADDM 7.1

To complete the installation of TADDM with a remote database, complete the following steps:

1. Log in to the AIX system as the root user.
2. Locate the installation media and copy it to the AIX system. We copied the installation media to the /code/TADDM_V710/TADDM directory.
3. Go to the directory where you copied the installation media and run the **setupAix.bin** command, for example:

```
# ./setupAix.bin
```

Issuing the **setupAIX.bin** command displays the InstallShield Wizard Welcome panel, which is shown in Figure 5-52 on page 145.

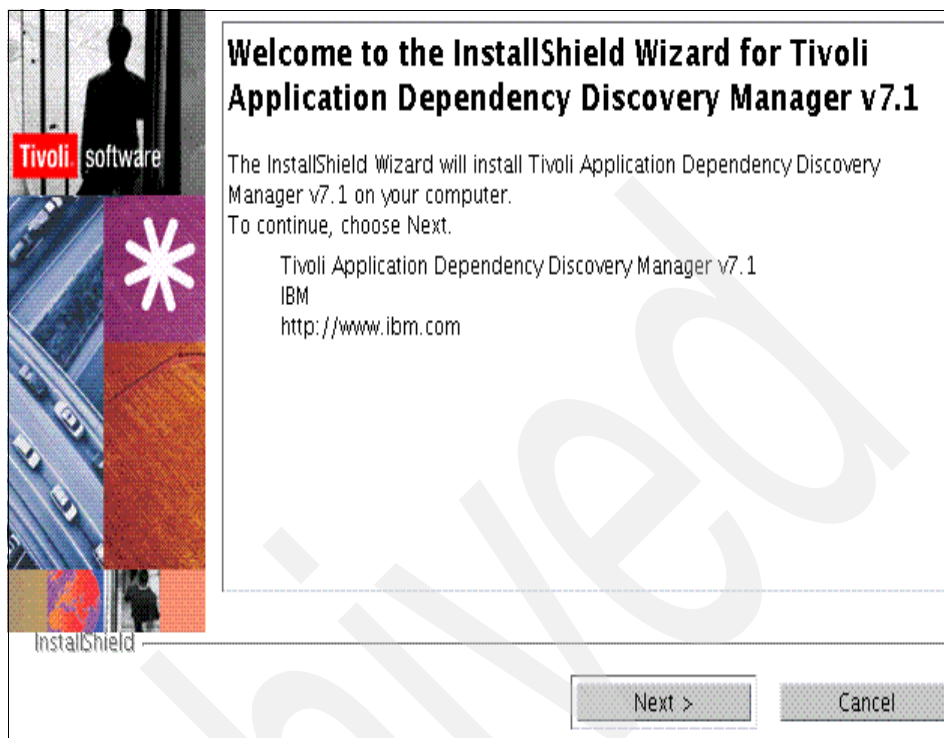


Figure 5-52 InstallShield Wizard Welcome panel

4. Click **Next**, and the International Program License Agreement panel, which is shown in Figure 5-53 on page 146, is displayed.

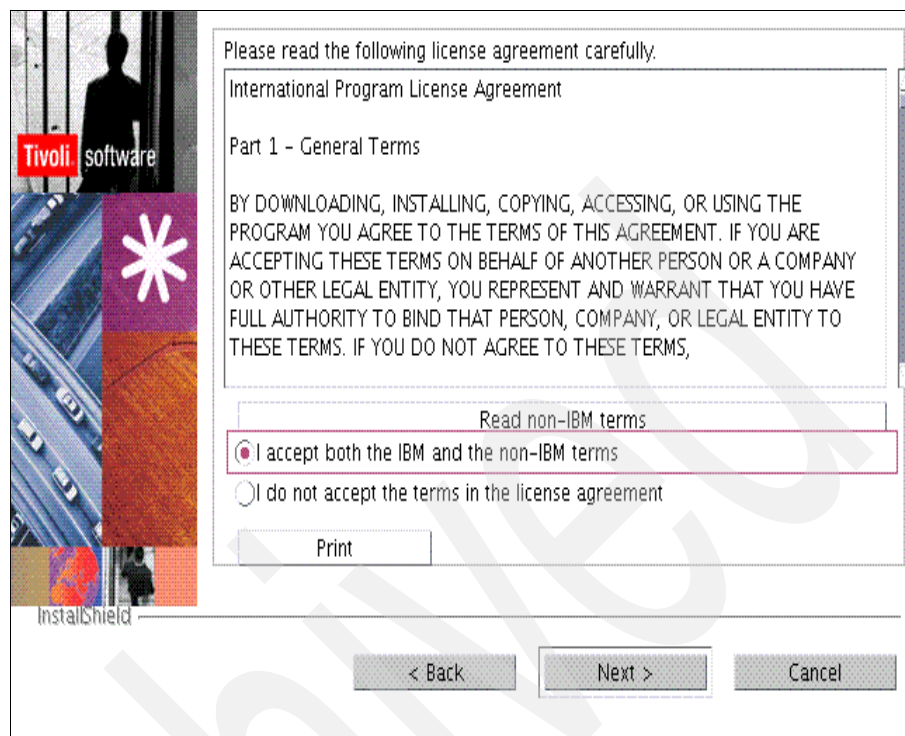


Figure 5-53 License Agreement

5. Read the licensing terms, and if you agree to the licensing terms, click **I accept both the IBM and the non-IBM terms**. You must accept the terms of the licensing agreement in order to continue the installation. Click **Next**, and the panel that is shown in Figure 5-53 is displayed.

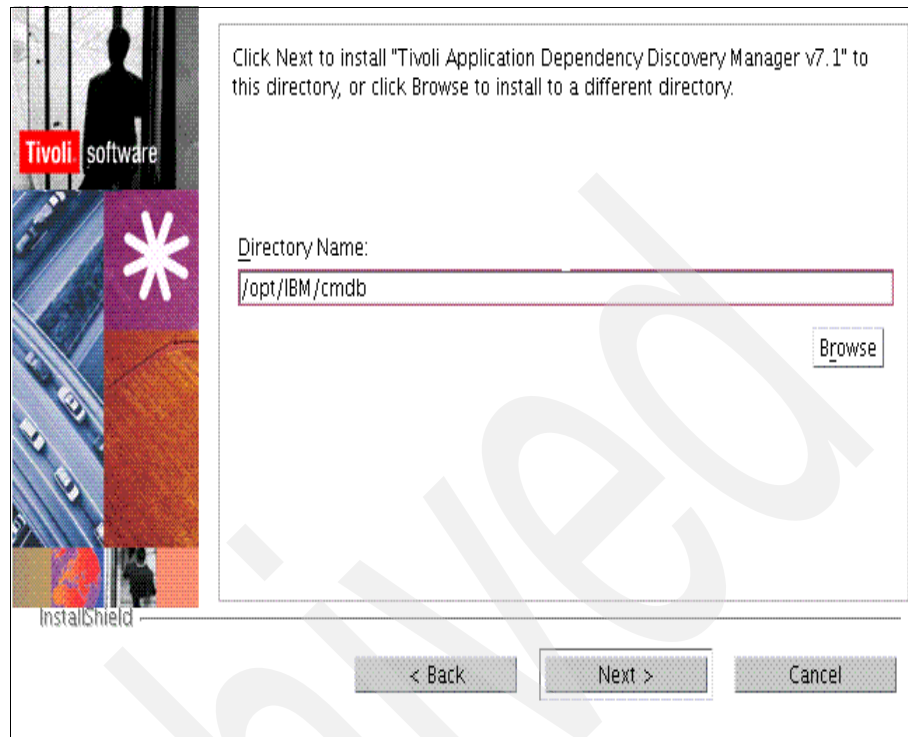


Figure 5-54 Installation directory

6. Enter the name of the directory where you want to install TADDM. Click **Next**, and the panel that is shown in Figure 5-55 on page 148 is displayed.



Figure 5-55 Defining a TADDM user

7. Enter the user that you want to start the TADDM Server process. This user must be a non-root user. If this user does not exist, you can request that the InstallShield Wizard create the user account for you by checking **Create user ID if it does not exist**, and clicking **Next**. The Choose the installation type panel that is shown in Figure 5-56 on page 149 is displayed.

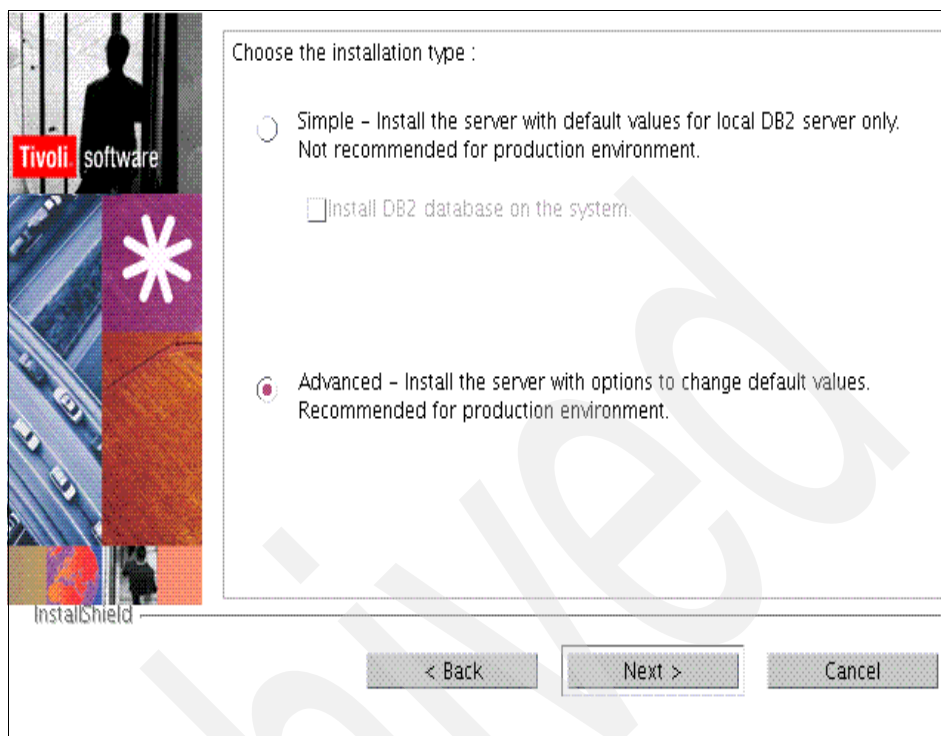


Figure 5-56 Choose the installation type

8. Choose the installation type: simple or advanced. A simple installation uses default values for a local DB2 database. We do not recommend that you use a simple installation for production environments.

We need to use the advanced installation type, because we plan to use a remote database. Select **Advanced - Install the server with options to change default values. Recommended for production environment**. Click **Next**, which displays the Select the server type panel that is shown in Figure 5-57 on page 150.

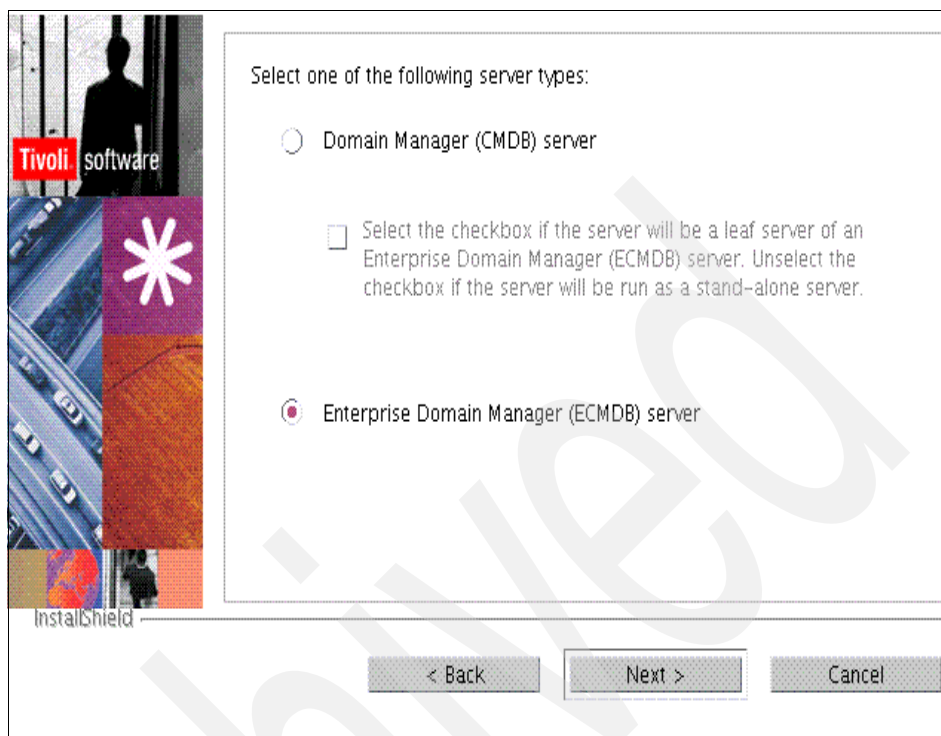
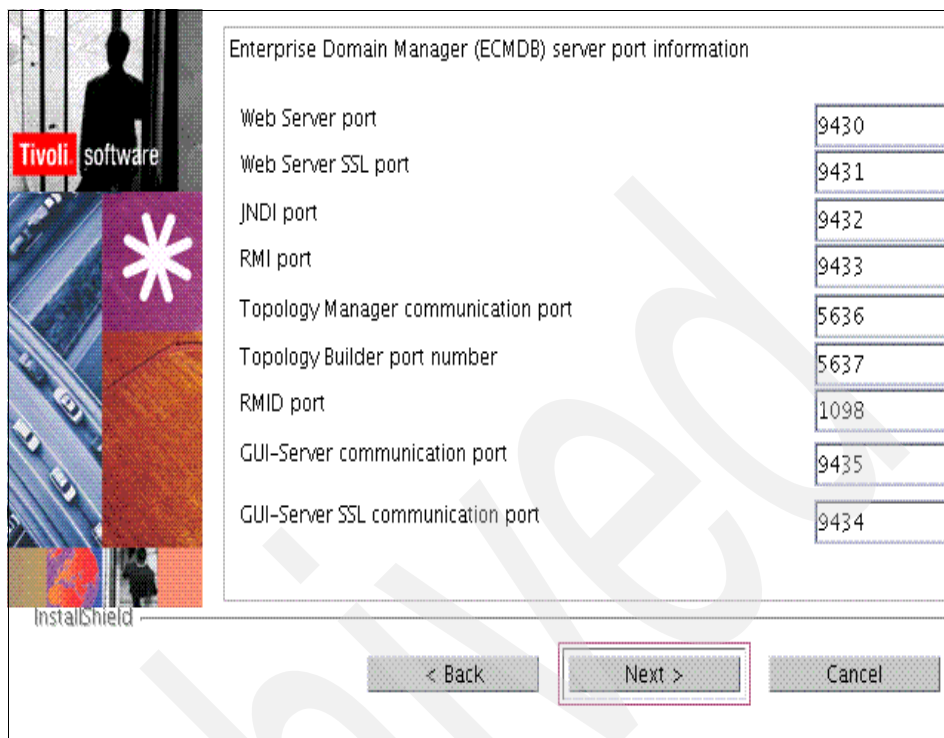


Figure 5-57 Select the server type

9. Select the server type for the TADDM Server that you are installing.

We were installing a TADDM enterprise manager; therefore, we selected **Enterprise Domain Manager (ECMDB) server**.

Clicking **Next** displays the panel that is shown in Figure 5-58 on page 151.



Enterprise Domain Manager (ECMDB) server port information	
Web Server port	9430
Web Server SSL port	9431
JNDI port	9432
RMI port	9433
Topology Manager communication port	5636
Topology Builder port number	5637
RMID port	1098
GUI-Server communication port	9435
GUI-Server SSL communication port	9434

InstallShield

< Back **Next >** Cancel

Figure 5-58 TADDM Server port information

10. Review the default port information and change any port numbers that need to be changed. We accepted all of the default port values. Click **Next**, and the panel shown in Figure 5-59 on page 152 is displayed.

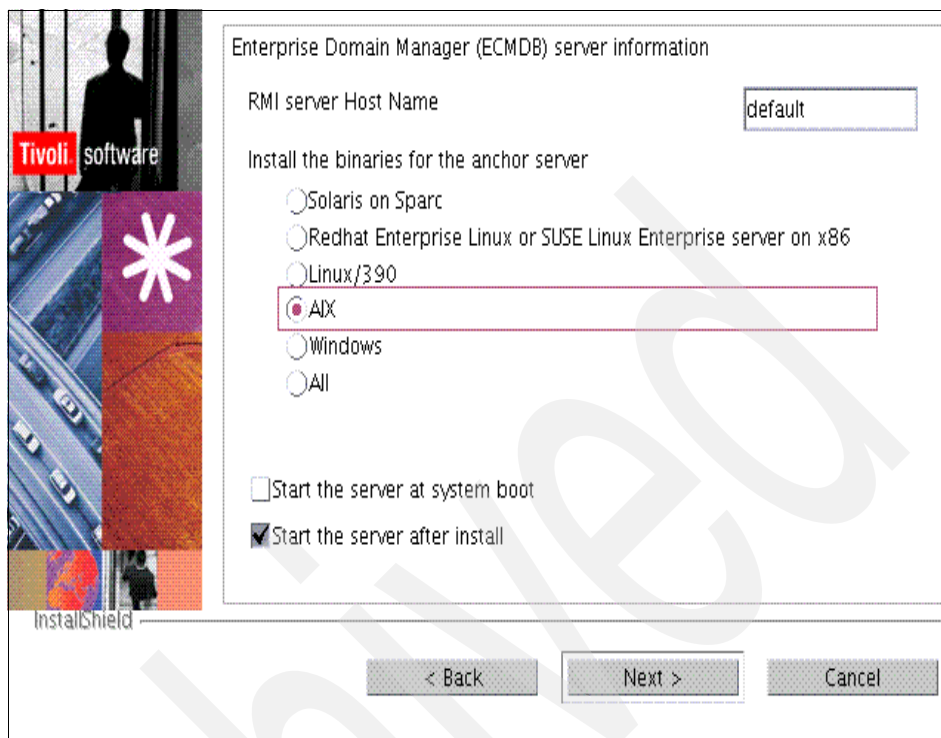


Figure 5-59 Specifying RMI information

11. You must enter certain information on this panel, such as the host name for the RMI server. The default value is default (use the default value if the RMI server resides on the same system as the CCMDB). If not, enter the IP address (not the host name) of the RMI server.

Next, select the platform binaries that you want installed. These binaries will be copied to Windows gateways and anchor servers. If you know on which OSs those gateways and anchors will run, choose only the binaries for those platforms. If you are unsure of which OSs the gateways and anchors will run on, select all of the platforms.

To start this TADDM Server when the system is started, select **Start the server at system boot**.

To start the server after the installation of TADDM is complete, select **Start the server after install**. Click **Next**, and the optional CCMDB host name and port panel, which is shown in Figure 5-60 on page 153, is displayed.

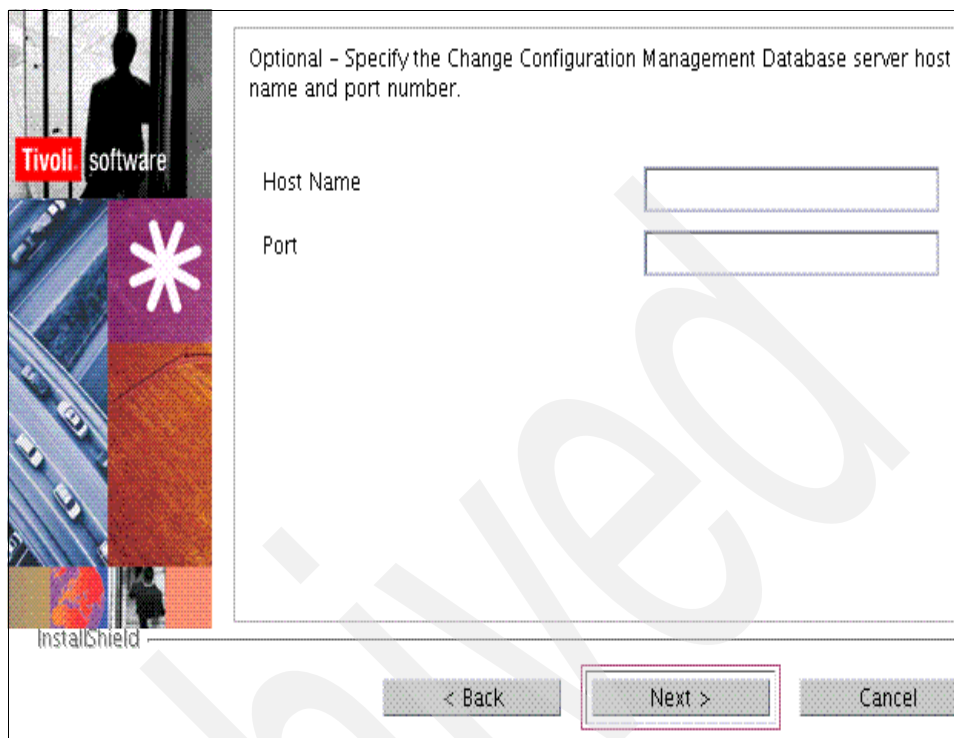


Figure 5-60 Optional CCMDB host name and port

12. This panel asks for the host name and port number of the Change and Configuration Management Database (CCMDB). We left this panel blank, because our implementation did not include a CCMDB. Click **Next**, and the Select the database type panel, which is shown in Figure 5-61 on page 154, is displayed.

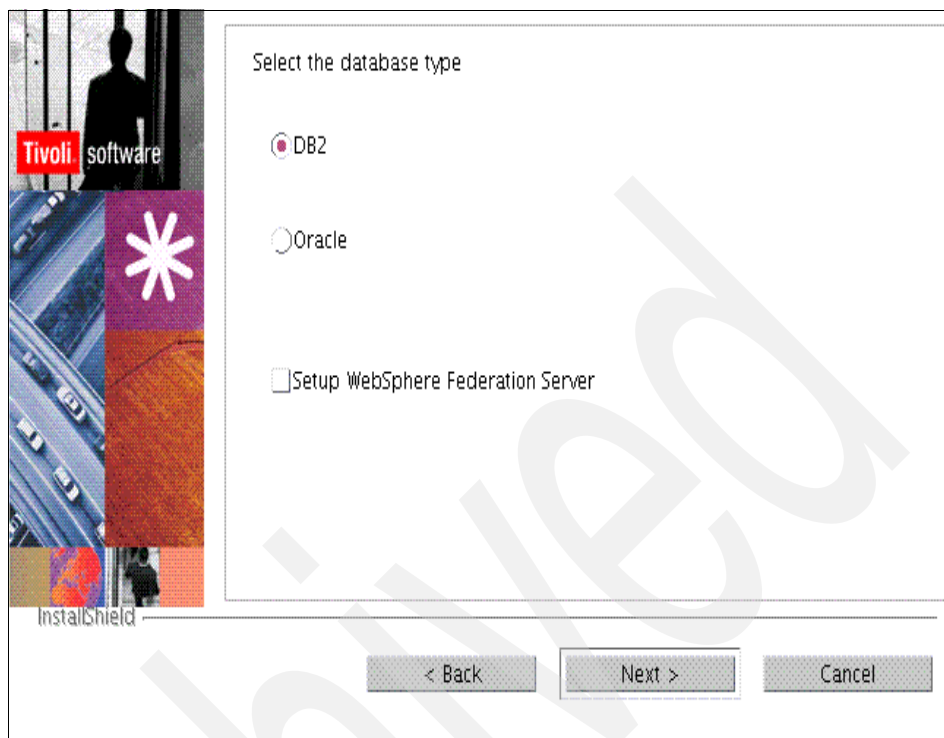


Figure 5-61 Select database type

13. Select the database type for the TADDM Database that you are installing. We are using DB2, so we selected **DB2**.

We chose not to set up the WebSphere Federation Server; therefore, we did not check the Setup WebSphere Federation Server check box.

Click **Next**, and the panel that is shown in Figure 5-62 on page 155 is displayed.

DB2 configuration information

Database Host Name: copenhagen

Database Port: 50030

Database node name (DB2 client only):

Database Name: ecmdb

DB2 Instance User ID: ecmdb

DB2 Instance User Password: *****

Additional User ID for database access: arecmdb

Additional Password for database access: *****

☐ Create the database during install. Local DB2 only.

Help with create database

< Back Next > Cancel

Figure 5-62 Database configuration information

14. This panel asks for information about the database. We created our database prior to installing TADDM. Enter the database information here.

Click **Next**, and the Select the User Registry Option panel, which is shown in Figure 5-63 on page 156, is displayed.

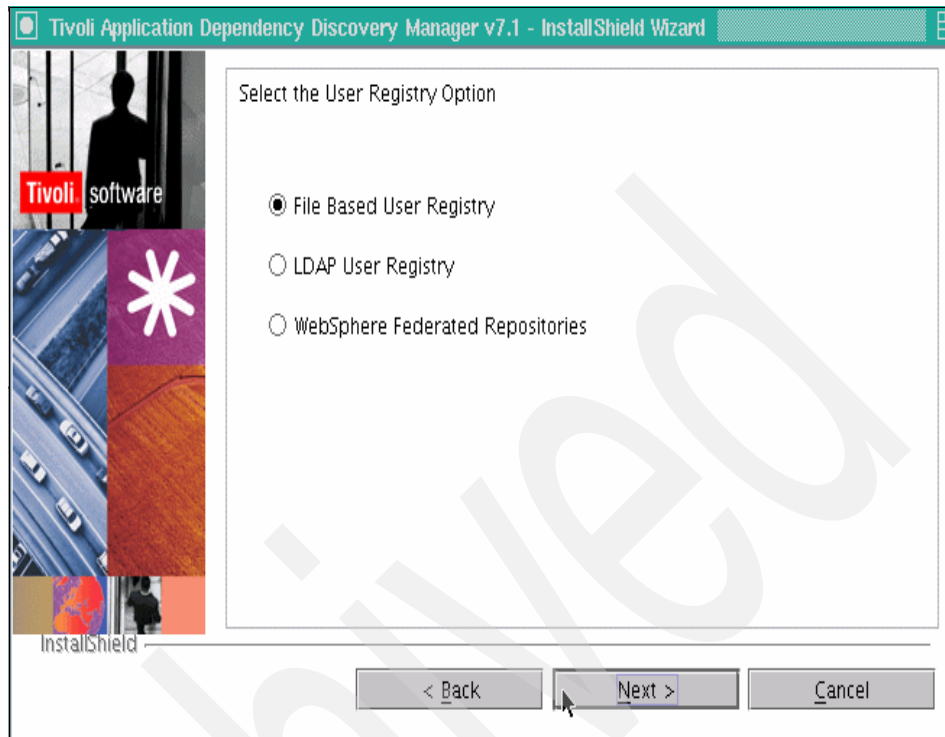


Figure 5-63 Select user registry

15. Select the option for the user registry that will be used with TADDM.

We will use LDAP for our user registry at the TADDM enterprise level, but we want to show how you can switch to use LDAP as your user registry after the installation, so we selected the **File Based User Registry**. Click **Next**, and the summary panel, which is shown in Figure 5-64 on page 157, is displayed.

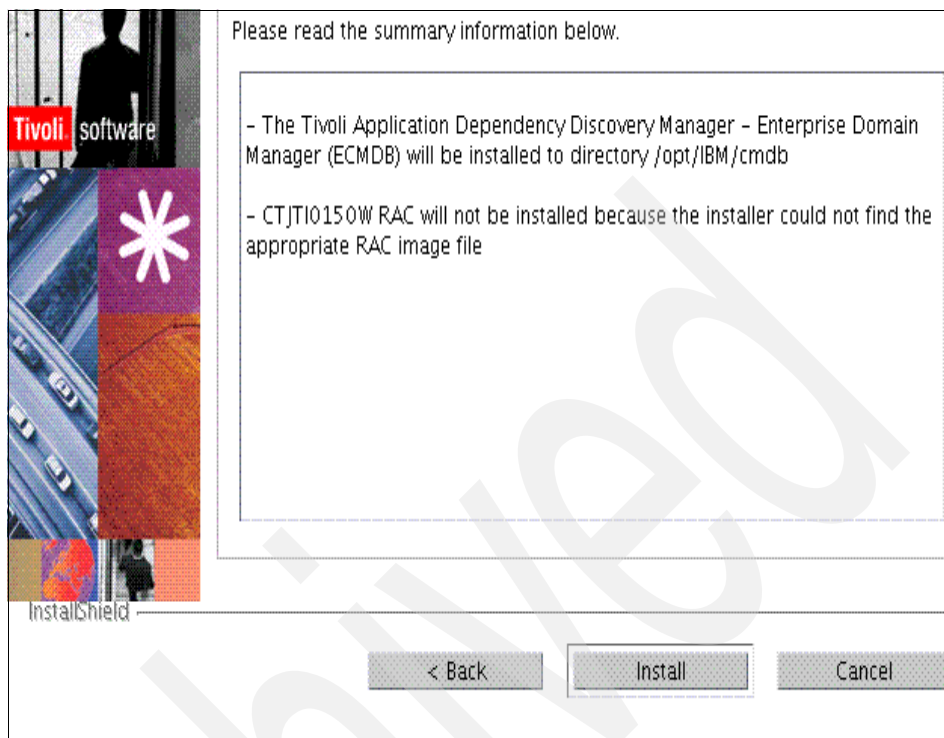
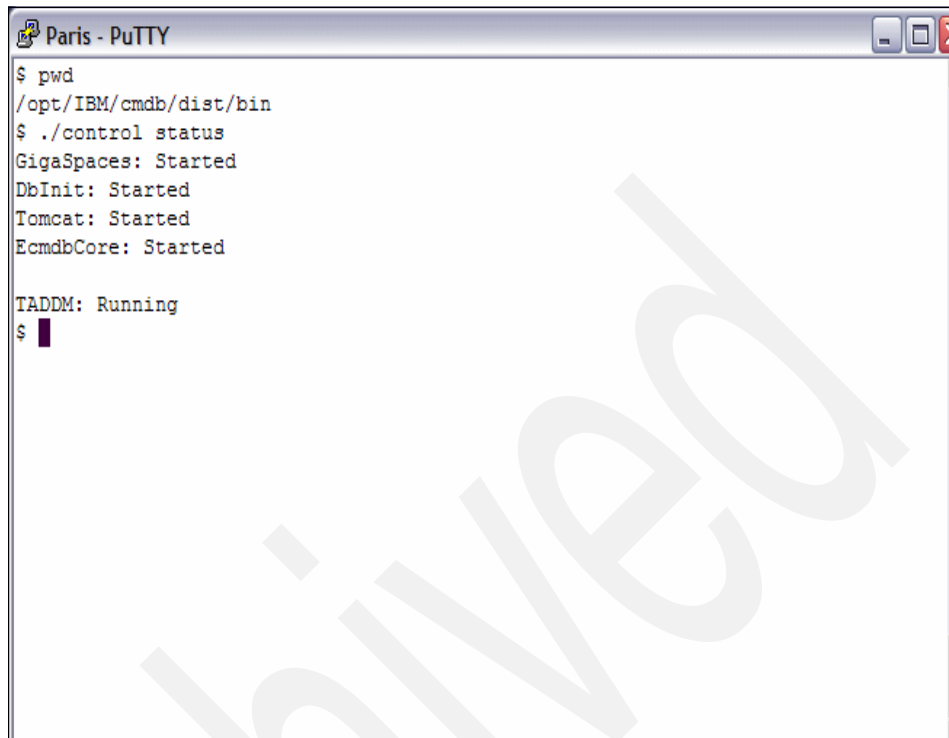


Figure 5-64 Summary information

16. Review the summary information, and if the information is correct, click **Next** to begin the installation.
17. When the installation completes, review the installation summary information, and click **Finish**. The installation of the TADDM enterprise server on AIX is now complete.
18. Before proceeding, we recommend that you verify that all of the TADDM services are running and that you can log on to the Domain Manager:
 - a. To verify that all of the TADDM services are running, log on to the TADDM Server with a non-root user ID, use the command `cd` to change to the `$COLLATION_HOME/bin` directory, and issue the following command:


```
./control status
```

The command and the expected output are shown in Figure 5-65 on page 158.



```
$ pwd
/opt/IBM/cmdm/dist/bin
$ ./control status
GigaSpaces: Started
DbInit: Started
Tomcat: Started
EcmdbCore: Started

TADDM: Running
$
```

Figure 5-65 The control status command and output

- b. Verify that the Product Console comes up successfully. Bring up your browser and enter a URL with the host name of the Linux server where TADDM was installed and port number 9430. For example:
`http://paris:9430`

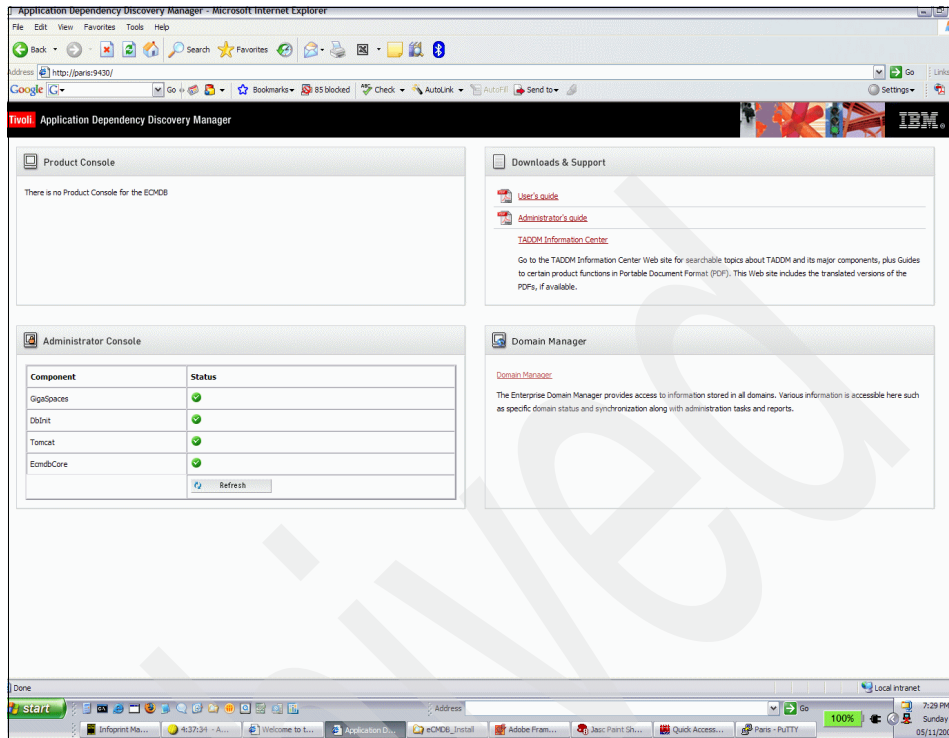


Figure 5-66 TADDM

Notice that the TADDM components and their associated status is displayed in the lower left corner of the panel. Clicking Refresh refreshes the status. All of the statuses need to be green.

19. Click **Domain Manager**, and when prompted, enter a valid user name and password. The Product Console panel, similar to Figure 5-67 on page 160, is displayed.

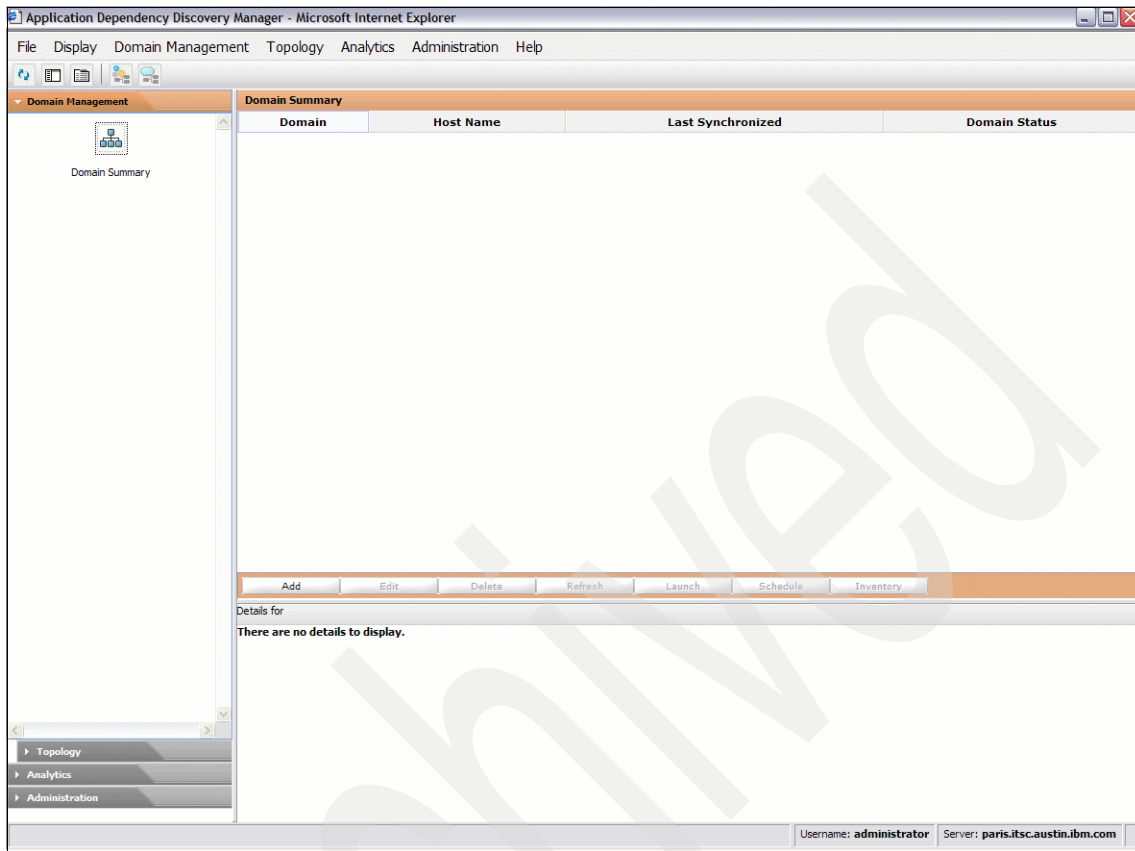


Figure 5-67 Domain Manager

5.5.2 Install interim fix 0007

At the time of writing this book, interim fix 0007 (IF0007) was the latest interim fix that was available for TADDM 7.1. To apply IF0007, obtain the image from the IBM support site and load it on to the AIX server. Log in to the AIX server as the root user, go to the directory where the IF0007 code has been copied, and type the command along with the directory where TADDM 7.1 exists. For example,

```
cd /root/code/ITADDM-IF0007/taddm_IF
installIF.sh /opt/IBM/cmdb
```

5.5.3 Configuring the eCMDB

The TADDM enterprise server, or eCMDB server, has been installed. Now, we need to configure the eCMDB to attach and synchronize with the TADDM Domain Servers by executing the following steps:

1. Open a Web browser and type the URL `http://<host name>:9430`, where `<host name>` is the host name of your eCMDB server, in the Address field. For example, our eCMDB server host name is Paris, so we type:

`http://paris:9430`

A Web page that is similar to Figure 5-68 is displayed.

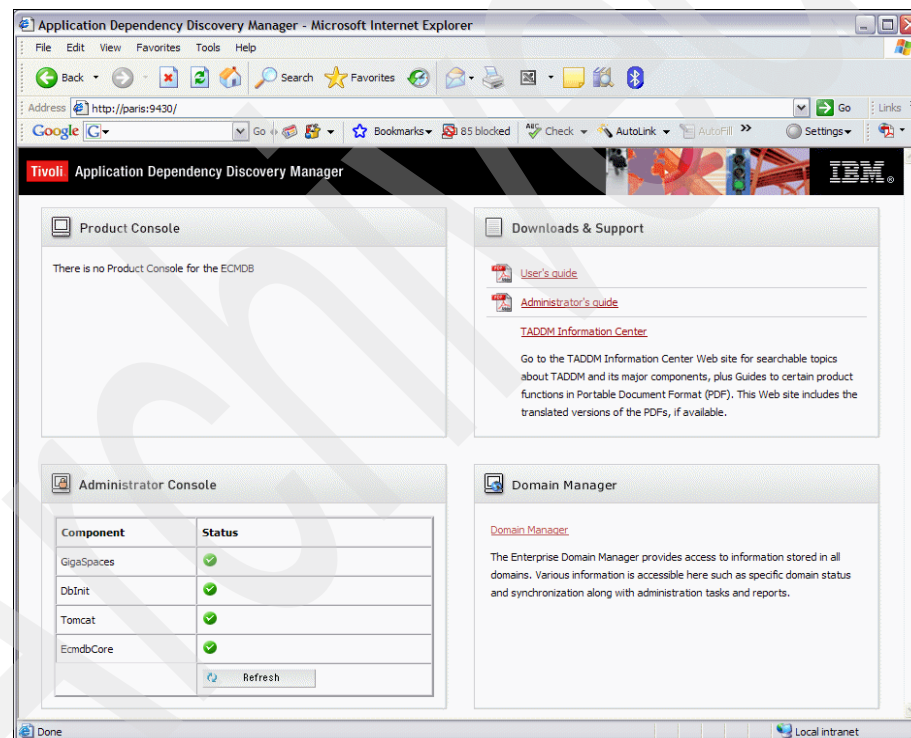


Figure 5-68 eCMDB

2. Click the **Domain Manager** to log on to the domain manager. When prompted, enter a valid user ID and password. The domain summary panel, which is similar to Figure 5-69 on page 162, is displayed.

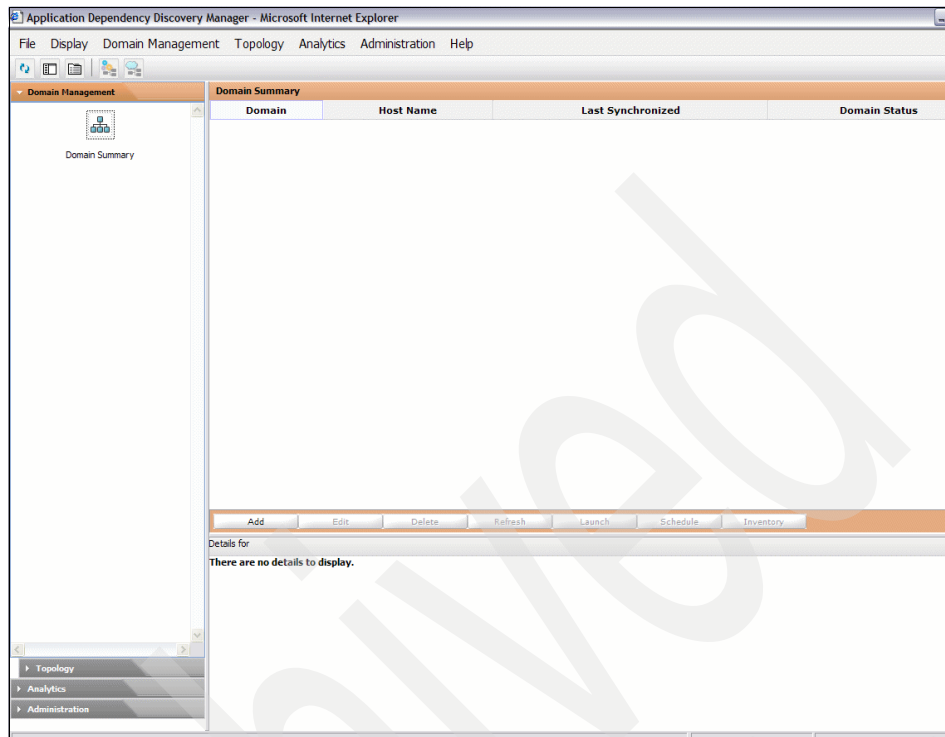


Figure 5-69 Domain summary

3. Click **Add**, and a panel that is similar to Figure 5-70 on page 163 is displayed.

Figure 5-70 Add Domains

4. To add a domain, enter the required information in the fields. In the Domain Password field, enter the `com.collation.sslpassphrase` value that is stored in the `collation.properties` file. For the Listening Port field, enter the `com.collation.jini.unicastdiscoveryport` value in the `collation.properties` file. To obtain this information, log in to the TADDM Domain Server and get these values from the `collation.properties` file.

To add our TADDM Domain Servers (Waco and Southend), we used the following steps:

- a. We logged in to the TADDM domain Waco and issued the commands in Example 5-1 on page 164 to get the `sslpassphrase` and `unicastdiscoveryport` values, as shown.

Example 5-1 Obtaining sslpassphrase and unicastdiscoveryport values

```
[root@waco etc]# cd /opt/IBM/cmdm/dist/etc
[root@waco etc]# cat collation.properties | grep sslpassphrase
com.collation.sslpassphrase=2733817229
[root@waco etc]# cat collation.properties | grep unicastdiscovery
com.collation.jini.unicastdiscoveryport=4160
[root@waco etc]# butt
```

These values can now be entered in the eCmdb Add Domain window. Refer to Figure 5-71.

Domain Management

Domain Summary

Topology

Analytics

Administration

Display Domain Management Topology Analytics Administration Help

Add Domain

Domain Details

*Domain Name: Waco *Domain Password:

*Fully Qualified Host Name/IP: waco.itsc.austin.ibm.com *Listening Port: 4160

Admin Details

Name: Contact:

Escalation Contact: Notes:

The fields marked with an asterisk * are required.

Add Domain Cancel

Details for

There are no details to display.

Username: administrator Server: paris.itsc.austin.ibm.com

Figure 5-71 Add Waco domain

- b. Click **Add Domain**. Figure 5-72 on page 165 shows the Domain Summary after the Waco domain was successfully added.

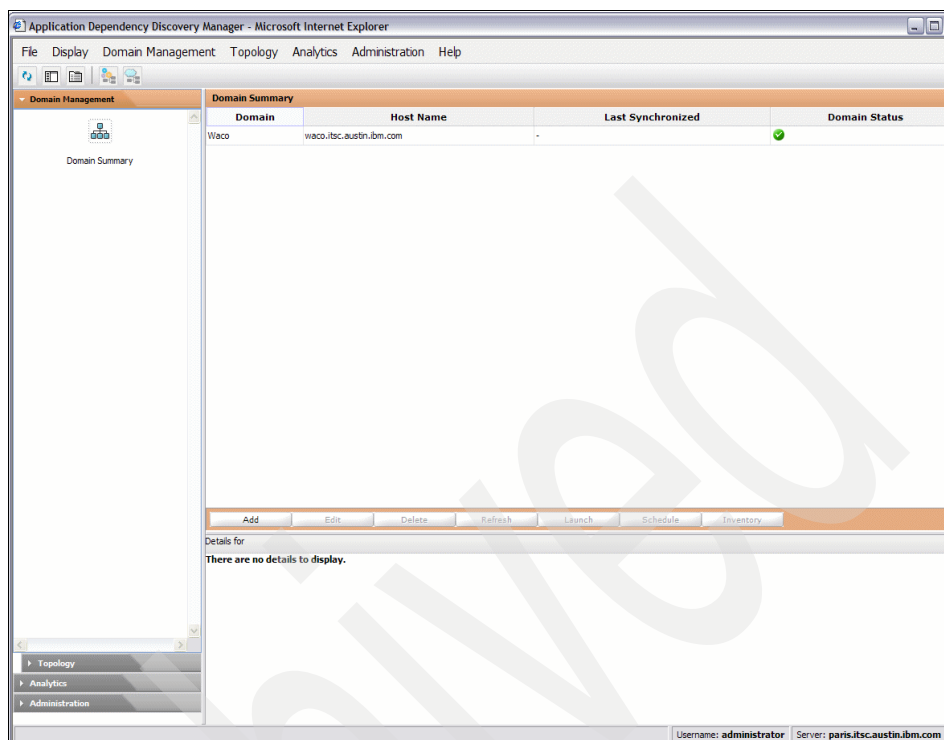


Figure 5-72 Waco domain added

c. Next, we added the Southend domain. Click **Add**.

Application Dependency Discovery Manager - Microsoft Internet Explorer

File Display Domain Management Topology Analytics Administration Help

Domain Management

Domain Summary

Add Domain

Domain Details

*Domain Name: Southend *Domain Password: ••••••••

*Fully Qualified Host Name/IP: thend.itsc.austin.ibm.com *Listening Port: 4160

Admin Details

Name: Contact:

Escalation Contact: Notes:

The fields marked with an asterisk * are required.

Add Domain Cancel

Details for

There are no details to display.

Topology Analytics Administration

Username: administrator Server: paris.itsc.austin.ibm.com

Figure 5-73 Add Southend domain

Search for the `com.collation.sslphrase` and `com.collation.jini.unicastdiscoveryport` values, and copy these values into the Domain Password and Listening Port fields. Enter the Domain Name and Fully Qualified Host Name/IP values, and then, click **Add Domain** to add this new domain.

- d. Both domains have now been added. Figure 5-74 on page 167 shows that both domains have been added.

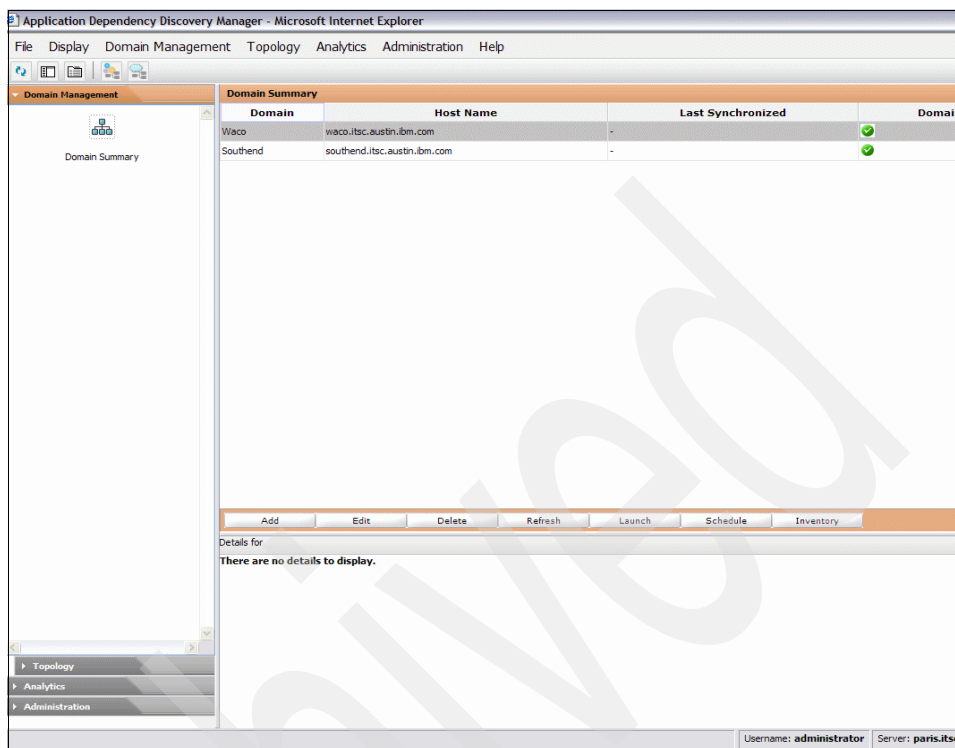


Figure 5-74 Waco and Southend domains added

5. Now, we need to perform the initial synchronization and schedule regular synchronization for both domains:
 - a. From the Domain Summary, highlight the Waco domain and click **Schedule**, which displays a panel similar to Figure 5-75 on page 168.

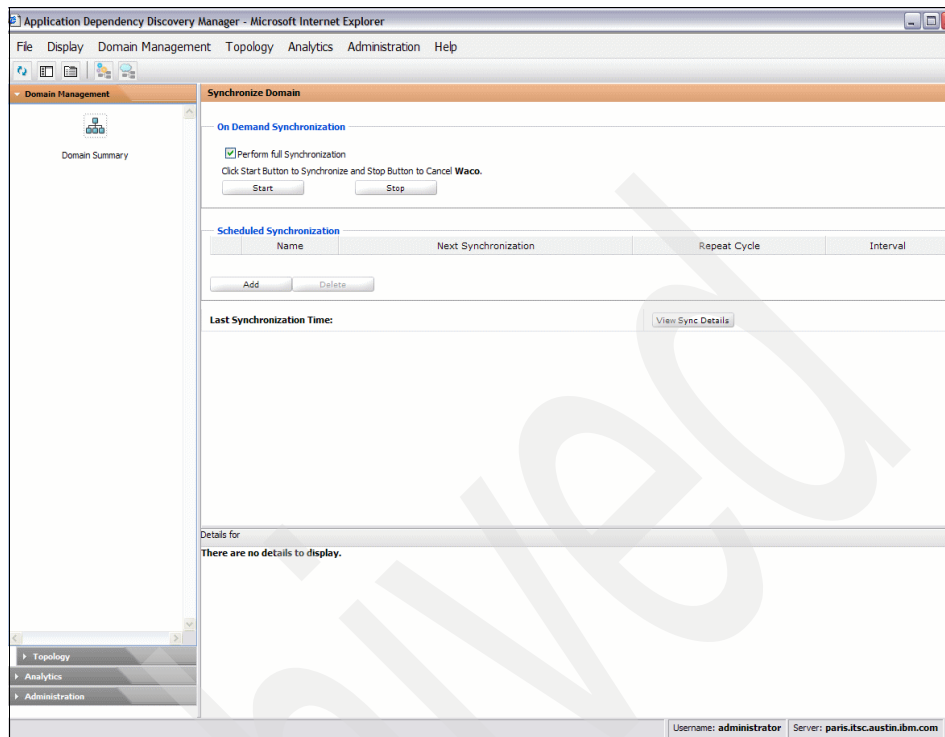


Figure 5-75 Synchronize Domain: Waco

- b. Click **Perform full Synchronization**, and click **Start**. A panel similar to Figure 5-76 on page 169 is displayed when the synchronization is complete.

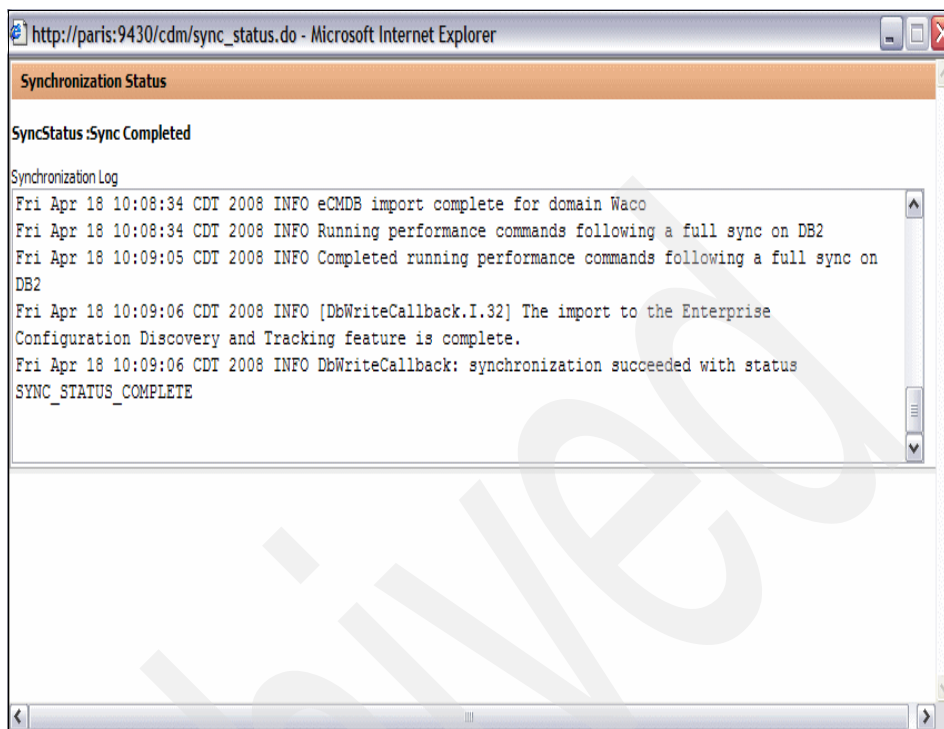


Figure 5-76 Waco full synchronization complete

- c. From the Synchronize Domain panel, click **Add** to schedule a synchronization for the Waco domain. A panel that is similar to Figure 5-77 on page 170 is displayed.

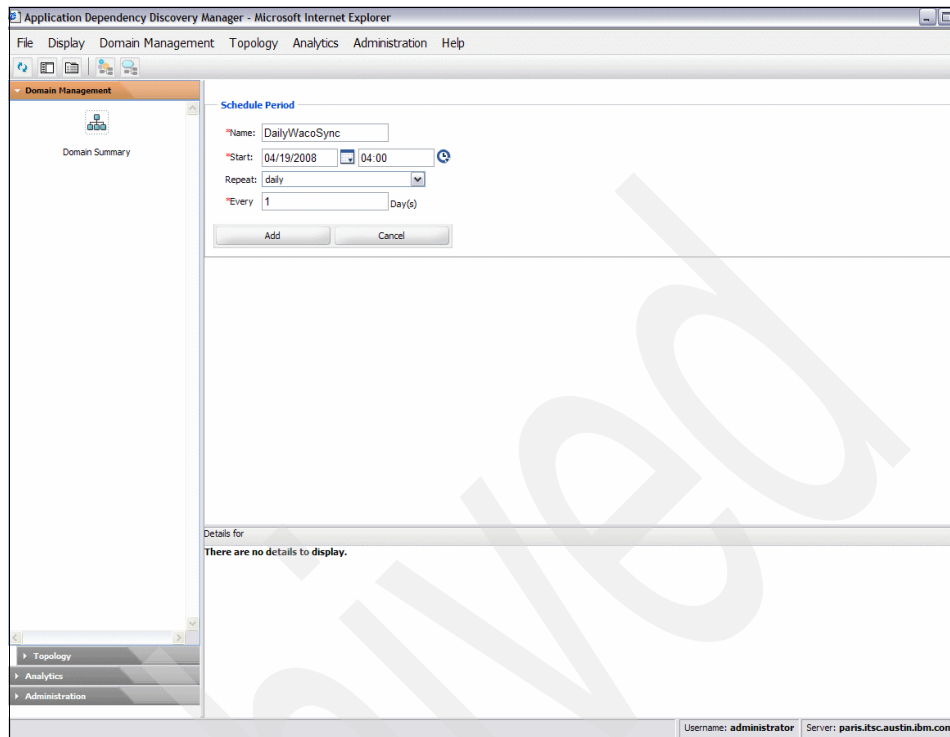


Figure 5-77 Schedule daily synchronization for Waco

- d. Give the schedule a name, and then, enter the other fields.

We named the synchronization for the Waco domain `DailyWacoSync`. We scheduled the synchronization to begin on 4/19/2008 and run every day at 04:00 AM. Click **Add** to create the scheduled synchronization. Figure 5-78 on page 171 shows that the daily synchronization is scheduled for the Waco domain.

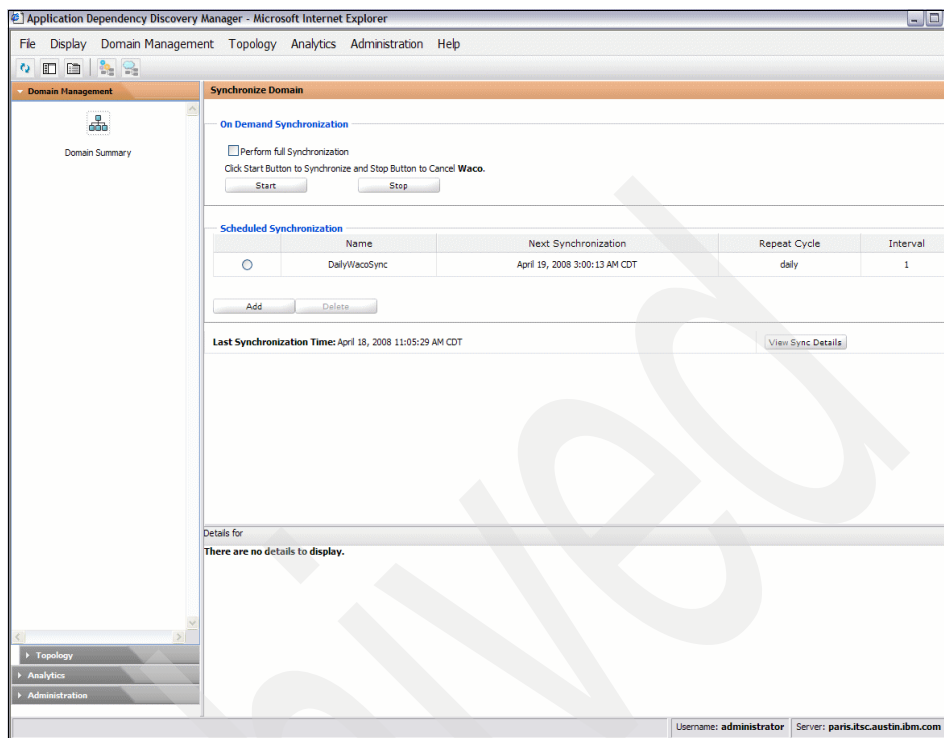


Figure 5-78 Waco daily synchronization scheduled

- e. The steps to perform the initial full synchronization for the TADDM domain Southend are similar to the previous steps a-d.

Both domains, Waco and Southend, have had the initial full synchronization done and a daily synchronization scheduled. Figure 5-79 on page 172 shows the resulting domain summary.

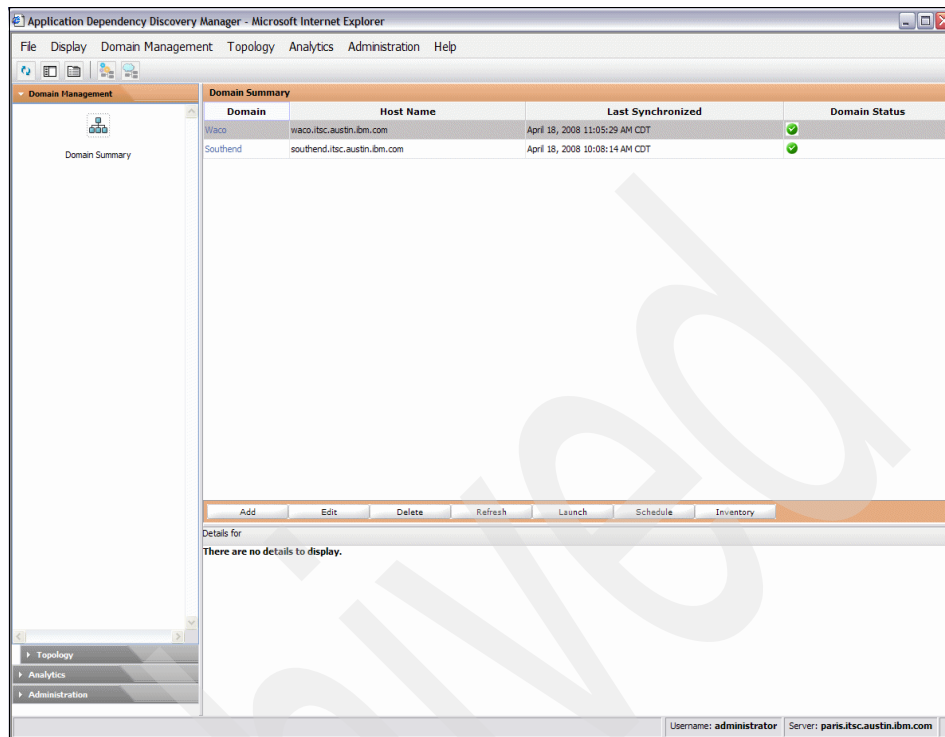


Figure 5-79 Domain summary showing scheduled synchronizations

5.6 Configuring LDAP

When a TADDM Domain Server is connected to the TADDM enterprise server, user authentication for the TADDM domain is done through the TADDM enterprise.

We used the following steps to configure our TADDM enterprise server to use LDAP for user authentication.

But first, we had to enter the TADDM users in our LDAP registry. Our LDAP registry was implemented using Tivoli Directory Server (TDS) running on a Windows Server 2003 server with the host name, Helsinki. The LDAP server port was port 1389. We added a new branch for our TADDM users to keep them separate from other projects that were using this LDAP server. The distinguished name for this branch was `dn: o=IBM,c=US,ou=ITS0`. We created two organizational units under this branch: one unit for users and one unit for groups.

We then added the users. Example 5-2 shows adding the relative distinguished name administrator to the user container.

Example 5-2 Adding the relative distinguished name administrator

```
dn: cn=administrator,ou=users,o=IBM,c=US,ou=ITS0
uid: administrator
userpassword: collation
objectclass: organizationalPerson
objectclass: inetOrgPerson
objectclass: person
objectclass: top
title: TADDM Administrator
sn: administrator
cn: administrator
```

After the definitions were completed with TDS, we restarted the Windows TDS service so that the changes that we made to the LDAP registry were active.

Next, we set several security-related properties in the collation.properties file. We set the following properties:

1. Specify the user management module that is used by this TADDM Server. Valid values are:
 - file for a file-based user registry. This is the default value.
 - ldap for an LDAP user registry.
 - vmm for a user registry that uses the federated repositories of WebSphere Application Server.

For example, in the \$COLLATION_HOME/etc/collation.properties file:

```
com.collation.security.usermanagementmodule=ldap
```

2. Set the LDAP authentication enabled value to true.

For example, in the \$COLLATION_HOME/etc/collation.properties file:

```
com.collation.security.auth.ldapAuthenticationEnabled=true
```

3. Set the host name and port number of the LDAP server.

For example, in the \$COLLATION_HOME/etc/collation.properties file:

```
com.collation.security.auth.ldapHostName=HELSINKI.itsc.austin.ibm.com
```

```
com.collation.security.auth.ldapPortNumber=1389
```

4. Specify the starting point in the LDAP hierarchy to begin searching.
For example, in the \$COLLATION_HOME/etc/collation.properties file:
`com.collation.security.auth.ldapBaseDN=ou=users,o=IBM,c=US,ou=ITS0`
5. Specify the name of the attribute that was used to represent users in LDAP.
For example, in the \$COLLATION_HOME/etc/collation.properties file:
`com.collation.security.auth.ldapBaseDN=cn=root`
6. Specify the user password that was used to authenticate to LDAP if simple authentication is used.
For example, in the \$COLLATION_HOME/etc/collation.properties file:
`com.collation.security.auth.ldapBindPassword=YqANUIiFCD4=`
7. Specify the LDAP object class and naming attribute that were used for naming a person.
For example, in the \$COLLATION_HOME/etc/collation.properties file:
`com.collation.security.auth.ldapUserObjectClass=person
com.collation.security.auth.ldapUIDNamingAttribute=cn`
8. Specify the LDAP object class and naming attribute that were used for naming a group.
For example, in the \$COLLATION_HOME/etc/collation.properties file:
`com.collation.security.auth.ldapGroupObjectClass=organizationalUnit
com.collation.security.auth.ldapGroupNamingAttribute=ou
com.collation.security.auth.ldapGroupMemberAttribute=uniqueMember`

After making changes to the \$COLLATION_HOME/etc/collation.properties file, stop and start TADDM so that the changes become active.

5.7 Deploying anchors and gateways

The TADDM Server uses Secure Shell (SSH) to directly communicate with computer hosts and the other components that it discovers. However, there are two cases when the TADDM Server must communicate through a proxy to collect system information:

- ▶ When using a firewall between the TADDM Server and other sections of your network
- ▶ When discovering and collecting information from Windows systems

In this section, we teach you how to enable discoveries across the firewall. The next section, 5.8, “Setting up Windows gateways” on page 182, will teach you how to enable the discovery of Windows systems.

In order to discover components, each discovery server must be able to communicate with other computer hosts and network devices. In cases when a firewall prevents direct access from the discovery server to certain hosts or devices, you can specify a computer system that does have access to the hosts or devices to be an anchor host. An *anchor host* acts as a proxy to assist in the discovery process.

You do not need to configure anchor hosts during the installation process, but you need to include anchor hosts in your installation plan and verify the system requirements for the candidate machines. Following the installation of the TADDM Server, you can use the TADDM Product Console to configure which hosts serve as anchor hosts in your environment.

5.7.1 Enabling discoveries across the firewall

If a firewall exists between the TADDM Server and hosts and network devices to be discovered, you must enable discoveries across the firewall. Complete the following steps to enable discoveries across the firewall:

1. Specify at least one computer system in the firewall zone that is adjacent to the firewall zone where the TADDM Server is located. Any operating system that can be used as a Domain Manager can be used as an anchor.

This computer system, which is called an *anchor*, runs a discovery subsystem to discover the components that are located in the section. There is nothing to install on the anchor system, because the TADDM product automatically deploys and manages the anchor server software on the anchors.

2. Configure the firewall to allow SSH access to the anchor.

The TADDM Server uses the SSH port on the Anchor Server to push and automatically run the discovery software on the anchor that is on the other side of the firewall. The discovery software on the anchor then returns information about the discovered components in its zone to the TADDM Server.

3. If there are multiple zones and firewalls, each firewall zone needs its own anchor so that the communications can be relayed from each anchor across each firewall.

You must also allow SSH traffic on each firewall.

4. When running a discovery that uses an anchor, make sure that the anchor is included in the discovery scope, which is a change from previous versions of TADDM. For example, to discover a target that is in a scope set (for example, scopeset1) that is assigned to an anchor, both the anchor and scopeset1 must be included in the discovery scope. The scope sets that are assigned to an anchor need to be only the IP addresses that are accessible by that anchor.

5.7.2 Defining an anchor host

To add an anchor host, complete the following steps:

1. Log on to the TADDM Product Console.
2. On the Discovery menu, click **Anchors and Gateways**.
3. Click **Add** to define a new anchor or gateway.
4. In the Add Anchor Host window, select **Type** to be **Anchor**. In our environment, we used Zaire as an anchor host.
5. Select **Address** or **Host Name** to set the anchor host that you are adding. If you select **Address**, ensure that the IP address of each anchor host computer system is contained within the scope.
6. Select **Entire Scope** or **Limit to selected scope** to set the scope.
If you select **Limit to selected scope**, select the scope set to use.
Remember, the anchor must be included in any scope assigned to it.
7. Click **OK**.

Editing the scope of an anchor

To edit the scope of an anchor, complete the following steps:

1. On the Discovery menu, click **Anchors and Gateways**.
2. Select an anchor.
3. Click **Edit Scope**.
4. Change the scope set for the anchor or gateway.
5. Click **OK**.

Setting an anchor port for an anchor server

To set an anchor port for an anchor server, complete the following steps:

1. On the Discovery menu, click **Anchors and Gateways**.
2. Select an anchor server.
3. Enter a port number.
4. Click **OK**.

Archived

Discovering an anchor

After you define your anchor, you can verify the setup (for example, the firewall has the SSH port open, credentials are correct, and the TADDM Server can distribute binaries to the anchor) by discovering that anchor. You can discover the anchor by following these steps:

1. Go to **Discovery** → **Scope**, and click **Add Set** to create a new Set called AnchorHosts, as shown in Figure 5-80. Click **OK**.

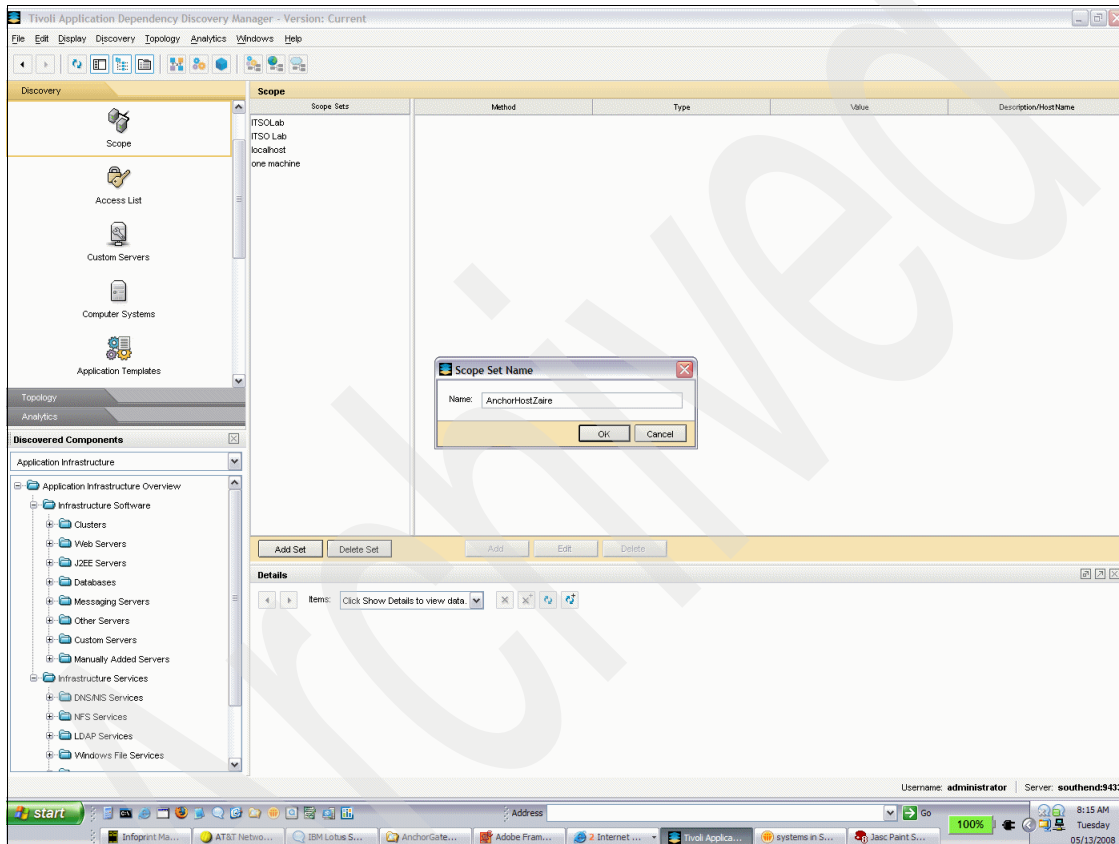
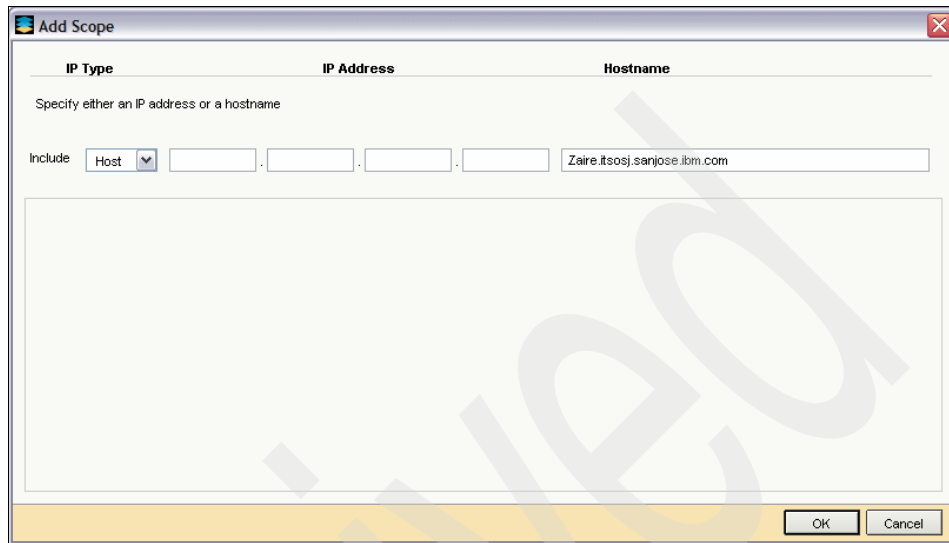


Figure 5-80 Adding a new Discovery Scope Set

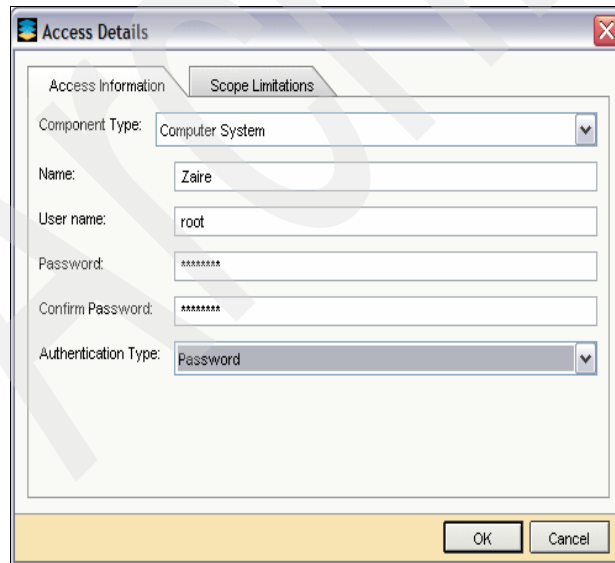
2. Click **Add** (Figure 5-81) to add the new target. We entered the host name `Zaire.itsosj.sanjose.ibm.com`. Then, click **OK**.



The 'Add Scope' dialog box is shown. It has a title bar with a close button. Below the title bar, there are three tabs: 'IP Type', 'IP Address', and 'Hostname'. The 'Hostname' tab is selected. Below the tabs, there is a text area with the instruction 'Specify either an IP address or a hostname'. Below this, there is a section labeled 'Include' with a dropdown menu set to 'Host'. To the right of the dropdown are four empty input fields for IP address segments, followed by a text field containing 'Zaire.itsosj.sanjose.ibm.com'. At the bottom right, there are 'OK' and 'Cancel' buttons.

Figure 5-81 Adding a new target

3. Go to **Discovery** → **Access List**, click **Add** (Figure 5-82) to add a new credential and complete the fields in the window with the correct information.



The 'Access Details' dialog box is shown. It has a title bar with a close button. Below the title bar, there are two tabs: 'Access Information' and 'Scope Limitations'. The 'Access Information' tab is selected. Below the tabs, there are several fields: 'Component Type' (dropdown menu set to 'Computer System'), 'Name' (text field containing 'Zaire'), 'User name' (text field containing 'root'), 'Password' (text field with masked characters), 'Confirm Password' (text field with masked characters), and 'Authentication Type' (dropdown menu set to 'Password'). At the bottom right, there are 'OK' and 'Cancel' buttons.

Figure 5-82 Entering Access List credentials for Zaire

4. Click the **Scope Limitations** tab (Figure 5-83), and choose the **AnchorHosts** scope that you previously created. Click **OK**.

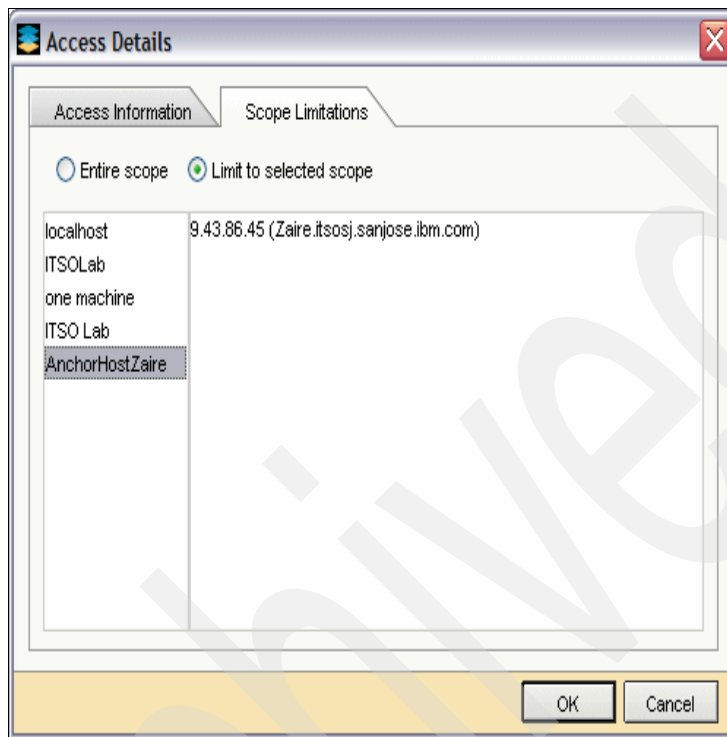


Figure 5-83 Entering scope limitations for anchor

5. Select **Discovery** → **Overview**, and run a new discovery to your new anchor (Figure 5-84), which in our case, was lincanc1. Click **OK**.

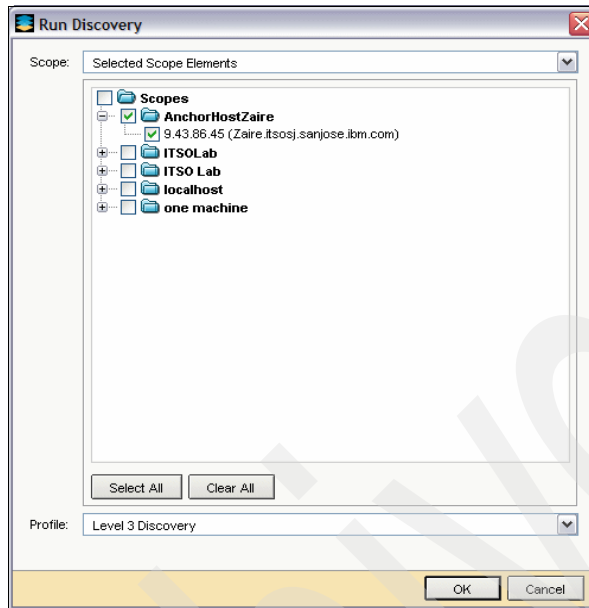


Figure 5-84 Starting a new discovery

Shutting down an anchor server

To shut down an anchor server, complete the following steps:

Note: If you want to shut down the TADDM Server, you must also shut down the anchor server. If you do not shut down the anchor server, unexpected behavior can occur, including the bad performance of certain discoveries.

1. Verify that the process for the anchor server is no longer running by using the following command:

```
% ps -ef |grep -i anchor
```

You get the following output, which identifies any anchor server processes that are still running:

```
coll 23751 0.0 0.0 6136 428 ? S Jun02 0:00 /bin/sh
local-anchor.sh 8494 <more information here>
```

2. Stop the process by typing the following command:

```
- % kill -9 23751
```

You get the following output if the anchor server process is gone, which also shows that there are no processes running:

```
% ps -ef | grep -i anchor
```

5.7.3 Open ports

To discover components, the TADDM Server must be able to use SSH to communicate with all of the computer hosts and any other devices that support SSH.

SSH is therefore required on all non-Windows computer hosts that you want discovered. A Windows gateway is required in order to discover Windows computer hosts and the Windows gateway must have an SSH client installed. In addition to SSH, the TADDM Server uses Simple Network Management Protocol (SNMP) and Java Management Extensions (JMX), among other methods, to communicate with computer hosts and devices.

As discussed earlier, an anchor is required in order to discover resources on the other side of a firewall. Multiple TADDM Servers can share a single anchor, but each TADDM Server needs to use a different RMI port on that shared anchor. The anchor server does not allow multiple TADDM Servers to use the same RMI port during discoveries.

If a firewall is preventing access to hosts or devices on these ports, you must configure an anchor host to properly allow discoveries. For more information about how to configure TADDM to use anchor hosts to allow for discoveries across firewalls, see the topic about using TADDM in secure environments in the *TADDM User Guide*. The link to the document is:

http://publib.boulder.ibm.com/infocenter/tivihelp/v10r1/topic/com.ibm.taddm.doc_7.1/cmdb_user.pdf

You have to enable SSH communication across the firewall.

5.8 Setting up Windows gateways

In this section, we teach you about discovering Windows systems.

If your network contains Windows-based systems that are to be discovered with TADDM, you must specify a Windows system to serve as a gateway server in order to discover information about the Windows-based systems that are running in your environment. This gateway server must be in the same firewall zone as

the discovered Windows hosts and must have SSH access from the discovery server.

You do not need to configure Windows gateways during the installation process, but you need to include gateways in your installation plan and verify the system requirements for candidate machines. Following installation, you can use the TADDM Product Console to configure which hosts will serve as Windows gateways on your network.

To configure the TADDM Server for Windows discovery, complete the following steps:

1. Specify a Windows gateway server that communicates with Windows systems.

If the gateway resides in a firewalled zone, the zone must contain an Anchor Server, or the firewall must be configured to allow SSH from the upstream TADDM Server (or anchor server) to the gateway.

2. Limit the gateway to specific scope entries.
3. Add authentication information for the gateway server and for the Windows hosts that are discovered.
4. Configure the firewall to allow SSH access to the anchor or the Windows gateway.

The TADDM Server uses the SSH port on the firewall to automatically run the discovery software on the Windows gateway that is on the other side of the firewall. The Windows gateway then returns information about the discovered components, in its section, to the TADDM Server.

5.8.1 Installing Cygwin SSH

The TADDM Server communicates with Windows gateways through SSH. The Windows gateway then communicates with Windows hosts using Windows Management Interface (WMI). In order for the Windows gateway to communicate with the TADDM Server via SSH, an SSH Server/Daemon (sshd) client must be installed on the Windows gateway. TADDM V7.1 supports two SSH clients: Bitvise WinSSHD 4.06a or higher and Cygwin SSH. In the following example, we install Cygwin SSH.

To install Cygwin SSH:

1. Download the cygwin setup program from the following Web site:

<http://www.cygwin.com/>

2. Run the setup program. A panel similar to Figure 5-85 is displayed.

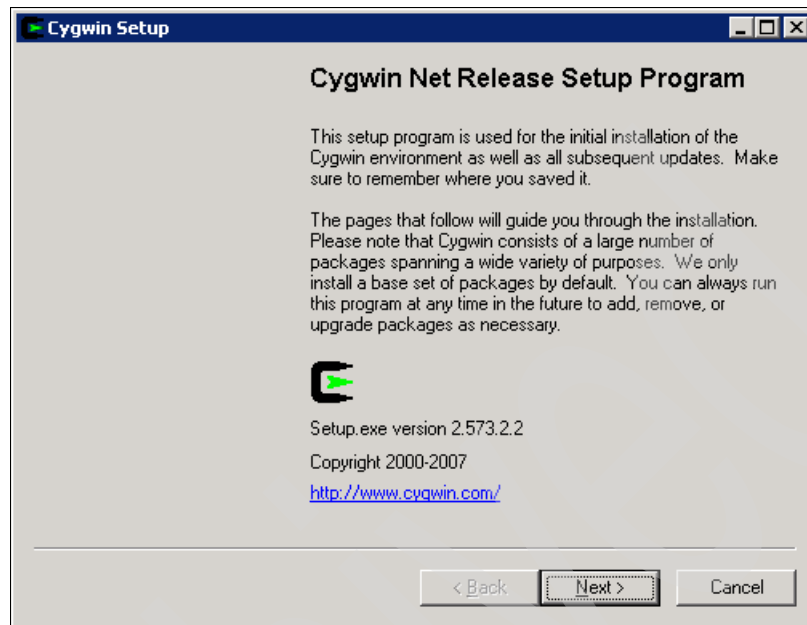


Figure 5-85 Cygwin NetRelease Setup Program

3. Click **Next**.

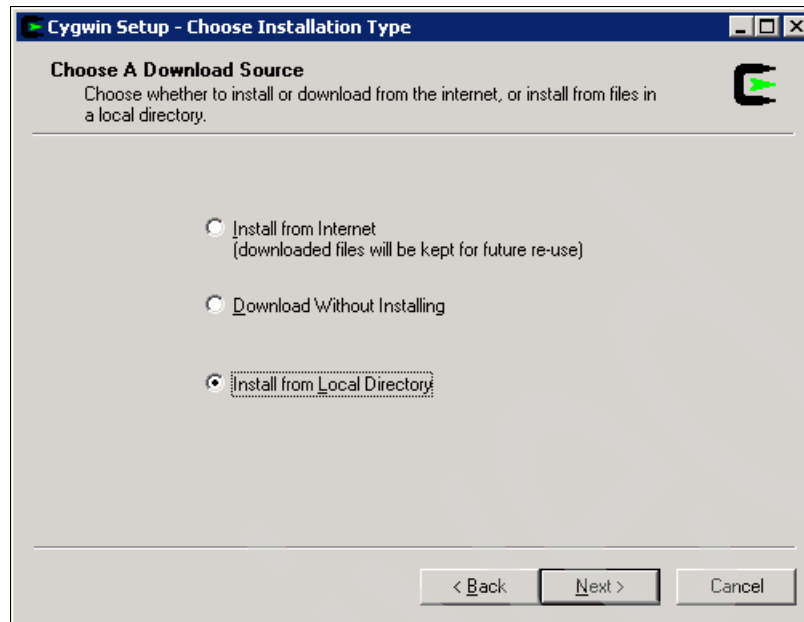


Figure 5-86 Choose A Download Source

4. Choose the appropriate option. Because we downloaded the package prior to running the installer, we choose **Install from Local Directory**. Click **Next**. Figure 5-87 on page 186 is displayed.

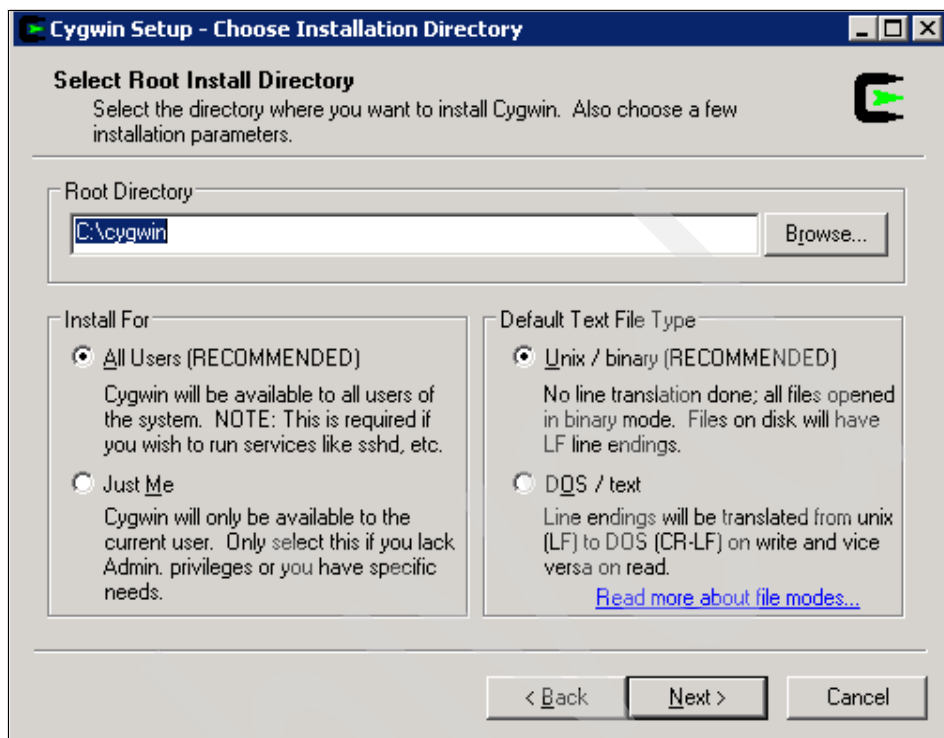


Figure 5-87 Choose Installation Directory

5. Set the Root Directory where you want Cygwin to be installed. We entered C:\cygwin. We took the defaults for the other options on this window. Click **Next**. Figure 5-88 on page 187 is displayed.

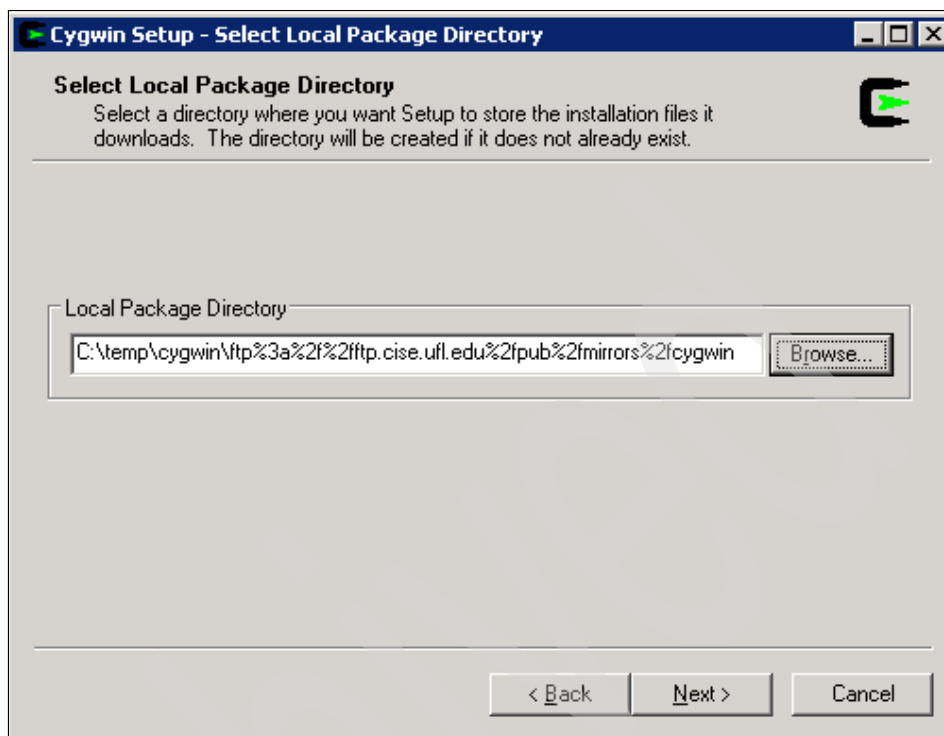


Figure 5-88 Select Local Package Directory

6. Enter the Local Package Directory where you want to store the Cygwin installation package. We downloaded Cygwin to the C:\temp\ directory; the setup program actually wanted a directory one level beneath that. If you put the wrong directory here, on the next panel (Figure 5-89 on page 188), when you expand the categories, you will not see the package that you need to install.

Click **Next**. Figure 5-89 on page 188 is displayed.

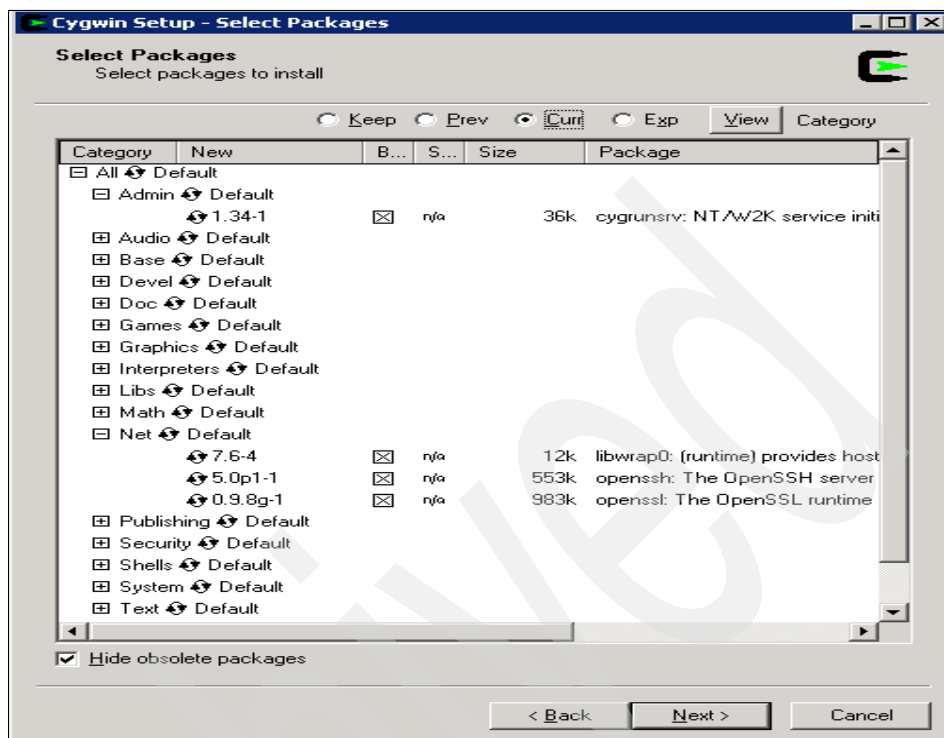


Figure 5-89 Select Packages

7. Select the packages to install.

Take all of the defaults, as well as:

- cygrunsrv from the admin category (Version 1.17-1 or later)
- opensshd from the net category (Version 4.6p 1-1 or later)

Click **Next**.

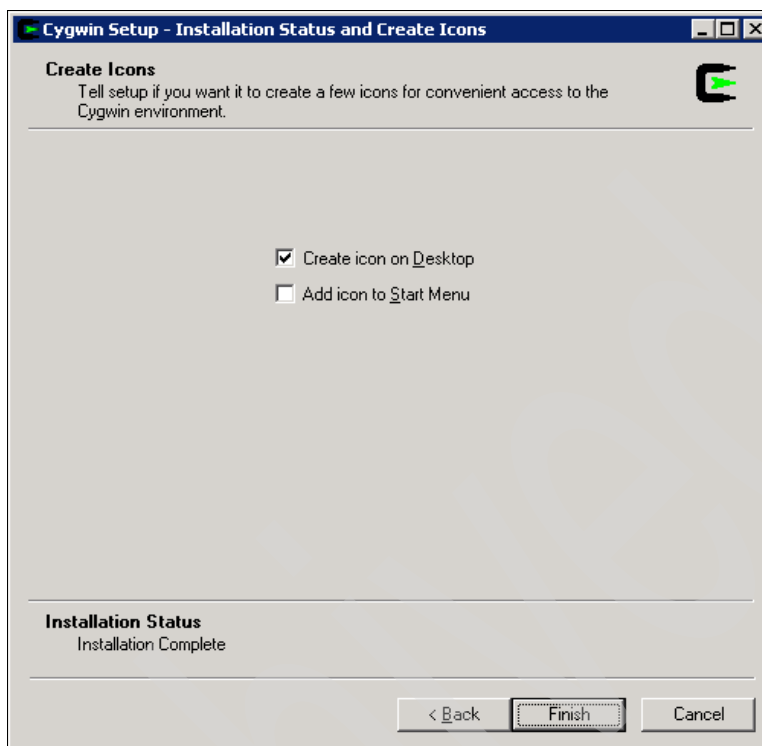


Figure 5-90 Installation Status and Create Icons

8. Notice the Installation Status. If you want to create an icon on the Desktop, or add an icon to the Start Menu, check the appropriate option.

Click **Finish**.

9. When the installation is finished, start the **cygwin bash** shell.

10. Use the **cygwin mkpasswd** utility to create an initial `/etc/passwd` from your system information. You can also use the **mkgroup** utility to create an initial `/etc/group`. For more information, refer to the *Cygwin User's Guide* at:

<http://sources.redhat.com/cygwin/cygwin-ug-net/cygwin-ug-net.html>

For example, the following command sets up the password file, `passwd`, from the local accounts on your system:

```
mkpasswd -1 > /etc/passwd
```

11. Run the **ssh-host-config** program and configure SSH. Figure 5-91 on page 190 shows the output when we ran **ssh-host-config** on Wisla. It also shows our answers to all of the questions that were asked by this config tool.

```
Administrator@Wisla ~
$ ssh-host-config
Generating /etc/ssh_host_key
Generating /etc/ssh_host_rsa_key
Generating /etc/ssh_host_dsa_key
Generating /etc/ssh_config file
Privilege separation is set to yes by default since OpenSSH 3.3.
However, this requires a non-privileged account called 'sshd'.
For more info on privilege separation read /usr/share/doc/openssh/README.privsep
.

Should privilege separation be used? <yes/no> yes
Warning: The following function requires administrator privileges!
Should this script create a local user 'sshd' on this machine? <yes/no> yes
Generating /etc/sshd_config file

Warning: The following functions require administrator privileges!

Do you want to install sshd as service?
<Say "no" if it's already installed as service> <yes/no> yes

You appear to be running Windows 2003 Server or later. On 2003 and
later systems, it's not possible to use the LocalSystem account
if sshd should allow passwordless logon (e. g. public key authentication).
If you want to enable that functionality, it's required to create a new
account 'sshd_server' with special privileges, which is then used to run
the sshd service under.

Should this script create a new local account 'sshd_server' which has
the required privileges? <yes/no> yes

Please enter a password for new user 'sshd_server'. Please be sure that
this password matches the password rules given on your system.
Entering no password will exit the configuration. PASSWORD=itso13sj

User 'sshd_server' has been created with password 'itso13sj'.
If you change the password, please keep in mind to change the password
for the sshd service, too.

Also keep in mind that the user sshd_server needs read permissions on all
users' .ssh/authorized_keys file to allow public key authentication for
these users!. <Re->running ssh-user-config for each user will set the
required permissions correctly.

Which value should the environment variable CYGWIN have when
sshd starts? It's recommended to set at least "ntsec" to be
able to change user context without password.
Default is "ntsec". CYGWIN=binmode ntsec tty

The service has been installed under sshd_server account.
To start the service, call 'net start sshd' or 'cygrunsrv -S sshd'.

Host configuration finished. Have fun!

Administrator@Wisla ~
$ -
```

Figure 5-91 ssh-host-config utility

5.8.2 Adding or changing a Windows gateway

In this section, we tell you how to add, change, and delete a Windows gateway.

Adding a Windows gateway

To add a Windows gateway, complete the following steps:

1. On the Discovery menu, click **Anchors and Gateways**.
2. Click **Add** to define a new gateway.
3. In the Add Anchor window (“Editing the scope of a gateway” on page 191), from the Type list box, select **Windows gateway**.
4. Select **Address** or **Host Name** to set the Windows host that you are adding.
5. Select **Entire Scope** or **Limit to selected scope** to set the scope, and click **OK**. If you select **Limit to selected scope**, select the scope set to use.

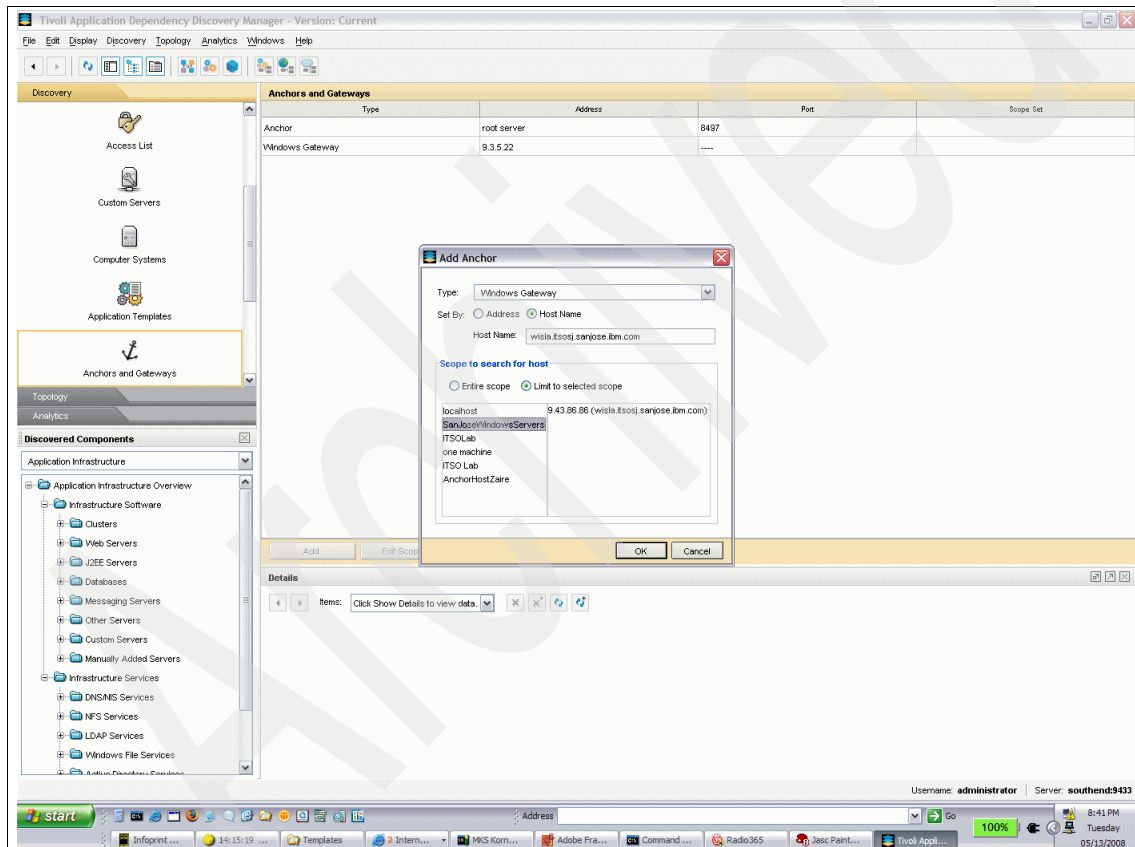


Figure 5-92 Adding a gateway and setting the scope

Editing the scope of a gateway

To edit the scope of a gateway, complete the following steps:

1. On the Discovery menu, click **Anchors and Gateways**.

2. Select an anchor or gateway.
3. Click **Edit Scope**.
4. Change the scope set for the anchor or gateway.
5. Click **OK**.

Deleting a gateway

To delete a gateway, complete the following steps:

1. On the **Discovery** menu, click **Anchors and Gateways**.
2. Select a gateway.
3. Click **Delete**.

Discovering your Windows gateway

After you define your Windows gateway, one way to verify that it is working (for example, credentials are valid, SSH client is set up properly, TADDM Server can distribute binaries, and so forth) is to run a discovery to discover the Windows gateway. You can discover the Windows gateway using the following steps:

1. Go to **Discovery** → **Scope**, and click **Add Set** to create a new set called WindowsGateway. Click **OK**.
2. Click **Add** to add the new target.
3. Go to **Discovery** → **Access List**, and click **Add** to add a new credential. Complete the fields in the window with the correct information.
4. Go to **Discovery** → **Overview**, and run a new discovery (Figure on page 183) of your new Windows gateway, in our case, Wilas.

Note: Before you can discover the Windows gateway, the SSH client must be installed. Refer to 5.8.1, “Installing Cygwin SSH” on page 183 for an example of installing Cygwin.

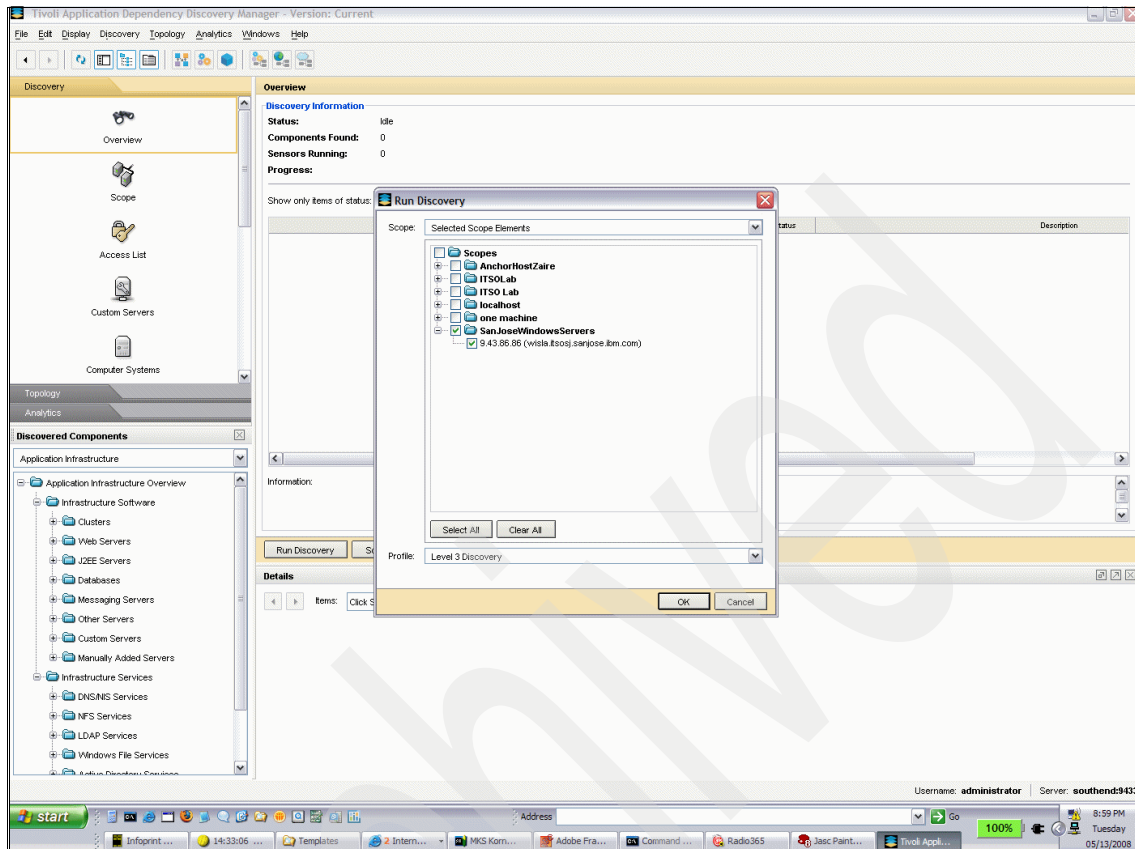


Figure 5-93 Running discovery of new Windows gateway

5.9 Troubleshooting

TADDM provides comprehensive logging features that allow you to isolate any problems that you might experience.

5.9.1 Server not started automatically

If the TADDM Server does not start automatically, check the following settings (execute as the CMDB user, not as root):

- ▶ To verify that your system was set up to start the TADDM Server during startup, use the **chkconfig --list collation** command to see at which OS-startup levels the service is active:

```
[root@taddm bin]# chkconfig --list collation
collation 0:off 1:off 2:on 3:on 4:on 5:on 6:off
```

- ▶ To disable automatic start of the TADDM Server at OS-startup levels 2, 3, 4, and 5, use the following command:

```
[root@taddm bin]# chkconfig --level 2345 collation off
```

- ▶ To verify your new settings, issue a new **chkconfig --list** command:

```
[root@taddm bin]# chkconfig --list collation
collation 0:off 1:off 2:off 3:off 4:off 5:off 6:off
```

Starting and stopping the TADDM Server

You can use the built-in service interface to start, stop, and query the status of your TADDM Server:

- ▶ To start the server, use:
control start
- ▶ To stop the server, use:
control stop
- ▶ To get the server status, use:
control status

Server start is slow

If your server starts slowly, there might be information left over from previous discoveries, so clean out the contents of:

`${COLLATION_HOME}/var/dwitem` and `${COLLATION_HOME}/var/scwitem`

5.9.2 Installation log files

The installation log files are located in the installLogs subdirectory in the location where you installed the product, which is typically /opt/IBM/cmdmb. The names of the files that are produced during installation are self-explanatory. The files are:

- ▶ cdb_install.log
- ▶ cdb_install_stdout.log
- ▶ cdb_start_server_sterr.log
- ▶ cdb_install_stderr.log
- ▶ cdb_start_server.log
- ▶ taddm_IF.log



Part 3

Discovery and Reporting Case Studies

In this part, we discuss discovery and reporting in Tivoli Application Dependency Discovery Manager and guide you through several case studies.

Discovery scenarios

In this chapter, we describes how components, configuration data, and dependencies are discovered and fed into the IBM Tivoli Application Discovery and Dependency Manager (TADDM) database from both the built-in sensors, the customizable templates, such as custom servers and computer systems, and loading discovery library books that were generated by external solutions.

TADDM provides a variety of specialized sensors that cover most of the infrastructure components that are used in large enterprises today. But in order to support custom applications and special cases, the openness of the Change and Configuration Management Database (CCMDB) architecture allows for additional ways to automatically feed data into the solution. This chapter describes these options, starting with feeds related to discoveries and then presenting how to manipulate the database from external systems.

The major topics that we discuss in this chapter are:

- ▶ “Discovery sensors” on page 200
- ▶ “Customizing and managing discoveries” on page 230
- ▶ “Reconciliation and prioritization” on page 250
- ▶ “Discovery Library Adapters” on page 263
- ▶ “Understanding the DLA APIs” on page 270
- ▶ “Example of Discovery Library Adapter” on page 282

6.1 Discovery sensors

TADDM discovers and collects configuration information for the entire application infrastructure, identifying deployed software components, physical servers, network devices, virtual LAN, and host data used in a runtime environment.

Discovery is performed using sensors that are currently built and deployed as part of the TADDM product. The sensor asks, figuratively, the host and the applications how they are configured and to whom they are talking.

Discovery sensors are the heart of TADDM and the primary way of populating the TADDM Database. In this section, we describe how data is fed into TADDM during discovery.

6.1.1 Discovery overview

The TADDM Agent-free discovery engine manages the overall discovery process. The discovery process collects the data that is needed to populate the Common Data Model to represent the specific data center infrastructure. Core to the discovery process are lightweight *discovery sensors*, which build upon the Common Data Model to comprehensively discover the infrastructure components, their configurations, and dependencies.

Sensors provide the TADDM product with an application discovery and dependency mapping facility to discover the relationships between components executing in your IT infrastructure.

6.1.2 Discovery components

The TADDM Discovery subsystem is composed of the following components:

- ▶ Discovery engine (Discovery Manager)
- ▶ Sensors
- ▶ Discover JavaSpace
- ▶ Discover observer
- ▶ Process Flow Manager

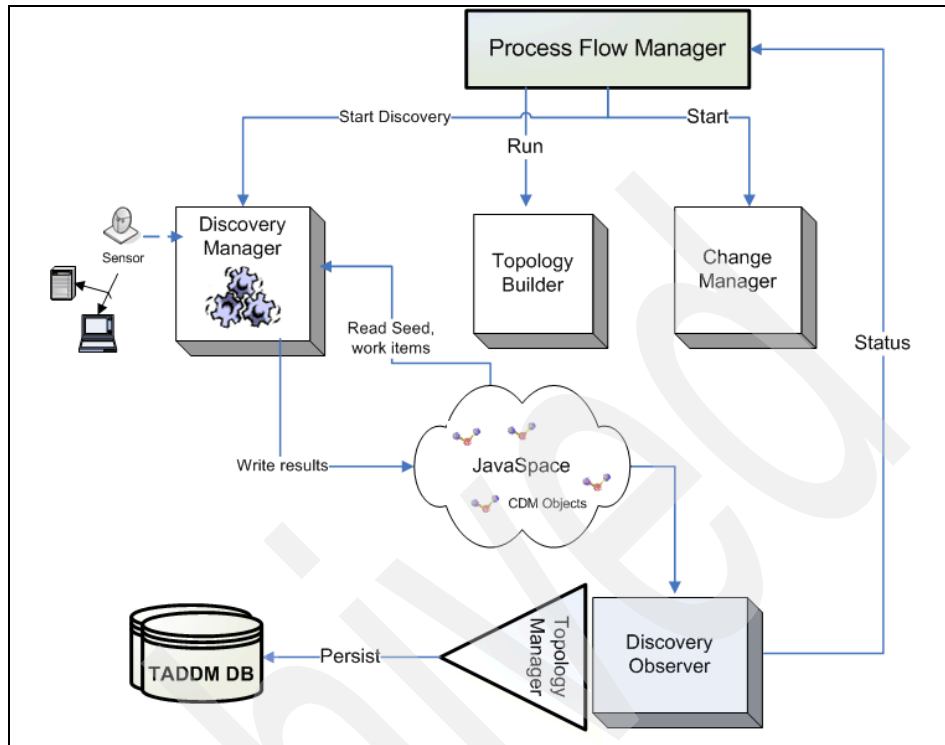


Figure 6-1 Discovery components

Discovery engine

The discovery engine service is responsible for discovering the contents of the datacenter, which it does by running discovery sensors. A *discovery workflow* determines which sensor to run against what target.

The discovery engine performs its discovery using a set of discover sensors. Each discover sensor has an input object called a *seed*, then a *discover step*, and an output called a *result*. Discover workflow is determined by a *converter*, which takes a result as input and produces a new seed object.

A discovery is started by providing a set of initial seeds. This initial seed is typically an IP address or a range of IP addresses to discover. This seed provides a starting point for the discovery engine, which triggers the initial sensors. The results from these sensors are then converted into new seeds, which trigger a new set of sensors, and so on, until no more result-to-seed conversions can be made.

Sensor

The sensor is the primary agent for discovery in TADDM. It is responsible for probing the remote system and discovering configuration and dependency relationship information about its characteristics. The data that the sensor discovers is mapped into model objects that then get saved to the database. For certain sensors, discovered results also cause new seeds to be created; thus, new sensors are spun off.

Discover JavaSpace

Discover JavaSpace is the integration point for the discovery workflow. It is used as a synchronization space for the implementation of the discovery logic as a master and worker pattern. During a discovery run, the discovery engine stores the seed and result objects in JavaSpace, which maintains a number of discovery tasks that the sensor threads perform:

- ▶ The sensor selects the seeds according to the discovery tasks that represent the target system.
- ▶ The sensor discovers the target system and returns results.
- ▶ The sensor threads create new discovery tasks from seeds.

JavaSpaces are closely tied to the Jini architecture. You can read about Jini and JavaSpaces at the following Web site:

<http://www.jini.org>

Discover observer

The *discover observer* service persists the results that the discover sensors discover by communicating with the Topology Manager. It also determines the status of a discovery by observing the contents of the JavaSpace.

Process Flow Manager

The *Process Flow Manager* service controls the discover state, and it exposes an application programming interface (API) to the external world that starts and cancels a discovery. It also takes care of running necessary post-discover steps (such as the Topology Builder that builds the relationships and dependencies among the discovered items).

6.1.3 Discovery process in detail

You can initiate the discovery process from the GUI or API by specifying an initial scope (seed). This initial seed is typically an IP address or a range of IP addresses to discover. A discovery workflow process is displayed in Figure 6-2 on page 203.

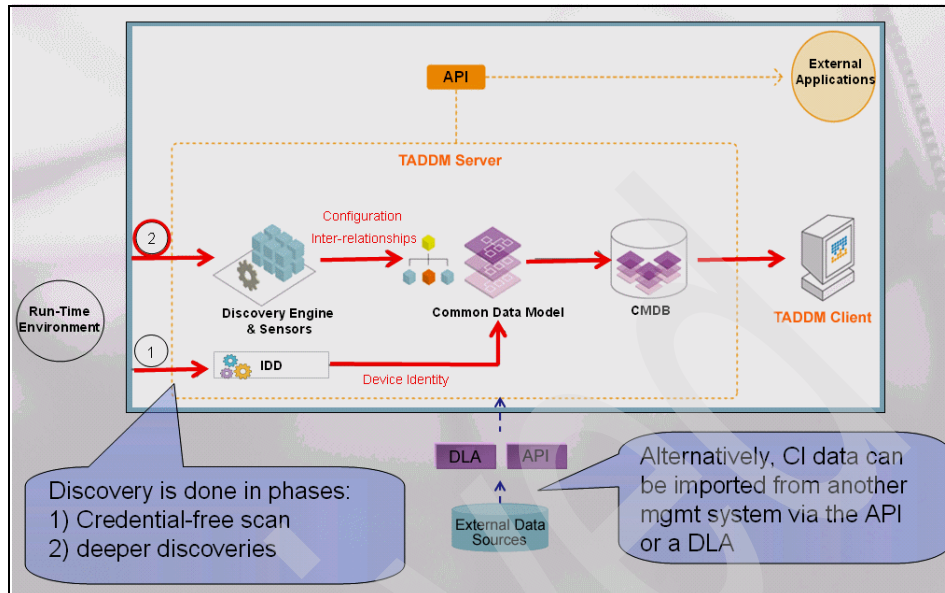


Figure 6-2 Discovery workflow process

Level 1 discovery

For a level 1 discovery profile, the *StakScan* sensor performs credential-less discovery. The *StakScan* sensor can discover active computer systems in the runtime environment. It discovers all computer systems in your environment and tries to determine the operating system that runs on each computer system with a certain level of confidence.

The *StakScan* sensor uses multiple ways to discover the computer systems in order to determine the type of computer system. It utilizes a heuristic-based approach to determine whether a given OS type is present or not. When launched, the sensor tries the following three techniques to collect information from a computer system:

- ▶ Host scanning rule-based OS fingerprinting to determine the type of OS
- ▶ The use of the Open Source tool Nmap and OS information
- ▶ Remote Execution and Access (RXA) to determine OS level details, such as OS name, level, and so on

Based on the information collected, the sensor guesses the installed OS type with a certain confidence level. If this confidence level is greater than a certain threshold, a computer system of that OS type is created; otherwise, the computer system is left as an IP device type to be classified appropriately later by a deeper discovery. You can configure the threshold for the confidence level.

Tip: The StakScan sensor uses remote anchors, such as other TADDM sensors, and no gateway access is needed for Windows discovery.

Level 2 and 3 discovery

Figure 6-3 displays the basic discovery flow for the level 2 and level 3 discovery profiles. Note that each box in Figure 6-3 represents a different native TADDM sensor.

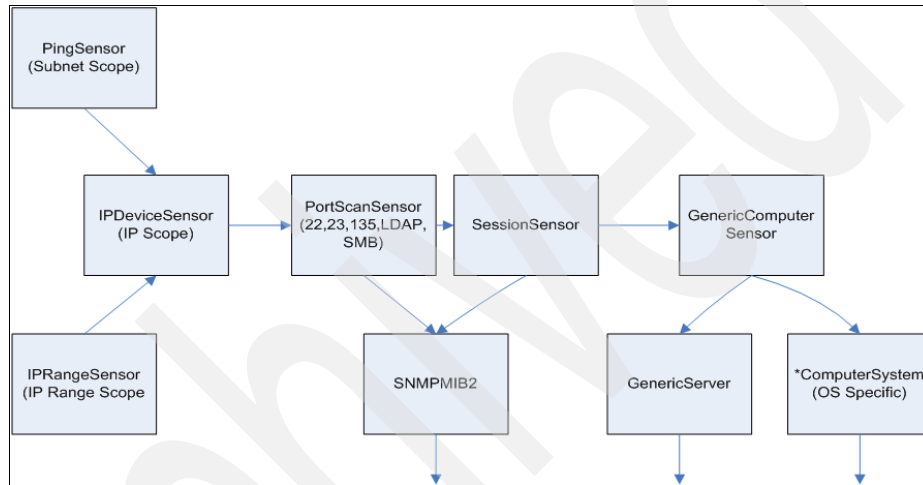


Figure 6-3 Basic discovery sensor sequence

The basic discovery flow actions are:

1. TADDM specifies an initial scope (seed) for a discovery run, either through the GUI or an API call. This initial scope becomes the first seed for the discovery run and is written to the Java Spaces.
2. TADDM identifies the active IP devices in the chosen scope:
 - TADDM attempts a Transmission Control Protocol (TCP) connection on several ports (such as 22 and 135) looking for a response.
 - Any response is enough to notify TADDM that the device exists.
 - An IP device is created, and a PortScan seed is created.

3. TADDM determines if there is a method of establishing a session to the IP device:
 - The PortSensor tries a TCP connection on several ports (including 22 and 135) to try to establish what technology TADDM uses to discover the host.
 - TADDM creates either a SessionSensor seed or an SnmpMib2Sensor seed.
4. The session sensor activities:
 - a. If the SSHPort was open, the session sensor tries to establish a Secure Shell (SSH) connection using credentials from the Access List.
 - b. The session sensor tries to use Access List entries of type computer system or Windows computer system, in sequence, from the Access List until an entry works or until the list is exhausted.
 - c. If the Windows Management Interface (WMI) port was open, the session sensor establishes an SSH connection with a gateway computer system (provided that a gateway computer system is found for the target).
 - d. If the session sensor cannot establish a session, an SnmpMib2Sensor seed is created.
 - e. If a session is established, a Generic Computer System Sensor seed is created.
5. The Generic Computer System Sensor:
 - Tries to determine what type of OS is installed, such as AIX, Linux, SunOS, Hewlett-Packard UNIX (HP-UX), Windows, Tru-64, OpenVMS, and so on.
 - Creates a seed that is appropriate for the OS and discovers other system components, such as sharing, storage, and so on, as shown in Figure 6-4 on page 206.
6. An OS-specific sensor is invoked and uses native commands, such as **ps**, **netstat**, and **lsof**, to find a list of all of the running processes that are listening on a socket.
7. The sensor proceeds to perform its application (ISS, Apache, and so on) discovery, and the discovered results are written back to the Java Space. Next, the discovery observer detects that a result was placed in the Java Spaces and extracts the result.

Figure 6-4 on page 206 illustrates the operating system and application discovery process.

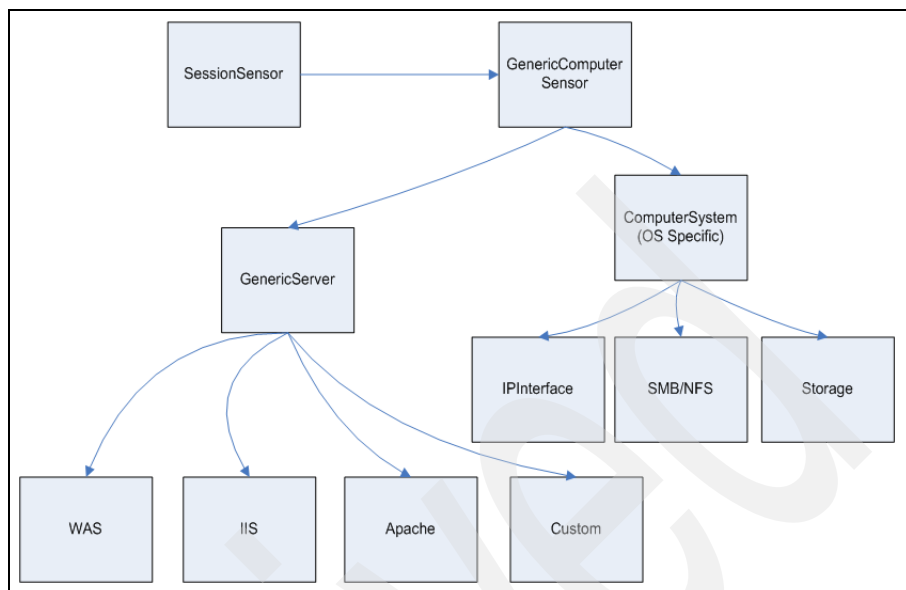


Figure 6-4 OS and application discovery

8. The discovery observer saves the result to the database. If there is a result converter associated with the sensor, the discovery observer passes the result to that result converter.
9. The result converter parses the result and might generate new seeds, which are put back in the Java Spaces.

TADDM is shipped with many built-in sensors. You cannot modify these sensors, but you can disable and enable them when needed. IBM continuously develops new sensors that are based on client needs.

One of the available sensors is the *CustomServerSensor*. Its objective is to discover (or exclude) custom-defined server processes that were defined through the definition (and enablement) of one or more Custom Server templates. We discuss Custom Server templates in 6.2.1, “Custom servers” on page 231.

Our discussion about the discovery process is complete. Although much of the functionality that we described occurs in the background, it is useful to have detailed knowledge when you dive into discovery tuning and troubleshooting. For details about how to run discoveries, refer to 6.1.7, “Discovery profiles” on page 218.

6.1.4 Dependency discovery

Relationships specify the connection among various entities more precisely and gives meaning to the type of information that is stored at each entity. Without relationships between data objects in the TADDM Database, the data store is simply a repository for discovered configuration information and does not provide a complete picture of the information technology enterprise. In this section, we discuss dependency and relationship discovery.

Dependency and relationship overview

As part of the discovery process, the discovery feature examines the configuration of each device and discovers the ports that are assigned to the applications. The discovery feature uses this information to determine the relationships and the dependencies between applications and other discovered components. A dependent component relies on data or configurations from another component, and a provider component provides information to a dependent component.

TADDM discovers dependencies in two ways:

- ▶ By looking at the TCP connections that the `lsotf` command lists
- ▶ By looking at the configuration of programs, such as Java Database Connectivity (JDBC) resources that are returned by Java Management Extensions (JMX)

In the first case, only TCP connections that are present at the time when the discovery runs the `lsotf` command are discovered.

Eventually, a discovery catches those connections that are established, and creates the dependencies. If you do not want to wait that long, you can manually create a connection. After the connection is actually discovered, the manual connection is promoted to a transactional dependency.

Dependency types

There are two types of dependencies: transactional and service.

Transactional dependencies occur between application components, such as Web servers, application servers, and databases. The dependent component issues requests to the provider component in order to perform certain functions, such as Java Database Connectivity (JDBC) calls from a Java 2 Platform, Enterprise Edition (J2EE) server to a database. In this case, the provider is often referred to as a *server* and the *dependent* as a consumer or client.

Service dependencies occur between application components and infrastructure services, such as Domain Name System (DNS), Lightweight Directory Access Protocol (LDAP), and Network File System (NFS). The provider is the infrastructure service, and the dependent component requests system services from the provider, such as a request to map a DNS name to an IP address.

6.1.5 Understanding sensors

Sensors work by emulating a user running locally to gather (discover) information. The sensor is able to gather discovery-related information without incurring any of the costs associated with the installation and maintenance of an agent locally on the client machines to be discovered. The sensors use secure network connections, encrypted access credentials, and host native utilities. In this way, a sensor is safe to use and provides the same data acquisition that you acquire by having the software residing locally on the client machine.

Using a discovery profile, you take control of what you discover. For example, you can configure individual sensors, manage multiple configurations of the same sensor, select the appropriate configuration based on a set of criteria, and manage sets of configurations of multiple sensors to be applied on a single discovery run. When you run a discovery, you must select a profile. If no profile is selected, the discovery is run against the default profile, which is the Level 3 discovery. By default, there are three levels of discovery profiles that you can choose, depending on the type of discovery that you want. The default profile can be changed in the Product Console. The profiles provide the following capabilities:

- ▶ You can use credential-less scanning to discover basic information about the active computer systems in the runtime environment.
- ▶ You can use OS credential scanning to gather detailed information about each operating system in the runtime environment.
- ▶ You can use full credential scanning that discovers the entire application infrastructure, deployed software components, physical servers, network devices, virtual LAN, and host data used in a runtime environment.

Sensor logging

There is a property in the \$COLLATION_HOME/etc/collation.properties file that improves readability of the logs by separating the logging into per-sensor log files for each discovery run. To enable this capability, set the following property as is shown in Example 6-1 on page 209.

Example 6-1 Sensor logging

```
com.collation.discover.engine.SplitSensorLog=true
```

Important: The default logging for all of the sensors is put in the `$COLLATION_HOME/log/services/DiscoveryManager.log` file if you do not set this property to true.

This property separates the logs into the following directory structure:

```
$COLLATION_HOME/log/sensors/<runid>/sensorName-IP.log
```

For example:

```
sensors/20070621131259/SessionSensor-10.199.21.104.log
```

Note:

- ▶ The *runid* includes the date of the discovery run, and the *log* file name includes the sensor name and IP address of the target.
- ▶ When using this option, the logs are not automatically cleared. You must clear the logs manually, if required.

6.1.6 Setting up discoveries

TADDM automatically discovers and provides a top-down, application-centric, topological map of the data center infrastructure. TADDM discovers many configuration items, including:

- ▶ Software applications, hosts, network devices, and software servers
- ▶ Configuration attributes of the discovered components
- ▶ Runtime dependencies of the various components

To optimize the breadth and depth of the information that TADDM gathers, there are setup tasks that are required within TADDM and in your environment.

For discoveries to run in your environment, you need to provide TADDM with three types of information:

- ▶ Discovery scope
- ▶ Access Lists (not required for credential-less discoveries)
- ▶ Schedule

In this section, we focus on how to prepare TADDM and your environment for running discoveries. Discovering your environment is an iterative process. A full understanding of your IT infrastructure develops through successive discoveries.

In the Product Console, either the Discovery tab or the Discovery menu gives you access to the items that we need in order to configure discoveries.

Discovery scope

A *discovery scope* identifies the devices, computer systems, and other components in your infrastructure that you want the server to access. You specify scopes using IP addresses, ranges of IP addresses, or subnets to define the boundary of the networks that can be accessed during discovery. A scope can be as small as a single IP address or as large as a range of IP addresses or a Class C network. You can also exclude specific devices from the scope.

When there is a firewall between the server and the systems, which are in another area of your network, that you want discovered, you must configure the firewall to allow access on the SSH port (port 22) and then set up an anchor. Refer to “Anchor servers” on page 52 for more information about anchor servers.

Table 6-1 describes the information that is displayed for a discovery scope in the Scope pane.

Table 6-1 *Discovery scope information*

Information	Description
Method	Specifies whether to include or exclude the IP address, IP address range, or subnet
Type	The type of address (from among the following options) that is specified: Subnet: An IP subnet, for example, 255.255.255.0 Range: IP address range, for example, 1.2.3.4 - 1.2.3.10 Host: IP address, for example, 1.2.3.4
Value	The actual IP address, IP address range, or subnet
Description	A user-supplied description or host name of the discovery scope

Setting the discovery scope

To configure a scope set and scope, complete the following steps from the Product Console:

1. From the menu bar, click **Discovery** → **Scope**. The Scope pane is displayed.
2. To define a new discovery scope set, click **Add Set**. The Scope Set Name window is displayed.
3. In the Name field, type the name for the new scope set.
4. Click **OK**. The new scope set appears in the Scope Sets list.
5. To add the scope and contents to the scope set, select the scope set that you just created and click **Add**. The Add Scope window is displayed.
6. To add the settings for the scope, complete one of the following steps:
 - f. Select Subnet from the IP Type list and type the IP address of the subnet in the IP Address field. This IP address must be a unique value within the scope set.
 - g. Select Range from the IP Type list and type the starting IP address and the ending IP address in the IP Addresses field. These IP addresses must be unique values within the scope set.
 - h. Select Host from the IP Type list and type the IP address of the host in the IP Address field or type the host name in the Hostname field. This IP address or host name must be a unique value within the scope set.
7. To exclude devices from your scope, click Add Exclusion and complete one of the following steps:
 - a. From the IP Type list, select Subnet and type the IP address of the subnet in the IP Address field.
 - b. From the IP Type list, select Range and type the starting IP address and the ending IP address in the IP Address field.
8. To save the scope, click **OK**. The new scope appears in the list.

Using the command line to load scopes

Rather than using the GUI, it might be faster and easier to set up scopes by creating a text file or files and loading them into TADDM using the following command line, located in:

```
loadscope.jy [-d] -u <username> -p <password> -s <scopeset> \  
load <scopefile>
```

Where:

The following definitions explain the command:

-d	Turns on verbose debug logging
-u username	User name under which the command runs
-p password	Password matching the user name
-s scopeset	Scope set to use for loading the scope elements
load	Loads the scope entries, and appends new entries to existing entries
scopefile	Name of the file containing the scope entries

In Example 6-2, we show you step-by-step how to use the **loadscope.jy** command:

Example 6-2 Using the loadscope.jy command

```
1) Create a file with the scope information  
$ cat ITSOLab.scope  
# ITS0 Lab Scope  
# Adding the entire ITSOLab Network, excluding the default gateway and  
the DNS server  
9.3.5.0/255.255.254.0,9.3.5.1: 9.3.5.2,ITSOLab Scope  
  
2) Go to COLLATION_HOME/bin directory:  
$cd $COLLATION_HOME/bin  
  
3) You are able to execute loadscope.jy command, as is showned:  
$ ./loadscope.jy -d -u administrator -p collation -s "ITS0 Lab" load  
/home/cmdbadmin/ITSOLab.scope  
Processing: # ITS0 Lab Scope  
Processing: # Adding the entire ITSOLab Network, excluding the default  
gateway and the DNS server  
Processing: 9.3.5.0/255.255.254.0,9.3.5.1: 9.3.5.2,ITSOLab Scope  
Successfully parsed  
DEBUG--> XML stored in /tmp/tmp1loadscope.xml
```

After the scope is loaded, you can check the results by using the GUI or directly opening the /tmp/tmp1loadscope.xml file.

Using the GUI, go to **Discovery** → **Scope** and click **ITSO Lab** Set Scope. You will see a panel as shown in Figure 6-5.

Scope				
Scope Sets	Method	Type	Value	
ITSO Lab	Include	Subnet	9.3.4.0/255.255.254.0	ITSO Lab Scope
	Exclude	Host	9.3.5.1	9.3.5.1
	Exclude	Host	9.3.5.2	9.3.5.2

Figure 6-5 Viewing the loaded scope using the GUI

To open the tmp1loadscope.xml file, you can use the **cat** command on the command line interface (CLI) of a Linux or UNIX machine. For example:

```
$ cat /tmp/tmp1loadscope.xml
```

After you enter this command, you see the content of the file, which is similar to Figure 6-6 on page 214.

```
cmdbadmin@waco:/opt/IBM/cmdb/dist/bin
<?xml version="1.0" encoding="UTF-8"?>
<results
  xmlns="urn:www-collation-com:1.0"
  xmlns:coll="urn:www-collation-com:1.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="urn:www-collation-com:1.0
urn:www-collation-com:1.0/results.xsd">
  <Scope array="1" xsi:type="coll:com.collation.platform.model.discovery.
    <displayName>ITSO Lab</displayName>
    <name>ITSO Lab</name>
    <elements array="1" xsi:type="coll:com.collation.platform.model
      <displayName>ITSO Lab Scope</displayName>
      <name>ITSO Lab Scope</name>
      <keyGuid>1209065794414120906579441412090</keyGuid>
    <excludes array="1" xsi:type="coll:com.collation.platform.model
      <displayName>9.3.5.1</displayName>
      <name>9.3.5.1</name>
      <keyGuid>1209065794414120906579441412090</keyGuid>
      <ip>9.3.5.1</ip>
    </excludes>
    <excludes array="2" xsi:type="coll:com.collation.platform.model
      <displayName>9.3.5.2</displayName>
      <name>9.3.5.2</name>
      <keyGuid>1209065794414120906579441412090</keyGuid>
      <ip>9.3.5.2</ip>
    </excludes>
      <ipAddress>9.3.4.0</ipAddress>
      <subnetMask>255.255.254.0</subnetMask>
    </elements>
  </Scope>
</results>
[cmdbadmin@waco bin]$
```

Figure 6-6 Viewing the loaded scope using the CLI

Access Lists

The *Access List* is a collection of all user names, passwords, and Simple Network Management Protocol (SNMP) community strings that the server uses when accessing entities, which are stored as configuration items in the database, in your infrastructure. You have to set up this list for the Configuration Items that you want to discover.

User names, passwords, and community strings, if needed, are categorized by each type of device or software application and optionally restricted by scope. For example, all user names and passwords for all computer systems are stored as a group, and all user names and passwords for all databases are stored as another group.

When accessing a device, the server sequentially uses each user name and password (or community string) in the group across a particular scope (IP address per subnet) until the device allows the server permission to access it. For example, when accessing a computer system, the server uses the first user name and password specified in the Access List for computer systems. If the user name and password are incorrect for a particular computer system, the

server automatically uses the next user name and password that are specified in the Access List for a computer system.

Because you enter a list of user names and passwords (or community strings) for each type of configuration item, you do not need to specify a user name and password for a particular configuration item. When you specify all of the user names and passwords for each type of device, define the scope for each user name and password pair. The server automatically tries each user name and password until the correct combination is found. The access list that you create is stored in the database in an encrypted file.

If the device that you are discovering is an SNMP network element, enter an SNMP community string in the Community field. If you are using SNMP for a Cisco device, you must select the SNMP network element and enter an SNMP community string in the Community field for the Cisco device.

For each Computer System entry in the Access List, you have the option to specify one of the following authentication types:

- ▶ Default
- ▶ Password
- ▶ Public key infrastructure (PKI)

If you use the default, SSH key-based authentication is attempted first, using the password for the key passphrase, if required. If key-based authentication does not succeed, the login name and password authentication is attempted. If password authentication type is selected, only password authentication is attempted. Similarly, if PKI is selected, only key-based authentication is attempted. We recommend that you set the authentication type for the new Access List entry being added if you know the type. If you do not know the authentication type, the default behavior can lead to multiple invalid login attempts that can sometimes result in the account being locked out.

In cases when your system administrator has set up SSH with the login and password authentication method, start the Product Console with the Establish a Secure (SSL) Session option enabled before you set up the Access List. This option encrypts all of the data, including Access List user names and passwords, before the data is transmitted between the Product Console and the server.

When using the StackScan sensor (Level 1 discovery profile), no Access List is required.

Note: If you only use the StackScan sensor, you also only get the bare minimum CI data from the entities in the environment.

Adding a new Access List

The steps for adding a new Access List entry vary based on the component type that you want to add. To add a new Access List entry, complete the following steps from the Product Console:

1. From the menu bar, click **Discovery** → **Access List**. The Access List pane is displayed.
2. To add a new entry into the Access List, click **Add**. The Access Details notebook is displayed.
3. From the Component Type list, select the component type that you want to discover.
4. For all component types other than Network Element (SNMP), complete the following steps:
 - a. In the Name field, type the name of the Access List entry.
 - b. In the User name field, type the user name to log in to the component that you want to discover.
 - c. In the Password field, type the password to log in to the component that you want to discover.
 - d. In the Confirm Password field, retype the password for confirmation.
5. Additional steps might be required based on the component type that you selected. Table 6-2 on page 217 identifies the component types and the additional fields and lists that you are required to complete for the Access List entry.

Table 6-2 Access details by components

Component types	Fields and lists
Application server, database, messaging servers	Vendor: The vendor of the server or database.
Cisco device	<p>Enable password: The enable password for the Cisco device, if you are using Telnet.</p> <p>Confirm enable password: The enable password for the Cisco device, if you are using Telnet.</p> <p>The Telnet Cisco sensor requires the SNMP sensor to be established and working against the device. If your Cisco device does not prompt for a user name, type default in the User name field.</p>
Computer System Management System (CCMS)	Client ID: The client ID of the SAP CCMS server.
Network Element (SNMP)	<p>Community string: The community string for the network device.</p> <p>Confirm community string: Confirm the community string for the network device.</p> <p>The SNMP network element must be configured to answer queries from the TADDM Server IP address.</p>
Network Element (SNMPV3)	<p>Private password: The password used if data encryption is set for SNMP.</p> <p>Confirm private password: The password used if data encryption is set for SNMP.</p> <p>Authentication protocol: The type of authentication protocol used by SNMP.</p>

- To configure the scope limitations, click the **Scope Limitations** tab. The Scope Limitations page is displayed.

7. On the Scope Limitations page, complete one of the following steps:
 - To use the access information across all of the components of the entire discovery scope, click Entire scope.
 - To restrict the application of the specific access information to certain systems, click Limit to selected scope and then select the scope to which you want to restrict access.
8. To save the new Access List entry, click **OK**.

6.1.7 Discovery profiles

A *discovery profile* is a set of rules that controls how discoveries run. Using discovery profiles, you can control what TADDM discovers. For example, you can configure individual discovery sensors, manage various configurations for the same sensor, select the appropriate sensor based on a set of criteria, and control sets of different sensor configurations to be applied on a single discovery run.

By default, there are three levels of discovery profiles, which are described in Table 6-3.

Table 6-3 *Discovery profiles by default in TADDM*

Level	Type of discovery	What is discovered
Level 1	credential-less	Basic information about active computer systems: host name, fully qualified domain name (FQDN), OS release level, IP address, and open ports. Network operating systems, such as Cisco and Alteon, are also discovered. On Windows and zLinux hosts, the Media Access Control (MAC) address is discovered.
Level 2	OS credentials only	Detailed information about all of the operating systems.
Level 3	Full credential	Entire application infrastructure: deployed software components, physical servers, network devices, virtual LANs, and hosts.

Level 1 discovery: IDD StackScan sensor

The *Intelligent Device Discovery® (IDD) StackScan* sensor provides credential-less discovery using stack classification for a less intrusive mapping of the installed operating system and open ports on a computer system. The

StackScan sensor can collect the type of operating system, the active IP interfaces, and the open ports without access credentials.

Note: IDD is an asset discovery and inventory tool for hardware and software in a heterogeneous, unknown systems, and networking enterprise environment. It was developed by IBM Zurich Labs. The StackScan sensor is the TADDM implementation of IDD.

The StackScan sensor labels each discovered computer system with a confidence level for the operating system. If the discovered operating system confidence level is higher than a threshold, the computer system is displayed under the appropriate operating system category. If the operating system confidence level is lower than the threshold, the operating system is modeled as a general computer system. The threshold is configured between 0 - 100. You can set the threshold by using the `confidenceThreshold` sensor configuration attribute.

To enable and disable the StackScan sensor and to set the `confidenceThreshold` attribute, use a discovery profile.

Note: The StackScan sensor is enabled by default in all discovery profiles.

The IDD StackScan sensor is not supported when the TADDM Server is running on an AIX operating system. There is no reliable Nmap version available. For supported Windows operating systems, the StackScan sensor needs raw socket support enabled on the operating system (for example, Windows Server 2003) where the TADDM Server is running. If the operating system does not provide raw socket support, the StackScan sensor cannot work, and data collection does not occur.

Note: Raw sockets are required to construct packets to send to remote hosts for reading the TCP stack and determining the operating system type. If absent, the sensor cannot formulate and send that information. If you do not know if you have raw socket support, check the Web site or support site for your operating system to determine if the raw socket support is enabled.

Configuring sudo access control

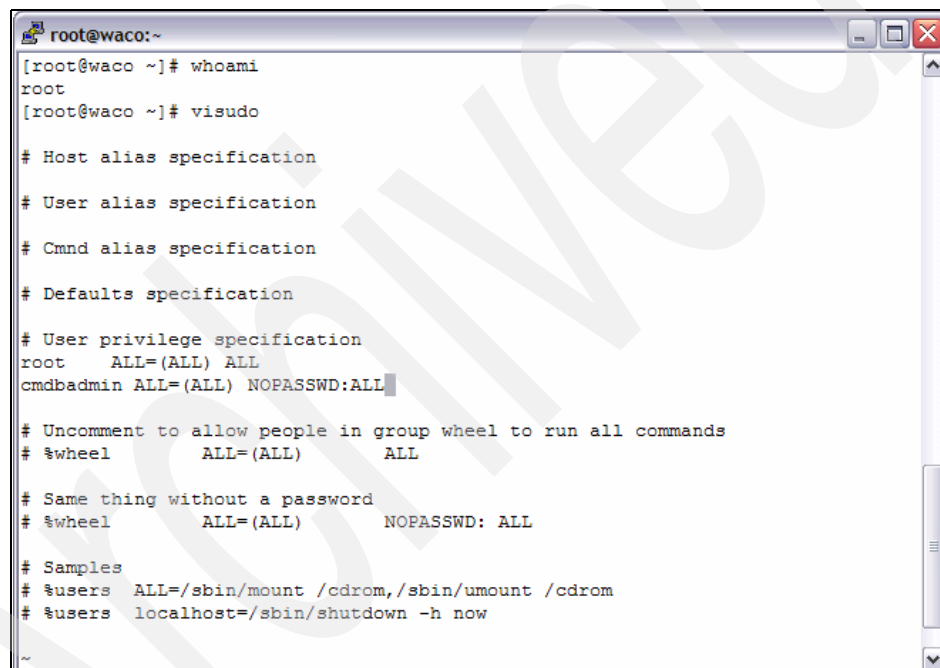
The StackScan sensor requires sudo access control to collect discovery information. For Windows operating systems, sudo access control is not needed.

To configure sudo access, complete the following steps for anchor hosts and TADDM Servers for other operating systems:

1. At a command prompt window, run the **su** command to switch to root authority on the local host.
2. Enter the **visudo** command.
3. Type the following line under User privilege specification:
`cmdbadmin ALL=(ALL) NOPASSWD:ALL`

The cmdbadmin is the non-root user ID that is used by the TADDM Server.

The output looks similar to Figure 6-7.



```
root@waco:~  
[root@waco ~]# whoami  
root  
[root@waco ~]# visudo  
  
# Host alias specification  
  
# User alias specification  
  
# Cmnd alias specification  
  
# Defaults specification  
  
# User privilege specification  
root    ALL=(ALL) ALL  
cmdbadmin ALL=(ALL) NOPASSWD:ALL  
  
# Uncomment to allow people in group wheel to run all commands  
# %wheel    ALL=(ALL)        ALL  
  
# Same thing without a password  
# %wheel    ALL=(ALL)        NOPASSWD: ALL  
  
# Samples  
# %users    ALL=/sbin/mount /cdrom,/sbin/umount /cdrom  
# %users    localhost=/sbin/shutdown -h now  
  
~
```

Figure 6-7 Configuring sudo access

Installing Nmap

If you use the StackScan sensor, the use of Nmap is optional. If you want to increase the accuracy of your results, use Nmap.

Nmap is an open source network exploration tool and security scanner. When using the StackScan sensor, install Nmap as a part of the Tivoli Application Dependency Discovery Manager installation process.

Nmap must be installed in the same operating system where your TADDM Server is installed. If you use an anchor server, Nmap must be installed on the anchor server, too.

To download the latest version of Nmap for your specific operating system, go to:

<http://nmap.org/download.html>

In our lab environment, we used Nmap 4.20 for Linux. We installed Nmap following the installation steps from:

<http://nmap.org/book/install.html>

When your Nmap installation is done, run the following command to validate the installation before proceeding with any tasks:

```
nmap -v -sS -O <Target IP Address>
```

The output must not contain any errors; for example, refer to Example 6-3.

Example 6-3 Executing Nmap after install

```
# nmap -v -sS -O 9.3.5.22
```

```
Starting Nmap 4.20 ( http://insecure.org ) at 2008-04-24 12:30 CDT
Initiating ARP Ping Scan at 12:30
Scanning 9.3.5.22 [1 port]
Completed ARP Ping Scan at 12:30, 0.01s elapsed (1 total hosts)
Initiating Parallel DNS resolution of 1 host. at 12:30
Completed Parallel DNS resolution of 1 host. at 12:30, 0.00s elapsed
Initiating SYN Stealth Scan at 12:30
Scanning newyork.itsc.austin.ibm.com (9.3.5.22) [1697 ports]
Discovered open port 3389/tcp on 9.3.5.22
Discovered open port 445/tcp on 9.3.5.22
Discovered open port 139/tcp on 9.3.5.22
Discovered open port 22/tcp on 9.3.5.22
Discovered open port 5900/tcp on 9.3.5.22
Discovered open port 1025/tcp on 9.3.5.22
Discovered open port 135/tcp on 9.3.5.22
Discovered open port 5800/tcp on 9.3.5.22
Completed SYN Stealth Scan at 12:30, 1.37s elapsed (1697 total ports)
Initiating OS detection (try #1) against newyork.itsc.austin.ibm.com
(9.3.5.22)
Retrying OS detection (try #2) against newyork.itsc.austin.ibm.com
(9.3.5.22)
Retrying OS detection (try #3) against newyork.itsc.austin.ibm.com
(9.3.5.22)
```

```

Retrying OS detection (try #4) against newyork.itsc.austin.ibm.com
(9.3.5.22)
Retrying OS detection (try #5) against newyork.itsc.austin.ibm.com
(9.3.5.22)
Host newyork.itsc.austin.ibm.com (9.3.5.22) appears to be up ... good.
Interesting ports on newyork.itsc.austin.ibm.com (9.3.5.22):
Not shown: 1689 closed ports
PORT      STATE SERVICE
22/tcp    open  ssh
135/tcp   open  msrpc
139/tcp   open  netbios-ssn
445/tcp   open  microsoft-ds
1025/tcp  open  NFS-or-IIS
3389/tcp  open  ms-term-serv
5800/tcp  open  vnc-http
5900/tcp  open  vnc
MAC Address: 00:11:25:06:F5:5D (IBM)
No exact OS matches for host (If you know what OS is running on it, see
http://insecure.org/nmap/submit/ ).
TCP/IP fingerprint:
OS: SCAN (V=4.20%D=4/24%OT=22%CT=1%CU=32480%PV=N%DS=1%G=Y%M=001125%TM=481
0C3D
OS: 4%P=i686-redhat-linux-gnu) SEQ (SP=106%GCD=1%ISR=107%TI=I%II=I%SS=S%TS
=0)S
OS: EQ (SP=106%GCD=1%ISR=106%TI=I%II=I%SS=S%TS=0) SEQ (SP=106%GCD=1%ISR=107
%TI=
OS: I%II=I%SS=S%TS=0) OPS (01=M5B4NWONNT00NNS%02=M5B4NWONNT00NNS%03=M5B4NW
ONNT
OS: 00%04=M5B4NWONNT00NNS%05=M5B4NWONNT00NNS%06=M5B4NNT00NNS) WIN (W1=4000
%W2=
OS: 4000%W3=4000%W4=4000%W5=4000%W6=4000) ECN (R=Y%DF=N%T=80%W=4000%0=M5B4
NWON
OS: NS%CC=N%Q=) T1 (R=Y%DF=N%T=80%S=0%A=S+%F=AS%RD=0%Q=) T2 (R=Y%DF=N%T=80%W
=0%S
OS: =Z%A=S%F=AR%0=%RD=0%Q=) T3 (R=Y%DF=N%T=80%W=4000%S=0%A=S+%F=AS%0=M5B4N
WONN
OS: T00NNS%RD=0%Q=) T4 (R=Y%DF=N%T=80%W=0%S=A%A=0%F=R%0=%RD=0%Q=) T5 (R=Y%DF
=N%T
OS: =80%W=0%S=Z%A=S+%F=AR%0=%RD=0%Q=) T6 (R=Y%DF=N%T=80%W=0%S=A%A=0%F=R%0=
%RD=
OS: 0%Q=) T7 (R=Y%DF=N%T=80%W=0%S=Z%A=S+%F=AR%0=%RD=0%Q=) U1 (R=Y%DF=N%T=80%
TOS=
OS: 0%IPL=B0%UN=0%RIPL=G%RID=G%RIPCK=G%RUCK=G%RUL=G%RUD=G) IE (R=Y%DFI=S%T
=80% OS: TOSI=Z%CD=Z%SI=S%DLI=S)

```

Network Distance: 1 hop
TCP Sequence Prediction: Difficulty=262 (Good luck!)
IPID Sequence Generation: Incremental

OS detection performed. Please report any incorrect results at
<http://insecure.org/nmap/submit/> .
Nmap finished: 1 IP address (1 host up) scanned in 11.224 seconds
Raw packets sent: 1898 (87.080KB) | Rcvd: 1778(84.224KB)

Running StackScan discovery

Now, we can run the StackScan discovery in our lab environment. Follow these steps:

1. Open the TADDM console.
2. The Level 1 profile includes, by default, SnmpLightSensor, but we do not want to execute it. So, we need to create a new discovery profile that is based on the Level 1 profile:
 - a. Go to **Discovery** → **Discovery Profiles** and click **New**. You see the next panel (Figure 6-8).
 - b. Enter the profile name, description, and profile that you want to clone, as shown in Figure 6-8, and click **OK**.

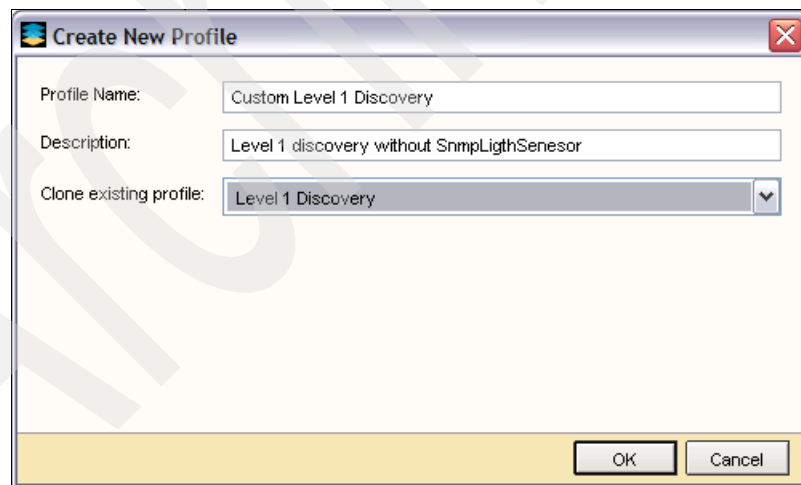


Figure 6-8 Creating a new discovery profile

- c. In the Discovery Profile list, you see the profile that you have just created. Highlight it, and remove the check mark from SnmpLightSensor (Figure 6-9 on page 224).

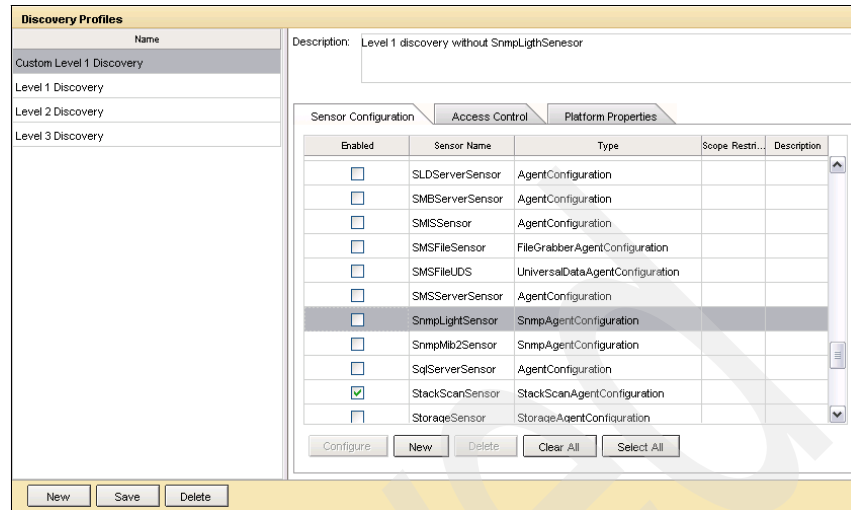


Figure 6-9 Deselecting SnmpLightSensor

- d. Finally, click **Save**.
3. Click **Discovery** → **Overview**, and click **Run Discovery**. Figure 6-10 on page 225 appears.
4. Check **ITSO Lab** under **Scopes**, and select **Custom Level 1 Discovery** from the Profile list box. Click **OK**.

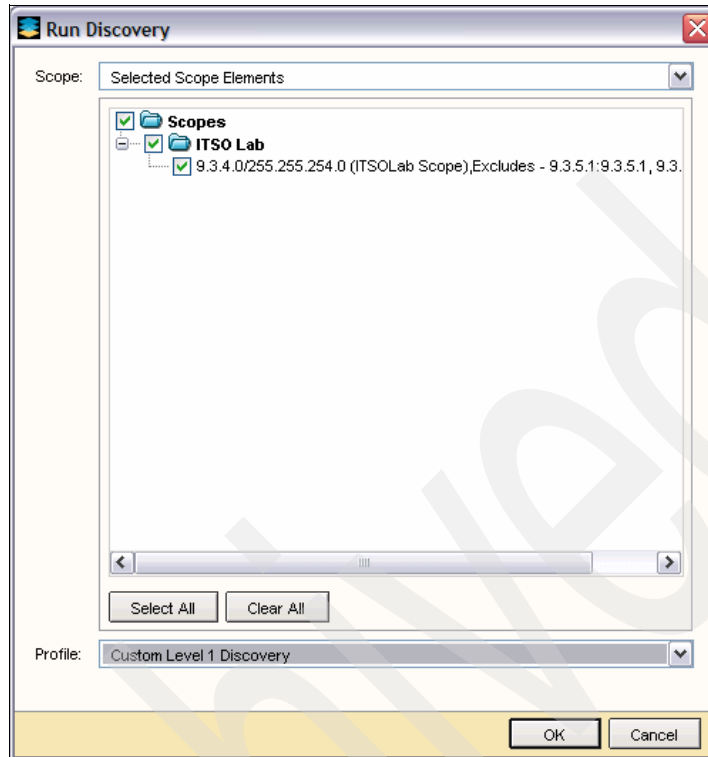


Figure 6-10 Executing StackStan discovery

5. Our discovery process runs. The amount of time that the discovery takes to complete depends on the number of machines included in our defined scope. You can follow the status by selecting **Discovery** → **Overview** (Figure 6-11 on page 226).

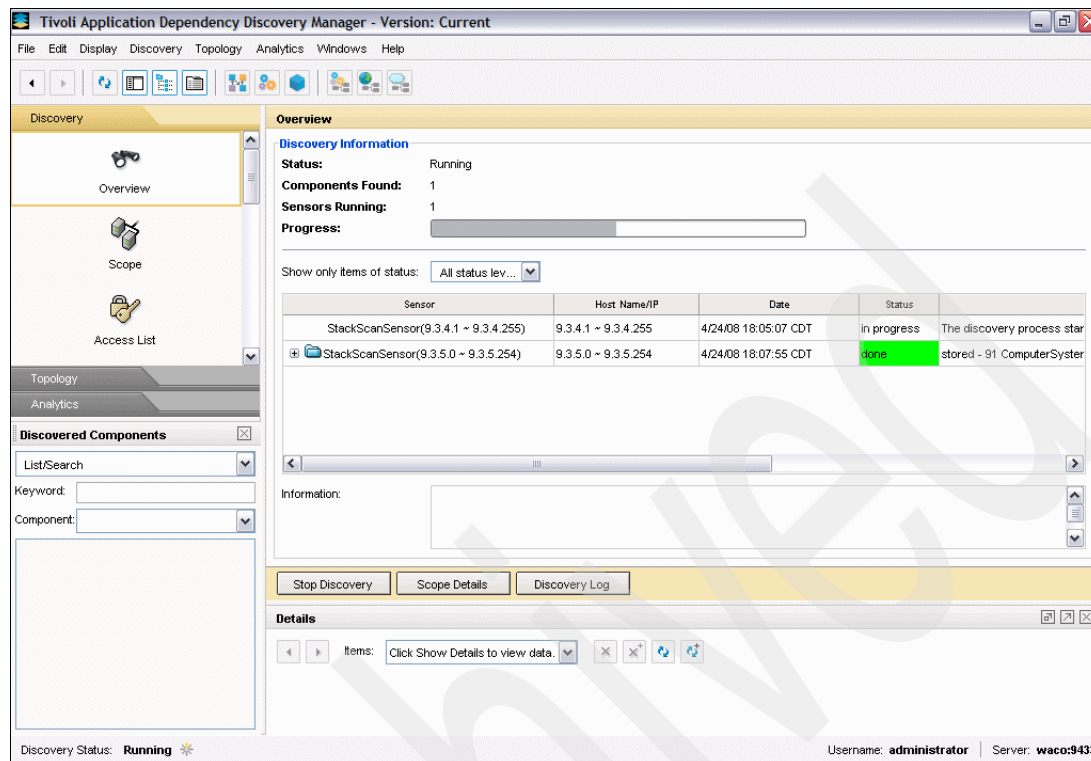


Figure 6-11 Checking the discovery status

6.1.8 Level 2 profile

You use this profile to discover detailed information about the active computer systems in the runtime environment. A Level 2 discovery discovers each computer system in your runtime environment, as well as running a detailed discovery of the operating system on each computer system. It requires that access credentials have been entered for each computer system in the runtime environment.

You can use the Level 2 profile to enable shallow discovery of applications that are running on a target system by using only the system credentials by adding the following property to the collation.properties file:

```
com.collation.internalTemplatesEnabled=true
```

If this property is set to true, you receive a CustomAppServer object representing the application running on the target machine. You do not need to provide application credentials to enable this property.

In our environment, we have already run the Level 1 StackScan. Therefore, we use the results of the Level 1 StackScan to feed into our new Level 2 scopes, which will help us to run Level 2 discoveries more quickly and effectively:

1. First, we create new scopes to help us run the Level 2 discovery. Follow these steps to create one scope set for each operating system that was discovered by the StackScan sensor:
 - a. Click **Analytics** → **Inventory**. Check all of the operating systems that were discovered (in our case, we use AIX, Linux, Sun SPARC, and Windows) by StackScan, and click **Run Report**. You can check the Systems Tier to see which Operating Systems have been discovered.
 - b. Now that you have the report's results, save these results to a comma-separated values (csv) file.
 - c. Click **Save**, choose a name for the file (for example, ScanStack), and leave csv as the file type. Click **Save**.
 - d. You have as many files as you have types of operating systems; in our case, we have four files.
 - e. Copy these csv files to the TADDM Server machine.
 - f. After the csv files are in the TADDM Server, we need to change them slightly to be able to import them as our new scopes. Use Example 6-4 to make the changes.

Example 6-4 Preparing csv files to import Level 2 Scopes

```
cat StackScan_AIX_Unitary_Computer_System.csv | tr '"' ' ' | awk
'{print $3,$1}' | grep -v "IP Name" | while read name ip; do echo
$name,, $ip >> AIXMachines.scope; done
```

- g. Repeat the same changes for all of your csv files.
 - h. Now, we have our .scope files ready to import by using the **loadscope.jy** command (Example 6-5).

Example 6-5 Listing scope files to import

```
$ ls -l *scope
-rw-r--r-- 1 cmdbadmin cmdbadmin 836 Apr 24 19:23
AIXMachines.scope
-rw-rw-r-- 1 cmdbadmin cmdbadmin 1293 Apr 24 19:24
LinuxMachines.scope
-rw-rw-r-- 1 cmdbadmin cmdbadmin 29 Apr 24 19:25
SunMachines.scope
-rw-rw-r-- 1 cmdbadmin cmdbadmin 1894 Apr 24 19:25
WindowsMachines.scope
```

- i. Change the directory to \$COLLATION_HOME/bin and use the **loadscope.jy** command to import the new scopes as shown in Example 6-6.

Example 6-6 Importing the new scopes

```
[cmdbadmin@waco bin]$ ./loadscope.jy -d -u administrator -p  
collation -s "AIX Machines" load  
/home/cmdbadmin/StackScanResultFiles/AIXMachines.scope
```

```
Processing: 9.3.5.44,,istanbul.itsc.austin.ibm.com  
Successfully parsed  
Processing: 9.3.5.45,,paris.itsc.austin.ibm.com  
Successfully parsed  
Processing: 9.3.5.111,,lpar01.itsc.austin.ibm.com  
Successfully parsed  
Processing: 9.3.5.117,,lpar07.itsc.austin.ibm.com  
Successfully parsed  
Processing: 9.3.5.196,,server3.itsc.austin.ibm.com  
Successfully parsed  
Processing: 9.3.5.197,,server4.itsc.austin.ibm.com  
Successfully parsed  
Processing: 9.3.5.208,,keyworth.itsc.austin.ibm.com  
Successfully parsed  
Processing: 9.3.5.57,,milan.itsc.austin.ibm.com  
Successfully parsed  
Processing: 9.3.5.54,,rome.itsc.austin.ibm.com  
Successfully parsed  
Processing: 9.3.5.108,,9.3.5.108  
Successfully parsed  
Processing: 9.3.5.203,,bari.itsc.austin.ibm.com  
Successfully parsed  
Processing: 9.3.5.205,,nottingham.itsc.austin.ibm.com  
Successfully parsed  
Processing: 9.3.5.143,,kramer_vio.itsc.austin.ibm.com  
Successfully parsed  
Processing: 9.3.5.173,,guadalupe.itsc.austin.ibm.com  
Successfully parsed  
Processing: 9.3.5.174,,brazos.itsc.austin.ibm.com  
Successfully parsed  
Processing: 9.3.5.175,,trinity.itsc.austin.ibm.com  
Successfully parsed  
Processing: 9.3.5.170,,nimrod.itsc.austin.ibm.com  
Successfully parsed  
Processing: 9.3.5.172,,nueces_vio.itsc.austin.ibm.com
```



```

Successfully parsed
Processing: 9.3.5.72,,madrid.itsc.austin.ibm.com
Successfully parsed
Processing: 9.3.5.73,,el Paso.itsc.austin.ibm.com
Successfully parsed
Processing: 9.3.5.127,,lpar15.itsc.austin.ibm.com
Successfully parsed
Processing: 9.3.5.33,,sydney.itsc.austin.ibm.com
Successfully parsed
DEBUG--> XML stored in /tmp/tmp3loadscope.xml

```

- j. Repeat the previous step with all of your scope files. Then, select **Discovery** → **Scope** to verify your loaded scopes. For more details, refer to “Discovery scope” on page 210.
 - k. Check your loaded scopes.
2. Now that we have created the scopes, we can create the Access Lists. For more information, refer to “Access Lists” on page 214. Follow these steps:
 - a. In our case, we create an Access List for each scope set that was created. Figure 6-12 shows the Access Lists that we created.

Access List			
Type	Name	Username	Scope Set Restriction
Sybase	default	sa	
Oracle	default	system	
Network Element (SNMP)	default		
WebLogic	default	weblogic	
Computer System	LinuxMachines	root	Linux Machines
Computer System	AIXMachines	root	AIX Machines
Computer System	SunMachines	root	Sun Machines
Computer System (Windows)	WindowsMachines	Administrator	Windows Machines

Figure 6-12 Access Lists

3. Next, select **Discovery** → **Overview**, and click **Run Discovery**. A new window appears. Check that your new scopes were created for Level 2 discovery. Then, select **Level 2 Discovery** under the Profile list box, and click **OK** (Figure 6-13 on page 230).

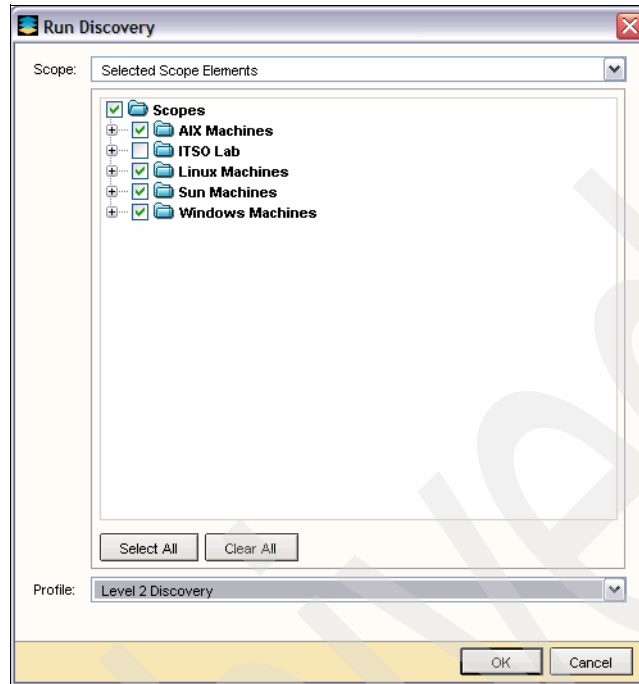


Figure 6-13 Selecting scopes for Level 2 discovery

In order to check for any errors during the Level 2 discovery, refer to Chapter 9, “Troubleshooting” on page 367.

Level 3 discovery

You can use Level 3 discovery to discover the entire application infrastructure, deployed software components, physical servers, network devices, virtual LANs, and host data that is used in a runtime environment. Just as with a Level 2 discovery process, Level 3 discovery requires that full credentials are defined for all required computer systems and applications. For example, unlike L2 discovery, you need to provide the application credentials (such as for WebSphere Application Server) in L3 discovery.

6.2 Customizing and managing discoveries

In this section, we discuss how to customize and manage discoveries.

6.2.1 Custom servers

You can create custom servers to discover and categorize servers that are not, by default, supported by TADDM. Creating custom servers is an advanced technique for configuring TADDM to discover software application servers that it does not know about by default.

Your infrastructure might contain software applications and server types, such as custom Java servers, that are not automatically categorized by TADDM. Any server process with a TCP listening port that is not recognized is categorized into an Unknown Server category. Unknown servers are not displayed in the topology and cannot take advantage of most of the functions. You do, however, receive basic information, such as the name and the runtime data of the unknown server.

You can define a custom server to create a template that sets up the membership rules for the custom server. During a discovery, any unknown server is automatically categorized as a custom server of this type if the runtime information matches the criteria that you have defined in the template. Any configuration files that are used by the custom server are also automatically captured if specified in the templates. Custom servers are displayed in the topology, and you can view details about them. Although these details are not as complete as those details provided for supported servers, defining custom servers allows all of the components in your infrastructure to participate in the topology and comparisons. You can manage custom servers by selecting **Discovery → Custom Servers**.

Several primary reasons that custom server templates are important in the use of TADDM include:

- ▶ Categorizing running software on a computer system
- ▶ Suppressing extraneous software servers from the topology
- ▶ Collecting configuration files from all computer systems of a certain type
- ▶ Populating a business application with the software that is matched by specific custom server templates

Table 6-4 on page 232 describes the information that is presented in the Custom Servers panel.

Table 6-4 Custom server information

Field	Description
Enabled	Specifies whether the custom server is to be included in the discovery. Values are true or false.
Icon	The icon associated with the custom server.
Name	The name of the custom server.
Type	The type of custom server: AppServer J2EE Server Web server Database Server
Action	The action to perform during the discovery: Discover: Include in the discovery. Ignore: Do not include in the discovery.
Config Files	The path to the configuration files with which the custom server is associated.

Identifying unknown server patterns

Before adding a server, run a basic discovery to check for unknown servers. You can run a report on unknown servers to help you identify patterns to use in the custom server template.

To identify patterns in unknown servers, click **Analytics** → **Inventory**, and the Inventory pane is displayed in the workspace. Then, select **Unknown Servers**, and click **Run Report**.

You can identify a pattern in the configuration of the unknown server, such as the program name, arguments, environment, and port. Use this pattern to create the identifying criteria for the custom server template.

In our environment, we use sendmail to create a new custom server. We highlight sendmail and click **Create Custom Server** (Figure 6-14 on page 233).

Inventory: Results					
<div> <div> <div></div> <div></div> </div> <div>Component Type: Unknown Servers</div> </div>					
db2tcpcom	0	HOME=/home/db2inst...	50000		oslo
sendmail:	accepting connections	ons	25		waco
sendmail:	accepting connections	ons	25		copenha
sendmail:	accepting connections	ons	25		ankara.it
System		COMSPEC=%SystemR...	139,139,139,445		NEWYOF
System		COMSPEC=%SystemR...	139,139,139,445		CHICAGO
System		COMSPEC=%SystemR...	139,139,445		ATHENS
System		BIM_PROLOG_DIR=/a...	139,445		SUMMIT
System		CLASSPATH=.CLUST...	139,445		LONDON
<div>Criteria:</div> <div> <div>Items per page: 331</div> <div>Go to page: 1 of 1</div> <div>refresh</div> <div><< previous</div> <div>next >></div> </div> <div> <div>Create Custom Server</div> <div>Show Details</div> </div>					

Figure 6-14 Identifying unknown server patterns

Adding custom servers

A *custom server template* contains descriptive criteria that are used to assign unknown server processes to the custom server. You specify this criteria when defining the template for the custom server.

The following information associated with running processes is parsed to match the process to a particular custom server:

- ▶ Program name: The name of the executable program
- ▶ Window Service name: The name of a window service
- ▶ Argument: The arguments passed to the program
- ▶ Environment: The environment variables set for the program
- ▶ Port: The TCP port number on which the process is listening

The custom server general information and criteria details include the name, the type of server, and the identifying criteria for the custom server. To view details about a specific unknown server, double-click the unknown server in the Topology, and click the **Runtime** tab.

You can then use this information to create search criteria for a custom server using the **General Info & Criteria** tab of the Custom Server Details window.

To add a custom server, complete the following steps from the Product Console:

1. In the Functions pane, click **Discovery** → **Custom Server**, and click **Add** from the Custom Servers pane. Use this step to create a new custom server from the beginning. If you want to create a new custom server from an unknown server, refer to “Identifying unknown server patterns” on page 232.
2. The Custom Server Details panel is displayed.
3. In the Name field, type the name of the custom server. In our lab, we used Sendmail - e-mail server.
4. From the Type list, select the type of custom server that you are adding. We selected AppServer.
5. Under Action, complete one of the following steps:
 - Click Discover if you want to discover all instances of the server.
 - Click Ignore if you want to suppress the discovery of all instances of the server.
6. To enable the custom server definition, select **Enabled**.
7. To select an icon to associate with the custom server, click **Browse** and select the icon that you want to use.
8. Under Identifying Criteria, complete one of the following steps:
 - To match all of the identifying criteria, click All Criteria.
 - To match any of the identifying criteria, click Any Criteria.
9. Define the criteria for the custom server. If you are creating a custom server from an unknown server, you will see at least one criterion that is set up. Figure 6-15 on page 235 shows our example.

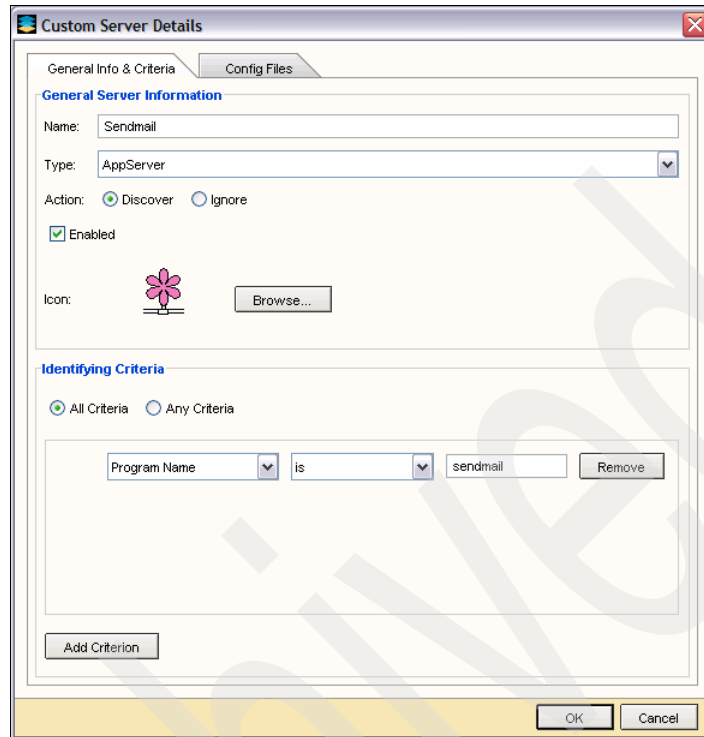


Figure 6-15 Creating a custom server

10. To add configuration files, click the **Config Files** tab. The Config Files page is displayed.
11. On the Config Files page, click **Add**. The Search Path for Capture File window is displayed.
12. From the Type list, select one of the file types to capture:
 - Config File
 - Software Module
 - Application Descriptor Directory/File
13. From the Search Path list, select one of the following search paths for the configuration file:
 - /, which is the root of the file system
 - \$PWD, which is the current working directory of the running program
 - \$Home, which is the home directory of the user ID of the running program
 - C:, which is a directory on your local computer

- %ProgramFiles%, which is the program files directory
 - %SystemRoot%, which is the system root directory
14. To capture the contents of the configuration file, click **Capture file contents**, and optionally, specify the maximum number of bytes of the captured configuration file.
 15. To recurse through the directory structure to search for the specified file, click **Recurse Directory Content**.
 16. To save the settings for your custom server, click **OK**. Refer to Figure 6-16.

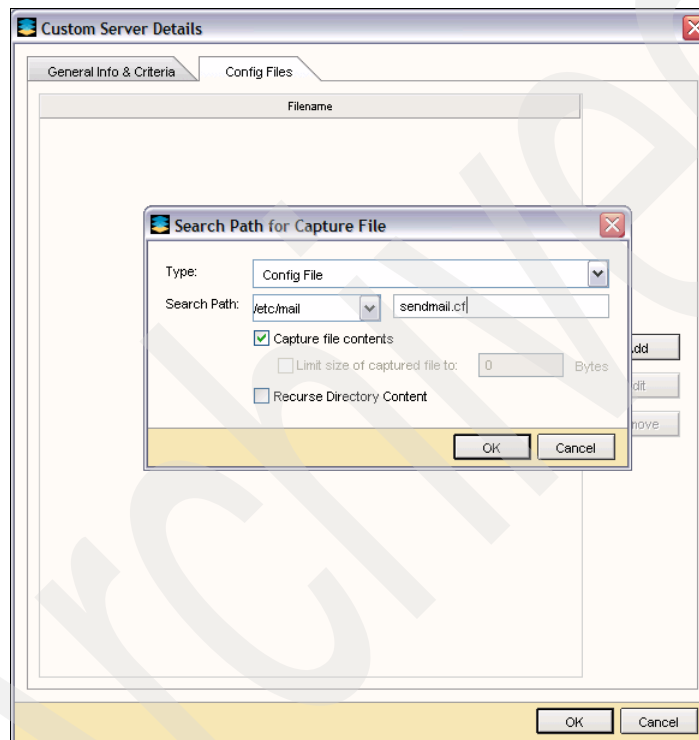


Figure 6-16 Selecting config files

Repositioning custom server entries

You can change the order in which custom servers are listed in the Custom Servers pane. The list order is important, because template matching is applied from top to bottom in the custom server list and stops at the first match. For example, a more generic template matches all servers of a specific type, and a more specific template matches only servers that have a specific string argument. After a server is matched to a server category, the custom server is removed from the unknown server list. A server cannot be a member of more

than one category at the same time, even if the server matches criteria from several custom servers in the list. Changing the order of the list can cause the server process to match to a different custom server.

To reposition entries in the Custom Servers pane, complete the following steps from the Product Console:

1. In the Functions pane, click **Discovery** → **Custom Servers**. The Custom Servers pane is displayed in the workspace.
2. In the Custom Servers pane, select the custom server that you want to reposition and complete one of the following steps:
 - a. To move the server up in the entry list, click Move Up.
 - b. To move the server down in the entry list, click Move Down.

Now, we can run a new discovery and check our new custom server that we have just created. Refer to 6.1.8, “Level 2 profile” on page 226 for more detail about how to run discoveries. After the new discovery is finished, check the results by clicking **Topology** → **Application Infrastructure** (Figure 6-17).

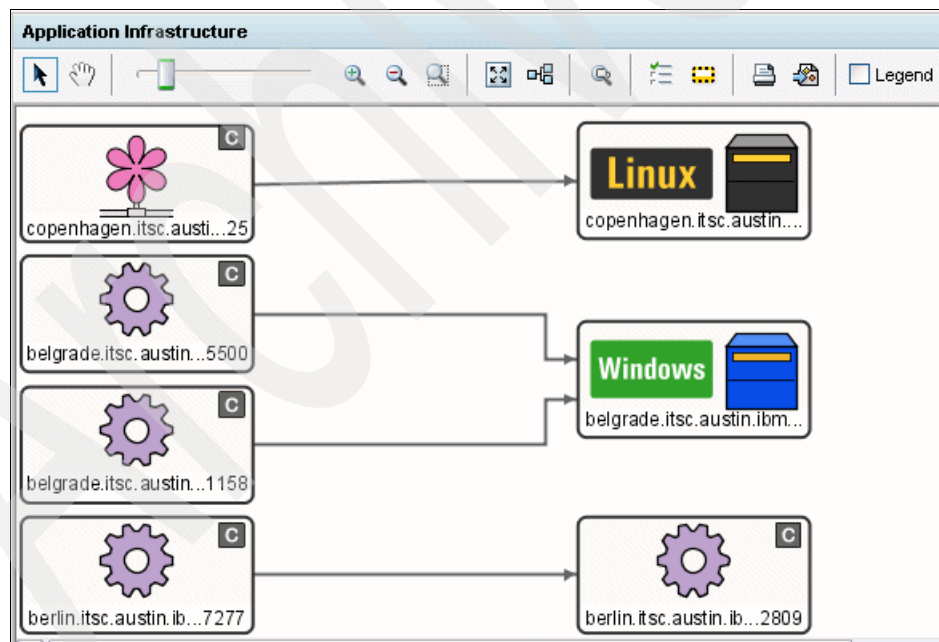


Figure 6-17 Sendmail custom server discovered

If you want to see more details, right-click in the discovered component and click **Show Details**. You can see all of the information related to the component, including the configuration file.

6.2.2 Custom server extensions

Creating a custom server template for an application also enables TADDM to subsequently display it as part of the topology. You can view details about the application, including the listening port, runtime information, and any config files or application descriptors that were collected.

In certain cases, this information might not be sufficient. For example, you might also need to access the product version. By default, TADDM is unable to collect version information for arbitrary custom server applications.

TADDM enables you to extend custom server templates to collect additional information, as required, using the following approaches:

- ▶ Execute commands on the target system to populate any attribute in the Common Data Model for the component. You can use this approach to set the `productVersion` attribute, for example.
- ▶ Execute commands on the target system and store the result as a config file for the component. One common use of this approach is to extract information from the Windows Registry.
- ▶ Execute a Jython script on the TADDM Server, which enables you to change any information about a component. The difference between this approach and the first approach is where the code executes.

After you have created the custom server, in order to extend it, you have to create a *directive file* that contains the commands to execute. You can add commands and scripts to the directive file, as required.

Use the format described in Table 6-5 on page 239 when specifying commands in the directive file.

Keep commands as simple as possible. If the command stops during execution, the sensor times out, and the component is not discovered.

The directive file must have the same name as the custom server template and must be stored in this directory: `$COLLATION_HOME/etc/templates/commands`. TADDM triggers directives in this directory by using the name of the custom server template.

Table 6-5 Directive file format

Directive	Description
<code>CMD:variable=</code> <code>path/command</code>	<p>Enables you to define an in-line command. For example:</p> <pre>CMD:productVersion=/usr/sbin/postconf awk '/^mail_version/ {print \$3}'</pre> <p>You must always specify absolute paths to commands, and you must use quotation marks (') around commands or arguments containing spaces.</p> <p>You can use environment variables associated with the process, specified by \$VARIABLE\$.</p>
<code>CMD:NOP=</code> <code>path/command</code>	<p>Enables you to execute the command without assigning results to a variable. For example:</p> <pre>CMD:NOP=reg export HKLM\Software Microsoft\InetStp c:\windows\temp\ iis.reg /y</pre>
<code>CMD:CONFCONTEN</code> <code>filename=path/command</code>	<p>Enables you to execute a command and store the results in the custom configuration file specified by file name. For example:</p> <pre>CMD:CONFCONTENT.iisREG=cmd.exe /c type c:\windows\temp\iis.reg</pre>
<code>SCRIPT: path/script</code>	<p>Enables you to invoke Jython (.py) scripts. For example:</p> <pre>SCRIPT:path/command.py</pre> <p>When the path starts with "/" TADDM assumes an absolute path; otherwise, the path is relative from \$COLLATION_HOME.</p>

Executing commands to populate the Common Data Model

You can execute commands on a target system to populate attributes in the Common Data Model. We used a command to store the sendmail version running in the ITSO lab machines. Therefore, you can use the following command to extract the version string (Example 6-7).

Example 6-7 Querying sendmail version

```
$ sendmail -d0.1 -bt < /dev/null | grep Version | awk '{print $2}'
8.13.1
$
```

To have TADDM execute the command, create a directive file that is stored in \$COLLATION_HOME/etc/templates/commands/Sendmail that contains the following line as described in Example 6-8.

Example 6-8 Directive file for Sendmail custom server

```
$ vi Sendmail
CMD:productVersion=sendmail -d0.1 -bt < /dev/null | grep Version | awk
'{print $2}'
```

When that command creates the directive file, we execute a new discovery. Populating an attribute in the Common Data Model does not make it appear in the Product Console Details pane for the component. The attribute is, however, stored in the database and can be retrieved using the TADDM SDK, which is described in Example 6-9.

Example 6-9 Querying updated Sendmail version

```
$/api.sh -u administrator -p collation find "select * from AppServer
where objectType starts-with 'Sendmail'" > Sendmail.xml
```

Finally, open the xml that was generated by the `api.sh` tool, and check the results. You see an output similar to Figure 6-18 on page 241.

?? xml	version="1.0" encoding="UTF-8"
[-] [e] results	
[a] xmlns	urn:www-collation-com:1.0
[a] xmlns:coll	urn:www-collation-com:1.0
[a] xmlns:xsi	http://www.w3.org/2001/XMLSchema-instance
[a] xsi:schemaLocation	urn:www-collation-com:1.0 urn:www-collation-com:
[-] [e] AppServer	
[a] array	1
[a] guid	F2A001E87FDA3CD49C4F2E4553BEA909
[a] lastModified	1209427118664
[a] xsi:type	coll:com.collation.platform.model.topology.app.AppServer
[e] productVersion	8.13.1
[e] status	1
+ [e] primarySAP	
[e] keyName	AppServer
+ [e] host	
+ [e] processPools	
+ [e] configContents	
[e] displayName	ankara.itsc.austin.ibm.com:25
[e] contextIp	9.3.5.55
[e] objectType	Sendmail
[e] bidiFlag	3
+ [e] AppServer	
+ [e] AppServer	

Figure 6-18 Checking updated product version

If you want to see this new field in the TADDM administration console, you need to edit the \$COLLATION_HOME/etc/detail/screencontent.xml file and add a new section as shown in Example 6-10.

Example 6-10 Editing the screencontent.xml file

```

.....
<tableContent
className="com.collation.platform.model.topology.app.AppServer"
name="AppServer.General">
  <field>
    <plain fieldName="displayName"/>
  </field>
  <!-- Lines added to see our product version -->
  <field>
    <plain displayName="Version"
fieldName="productVersion"/>
  </field>
  <!-- End -->
.....

```

After you change the screencontent.xml file, restart the TADDM Server, and then go to the Product Console to see the details of the discovered Sendmail custom server. Your custom server is similar to Figure 6-19.

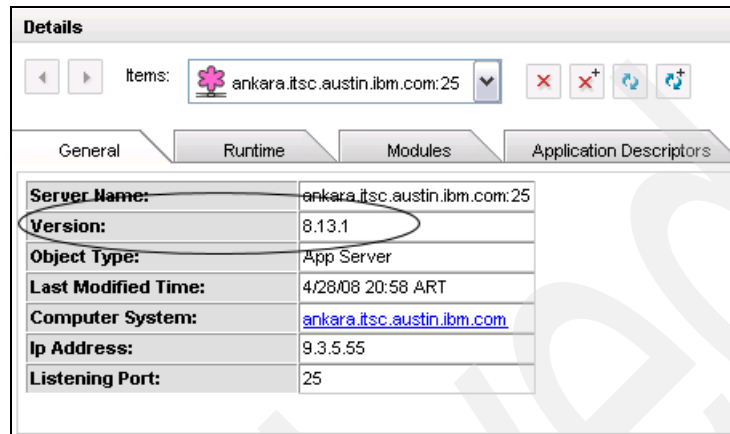


Figure 6-19 Checking updated product version using TADDM Product Console

Executing commands to create a custom configuration file

You can execute commands on a target system and store the results in a custom configuration file. Saving the results in a custom configuration file enables you to access the information using the Product Console.

In order to continue with our previous example, we added a new line in the Sendmail file that is located in \$COLLATION_HOME/etc/templates/commands (Example 6-11).

Example 6-11 Adding the command to create a configuration file

```
$ cat Sendmail
CMD:productVersion=sendmail -d0.1 -bt < /dev/null | grep Version | awk
'{print $2}'
CMD:CONFCONTENT:SendmailVersion=sendmail -d0.1 -bt < /dev/null
```

After you run a new discovery, click **Topology** → **Application Infrastructure**, right-click Sendmail custom server, and click **Show Details**. Then, under the **Config files** tab, click **Sendmail/SendmailVersion** to see the result (a new panel similar to Figure 6-20 on page 243).

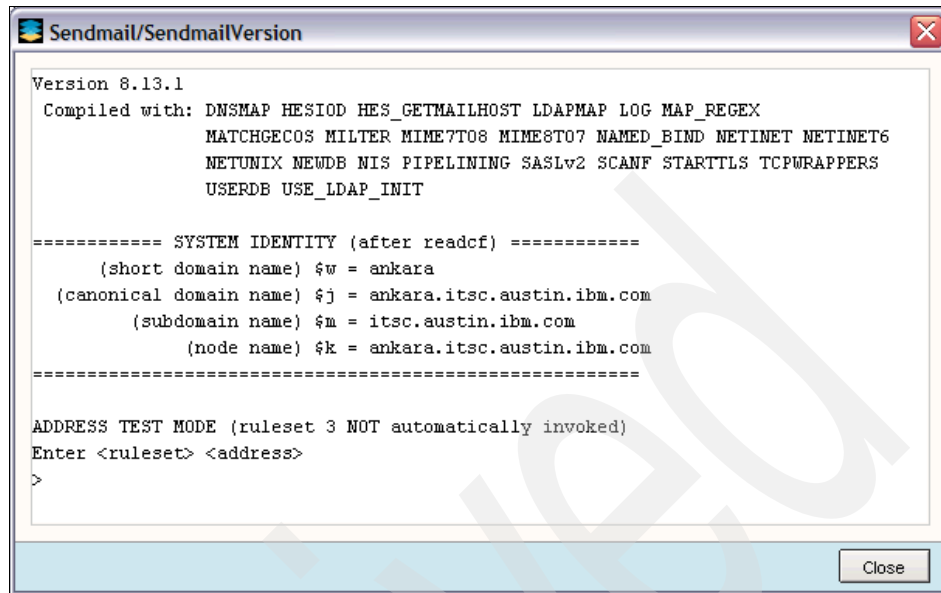


Figure 6-20 New Config File added

Executing Jython scripts

You can invoke Jython (.py) scripts on a target system. TADDM automatically detects the script language and passes the context of the custom server or computer system to the scripting language using a hashmap. This function enables Jython to manipulate Java objects.

The hashmap, also called *targets*, has predefined objects that can be used by the script for processing and for passing back results.

For custom servers, targets contain:

- ▶ Result: The CustomAppServerResult objects
- ▶ Seed: The CustomAppServerSeed object
- ▶ Environment: The environment variables hashmap
- ▶ Oobject: The oobject (which can be used to execute remote commands)

For custom systems, targets contain:

- ▶ Result, seed, and oobject as in custom servers
- ▶ System: The ComputerSystem
- ▶ Contentarray: The ComputerSystem ConfigContent

For example, you can execute the Jython script `myscript.py` by including the following command in the directive file:

```
SCRIPT:myscript.py
```

6.2.3 Computer system templates

You use computer system templates for the same purpose as custom server templates. Also, you can use computer system template extensions in the same manner that you use custom server template extensions.

In this section, we include procedures for adding and copying computer system templates. These procedures are similar to custom server procedures and are based on the same concept.

Computer system templates are an advanced technique for configuring TADDM to collect additional configuration files from specific computer system types.

Adding computer system templates

To add a computer system template, complete the following steps from the Product Console:

1. In the Functions pane, click **Discovery** → **Computer Systems**. The Computer Systems pane is displayed in the workspace.
2. In the Computer Systems pane, click **Add**. The Computer System Details notebook is displayed.
3. In the Name field, type the name of the computer system.
4. To enable the discovery of the computer system template, click **Enabled**.
5. To select an icon to associate with the computer system template, click **Browse** and select the icon that you want to use.
6. Under Identifying Criteria, select the type of operating system to associate with the computer system template.
7. To add configuration files, click the **Config Files** tab. The Config Files page is displayed.
8. On the Config Files page, click **Add**. The Search Path for Capture File window is displayed.
9. From the Type list, select one of the file types to capture:
 - Config File
 - Software Module
 - Application Descriptor Directory/File

10. From the Search Path list, select one of the following search paths for the configuration file:
 - /, which is the root of the file system.
 - \$PWD, which is the current working directory of the running program
 - \$Home, which is the home directory of the user ID of the running program
 - C:, which is a directory on your local computer
 - %ProgramFiles%, which is the program files directory
 - %SystemRoot%, which is the system root directory
11. To capture the contents of the configuration file, click **Capture file contents**, and optionally, specify the maximum number of bytes of the captured configuration file.
12. To recurse through the directory structure to search for the specified file, click **Recurse Directory Content**.
13. To save the settings for your computer system, click **OK**.

Copying a computer system template

You can create a new computer system template that is based on an existing computer system template by copying a template listed in the Computer Systems pane and issuing it a unique name.

To copy a computer system template, complete the following steps from the Product Console:

1. In the Functions pane, click **Discovery** → **Computer Systems**. The Computer Systems pane is displayed in the workspace.
2. In the Computer Systems pane, select the template that you want to copy and click **Copy**. The Set Name window is displayed.
3. In the Name field, type the name for the new template.
4. To save the new template, click **OK**.

Repositioning computer system template entries

You can change the order in which computer system templates are listed in the Computer Systems pane. The list order is important, because template matching is applied from top to bottom in the template list and stops at the first match. For example, a more generic template matches all servers of a specific type, and a more specific template matches only servers that have a specific string argument.

To reposition entries in the Computer Systems pane, complete the following steps from the Product Console:

1. In the Functions pane, click **Discovery** → **Computer Systems**. The Computer Systems pane is displayed in the workspace.
2. In the Computer Systems pane, select the template that you want to reposition and complete one of the following steps:
 - To move the template up in the entry list, click **Move Up**.
 - To move the template down in the entry list, click **Move Down**.

6.2.4 The bulkload program

The Tivoli collection of books that can load TADDM with data from other Tivoli products can be found in the IBM Tivoli Open Process Automation Library (OPAL) at:

<http://catalog.lotus.com/wps/portal/tccmd>

The bulkload program (the `loadidm1.sh` script) loads Discovery Library books into the Domain Database.

The `loadidm1.sh` script reads the books, imports the data into the Domain Database, and logs the results in the results directory for the bulkload program. In addition, the bulkload program logs error messages in the `$COLLATION_HOME/log/bulkloader.log` file.

To ensure data consistency, only one bulkload program can run at a time. The bulkload program is designed to be run at the TADDM Server. To ensure proper authorization, the bulkload program must be run by the same user ID that runs the TADDM Server processes.

All of the directories that you use to store log files and results files must exist prior to running the bulkload program. You can customize these directories by modifying the configuration settings that are defined in the `$COLLATION_HOME/etc/bulkloader.properties` file.

The following list describes the properties in the `bulkloader.properties` file and its default values. You must not change anything in the file if you want to accept the defaults:

- ▶ **com.ibm.cdb.bulk.workdir=bulk:** This directory is the directory that the bulkload program uses to copy files before loading them. See the "-c" option and `com.ibm.cdb.bulk.createworkingcopy` property. The default directory is relative to the top-level directory of the directory referenced by the `$COLLATION_HOME` environment variable. This variable is usually `/opt/IBM/cmdb/dist`.

The International Development Markup Language (IDML) file to be loaded cannot reside in this directory; otherwise, the file fails to load.

- ▶ **com.ibm.cdb.bulk.workdir.cleanup=false:** Specifies whether the working directory must be cleaned up after the load completes.
- ▶ **com.ibm.cdb.bulk.processedfiles.cleanup=30:** Number of days to keep files in the processed files list.
- ▶ **com.ibm.cdb.bulk.retrycount=5:** Number of times to retry loading a file if a discovery is currently in progress.
- ▶ **com.ibm.cdb.bulk.retrydelay=600:** Number of seconds in between retries while a discovery is in progress.
- ▶ **com.ibm.cdb.bulk.resultsdir=bulk/results:** Directory to write the results files created during the load of a file. The default directory is relative to the top-level directory referenced by the `$COLLATION_HOME` variable. This variable is usually `/opt/IBM/cmdb/dist`.
- ▶ **com.ibm.cdb.bulk.stats.enabled=false:** Whether statistics gathering of the bulkload program is performed. Turning on statistics decreases performance and increases log and results file sizes.
- ▶ **com.ibm.cdb.bulk.log.success.results=true:** Whether successfully written objects are logged into the results file. Reduced logging can improve performance by reducing output.
- ▶ **com.ibm.cdb.bulk.cachesize=200:** Number of objects to be processed in a single write operation when performing graph writing. Increasing this number improves performance at the risk of running out of memory either on the client or at the server. Only alter the number when specific information is available to indicate that processing a file with a larger cache provides benefit in performance.
- ▶ **com.ibm.cdb.bulk.createworkingcopy=true:** First, copy the IDML source file to the bulk directory and then continue to process the copied file.

Running the bulkload program

To run the bulkload program, complete the following steps:

1. Check the \$COLLATION_HOME/etc/bulkloader.properties file for accuracy. You do not need to change anything in the file if you want to accept the defaults.
2. Verify that the working directory and the results directory that are mentioned in the bulkloader.properties file are valid.

The directories must be created using the same user account that starts and stops the TADDM Server. If the bulkload program does not have permissions to read and write from the working and results directories, the bulkload program cannot run.

3. Run the bulkload program, using the following command (flag descriptions are in Table 6-6):

```
./loadidml.sh -f <path_to_idml_file> -h <host name> -u <userid>  
-p <passwd> -g -c -o -b <bidirectional format on or auto>
```

Table 6-6 Reference for the loadidml.sd command

Option	Description
-f <path_to_idml_file>	Specifies the fully qualified path to the input file or a directory that contains input XML files. The directory where the input file is placed must not be the same as the working directory of the bulkload program. If a shared directory is used to stage the input file, or, if files are copied to a local directory, the directory where input files are staged cannot be the same as the working, results, or log directory of the bulkload program. This is a requirement.
-h <host name>	Specifies the host name of the TADDM Server. This is usually not required.
-u <userid>	Specifies the user ID to be used to authenticate with the TADDM Server. This is usually not required.
-p <password>	Specifies the password used to authenticate with the TADDM Server. This is usually not required.
-o	Instructs the bulkload program to override the processed files file and load the IDML files.
-b	Specifies if bidirectional support is enabled, disabled, or automatically configured.
-g	Specifies to use the graph writing algorithm to persist data into the database.

Option	Description
-c	Specified to copy the IDML source files to the working bulk directory and process them there. This might lead to delays when copying large files.

For more information about `loadidml.sh` options, refer to the *Tivoli Application Dependency Discovery Manager User's Guide, Version 7.1* at:

http://publib.boulder.ibm.com/infocenter/tivihelp/v10r1/topic/com.ibm.taddm.doc_7.1/cmdb_user.pdf

4. After the bulkload program runs, check the results file for problems that might have occurred during the bulkload program run. The results file is located in the `resultsdir` directory that is configured in the `bulkloader.properties` file.

Look for a file with a results extension and the same name as the IDML. If, for example, the name of the imported IDML file is `test.xml`, the name of the results file is `test.results`. Important entries in this file are marked with `SUCCESS` and `FAILURE` tags. "Percentage successful" messages are also recorded if statistics are enabled. `FAILURE` tags are for individual objects and do not necessarily indicate a failure of the entire file.

5. To process the same book again after the first initial load, either use the `-o` flag or remove the specific entry from the `processedfiles.list` file. The `processedfiles.list` file is located in the working directory that is specified in the `bulkloader.properties` file.
6. If the bulkload program indicates another bulkload program is running, and you know this is not the case, go to the working directory and delete the `.block` file and run the bulkload program again. The `.block` file is a hidden file on UNIX systems, because it starts with a period. Only delete this file if you are sure that another bulkload program is not already running.

Also, you need to read the information in the `bulkload.log` file. The log file can contain details about messages that are displayed.

Best practices for using the bulkload program

There are two approaches to controlling the order in which multiple input files from a directory are loaded. One option is to load each file individually, loading the files in the correct order. This approach might be necessary if the only difference between the file names is a time stamp in the file name. A second approach is to alter the names of the files to include alphabetic ordering strings. These ordering strings are then defined to the bulkload program using the `processOrder.list` file. The `processOrder.list` file does not exist until you manually create it. The bulkload program processes files that match the first ordering string first, the second ordering string second, and so forth. If more than one file

matches the same ordering string, a processing order within that group is not ensured.

For refresh files, typically only the latest refresh file needs to be loaded. For refresh and delta files, the refresh file is typically loaded first, and then, the delta files are loaded in the sequence in which they were generated. For just delta files, they need to be loaded in the sequence in which they were generated.

A shared directory that is used for input files must be properly maintained. Loaded input files must be removed from the shared directory before they are expired from the processed list file. If a file remains in the directory after being expired from the processed list, it is reloaded, perhaps loading older data.

6.3 Reconciliation and prioritization

The *attribute prioritization feature* allows you to prioritize sources of data by defining rules that describe which data sources take priority over other data sources when updating Configuration Item (CI) attributes.

Data for CIs can be supplied to TADDM from multiple sources. The multiple sources can include a variety of sensors, the Product Console, the API, or Discovery Library Adapters (DLAs). You might trust the data from one source more than the data from another source. You might trust different pieces of information from different data sources for the same CI.

Attribute prioritization consists of two concepts: data sources and priority rules:

- ▶ *Data sources* are abstract definitions of data providers, including sensors, which write data into the TADDM Database. Data sources are the building blocks of priority rules.
- ▶ A *priority rule* is a simple, ordered list of data sources.

You must follow these rules for prioritizing your CI attributes:

- ▶ Prioritization takes place only when data is written and does not affect existing data.
- ▶ Prioritization is done between data sources loading data into a single Configuration Management Database (CMDB). Cross-domain data, such as data combined at the Enterprise Configuration Management Database (eCMDB), is not covered.
- ▶ For consistent results, you need to define the priority rules when data is not being loaded into the TADDM Database.

- ▶ You only need to define the data sources for providers of data that needs to be prioritized. A provider of CI data, which does not need to be prioritized, does not need a data source definition in order to save the data into TADDM.
- ▶ You must only define new data sources and priority rules when the system is “quiet”. Defining new priority rules while a discovery is running changes the priority rules halfway through the discovery.
- ▶ Data sources and priority rules can be defined before or after data from a particular source is loaded into the TADDM Database. Deleting a data source does not affect data in the database and does not affect the ability of a data provider to write new data to the database.
- ▶ Priority rules for data sources can be defined for the entire class or on an individual attribute for a class. You must choose between class level and attribute level prioritization, because you cannot use both class level and attribute level prioritization at the same time on a single CI class:
 - Priority rules can be changed after definition from class level to attribute level or attribute level to class level for a particular CI class.
 - Changing priority rules or priority levels alters the definitions immediately in the TADDM Database. As the new rules are applied to incoming data, there can be a latency period before the system completely reflects the changed priority definitions for a given CI class.
- ▶ Class level priority rules provide an ordered list of data sources for the entire class. The priority of the data sources is determined by their position in the list, with the first data source in the list having the highest priority, the second data source in the list having the second highest priority, and so on. A data provider not in a priority rule can still write data for a CI; however, its data can be overwritten by any data source defined in the priority rule, because it is considered to have a low priority for that class.
- ▶ Individual attribute priority rules behave the same way as class level rules, except the specific ordered list of data sources applies only for a particular attribute on the class. Each attribute in the class can have a different ordered list of data sources. The number of attributes for each class that can be prioritized using attribute level prioritization is based on the 192 character limitation and the length of the attribute name, with the actual numbering varying depending on the attributes that are selected to be prioritized. The Product Console enforces the limit and informs you if an attempt is made to prioritize too many individual attributes.
- ▶ If you change the prioritization rules from attribute to class level after data is written in the database, the information about what data source provided which attribute value is not preserved the next time the data is loaded. Instead, one of the providing data sources for the existing data (not determinate) is selected as the owner for all of the data in the CI. This data

source is used in the comparison with the incoming data source to determine if the data in the CI needs to be updated. An internal algorithm is used for the selection. This selection occurs because class level prioritization requires that all data in a CI comes from a single provider.

- ▶ If no priority rules are defined, TADDM allows the latest data coming into the system to update the existing data.
- ▶ If priority rules are added after data is in the database, the existing data is treated as having the lowest priority in the system and is overwritten by the incoming data, no matter what priority the incoming data sources have. After data is written using the new rules, prioritization applies to future updates.
- ▶ If priority rules are deleted after the data is in the database, any incoming data is allowed to update the existing data.

It is important to understand that prioritization is a processing intensive feature. Use it sparingly to prioritize only important attributes on important CIs for the most important data sources. In many cases, it might not be necessary to provide any data sources or priority rules for a CI. Particularly, if only one data source reports information on that CI. Creating rules might also be unnecessary if only a few data sources provide data for a class, and they are all equally trusted.

When data is saved into the TADDM Database, a management software system (MSS) identifier is also provided to define the identity of the data provider. The system automatically attempts to match the MSS identifier of a data provider with data source definitions defined for prioritization. If a match is made between a MSS identifier and a data source, all priority rules that contain that data source are applied to the incoming data.

The system compares the priority of the data source that provided data already in the database with the priority of the incoming data. Whenever the incoming data source has a higher priority, it is allowed to update the data in the database (either the entire class or a particular attribute). If the incoming data source has a lower priority than the data source that owns the data in the database, the incoming data is ignored, either for the class or for a particular attribute.

Prioritization can only be applied between two CIs that are recognized by the system as the same item. For TADDM, this recognition happens when attribute values supplied for naming rule attributes match. If the same CI is written to the database using different naming rules or different values for the attribute for naming rules, the system views the CI as two separate things. Prioritization is not designed to prioritize data between two CIs, and, as a result, prioritization in this use case is not applied. This use case frequently occurs with TADDM discovery sensors. The sensors write the same CI using different naming rules in many cases. To eliminate duplicates from appearing in the Product Console, TADDM

runs a cleanup process (Topology Builder agents) that removes duplicate CIs from the database. It is important to understand when a duplicate CI is deleted as opposed to two CIs being correlated and prioritization applied.

The Details panel in the Product Console displays the MSSs that provided data on a particular CI. This information can be used to guide which MSSs might need data sources in the prioritization Product Console.

Migrating from an older version of TADDM to the 7.1 release of TADDM means that all of the existing data is saved at the equivalent of the lowest priority in TADDM 7.1. In addition, previously saved data is not attached to an MSS. Before prioritization can be applied, new data must be loaded into the system for CIs so that MSS information is recorded for the CIs.

6.3.1 Manually merging discovered configuration items

Manual merging is the process where you decide to combine two or more configuration item (CI) objects that are displayed in the Product Console into one CI.

The fact that the CIs are displayed separately indicates that they do not have overlapping naming rules, and as far as TADDM is concerned, the CIs are different. If you are certain that the two CIs represent the same actual CI, you can select the CIs in the Product Console and direct the system to combine them into one CI.

When merging CIs, a single CI is selected from the list of CIs to be merged. This CI is called the *durable CI* and is retained at the end of the merge operation. The other CIs are called *transient CIs* and are deleted at the end of the merge operation.

The following rules apply to manually merging CIs:

- ▶ When CIs are merged, only the attributes of primitive types (string, integer, and so on) are transferred from the transient CI to the durable CI, and only if the durable CI does not already have a value for that attribute. Arrays and objects associated with the transient CI are not transferred.
- ▶ When a transient CI is deleted, all of its related CIs are deleted as well. For example, if a ComputerSystem CI is deleted, the operating system CI running on the computer system is also deleted, in addition to all of the software installations on the operating system.
- ▶ Merging is not currently supported for business systems or business applications.

If a CI that is designated as a transient object in a manual or automated merge is later rediscovered or reloaded through the bulkload facility, it updates (combines with) the durable object with which it was originally merged and does not result in a second instance of the CI.

However, objects that are contained by the transient object (which are called *child objects*) are treated differently. Because only a shallow merge is performed, which combines only the top level objects, the child CIs of the transient object might still not be recognized as identical objects. The result is potentially multiple instances of a child CI. If multiple child CI instances do result after a merge, or on subsequent loads of data of merged objects, the extra copies might be deleted. For example, a bulkload operation results in computer systems CS1, CS2, and operating system OS1 being stored with OS1 installed and running on CS1. If CS1 and CS2 are then merged with CS1 as the durable CI, only CS1 and OS1 will remain, but TADDM does recognize that CS2 is the same CI as CS1. If another bulkload operation results in CS2 and OS2 being added with OS2 installed and running on CS2, the existing CS1 will be updated by the information in CS2, but OS1 and OS2 might remain as separate CIs. In this case, the proper action to take is to delete both operating systems, and when the operating system is rediscovered the next time, it is created so that duplicates do not exist in the future. In very rare circumstances where the durable and the transient CI share the identical child object (not just having different representations of the same child object), the child object might be deleted as a result of the merge. Rediscovering or reloading the durable object (with its children) restores the child object.

If two CIs are mistakenly merged together attempting to rediscover or reload, the transient object results in updating the durable object and does not recreate the original transient CI. To rectify this situation, the durable CI must be deleted, and the durable and transient CIs must be rediscovered or reloaded. At this point, they again are treated as separate CIs.

Manual merging example

In the next example, we show merging two CIs manually:

1. First, we imported two AIX computer systems into the TADDM Database, using the `loadidml.sh` command. For information about using the `loadidml.sh` command, refer to 6.2.4, “The bulkload program” on page 246. You can see both IDML books in Example 6-12 and Example 6-13 on page 255.

Example 6-12 AixUnitaryComputerSystem ServerA

```
<?xml version="1.0" encoding="UTF-8"?>
<idml:idml
  xmlns:idml="http://www.ibm.com/xmlns/swg/idml"
```

```

xmlns:cdm="http://www.ibm.com/xmlns/swg/cdm"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.ibm.com/xmlns/swg/idml
idml.xsd"
>
  <idml:source>
    <cdm:process.ManagementSoftwareSystem
id="AIXSource_A.marinom.argentina.ibm.com" >

<cdm:MSSName>ibm-cdm:///CDMMSS/Hostname=marinom.argentina.ibm.com+Ma
nufacturerName=IBM+ProductName=SourceA</cdm:MS
SName>

      <cdm:ManufacturerName>IBM</cdm:ManufacturerName>
      <cdm:ProductName>SourceA</cdm:ProductName>
      <cdm:Hostname>marinom.argentina.ibm.com</cdm:Hostname>
      <cdm:ProductVersion>1.1</cdm:ProductVersion>
    </cdm:process.ManagementSoftwareSystem>
  </idml:source>
  <idml:operationSet opid="single transaction">
    <idml:create timestamp="2008-05-07T14:30:57Z">
      <cdm:CDM-ER-Specification>
        <cdm:sys.aix.AixUnitaryComputerSystem id="1" >
          <cdm:AssetID>AssestA</cdm:AssetID>
          <cdm:Signature>9.3.4.88</cdm:Signature>
          <cdm:Fqdn>ServerA.itso.ibm.com</cdm:Fqdn>
          <cdm:CPUSpeed>30000</cdm:CPUSpeed>
        </cdm:sys.aix.AixUnitaryComputerSystem>
      </cdm:CDM-ER-Specification>
    </idml:create>
  </idml:operationSet>
</idml:idml>

```

Example 6-13 AixUnitaryComputerSystem ServerB

```

<?xml version="1.0" encoding="UTF-8"?>
<idml:idml
  xmlns:idml="http://www.ibm.com/xmlns/swg/idml"
  xmlns:cdm="http://www.ibm.com/xmlns/swg/cdm"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.ibm.com/xmlns/swg/idml
idml.xsd"
>
  <idml:source>
    <cdm:process.ManagementSoftwareSystem
id="AIXSource_B.marinom.argentina.ibm.com" >

```

```

<cdm:MSSName>ibm-cdm:///CDMMSS/Hostname=marinom.ibm.com+Ma
nufacturerName=IBM+ProductName=SourceB</cdm:MS
SName>
    <cdm:ManufacturerName>IBM</cdm:ManufacturerName>
    <cdm:ProductName>SourceB</cdm:ProductName>
    <cdm:Hostname>marinom.ibm.com</cdm:Hostname>
    <cdm:ProductVersion>1.1</cdm:ProductVersion>
</cdm:process.ManagementSoftwareSystem>
</idml:source>
<idml:operationSet opid="single transaction">
    <idml:create timestamp="2008-05-07T19:16:43Z">
        <cdm:CDM-ER-Specification>
            <cdm:sys.aix.AixUnitaryComputerSystem id="1" >
                <cdm:AssetID>AssestB</cdm:AssetID>
                <cdm:Manufacturer>IBM</cdm:Manufacturer>
                <cdm:Fqdn>ServerB.itso.ibm.com</cdm:Fqdn>
                <cdm:CPUType>TypeB</cdm:CPUType>
                <cdm:CPUSpeed>3000000</cdm:CPUSpeed>
                <cdm:Model>ModelB</cdm:Model>
                <cdm:SerialNumber>BBBB</cdm:SerialNumber>
            </cdm:sys.aix.AixUnitaryComputerSystem>
        </cdm:CDM-ER-Specification>
    </idml:create>
</idml:operationSet>
</idml:idml>

```

2. Then, we checked the new CIs that were imported by using the TADDM console, as shown in Figure 6-21 on page 257.

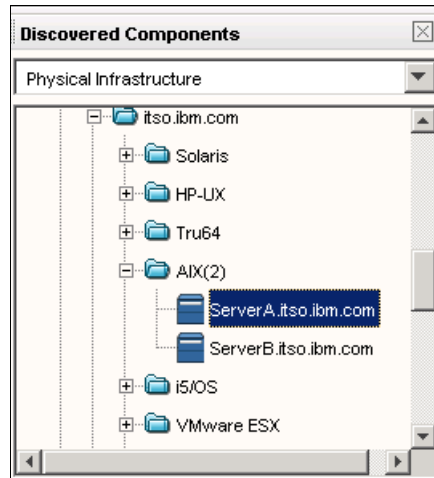


Figure 6-21 Using TADDM console to check the newly imported CIs

3. We know that ServerA and ServerB are the same machine, but TADDM recognizes them as two separate machines, because the naming rules created by the MSSs differ. If we manually delete one of these CIs, the problem is not fixed, and in the next import, the same problem occurs.
To fix this situation, we merged both machines as described in the following steps.
4. From the TADDM console, highlight ServerA and ServerB, right-click one of them, and select the option to **Merge**.

In the new panel, select the durable CI and the transient CI, and click **OK**, as shown in Figure 6-22.

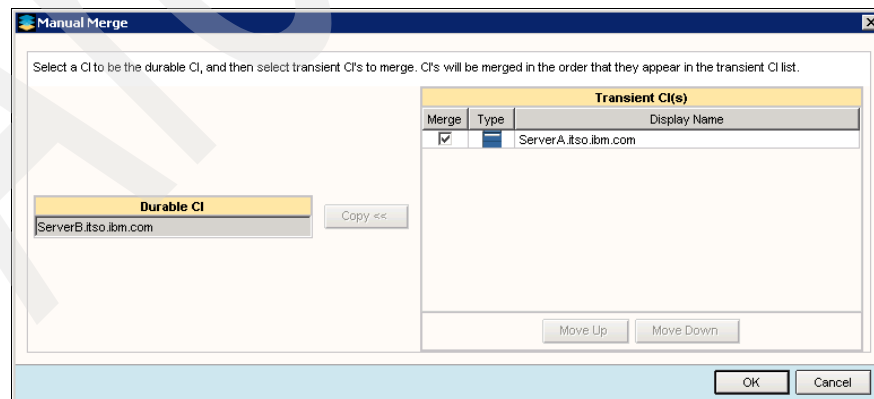


Figure 6-22 Merging ServerA and ServerB

5. Finally, you see only one CI as the result of the merger, as shown in Figure 6-23.

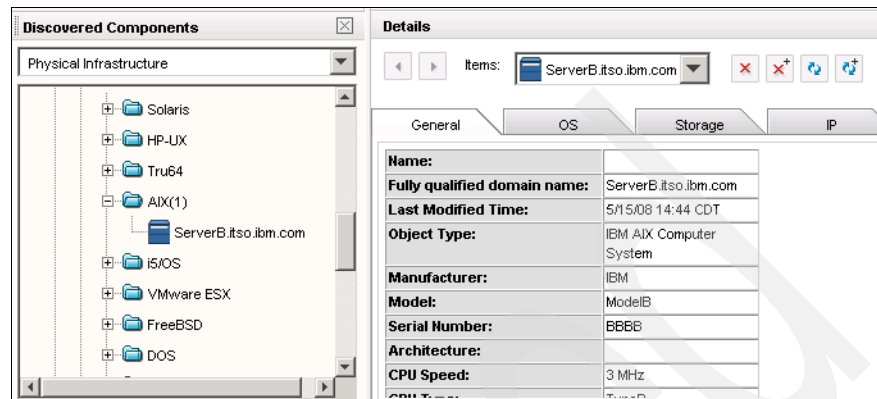


Figure 6-23 ServerB after manual merge

Note that, if you import ServerA again, the attributes will be updated without duplicating the existing CI. You can create prioritization rules in order to control the CI updates. Refer to 6.3.2, “Adding prioritization rules to your configuration items” on page 258 for more information.

6.3.2 Adding prioritization rules to your configuration items

Data sources are created by selecting **Edit** → **Prioritization Rules** from the menu bar, and then, selecting **Add** in the TADDM Data Sources List pane. You have three options when creating data sources in the Product Console:

- ▶ Create a data source for discovery
- ▶ Create a data source for topology
- ▶ Create a custom data source

In most cases, you create custom data sources for a few individual DLAs or sensors, or both, and use these custom data sources in priority rules.

Note: The product and host name fields of a custom data source must be completed carefully to ensure that the data matches the actual data source exactly, including capitalization.

Figure 6-24 on page 260 shows the Attribute Prioritization window with the various colors highlighted.

Note: You must view a Product Console with this window displayed in order to see the actual colors. Refer to the steps in the following section regarding attribute prioritization to see how to get to the Attribute Prioritization window.

The colors in the Attribute Prioritization window have the following descriptions:

- ▶ In the upper left pane, Configuration Items, the CI name is highlighted in blue if any of its attributes have a priority rule assigned to it. If a CI name is not highlighted, it means that it has no rules defined for any of its attributes.
- ▶ In the lower left pane, TADDM Data Sources List, the data source text is highlighted in green if it has an MSS associated with it. It is highlighted in blue if it does not have an MSS associated with it.
- ▶ In the upper right pane, in the Attribute Name and Source Object columns, the attribute line is highlighted according to the following schema:
 - a. The attribute line is highlighted in blue if it has priority rules associated with it.
 - b. The attribute line is highlighted in yellow if the “source object” from which it inherits the attribute has priority rules associated with it.
 - c. It is not highlighted if it does not have any priority rules assigned to it, and it does not inherit any rules.

In Figure 6-24 on page 260, the `adminState` attribute is highlighted in yellow, because the CI named `Agent` has priority rules defined for the attribute `adminState`, and the CI, `TWSAgent`, inherits its attribute `adminState` from that CI (`Agent`). The attribute `accessMethod` has no rules assigned to it and does not inherit any rules.

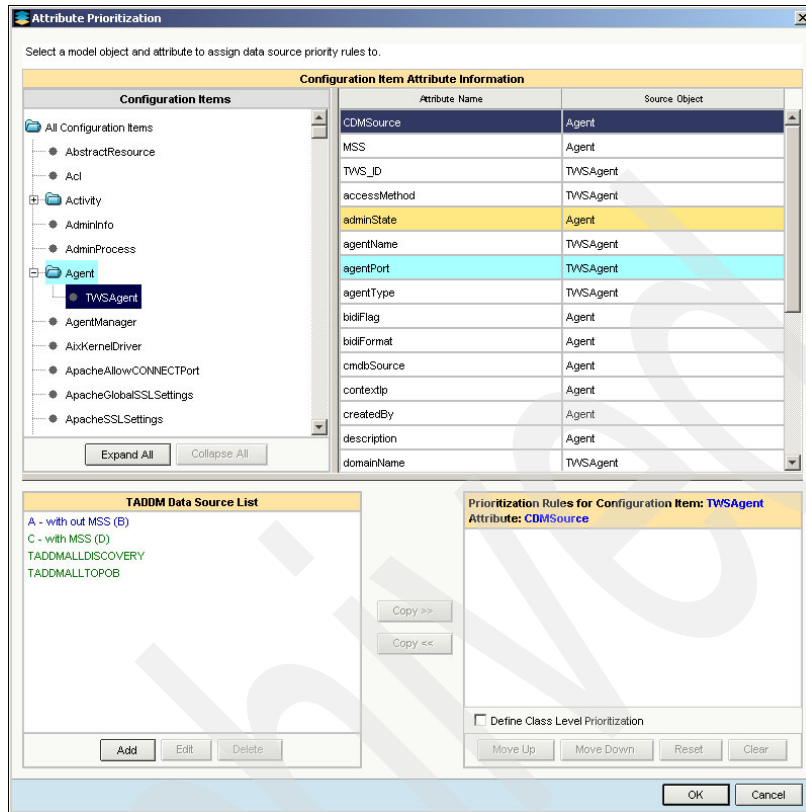


Figure 6-24 Attribute Prioritization window

To prioritize the attributes for configuration items (CIs), complete the following steps:

1. In the Product Console, select **Edit** → **Prioritization Rules**.
2. In the lower left side of the Attribute Prioritization pane, click **Add** in the TADDM Data Source List pane.
3. You can select the data source type (there are three types) in the Add Data Source pane:

- a. Create a Data Source for Discovery

This type of data source is a special data source. If you choose this selection, all input fields for the definition of a data source are filled in with appropriate default information and the values cannot be changed. Only one data source of this type can be created in the system. The name of the data source is TADDMALLDISCOVERY.

TADDM has many sensors that provide CI data. Each of these sensors is considered a source of data and can be individually prioritized using its own data source. However, there can be cases where you want to treat all sensors as one logical source of data. Creating a Data Source for Discovery, TADDMALLDISCOVERY, allows all sensors to be manipulated in a priority list as a group. Adding the TADDMALLDISCOVERY data source to the top of a priority list means that all sensors are prioritized ahead of any other data source in the list. This approach simplifies prioritization rules in cases where you want to prioritize all sensors over a Discovery Library Adapter or prioritize a Discovery Library Adapter over all sensors.

It is important to understand that the TADDMALLDISCOVERY data source only contains sensors that do not already have a specific custom data source defined for them. For example, assume that:

- TADDM only has three sensors: A, B, and C.
- A custom data source is defined for sensor A.
- The TADDMALLDISCOVERY data source will only include sensors B and C.

For this reason, TADDM does not pre-populate individual data sources for each sensor.

b. Create a Data Source for Topology

This data source is a special data source. The data source for topology is infrequently used, but it allows TADDM Topology Builder agents that create data to be prioritized if it becomes necessary. If you choose this selection, all input fields for the definition of a data source are filled in with appropriate default information, and the values cannot be changed. This data source behaves in the same way as the Data Source for Discovery described previously, except that it groups all TADDM Topology Builder agents into one data source. In most cases, a data source of this type is not needed. However, in certain circumstances, TADDM Topology Builder agents can create and update attributes on CIs. If it becomes necessary to prioritize these agents, a data source of this type can be created. As with sensors, Topology Builder agents can also have individual data sources defined. When individual data sources are defined for agents, they are not included in the grouped Data Source for Topology.

c. Create a Custom Data Source

Fill in the following fields:

- i. Product Name (required)
- ii. Host Name (required). For example: xyz.tivlab.raleigh.austin.ibm.com
- iii. Manufacturer Name (optional)

- iv. Description (optional)
- v. MSS Assignment (optional)

The Product Name and Host Name fields in the data source definition must exactly match the capitalization, spelling, and punctuation of an MSS that this data source represents.

Note: You must view a Product Console with this window displayed in order to see the actual colors. Refer to the following steps regarding attribute prioritization to determine how to get to this window.

This list includes the MSS name that is associated with each sensor. By default, all sensors have MSS names, but other MSSs can show up in this list if you have loaded data from IDML books or from the bulkload program. All MSSs that you have defined are in this list. You can prioritize which data has the highest priority (for example, if you want your WebSphere Application Server sensor information to get the highest priority, you choose the WebSphere Application Server sensor MSS in this field).

- 4. Select **OK** to save the new data source.
- 5. In the Attribute Prioritization pane, select the Model Object. The attributes that are associated with this model object appear in the adjacent pane. Select the Attribute Name to which to assign the new data source. Notice that the Model Object and Attribute Name or names that you have selected are highlighted in blue at the top of the Prioritization Rules for Configuration Item and Attribute pane.
- 6. In the TADDM Data Source List pane, select the data source that you want to apply to the model object and the attribute name that you have selected.
- 7. Click **Copy**. The data source moves to the Prioritization Rules for Configuration Item and Attribute pane.

You can select Define Class Level Prioritization in order to apply this prioritization to all of the attributes for the selected model object. The attributes turn blue when a prioritization has been assigned to them.

- 8. In the Prioritization Rules for Configuration Item and Attribute pane, if you have several data sources listed, you can use Move Up or Move Down to move the data sources to the positions that you want. The higher the position in this pane, the higher the priority for that data source.
- 9. Click **OK** to save your information.

6.4 Discovery Library Adapters

The *Discovery Library* provides an integration mechanism for communicating and sharing information about discovered resources and relationships within the enterprise.

IBM Discovery Library technology is based on the Identity Markup Language (IDML) industry standard format, which is supported in TADDM to enable users to exchange component data with other management solutions. It provides an integration mechanism for communicating and sharing information about discovered resources and relationships within the enterprise.

The Discovery Library consists of the following components:

- ▶ Discovery Library XML schema specification: This schema is called the Identity Markup Language (IDML), which defines a set of operations for creating, updating, and deleting objects in the Common Data Model (CDM).
- ▶ Discovery Library Adapter (DLA): DLAs are application code that is written to extract discovered resource and relationship data and then are transformed to the IDML specification.
- ▶ Discovery Library books: These are XML files, which are formatted according to the IDML, that contain discovery information, including the identity of resources and their relationships.
- ▶ Discovery Library File Store (DLFS): This is a repository for Discovery Library books.

The following sequence describes the Discovery Library information flow:

1. A Management Software System (MSS) discovers resources and relationships in an enterprise environment.
2. A DLA creates an IDML representation of the MSS application data (resources and relationships). The DLA can also request discovery updates, as required.
3. The DLA copies the IDML book to the DLFS enabling readers, such as the bulkload program, to access the resource and relationship information.

6.4.1 Discovery Library Adapter concepts

In this section, we describe several of the Discovery Library Adapter concepts and components, including IDML schema representation, Discovery Library Adapters, IDML books, IDML operation semantics, and Discovery Library File Store.

IDML schema representation

IDML is the Discovery Library XML schema specification. Discovery Library Adapters output files in IDML format that contain information about the Management Software System (MSS) and operation sets that define groups of operations for creating, updating, and deleting objects in the Common Data Model (CDM).

The IDML schema references the Common Data Model schema, which describes CDM model objects and relationships and their corresponding representations in XML format.

Discovery Library Adapters

A Discovery Library Adapter is a runtime component in the Discovery Library that exploits mechanisms native to Management Software Systems to extract specific details about resources and resource relationships. Discovery Library Adapters transform this information into files conforming to the IDML schema and store the resulting IDML books in the Discovery Library File Store. The purpose of Discovery Library Adapters, therefore, is to discover and keep current sets of resources and relationships that comprise business applications and support business and IT processes.

IDML books

IDML books, also known as Discovery Library books, are XML files containing details about resources and resource relationships written to conform to the IDML schema. Each IDML book represents the distinct view of the resources and relationships at a point in time. Collections of IDML books will therefore often represent overlapping views of the environment. Readers of IDML books are therefore responsible for merging these views into a consistent whole that is meaningful in the context of the application.

IDML books uniquely identify the author of the discovery data and contain timestamp information enabling applications to chronologically sequence multiple books. An IDML book can describe either delta or complete (also known as *refresh*) discoveries. Refer to “Understanding IDML operation semantics” on page 264 for more information. The Discovery Library provides an application programming interface (API) for the creation of well-formed IDML books so that Discovery Library Adapter developers can focus on the extraction and transformation of data.

Understanding IDML operation semantics

Operation sets stored in IDML books can represent the following semantics:

- ▶ *Delta*: Operations in the IDML book represent changes and updates to existing data imported during previous runs of IDML books for a particular Management Software System.

- ▶ *Refresh*: Operations in the IDML book represent a replacement of existing data imported from previous runs of IDML books for a particular Management Software System. Resources present from prior runs but not present in the refresh operation are removed. Refresh files represent a snapshot in time and replace existing information with new data.

The operation semantics apply to entire IDML books; you cannot specify both delta and refresh semantics for operation sets within the same book. By default, IDML books define delta operations for data. To specify refresh operations, the Discovery Library Adapter must include refresh as a suffix to the name of the book. Refer to 6.4.2, “File naming conventions” on page 265 for more information about naming IDML books.

Discovery Library File Store

A Discovery Library File Store (DLFS) is a repository for Discovery Library (IDML) books. A DLFS can reside on a local system or be accessible through a network connection. After a Discovery Library Adapter has written a book to a DLFS, the book must not be modified.

6.4.2 File naming conventions

IDML books are stored in plain text XML files, which must follow a consistent file naming convention. The file name includes information to uniquely identify the book within the Discovery Library File Store and to help developers and administrators quickly identify the source and creation date of the discovery data.

IDML book names consist of the following segments:

- ▶ The application code of the Management Software System (MSS)
Every Discovery Library Adapter requires an application code (10 character maximum). Include the short name of the application together with the version.
- ▶ The host name of the MSS
- ▶ An ISO 8601 time stamp UTC (Coordinated Universal Time), with colons (:) replaced by dots (.)
- ▶ The text refresh when the book contains a refresh operation
- ▶ A file name extension of .xml

The following file name example is for an IDML book that is in the Discovery Library:

AppAv1.3.host.abcxyz.com.2006-03-07T12.05.00Z.xml

The following file name example is for an IDML book that is in the Discovery Library that contains a refresh operation:

AppAv1.3.host.abcxyz.com.2006-03-07T12.05.00Z.refresh.xml

6.4.3 Integration overview

Integrating a Management Software System (MSS) with the TADDM Database consists of the following procedures:

- ▶ Mapping the MSS data to the Common Data Model (CDM)
- ▶ Creating a DLA that implements the model mapping and generates an IDML book

Mapping the MSS data to the CDM begins with collecting and analyzing the source data, as assembled by the MSS, with the intent of understanding the content and purpose of the information. After you collect and analyze the source data, you can identify corresponding model objects within the CDM and determine how to apply the CDM naming rules to create unique instances of the data.

In the process of defining model objects, you can look for additional relationships between objects to capture the maximum information about the environment. As part of model mapping, you must also verify that the MSS data is consistent with attribute conventions within the CDM.

Note: The MSS is responsible for performing discovery, monitoring resources, and capturing the data.

6.4.4 Creating a Discovery Library Adapter

In this section, we describe the procedure for creating a Discovery Library Adapter, how to develop a model map using data from the Management Software System (MSS), and how to use the DLA API to build the IDML book.

To create a DLA, complete the following steps:

1. Collect a representative sample (complete, if possible) of the type of data that is generated by the MSS.
2. Determine the type of resources and relationships supplied by the MSS.

It is important to understand the content and purpose of the data generated by the MSS. For each item, determine:

- If the item is a specific resource, a category of resources, or a relationship

- How the MSS uses the item
- How the item was discovered

You can use this information to more accurately identify the type of model objects that will represent the item.

3. Identify model objects within the Common Data Model (CDM) that correspond to entities within the MSS data

Identifying model objects is part of the process of creating a mapping between the CDM and the MSS data. For example, if the MSS contains a data item with an attribute of Windows, you can begin by deriving the following information to represent this data item:

- There is an operating system of type Microsoft Windows.
- There is a host computer system.
- There is a relationship between the operating system and the computer system.

Continue this process of identifying model objects present within the MSS data by examining the targets of explicit and implied relationships. For instance, if there is a source data item representing a computer system, determine additional characteristics (such as the IP address) of the computer system represented in the data. Document information about the MSS data and potential model objects for future reference.

4. For each model object that you identify, use the CDM naming rules to determine the attributes and relationships that are required to create an instance of a resource.

CDM naming rules define a set of attributes and relationships that provide the necessary naming context to create a unique identity for a model object. Potentially, there are multiple naming rules for each model object; each object instance must use at least one of the rules for each object type when mapping application resource data.

In certain instances, the MSS data might not include enough information to provide the unique identification of a resource. In this case, naming rules require that the resource is named not only with characteristics that are specific to it, but also with characteristics of the resource within the context of another instance.

For example, one of the naming rules for operating system type and operating system name specifies that the naming context be an instance of a computer system. Therefore, in order to create a mapping to an operating system name and operating system type, you must also define an instance of the associated computer system, along with the relationship between the computer system and the operating system.

Naming contexts are always specified in terms of other resources, relationships to other resources, or the attributes of other resources.

5. Apply CDM relationships between currently identified model objects, as appropriate.

Refer to the Uniform Modelling Language (UML) diagram for the CDM to determine potential model object relationships. Note that relationships are hierarchical, which means that relationships between model objects are automatically valid for subclasses of the model objects. For example, a runsOn relationship between an operating system and a computer system is valid for operating system subclasses. You do not need to define explicit relationships that mirror the hierarchy of the CDM.

6. Verify that the MSS data is consistent with the attribute conventions that are used to store existing information in the CDM and reconcile them as necessary.

Attribute content consistency is critical. When verifying attribute content consistency, consider the following points:

- The format of the data, including the use of dashes, dots, and other delimiters
- Whether special characters are present
- Units of measure, if appropriate
- Case sensitivity and whether the data is typically in upper, lower, or mixed case

For example, consider the case of serial numbers for computer systems. One technology might require the use of only capital letters and dashes, while another technology might consider dashes to be restricted characters. Make note of any data processing requirements that are uncovered during the model mapping stage.

7. Create a model mapping document and map the data from the MSS to the CDM.

Use the Data Model Template as a guide for creating this document. Complete the following steps to creating a model mapping document:

- a. Define a usage scenario.

You must create at least one scenario to describe the data usage in the mapping document. The scenario helps you to validate that you are gathering the necessary information from the MSS. For example, the MSS might be collecting information about operating systems, but you might also need to know about running instances of application servers.

A sample scenario might read as follows: The instance data based on classes x and y enables Application A to automate application mapping in the provisioning module.

- b. List all CDM attribute content conventions.
 - c. Specify the CDM classes (model objects) and associated attributes used.
 - d. List the relationships that are provided in the MSS data.
 - e. List the CDM classes with their associated naming policy and naming rules.
8. Use the model mapping document to define operations and operation sets for creating, modifying, and deleting model objects (managed elements).
 9. Use the DLA API to build the IDML book.

The DLA API consists of an adapter API and a book production API. You are not required to use the DLA API, but the production API offers considerable assistance in creating well-formed IDML books conforming to the IDML schema. Similarly, the adapter API offers common interfaces for command and control, and other operations, such as starting and stopping discoveries. Refer to 6.5, “Understanding the DLA APIs” on page 270 for more information.

As part of creating the IDML book, you must also assign a file name to the Discovery Library book. Refer to 6.4.2, “File naming conventions” on page 265 for complete information about Discovery Library book file naming conventions.

10. Save the book to the Discovery Library File Store by completing the following steps. You must have write and file rename permissions on the file store:
 - a. Append a .partial suffix to the name of the book when saving it to the Discovery Library.

Books copied or delivered to the Discovery Library File Store must include the .partial suffix during the copy operation, for example:

```
APPAv1.1.host.abcxyz.com.2006-03-07T12.05.00Z.xml.partial
```
 - b. After the book is completely written to the Discovery Library, remove the .partial suffix from the file name.

6.4.5 When to use a Discovery Library Adapter

Consider creating a DLA in the following cases:

- A discovery tool exists that can create a data file that contains discovered resources and relationships.

- ▶ The solution requires a loose integration with existing management technology.
- ▶ The environment demands a quick integration solution.
- ▶ There is a need to use an existing discovery scheduler.
- ▶ There is a requirement to minimize native environment interruptions.

Alternatively, consider using the TADDM API as an integration solution in the following cases:

- ▶ The environment requires real-time storage of information in the TADDM Database.
- ▶ The solution will benefit from making interactive calls to the TADDM Database to store information.
- ▶ The system requires synchronous acknowledgement that creates, updates, and deletes have completed successfully in the TADDM Database.
- ▶ There is a need to reduce the overhead and delay of processing books and data.

In general, the TADDM APIs provide more timely, synchronous, programmatic integration. DLAs provide more loosely coupled, asynchronous implementations and offer greater flexibility in many environments. Using DLAs, you can also maintain a degree of technology independence from the TADDM implementation, including the TADDM API, the programming model, and specific runtime aspects.

6.5 Understanding the DLA APIs

The Discovery Library provides the following application programming interfaces (APIs) to facilitate the integration of discovery data from a Management Software System into the Discovery Library:

- ▶ *Adapter API*: Use this API to start and stop discoveries, as well as to create transient or long running Discovery Library Adapters. Refer to 6.5.1, “Using the DLA adapter API” on page 271 for more information.
- ▶ *Book Production API*: Use this API to create well-formed IDML books that conform to the IDML schema. Refer to 6.5.5, “Using the DLA Book Production API” on page 275 for more information.

You can use the adapter and book production APIs either in conjunction or independently of each other.

6.5.1 Using the DLA adapter API

Each Management Software System (MSS) typically has its own conventions and requirements regarding configuration, deployment, control, and security. You can use the DLA adapter API to develop discovery code that can interface with an MSS and be reused in multiple runtime environments. The adapter API is implemented through the `DiscoveryLibraryAdapter` abstract class that provides methods for all supported Discovery Library Adapter functions.

To create a DLA for an MSS, complete the following steps:

1. Extend the `DiscoveryLibraryAdapter` abstract class and override the implementations for the `getCapabilities` and `getConfigParams` class scope methods and the `getState` and `stopDiscovery` methods.
2. Provide implementations for the abstract `setConfigParams` and `startDiscovery` methods. These methods are described in 6.5.2, “Managing configuration parameters and discoveries” on page 272.
3. Optionally, override the `addPropertyChangeListener` and `removePropertyChangeListener` methods.

You are not required to override these methods, because `addPropertyChangeListener` and `removePropertyChangeListener` are concrete methods in the `DiscoveryLibraryAdapter` class. Refer to 6.5.3, “Managing property change listeners” on page 273 for a description of the property change listener methods.

4. Implement the `start`, `pause`, `resume`, and `shutdown` methods for long running DLAs.

A DLA can run in either transient or long running mode. Think of a *transient DLA* as running a one-time discovery, initializing, performing the discovery operation according to the configuration properties, and, when completed, shutting down. Transient DLAs do not maintain an internal state that can be interrogated.

A long running DLA maintains an internal state over time, which you can control by using the state manipulation methods, including `start`, `pause`, `resume`, and `shutdown`. A long running DLA can perform discoveries when it is in the running state through a call to the `startDiscovery` method. We describe these methods for writing long running DLAs in 6.5.4, “Managing Discovery Library Adapter states” on page 274.

6.5.2 Managing configuration parameters and discoveries

You can use the methods described in this section to determine and set the configuration parameters required by an instance of a DLA. You can also use the methods to start and stop a discovery, and determine the state of the DLA.

Table 6-7 describes the methods for managing configuration parameters and discoveries.

Table 6-7 Configuration parameters and discovery methods

Method	Description
getCapabilities()	This is a class scope method that returns a set of properties indicating the capabilities of the DLA. These capabilities include the types of discoveries that are supported, along with whether the DLA can support transient or long running behaviors. By default, this method returns null (indicating that the capabilities of the DLA are unknown).
getConfigParams()	This class scope method returns a structure specifying the configuration parameters required by an instance of the DLA. By default, this method returns null (indicating that the configuration parameters of the Discovery Library Adapter are unknown).

Method	Description
getState()	<p>Retrieve the current state of the DLA. The state values for a DLA are:</p> <p>0 (Stopped): The DLA is not currently performing any function and will continue in this state until started by using the start() method. By default, this is the initial state of a DLA.</p> <p>1 (Starting): The DLA is in the process of starting as a result of a call to the initialize or start methods.</p> <p>2 (Running): The DLA is running. This does not imply that a discovery is in progress, only that the module is active and can perform discoveries.</p> <p>3 (Pausing): The DLA is in the process of pausing as a result of a call to the pause() method.</p> <p>4 (Paused): The DLA is paused. You can use this state with DLAs that maintain internal state variables whose values persist across calls to the pause() and resume() methods.</p> <p>5 (Resuming): The DLA is in the process of resuming as a result of a call to the resume() method.</p> <p>6 (Stopping): The DLA is in the process of stopping as a result of a call to the shutdown() method.</p> <p>7 (Recovering): A DLA that needs to be running is attempting to recover from an internal error.</p> <p>8 (Aborted): The DLA ended abnormally and is not attempting to recover.</p> <p>9 (Discovering): The DLA is currently in the process of performing a discovery.</p>
setConfigParams(configParams)	Initialize the API by providing a completed set of configuration properties that contain sufficient information for the API to connect to and extract data from its data sources.
startDiscovery()	Start a discovery using the set of parameters specified using the setConfigParams() method. If successful, the DLA returns a value identifying the discovery that has been started.
stopDiscovery(discoveryId)	Stop a previously started discovery.

6.5.3 Managing property change listeners

The methods described in this section enable you to add and remove listeners for notification when DLA properties change. The DiscoveryLibraryAdapter class defines a single property, although you can add additional properties that support notification.

Table 6-8 describes the methods for managing configuration parameters and discoveries.

Table 6-8 Property change listener methods

Methods	Description
addPropertyChangeListener (propertyName, listener)	Register an object for notification when a DLA property changes.
removePropertyChangeListener (propertyName, listener)	Unregister an object from the list of those objects to be notified when a DLA property changes.

6.5.4 Managing Discovery Library Adapter states

You can use the methods described in this section to start, pause, resume, and shut down a long running DLA.

Table 6-9 describes the methods for managing the state of DLAs.

Table 6-9 State methods

Methods	Description
pause()	Pause a long running DLA. The return value is the state of the DLA after executing the call, either 3 (Pausing) or 4 (Paused) depending on whether the DLA is capable of immediately pausing. By default, this method returns the current value of the state attribute.
resume()	Resume a long running DLA. The return value is the state of the DLA after executing the call, either 5 (Resuming) or 2 (Running) depending on whether the DLA is capable of immediately resuming from a paused state. By default, this method returns the current value of the state attribute.
shutdown()	Shut down a long running DLA. The return value is the state of the DLA after executing the call, either 6 (Stopping) or 0 (Stopped) depending on whether the DLA is capable of immediately stopping from its current state. By default, this method returns the current value of the state attribute.

Methods	Description
start()	Start a DLA that has already been initialized. The return value is the state of the DLA after executing the call, either 1 (Starting) or 2 (Running) depending on whether the DLA is capable of immediately entering a running state from a stopped state. By default, this method returns the current value of the state attribute.

6.5.5 Using the DLA Book Production API

You can use the DLA book production API to create well-formed IDML books conforming to the IDML schema.

To create IDML books, complete the following steps:

1. Create an instance of the `IdMLBook` class and initialize it by calling the `create()`, `getBookName()`, and `openBook()` methods.
Refer to 6.5.6, “Book properties and methods” on page 275 for more information.
2. Add operation sets to the book by calling the `openOperationSet()` and `closeOperationSet()` methods. Refer to 6.5.6, “Book properties and methods” on page 275 for more information.
3. Add operations to the operation sets by calling the `openCreateOperation()`, `openDeleteOperation()`, `openModifyOperation()`, and `openRefreshOperation()` methods. Refer to 6.5.6, “Book properties and methods” on page 275 for more information.
4. Within each operation, such as create or delete, add managed elements and relationships using the `addManagedElement()` and `addRelationship()` methods. Refer to 6.5.7, “Managed element properties and methods” on page 278 and 6.5.9, “Relationship properties and methods” on page 281 for more information.
5. Call the appropriate close method to complete each operation and operationSet, as required.
6. Close the book and complete production by calling the `closeBook()` method.

6.5.6 Book properties and methods

You can use the book production properties and methods to open and close IDML books, define and specify operation sets and operations, and add managed elements and relationships to operations.

Properties

Table 6-10 describes the book properties for the DLA production API.

Table 6-10 Book production properties

Property	Description
source	The <code>cdmManagementSoftwareSystem</code> instance that identifies the source of the discovery data contained in the book
timestamp	The IDML book creation time, specified using UTC

Methods

Table 6-11 describes the book production methods.

Table 6-11 Book production methods

Method	Description
<code>addManagedElement(managedElement)</code>	Append an <code>IdMLManagedElement</code> to the current operation in the current <code>operationSet</code> . It is an error to call this method if there is no current operation. The method returns a reference to the <code>IdMLBook</code> .
<code>addManagedElements(managedElements)</code>	Append a list of <code>IdMLManagedElements</code> to the current operation in the current <code>operationSet</code> . It is an error to call this method if there is no current operation. The method returns a reference to the <code>IdMLBook</code> .
<code>addRelationship(relationship)</code>	Append an <code>IdMLRelationship</code> to the current operation in the current <code>operationSet</code> . It is an error to call this method if there is no current operation. The method returns a reference to the <code>IdMLBook</code> .
<code>addRelationships(relationships)</code>	Append a list of <code>IdMLRelationships</code> to the current operation in the current <code>operationSet</code> . It is an error to call this method if there is no current operation. The method returns a reference to the <code>IdMLBook</code> .
<code>closeBook()</code>	Complete the book and close the file. It is an error to call this method unless the book has previously been opened using the <code>openBook()</code> method. It is also an error to call this method if the last call to <code>openOperationSet()</code> was not followed by a call to <code>closeOperationSet()</code> . The method returns a reference to the <code>IdMLBook</code> .

Method	Description
closeOperation()	Complete the current operation, as specified by the most recent call to the openCreateOperation(), openDeleteOperation(), or openModifyOperation() method. Following this call, there is no current operation defined until a subsequent call to open an operation. It is an error to call this method if there is no open operation. The method returns a reference to the IdMLBook.
closeOperationSet()	Complete the current operationSet. Following this call, there is no current operationSet defined until a subsequent openOperationSet call. It is an error to call this method if there is no open operationSet. The method returns a reference to the IdMLBook.
create(source, timestamp, modelSchemaURI, modelSchemaVersion)	This is a class scope method that creates an instance of an IdMLBook. The method accepts the following parameters: source: The cdmManagementSoftwareSystem instance that identifies the source of discovery data in the book. timestamp: The UTC time when the book was created. modelSchemaURI: The URI for the schema to be used to validate the instances of managed elements and relationships in the operations. modelSchemaVersion: The version.
create(source, timestamp)	This is a class scope method that creates an instance of an IdMLBook. Similar to create(source, timestamp, modelSchemaURI, modelSchemaVersion), this method uses the base Common Data Model schema for validation instead of the user-specified modelSchemaURI and modelSchemaVersion.
getBookName()	Retrieve a character string with a name that conforms to the conventions for IDML book file names, based on the type, source, and timestamp properties.
getSource()	Retrieve the value of the source property.
getTimestamp()	Retrieve the value of the timestamp property.

Method	Description
openBook(outputStream)	Retrieve a reference to the IdMLBook, which serves as the output stream to which the contents will be written. You can only call this method one time for a particular IdMLBook.
openCreateOperation(timestamp)	Begin a create operation. You can specify a timestamp value for the operation. It is an error to call this method if there is no current operationSet. The method returns a reference to the IdMLBook.
openDeleteOperation(timestamp)	Begin a delete operation. You can specify a timestamp value for the operation. It is an error to call this method if there is no current operationSet. The method returns a reference to the IdMLBook.
openModifyOperation(timestamp)	Begin a modify operation. You can specify a timestamp value for the operation. It is an error to call this method if there is no current operationSet. The method returns a reference to the IdMLBook.
openOperationSet(transactional)	Open a new operationSet. The transactional parameter specifies whether the reader of the book must treat all operations in the operationSet as one transaction. You must call the openBook() method before calling openOperationSet. It is also an error to call the openOperationSet() method if there is an existing operationSet open. The method returns a reference to the IdMLBook.
openRefreshOperation(timestamp)	Begin a refresh operation. You can specify a timestamp value for the operation. It is an error to call this method if there is no current operationSet. The method returns a reference to the IdMLBook.

6.5.7 Managed element properties and methods

You can use the properties and methods that are described in this section to create managed elements and add and retrieve associated attributes.

Properties

Table 6-12 on page 279 describes the managed element properties for the DLA book production API.

Table 6-12 Managed element properties

Property	Description
type	The class type of the managed element.
id	A string that uniquely identifies the managed element within the IdMLBook.
attributes	A list of scalar attributes of the managed element. Each item in the list must be of type IdMLAttribute.

Methods

Table 6-13 describes the managed element methods.

Table 6-13 Managed element methods

Method	Description
addAttribute(attribute)	Add an attribute to the IdMLManagedElement attribute list. The method returns a reference to the IdMLManagedElement.
addAttribute(name, value)	Add an attribute with the specified name and value to the IdMLManagedElement attribute list. The method returns a reference to the IdMLManagedElement.
create(type, id)	This is a class scope method that creates an instance of the IdMLManagedElement class using the following parameters: type: The class of the managed element. It is the responsibility of the caller to ensure that this is a defined class in the model schema of the IdMLBook to which the managed element will be written. id: Uniquely identifies the managed element within the IdMLBook. The source and target properties of the IdMLRelationship object refers to this value. The identifier does not need to be globally unique, because it is used only within the IdMLBook. While the size of the id string is not bounded, you need to use small strings to identify managed elements within an IdMLBook.
create(type, id, attributes)	This is a class scope method that accepts a list of attributes as a parameter and creates an instance of the IdMLManagedElement class. The list is copied into the IdMLManagedElement instance attribute list.

Method	Description
getAttributes()	Retrieve the list of attributes of the IdMLManagedElement.
getId()	Retrieve the ID property of the IdMLManagedElement.
getType()	Retrieve the type property of the IdMLManagedElement.

6.5.8 Attribute properties and methods

You can use the properties and methods that are described in this section to create attributes that are associated with managed elements.

Properties

Table 6-14 describes the attribute properties for the DLA production API.

Table 6-14 Attribute properties

Property	Description
name	The attribute name
value	The attribute value

Methods

Table 6-15 describes the attribute methods.

Table 6-15 Attribute methods

Method	Description
create(name, type, value)	This is a class scope method that creates an instance of IdMLAttribute using the following parameters: name: The name of the attribute. It is the responsibility of the caller to ensure that the name is valid for the IdMLManagedElement to which the IdMLAttribute is being added. type: The type of the attribute. value: The value of the attribute.
getName()	Retrieve the name property.
getValue()	Retrieve the value property.

6.5.9 Relationship properties and methods

You can use the properties and methods that are described in this section to create relationships between managed elements.

Properties

Table 6-16 describes the relationship properties for the DLA production API.

Table 6-16 Relationship properties

Property	Description
type	The relationship type.
source	The identifier of the source <code>IdMLManagedElement</code> of the relationship.
target	The identifier of the target <code>IdMLManagedElement</code> of the relationship.

Methods

Table 6-17 describes the relationship methods.

Table 6-17 Relationship methods

Methods	Description
<code>create(type, source, target)</code>	<p>This is a class scope method that creates an instance of <code>IdMLRelationship</code> using the following parameters:</p> <p>type: The type of the relationship. It is the responsibility of the caller to ensure that this value is a defined relationship type in the model schema of the <code>IdMLBook</code> to which the relationship is being written.</p> <p>source: The ID property of the <code>IdMLManagedElement</code> that is the source of the relationship. It is the responsibility of the caller to ensure that a relationship is written to the <code>IdMLBook</code> after both the source and the target <code>IdMLManagedElements</code> have been written.</p> <p>target: The ID property of the <code>IdMLManagedElement</code> that is the target of the relationship. It is the responsibility of the caller to ensure that a relationship is written to the <code>IdMLBook</code> after both the source and the target <code>IdMLManagedElements</code> have been written.</p>

Methods	Description
create(type, source, target)	This is a class scope method that creates an instance of IdMLRelationship. This method is similar to create(type, source, target) except that the source and target parameters are specified as strings instead of identifiers.
getSource()	Retrieve the property containing the source identifier.
getTarget()	Retrieve the property containing the target identifier.
getType()	Retrieve the type property.

6.6 Example of Discovery Library Adapter

In this section, we show you an example of creating an IDML Book using the DLA Book Production API, validating that the IDML Book was created, and loading it into TADDM using the `loadidml.sh` command.

Here are the steps to create, validate, and load a new IDML Book:

1. First, we collected the information that was generated by our MSS by using an Excel file called `LinuxComputerSystem.xls`, as described in Figure 6-25.

	A	B	C	D	E	F	G	H
1	AssetID	Signature	Manufacturer	FQDN	CPUType	CPU Speed	Model	SerialNumber
2	ITSO001	192.168.100.101	IBM	serv101.itso.ibm.com	CPU101	300000	MODEL101	111AAA101
3	ITSO002	192.168.100.102	IBM	serv102.itso.ibm.com	CPU102	200000	MODEL102	111AAA102
4	ITSO003	192.168.100.103	IBM	serv103.itso.ibm.com	CPU103	100000	MODEL103	111AAA103
5	ITSO004	192.168.100.104	IBM	serv104.itso.ibm.com	CPU104	300000	MODEL104	111AAA104
6	ITSO005	192.168.100.105	IBM	serv105.itso.ibm.com	CPU105	200000	MODEL105	111AAA105
7	ITSO006	192.168.100.106	IBM	serv106.itso.ibm.com	CPU106	100000	MODEL106	111AAA106
8	ITSO007	192.168.100.107	IBM	serv107.itso.ibm.com	CPU107	300000	MODEL107	111AAA107
9	ITSO008	192.168.100.108	IBM	serv108.itso.ibm.com	CPU108	200000	MODEL108	111AAA108
10	ITSO009	192.168.100.109	IBM	serv109.itso.ibm.com	CPU109	100000	MODEL109	111AAA109
11	ITSO010	192.168.100.110	IBM	serv110.itso.ibm.com	CPU110	400000	MODEL110	111AAA110
12								

Figure 6-25 MSS source `LinuxComputerSystem.xls`

2. We created a simple Java application project, which contained two classes:
 - ExcelDiscoveryLibraryAdaptor.java
 - ExcelConnection.java

We included the following libraries to be able to implement the DLA Book Production API:

- dll_core.jar
- idml_schema_2.4.jar

These files are available in \$COLLATION:_HOME/sdk/sdk.zip. You need to copy the zip file onto your machine and extract it. Then, search the libraries in `<unzip_sdk_home>/dla/dla_utility`.

The source code is shown in Example 6-14 and Example 6-15 on page 287.

Example 6-14 ExcelConnection.java

```
package DLACreate.bin;

import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.net.InetAddress;
import java.net.UnknownHostException;
import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.sql.Timestamp;
import java.util.Date;
import DLACreate.connection.ExcelConnection;
import com.ibm.dl.production.IDMLBook;
import com.ibm.dl.production.IDMLInvalidOperationException;
import com.ibm.dl.production.IDMLManagementSoftwareSystem;
import com.ibm.dl.production.utils.OutputStreamBookWriter;
import com.ibm.dl.schema.cdm.sys.linux.IDML_LinuxUnitaryComputerSystem;

public class ExcelDiscoveryLibraryAdaptor {

    private static final String OUT_FILE =
"c:\\Desarrollo\\LinuxComputerSystem.xml";

    private static final String BOOK_IDENTIFIER =
"LinuxComputerSystem.xls";
```

```

private static final String PRODUCT_NAME = "AssetsManagement";

private static final String MSS_NAME_PRE = "ibm-cdm:///CDMMSS/";

private static final String MSS_NAME_POST =
"+ManufacturerName=IBM+ProductName="
    + PRODUCT_NAME;

private static final String MSS_NAME_HOSTNAME = "Hostname=";

private static final Object VERSION = "1.1";

public static void main(String[] args) {
    InetAddress localMachine = null;
    try {
        localMachine = InetAddress.getLocalHost();
    } catch (UnknownHostException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
    if (localMachine == null || localMachine.getHostName() == null
        || localMachine.getCanonicalHostName() == null)
        return;

    IDMLManagementSoftwareSystem source = new
IDMLManagementSoftwareSystem(
        BOOK_IDENTIFIER + "." +
localMachine.getCanonicalHostName());

    String cdmName = MSS_NAME_PRE + MSS_NAME_HOSTNAME
        + localMachine.getCanonicalHostName() + MSS_NAME_POST;
    source.addAttribute("cdm:MSSName", cdmName);
    source.addAttribute("cdm:ManufacturerName", "IBM");
    source.addAttribute("cdm:ProductName", PRODUCT_NAME);
    source
        .addAttribute("cdm:Hostname", localMachine
            .getCanonicalHostName());
    source.addAttribute("cdm:ProductVersion", VERSION);

    IDMLBook idmlBook = IDMLBook.create(source, new Timestamp(new
Date()
        .getTime()));

    idmlBook.setIndent(true);
    idmlBook.setUsesRefreshOperation(true);

```



```

        String fileName = idmlBook.getBookName();
        OutputStreamBookWriter bookWriter = null;
        try {
            bookWriter = OutputStreamBookWriter.create(new
FileOutputStream(
                OUT_FILE));
        } catch (FileNotFoundException e1) {
            // TODO Auto-generated catch block
            e1.printStackTrace();
        }
        try {

            idmlBook.openBook(bookWriter);
            idmlBook.openOperationSet("single transaction");
            idmlBook.openCreateOperation(new Timestamp(System
                .currentTimeMillis()));
            addManagedElements(idmlBook);
            idmlBook.closeOperation();
            idmlBook.closeOperationSet();
            idmlBook.closeBook();

            System.out
                .println("idmlBook created: " + OUT_FILE);
        } catch (FileNotFoundException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (IDMLInvalidOperationException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }

    public static void addManagedElements(IDMLBook idmlBook) {

        Connection conn = ExcelConnection.getConnection();
        try {
            Statement st = conn.createStatement();
            String query = "Select * from [Linux$]";
            ResultSet rs = st.executeQuery(query);
            int id = 1;
            while (rs.next()) {

```

```

        IDML_LinuxUnitaryComputerSystem linuxCompSystem = new
IDML_LinuxUnitaryComputerSystem(
            String.valueOf(id));
        linuxCompSystem.addAttribute(
            IDML_LinuxUnitaryComputerSystem.ATTR_AssetID, rs
                .getString(1));
        linuxCompSystem.addAttribute(
            IDML_LinuxUnitaryComputerSystem.ATTR_Signature, rs
                .getString(2));
        linuxCompSystem.addAttribute(
            IDML_LinuxUnitaryComputerSystem.ATTR_Manufacturer,
rs
                .getString(3));
        linuxCompSystem.addAttribute(
            IDML_LinuxUnitaryComputerSystem.ATTR_Fqdn, rs
                .getString(4));
        linuxCompSystem.addAttribute(
            IDML_LinuxUnitaryComputerSystem.ATTR_CPUType, rs
                .getString(5));
        linuxCompSystem.addAttribute(
            IDML_LinuxUnitaryComputerSystem.ATTR_CPUSpeed, rs
                .getInt(6));
        linuxCompSystem.addAttribute(
            IDML_LinuxUnitaryComputerSystem.ATTR_Model, rs
                .getString(7));
        linuxCompSystem.addAttribute(
            IDML_LinuxUnitaryComputerSystem.ATTR_SerialNumber,
rs
                .getString(8));
        idmlBook.addManagedElement(linuxCompSystem);
        id = id + 1;
    }
} catch (SQLException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
} catch (IDMLInvalidOperationException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
} catch (IOException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}
}
}

```

Example 6-15 ExcelConnection.java

```
package DLACreate.connection;

import java.sql.DriverManager;
import java.sql.SQLException;

public class ExcelConnection {

    static public java.sql.Connection getConnection() {
        // Carga de Drivers
        try {
            Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
        }

        catch (ClassNotFoundException e) {
            System.out.println(e.toString());
        }
        // Realizo la conexion
        try {
            java.sql.Connection conn = DriverManager
                .getConnection(
                    "jdbc:odbc:Driver={Microsoft Excel Driver
(*.xls)}; DBQ=C:\\Desarrollo\\LinuxComputerSystem.xls",
                    "", "");
            return conn;
        } catch (SQLException e) {
            System.out.println(e.toString());
        }
        return null;
    }
}
```

3. After we created our Java application, we ran it. Our IDML Book was created, and the LinuxComputerSystem.xml file is shown in Example 6-16.

Example 6-16 LinuxComputerSystem.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<idml:idml
  xmlns:idml="http://www.ibm.com/xmlns/swg/idml"
  xmlns:cdm="http://www.ibm.com/xmlns/swg/cdm"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.ibm.com/xmlns/swg/idml idml.xsd"
>
  <idml:source>
```

```

        <cdm:process.ManagementSoftwareSystem
id="LinuxComputerDataSource.marinom.argentina.ibm.com" >

<cdm:MSSName>ibm-cdm:///CDMMSS/Hostname=marinom.argentina.ibm.com+Ma
nufacturerName=IBM+ProductName=AssetsManagement</cdm:MSSName>
    <cdm:ManufacturerName>IBM</cdm:ManufacturerName>
    <cdm:ProductName>AssetsManagement</cdm:ProductName>
    <cdm:Hostname>marinom.argentina.ibm.com</cdm:Hostname>
    <cdm:ProductVersion>1.1</cdm:ProductVersion>
</cdm:process.ManagementSoftwareSystem>
</idml:source>
<idml:operationSet opid="single transaction">
    <idml:create timestamp="2008-05-02T16:51:50Z">
        <cdm:CDM-ER-Specification>
            <cdm:sys.linux.LinuxUnitaryComputerSystem id="1" >
                <cdm:AssetID>ITS0001</cdm:AssetID>
                <cdm:Signature>192.168.100.101</cdm:Signature>
                <cdm:Manufacturer>IBM</cdm:Manufacturer>
                <cdm:Fqdn>serv101.itso.ibm.com</cdm:Fqdn>
                <cdm:CPUType>CPU101</cdm:CPUType>
                <cdm:CPUSpeed>300000</cdm:CPUSpeed>
                <cdm:Model>MODEL101</cdm:Model>
                <cdm:SerialNumber>111AAA101</cdm:SerialNumber>
            </cdm:sys.linux.LinuxUnitaryComputerSystem>
            <cdm:sys.linux.LinuxUnitaryComputerSystem id="2" >
                <cdm:AssetID>ITS0002</cdm:AssetID>
                <cdm:Signature>192.168.100.102</cdm:Signature>
                <cdm:Manufacturer>IBM</cdm:Manufacturer>
                <cdm:Fqdn>serv102.itso.ibm.com</cdm:Fqdn>
                <cdm:CPUType>CPU102</cdm:CPUType>
                <cdm:CPUSpeed>200000</cdm:CPUSpeed>
                <cdm:Model>MODEL102</cdm:Model>
                <cdm:SerialNumber>111AAA102</cdm:SerialNumber>
            </cdm:sys.linux.LinuxUnitaryComputerSystem>
            <cdm:sys.linux.LinuxUnitaryComputerSystem id="3" >
                <cdm:AssetID>ITS0003</cdm:AssetID>
                <cdm:Signature>192.168.100.103</cdm:Signature>
                <cdm:Manufacturer>IBM</cdm:Manufacturer>
                <cdm:Fqdn>serv103.itso.ibm.com</cdm:Fqdn>
                <cdm:CPUType>CPU103</cdm:CPUType>
                <cdm:CPUSpeed>100000</cdm:CPUSpeed>
                <cdm:Model>MODEL103</cdm:Model>
                <cdm:SerialNumber>111AAA103</cdm:SerialNumber>
            </cdm:sys.linux.LinuxUnitaryComputerSystem>
            <cdm:sys.linux.LinuxUnitaryComputerSystem id="4" >

```

```

    <cdm:AssetID>ITS0004</cdm:AssetID>
    <cdm:Signature>192.168.100.104</cdm:Signature>
    <cdm:Manufacturer>IBM</cdm:Manufacturer>
    <cdm:Fqdn>serv104.itso.ibm.com</cdm:Fqdn>
    <cdm:CPUType>CPU104</cdm:CPUType>
    <cdm:CPUSpeed>300000</cdm:CPUSpeed>
    <cdm:Model>MODEL104</cdm:Model>
    <cdm:SerialNumber>111AAA104</cdm:SerialNumber>
  </cdm:sys.linux.LinuxUnitaryComputerSystem>
  <cdm:sys.linux.LinuxUnitaryComputerSystem id="5" >
    <cdm:AssetID>ITS0005</cdm:AssetID>
    <cdm:Signature>192.168.100.105</cdm:Signature>
    <cdm:Manufacturer>IBM</cdm:Manufacturer>
    <cdm:Fqdn>serv105.itso.ibm.com</cdm:Fqdn>
    <cdm:CPUType>CPU105</cdm:CPUType>
    <cdm:CPUSpeed>200000</cdm:CPUSpeed>
    <cdm:Model>MODEL105</cdm:Model>
    <cdm:SerialNumber>111AAA105</cdm:SerialNumber>
  </cdm:sys.linux.LinuxUnitaryComputerSystem>
  <cdm:sys.linux.LinuxUnitaryComputerSystem id="6" >
    <cdm:AssetID>ITS0006</cdm:AssetID>
    <cdm:Signature>192.168.100.106</cdm:Signature>
    <cdm:Manufacturer>IBM</cdm:Manufacturer>
    <cdm:Fqdn>serv106.itso.ibm.com</cdm:Fqdn>
    <cdm:CPUType>CPU106</cdm:CPUType>
    <cdm:CPUSpeed>100000</cdm:CPUSpeed>
    <cdm:Model>MODEL106</cdm:Model>
    <cdm:SerialNumber>111AAA106</cdm:SerialNumber>
  </cdm:sys.linux.LinuxUnitaryComputerSystem>
  <cdm:sys.linux.LinuxUnitaryComputerSystem id="7" >
    <cdm:AssetID>ITS0007</cdm:AssetID>
    <cdm:Signature>192.168.100.107</cdm:Signature>
    <cdm:Manufacturer>IBM</cdm:Manufacturer>
    <cdm:Fqdn>serv107.itso.ibm.com</cdm:Fqdn>
    <cdm:CPUType>CPU107</cdm:CPUType>
    <cdm:CPUSpeed>300000</cdm:CPUSpeed>
    <cdm:Model>MODEL107</cdm:Model>
    <cdm:SerialNumber>111AAA107</cdm:SerialNumber>
  </cdm:sys.linux.LinuxUnitaryComputerSystem>
  <cdm:sys.linux.LinuxUnitaryComputerSystem id="8" >
    <cdm:AssetID>ITS0008</cdm:AssetID>
    <cdm:Signature>192.168.100.108</cdm:Signature>
    <cdm:Manufacturer>IBM</cdm:Manufacturer>
    <cdm:Fqdn>serv108.itso.ibm.com</cdm:Fqdn>
    <cdm:CPUType>CPU108</cdm:CPUType>

```

```

        <cdm:CPUSpeed>200000</cdm:CPUSpeed>
        <cdm:Model>MODEL108</cdm:Model>
        <cdm:SerialNumber>111AAA108</cdm:SerialNumber>
    </cdm:sys.linux.LinuxUnitaryComputerSystem>
    <cdm:sys.linux.LinuxUnitaryComputerSystem id="9" >
        <cdm:AssetID>ITS0009</cdm:AssetID>
        <cdm:Signature>192.168.100.109</cdm:Signature>
        <cdm:Manufacturer>IBM</cdm:Manufacturer>
        <cdm:Fqdn>serv109.itso.ibm.com</cdm:Fqdn>
        <cdm:CPUType>CPU109</cdm:CPUType>
        <cdm:CPUSpeed>100000</cdm:CPUSpeed>
        <cdm:Model>MODEL109</cdm:Model>
        <cdm:SerialNumber>111AAA109</cdm:SerialNumber>
    </cdm:sys.linux.LinuxUnitaryComputerSystem>
    <cdm:sys.linux.LinuxUnitaryComputerSystem id="10" >
        <cdm:AssetID>ITS0010</cdm:AssetID>
        <cdm:Signature>192.168.100.110</cdm:Signature>
        <cdm:Manufacturer>IBM</cdm:Manufacturer>
        <cdm:Fqdn>serv110.itso.ibm.com</cdm:Fqdn>
        <cdm:CPUType>CPU110</cdm:CPUType>
        <cdm:CPUSpeed>400000</cdm:CPUSpeed>
        <cdm:Model>MODEL110</cdm:Model>
        <cdm:SerialNumber>111AAA110</cdm:SerialNumber>
    </cdm:sys.linux.LinuxUnitaryComputerSystem>
</cdm:CDM-ER-Specification>
</idml:create>
</idml:operationSet>
</idml:idml>

```

4. After the IDML Book is created, we validated it. In order to validate the IDML Book, use the idmlcert.jar, which is in <unzip_sdk_home>dla\validator\v2. Follow these steps (Example 6-17).

Example 6-17 Validating the IDML Book

```

C:\Desarrollo\TADDM\sdk\dla\validator\v2>java -jar idmlcert.jar
c:\Desarrollo\LinuxComputerSystem.xml
IBM Discovery Library Certification Tool
Version 2.4.2

```

```

=====
File: c:\Desarrollo\LinuxComputerSystem.xml
=====

```

```

Certification tool found:
    11 Managed elements

```

0 Relationships

[PASS] - TEST 00 (XML Parse)
[PASS] - TEST 01 (All MEs have a valid ID)
[PASS] - TEST 02 (superior reference IDs in book)
[PASS] - TEST 03 (Attributes are valid)
[PASS] - TEST 04 (All managed elements have a valid naming rule)
[PASS] - TEST 05 (All managed elements are valid)
[PASS] - TEST 06 (All relationships are valid)

Book passed all certification tests
Elapsed time: 3,2 seconds

5. When the IDML Book has been created and validated, we loaded it into TADDM using the **loadidml.sh** command. For more information about how to use the **loadidml.sh** command, refer to 6.2.4, “The bulkload program” on page 246. Example 6-18 shows how we loaded the IDML Book.

Example 6-18 Loading the IDML Book

```
$ ./loadidml.sh -u administrator -p collation -f  
/home/cmdbadmin/LinuxComputerSystem.xml  
Bulk Load Program starting.  
Bulk Load Program running.  
Bulk Load Program running.  
Bulk Load Program running.  
Bulk Load Program succeeded. Return code is: 0  
0  
Bulk Load Program ending.
```

6. Finally, check whether these machines are already loaded into TADDM. There are several methods to check whether these machines are already loaded into TADDM. One method, using the TADDM console, is shown in Figure 6-26 on page 292 and Figure 6-27 on page 292.

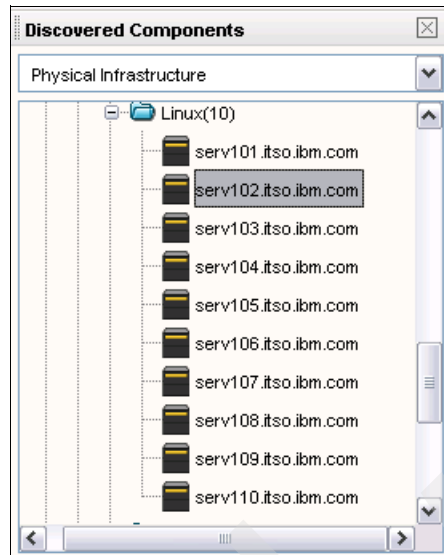


Figure 6-26 Viewing loaded Linux computer systems from the IDML Book

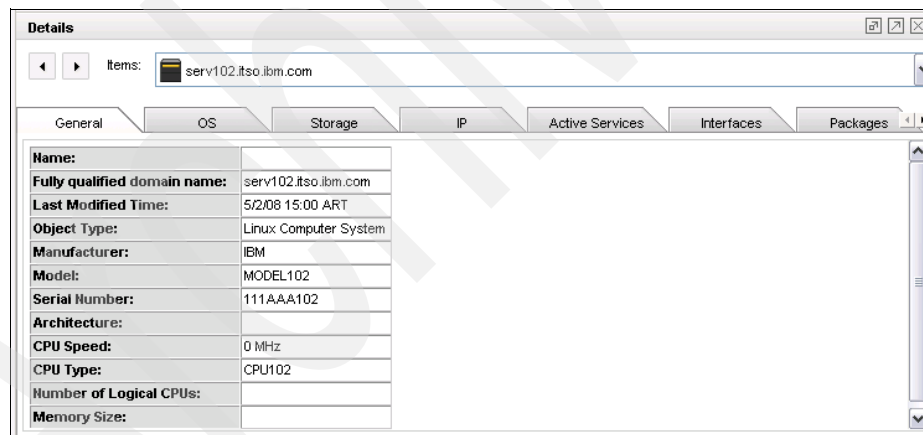


Figure 6-27 Viewing loaded Linux computer system detail from the IDML Book

Reporting scenarios

In this chapter, we present several reporting scenarios that focus on how to use an external reporting engine, such as Business Intelligence Reporting Tool (BIRT) to generate various custom reports.

We introduce BIRT and describe how to deploy it on the IBM Tivoli Application Dependency Discovery Manager (TADDM) server. We then describe scenarios that require particular reports and show you how to create those reports.

We describe multiple ways to access the TADDM configuration management database (CMDB) data from BIRT and explain the advantages and disadvantages of each method.

In this chapter, we include:

- ▶ “Introducing BIRT” on page 294
- ▶ “Deploying BIRT Report Viewer on TADDM” on page 294
- ▶ “Designing TADDM Reports with BIRT” on page 296
- ▶ “Disaster recovery and validation” on page 334
- ▶ “Root cause analysis with tracking changes” on page 338

7.1 Introducing BIRT

BIRT is an Eclipse-based open source reporting system for Java-based stand-alone and Web applications.

BIRT has two major components:

- ▶ A Report Designer that is based on Eclipse, which is used to design and format reports
- ▶ A runtime component that you can add to your application server or stand-alone application as a library, which is used to render report designs into actual reports in HTML or PDF format

BIRT also offers a charting engine that lets you add charts to your own application.

This chapter assumes a basic knowledge of BIRT. We use the BIRT 2.2.x release. You can also use this procedure with 2.1.x releases.

Further discussion about BIRT is beyond the scope of this chapter. For more information about BIRT, refer to the BIRT Web site, for proper introduction, at:

<http://www.eclipse.org/birt/>

7.2 Deploying BIRT Report Viewer on TADDM

The Viewer is used to render the report design. The BIRT Report Viewer WAR file is released as part of the Report Engine Runtime. To deploy the Viewer on TADDM Server, follow these steps:

1. Using your browser, navigate to the following URL:
<http://download.eclipse.org/birt/downloads/>
2. Click **Runtime**, and then, select the download mirror that is closest to you.
3. Download birt-runtime-2_2_2.zip to your system.
4. Extract the archive file on your local machine.
5. Copy birt.war from the extracted folder to the TADDM Server at `<collation_home>/dist/deploy_tomcat`.

Tomcat will automatically deploy birt.war, and you will see a directory called birt that is created under deploy_tomcat.

6. Using your browser, navigate to the following URL:

http://<TADDM_Host>:9430/birt/

If you see the panel that is shown in Figure 7-1, it means that you have successfully deployed birt on the TADDM Tomcat server.

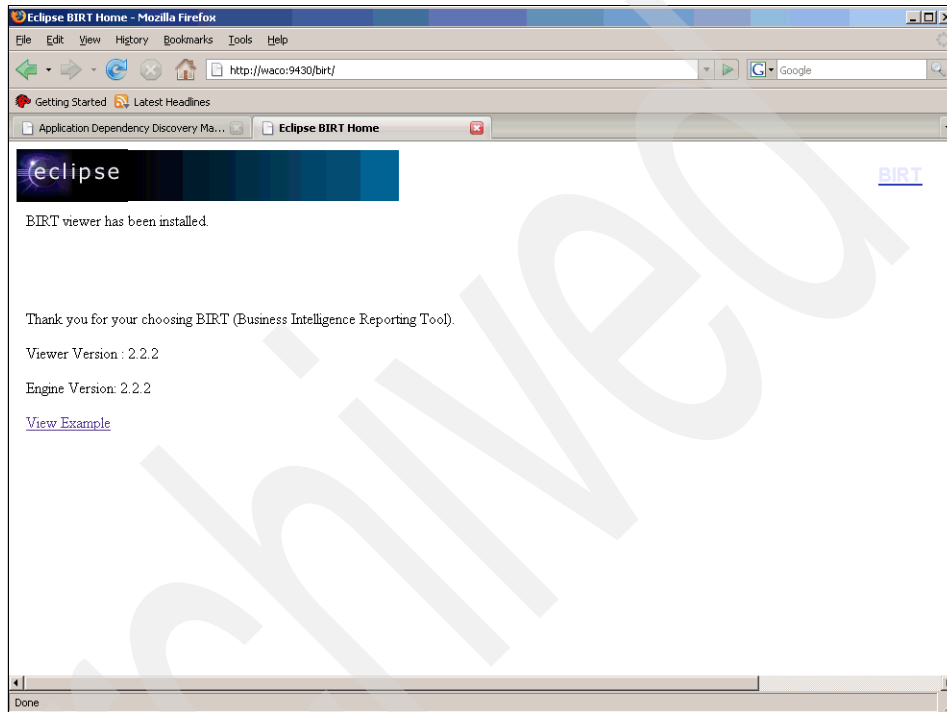


Figure 7-1 Introduction panel for BIRT Report Viewer

7. Click **View Example** to see a test report, such as the report in Figure 7-2 on page 296. Notice that you can use the buttons at the top left corner of the report to print the report or export it to PDF, PPT, Word, or Excel.

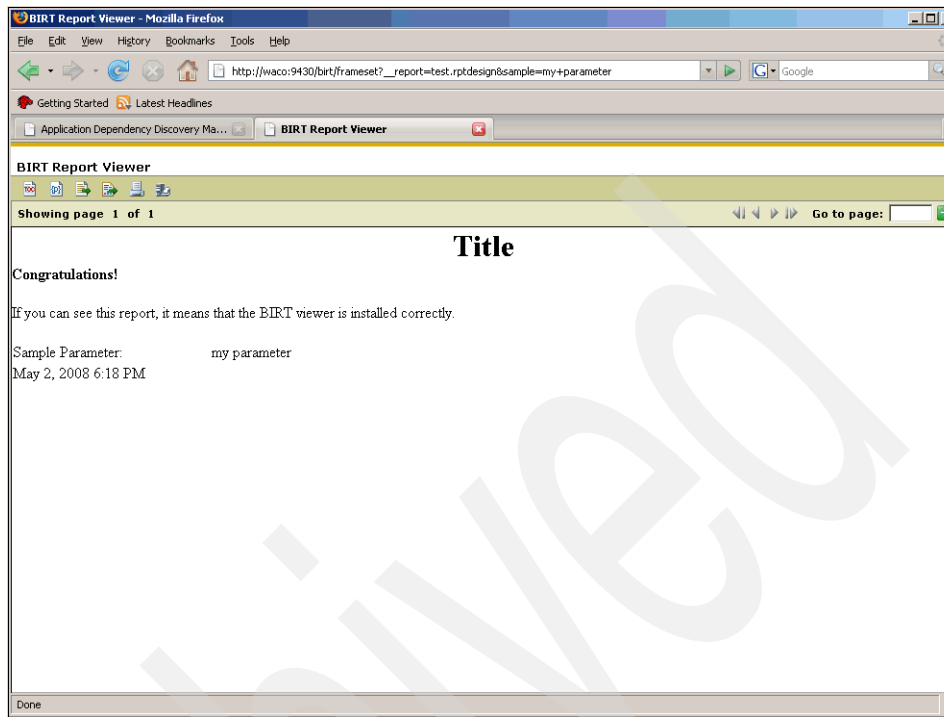


Figure 7-2 A test report after it was rendered as HTML

Now, you are ready to deploy your own TADDM BIRT reports.

7.3 Designing TADDM Reports with BIRT

XYZ is a Service Provider company that was recently acquired by GobaCo. The management of GlobalCo wants to generate an inventory report of all XYZ IT assets.

In this scenario, GlobalCo can use TADDM Level 1 discovery to quickly scan the XYZ infrastructure to produce an inventory report about the number of computer systems that the company has, as well as the types of operating systems that the company runs. Level 1 is a credential-less discovery method that can be used to discover active computer systems in the environment.

This report will help GlobalCo quickly and fairly accurately estimate the value of the XYZ IT assets while knowing very little about XYZ IT infrastructure.

TADDM can show the result using the built-in Inventory report in the analytics section. In this scenario, we show you how to use BIRT to show those same reports in a graphical, more appealing view using a pie chart format.

7.3.1 Designing reports with scripted data source

BIRT supports accessing a data source using Javascript code. We refer to the Javascript code type of data source as a *scripted data source*. Scripted data sources are used to access data from sources other than an SQL, XML, or text file. The Javascript code implements the data source by wrapping Java objects, such as EJB, XML Stream, or any other Java object that retrieves data.

In the case of TADDM, we create the scripted data source that wraps the Configuration Management Database (CMDB) application programming interfaces (APIs). The data must be returned in tabular format so that BIRT can perform various operations, such as sorting and grouping. CMDB APIs do not return the data in that format. So, we need to create an intermediary Java object that drives the CMDB APIs and convert the returned data to tabular format. The Javascript code only needs to work with this Java object.

In order to develop BIRT reports for TADDM using scripted data source, we need the BIRT Report Designer, as well as the Eclipse development platform. We can get both the BIRT Report Designer, as well as the Eclipse development platform, by downloading and installing the BIRT All-in-One package.

Setting up the BIRT 2.2.x All-in-One package

Complete the following procedure to download and install the BIRT 2.2.x All-in-One package. Replace *x* with the latest release number, which was BIRT 2.2.2 at the time of our writing this book:

1. Using your browser, navigate to the following URL:
<http://download.eclipse.org/birt/downloads/>
2. Click **All-in-One**, and then, select the download mirror that is closest to you.
3. Download `birt-report-designer-all-in-one-2_2_x.zip` to your system.
4. Extract the archive file on your local machine.
5. Go to the extracted folder and run the `eclipse.exe` application.

Creating your BIRT project

1. In Eclipse, click **File** → **New** → **Project**.
2. In the New Project window (Figure 7-3 on page 298), expand **Business Intelligence and Reporting Tools** → **Report Project**. Click **Next**.

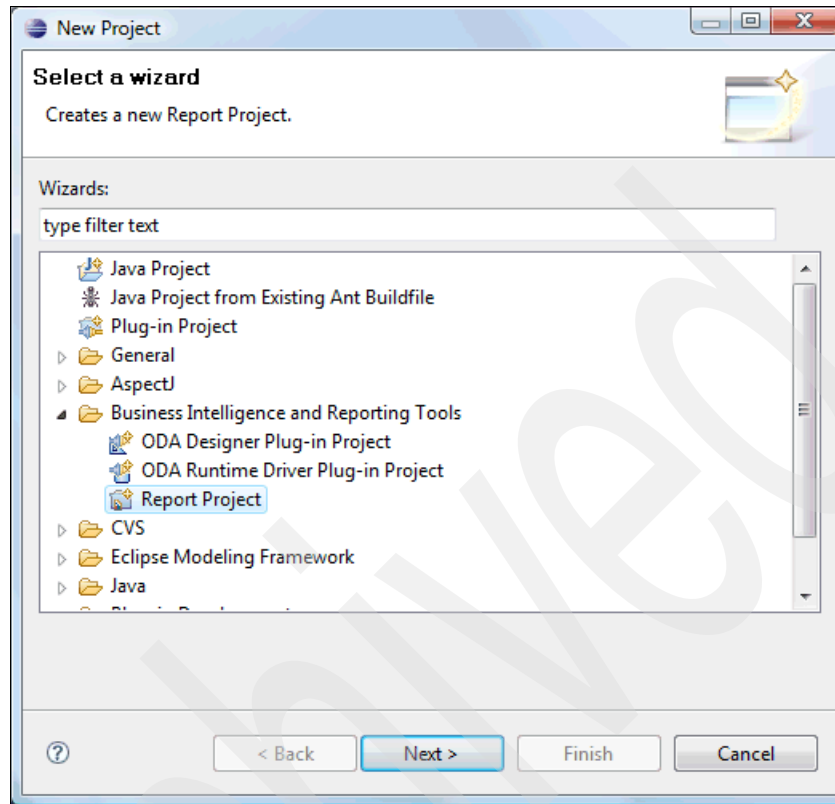


Figure 7-3 Creating new Report Project

3. Enter TADDM Reports, for example, in the Project Name field (Figure 7-4 on page 299). Click **Finish**.

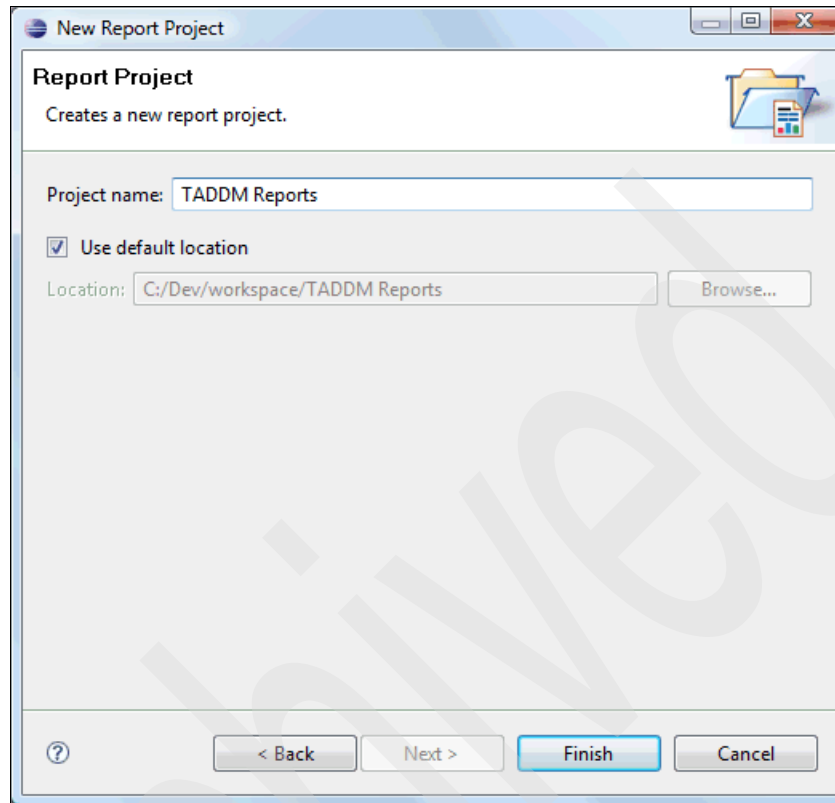


Figure 7-4 Adding the project name

4. Right-click the project name under the Navigator panel, and then select **New** → **Report** (Figure 7-5 on page 300).

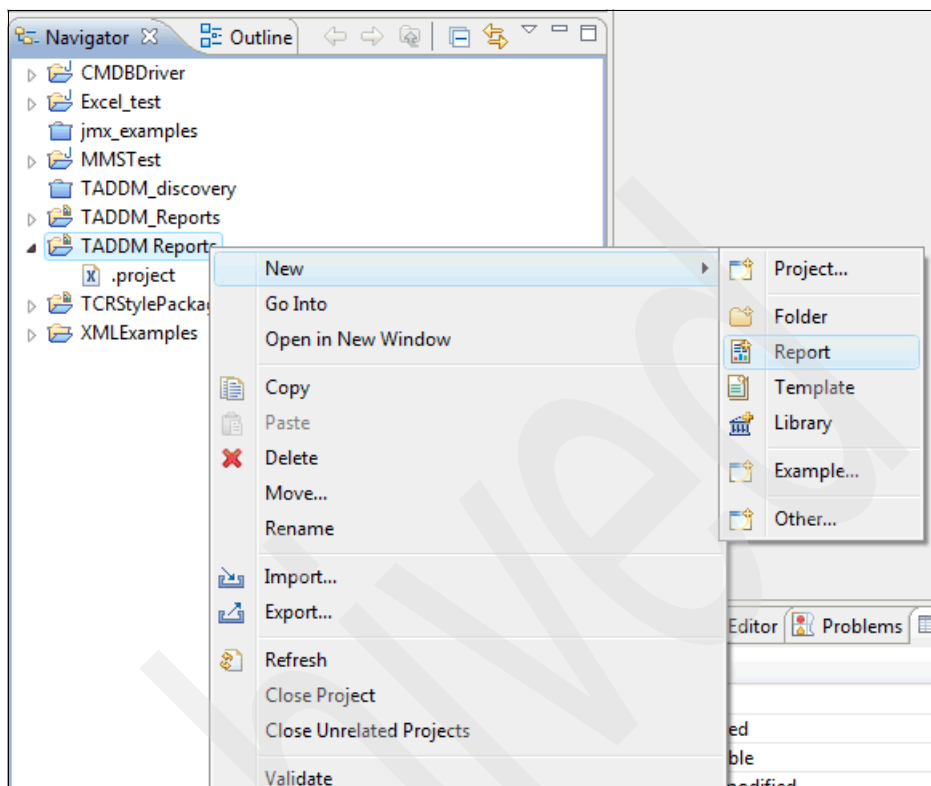


Figure 7-5 Creating a new report

5. Enter the name of the report as shown in Figure 7-6 on page 301. Then, click **Finish**.

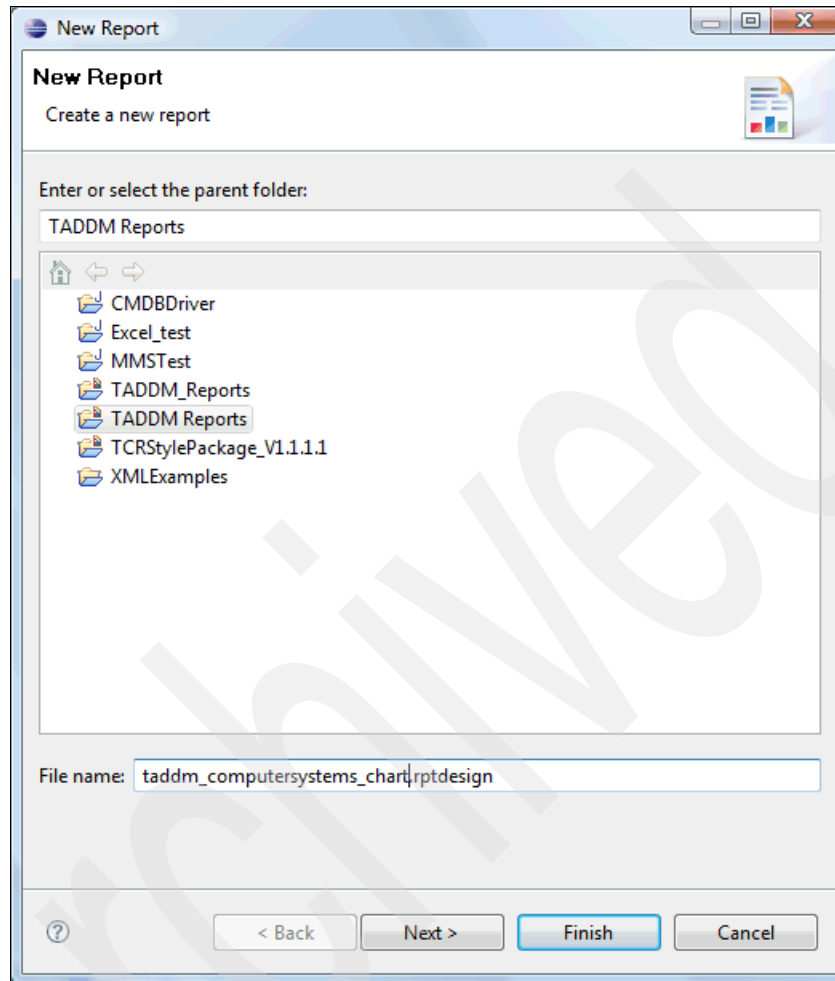


Figure 7-6 New report wizard

Creating and implementing a scripted data source

To create and implement a scripted data source:

1. In the Data Explorer panel, right-click **Data Sources** and choose **New Data Source**. The “Select a Data Source Type” dialog appears.
2. Select **Scripted Data Source** from the drop-down list.
3. In Data Source Name, type `CMDB_API`.
4. Choose **Finish**.

5. In Data Explorer panel, right-click **Data Sets**. Choose **New Data Set**. The New Data Set dialog appears, as shown in Figure 7-7.

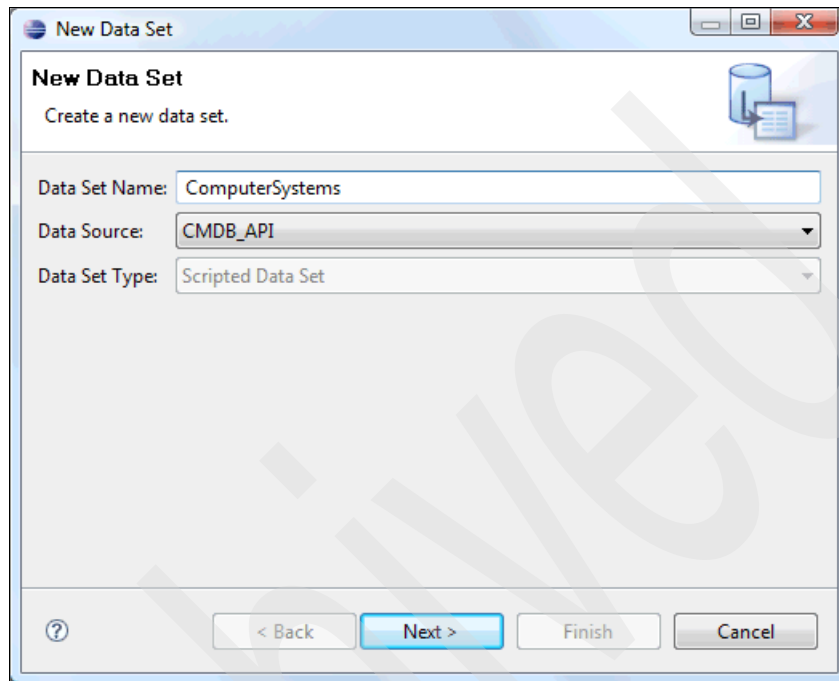
The image shows a 'New Data Set' dialog box with a title bar containing a question mark icon and the text 'New Data Set'. Inside the dialog, there is a subtitle 'Create a new data set.' and an icon of a database cylinder and a document. Below this, there are three input fields: 'Data Set Name' with the text 'ComputerSystems', 'Data Source' with a dropdown menu showing 'CMDB_API', and 'Data Set Type' with a dropdown menu showing 'Scripted Data Set'. At the bottom of the dialog, there is a row of four buttons: a help button (question mark icon), '< Back', 'Next >' (highlighted in blue), 'Finish', and 'Cancel'.

Figure 7-7 New data set for a scripted data source

6. In the Data Set Name field, type ComputerSystems.
7. Click **Next**. Type OSRunning for the first output column name.
8. Choose the Type **String**.
9. Type NumberOfMachines for the second output column name.
10. Choose the Type as **Integer**. Your column definitions appear in Figure 7-8 on page 303.

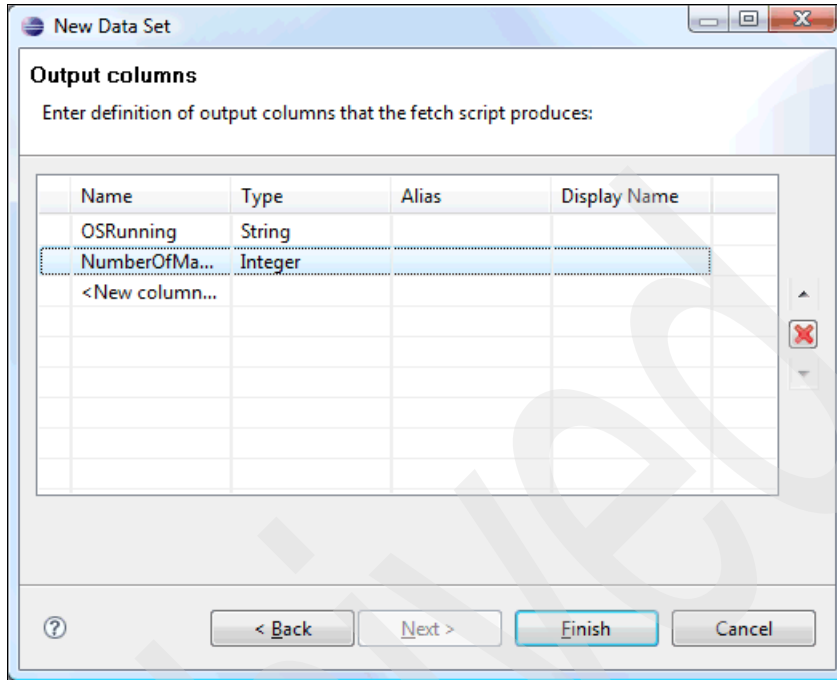


Figure 7-8 Column definitions

11. Click **Finish**.

The script window for the data set appears, as shown in Figure 7-9.

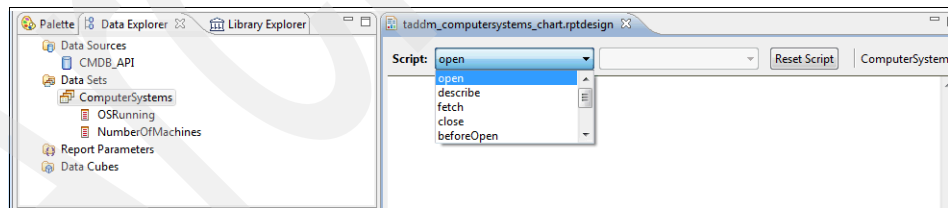


Figure 7-9 Script window for ComputerSystems data set showing available methods

Implementing the scripted data source and data set methods

Before implementing the data source and data set methods, we need to implement the intermediary Java class. We use this intermediary Java class to drive the CMDB APIs and convert the results to tabular form, which will be consumed by the scripted data set methods.

The sequence diagram in Figure 7-10 shows how the components of the scripted data source and the data set are related and how the intermediary Java class is used.

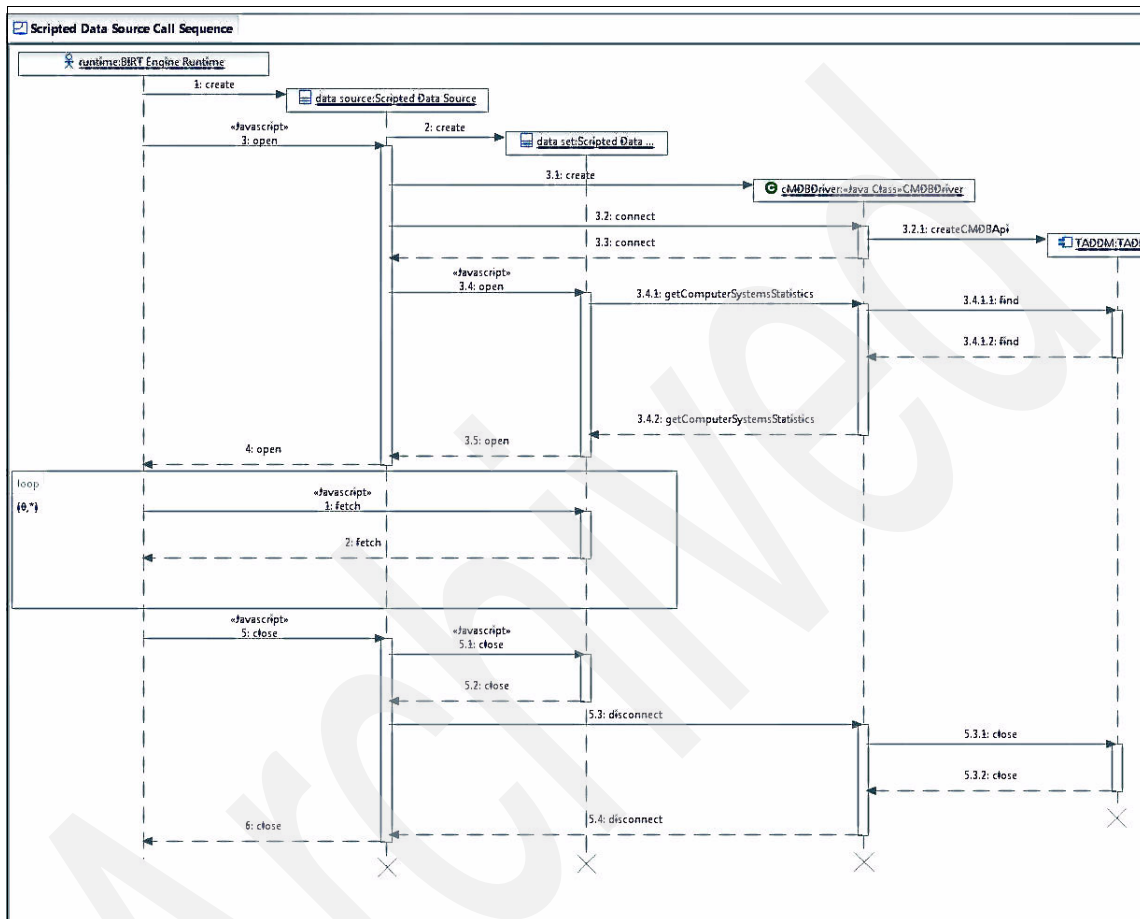


Figure 7-10 Scripted data source sequence diagram

Follow these steps to implement your first TADDM report:

1. Use Eclipse Java Platform, or any other Java development tool with which you are comfortable, to create a Java class as shown in Example 7-1.

Example 7-1 CMDBDriver.java class

```

import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;
  
```

```

import com.collation.platform.model.ModelObject;
import com.collation.platform.model.topology.sys.ComputerSystem;
import com.collation.proxy.api.client.ApiConnection;
import com.collation.proxy.api.client.ApiException;
import com.collation.proxy.api.client.ApiSession;
import com.collation.proxy.api.client.CMDBApi;
import com.collation.proxy.api.client.DataResultSet;
import com.ibm.cdb.api.ApiFactory;

public class CMDBDriver {

    public static final String DEFAULT_HOST = "localhost";

    public static final int DEFAULT_PORT = 9530;

    public static final String DEFAULT_ADMIN_USER = "administrator";

    public static final String DEFAULT_ADMIN_PASSWORD = "collation";

    private CMDBApi api = null;

    private ApiSession session = null;

    private ApiConnection connection = null;

    public void connect() {
        connect(DEFAULT_HOST, DEFAULT_PORT, DEFAULT_ADMIN_USER,
            DEFAULT_ADMIN_PASSWORD);
    }

    public void connect(String host) {
        connect(host, DEFAULT_PORT, DEFAULT_ADMIN_USER, DEFAULT_ADMIN_PASSWORD);
    }

    public void connect(String host, int port, String username, String password) {
        try {
            connection = ApiFactory.getInstance().getApiConnection(host, port,
                null, false);
            session = ApiFactory.getInstance().getSession(connection, username,
                password, ApiSession.DEFAULT_VERSION);
            api = session.createCMDBApi();
        } catch (ApiException e) {
            System.err.println("exception:" + e);
            e.printStackTrace();
        }
    }
}

```

```

    }
}

public void disconnect() {
    try {
        if (api != null) {
            api.close();
            api = null;
        }
        if (session != null) {
            session.close();
            session = null;
        }
        if (connection != null) {
            connection.close();
            connection = null;
        }
    } catch (Exception ex) {
        System.err.println("exception:" + ex);
        ex.printStackTrace();
    }
}

public List<String[]> getComputerSystems(String OSName) {

    if (api == null) {
        return null;
    }

    List<String[]> list = null;

    try {
        ModelObject[] mos = null;
        if (OSName.equals(""))
            mos = api.find("SELECT * from ComputerSystem", 2, null, null);
        else
            mos = api.find(
                "SELECT * from ComputerSystem WHERE OSRunning.OSName contains
"
                + "'" + OSName + "'", 2, null, null);

        list = new ArrayList<String[]>(mos.length);
        for (int i = 0; i < mos.length; i++) {
            ComputerSystem system = (ComputerSystem) mos[i];
            list.add(new String[] {

```

```

        system.getDisplayName(),
        system.getGuid().toString(),
        system.hasOSRunning() ? system.getOSRunning()
            .getOSName() : "N/A" });
    }

    } catch (ApiException ae) {
        System.err.println("api exception:" + ae);
        ae.printStackTrace();
    } catch (Exception ex) {
        System.err.println("exception:" + ex);
        ex.printStackTrace();
    } finally {
        disconnect();
    }

    return list;
}

public List<Object[]> getComputerSystemsStatistics() {

    List<Object[]> results = new ArrayList<Object[]>();
    List<String[]> systems = getComputerSystems("*");
    if (systems != null) {
        Iterator<String[]> it = systems.iterator();
        int windowsScore = 0;
        int linuxScore = 0;
        int aixScore = 0;
        int othersScore = 0;
        while (it.hasNext()) {
            String[] row = it.next();
            if (row[2].indexOf("Windows") != -1)
                windowsScore++;
            else if (row[2].indexOf("Linux") != -1)
                linuxScore++;
            else if (row[2].indexOf("AIX") != -1)
                aixScore++;
            else
                othersScore++;
        }
        results.add(new Object[] { "Windows", windowsScore });
        results.add(new Object[] { "Linux", linuxScore });
        results.add(new Object[] { "AIX", aixScore });
        results.add(new Object[] { "Others", othersScore });
    }
}

```

```

    }

    return results;
}

public static void main(String[] args) {
    CMDBDriver driver = new CMDBDriver();
    driver.connect("9.3.5.51");
    List<String[]> systems = driver.getComputerSystems("*");
    if (systems != null) {
        Iterator<String[]> it = systems.iterator();
        while (it.hasNext()) {
            Object[] row = it.next();
            System.out.println("Name: " + row[0] + "\tGUID: " + row[1]);
        }
    }
    driver.disconnect();
}
}

```

2. To build this class, you need three TADDM Client API JARs in the Eclipse Java project build path, api-client.jar, api-dep.jar, and platform-model.jar, which are under <COLLATION_HOME>/dist/sdk/lib. If you do not have access to the Eclipse Java development platform, you can simply compile the class in Example 7-1 on page 304 by using the Java 5.0 compiler from the command line, which is shown in Example 7-2.

Example 7-2 Compiling CMDBDriver.java class using the Windows command line

```

#set the COLLATION_HOME environment variable
set COLLATION_HOME=C:\IBM\cmdb

#add java to PATH
set PATH=%COLLATION_HOME%\external\jdk-1.5.0-Windows-i386\bin;%PATH%

#add TADDM API JARs to CLASSPATH
set
CLASSPATH=.;%COLLATION_HOME%\sdk\lib\api-client.jar;%COLLATION_HOME%\sd
k\lib\api-dep.jar;%COLLATION_HOME%\sdk\lib\platform-model.jar
#compile the class
javac -d . CMDBDriver.java

```

3. Go back to Eclipse Report Designer. In the Script view, as shown in Figure 7-9 on page 303, click **CMDB_API** under the data source, and then, select **open** from the method list box in the Script panel on the right. Enter the following Javascript code in the scripting space:

```
cmdbDriver = new Packages.CMDBDriver();  
cmdbDriver.connect("9.3.5.51");
```

Note: The TADDM Server in our lab was installed on 9.3.5.51. Change to the IP address or host name that is relevant for your environment, or just use `cmdbDriver.connect()` if the TADDM Server is running on the localhost.

4. Select **close** for the method from the CMDB_API data source and enter the following line of code in the scripting space:

```
cmdbDriver.disconnect();
```

The panel in Figure 7-11 shows where the code needs to be written.

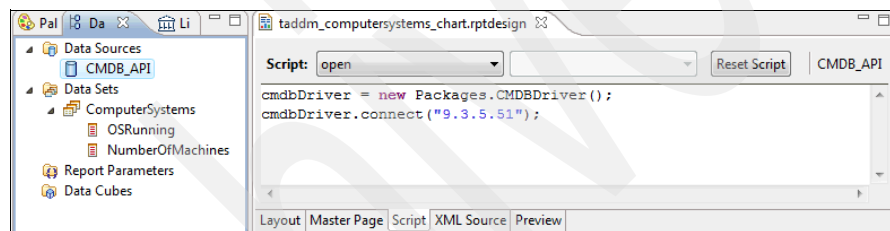


Figure 7-11 Adding Javascript code to the data source

5. Select **ComputerSystems** for the data set and select **open** in the scripting panel. Add the following code:

```
stats = cmdbDriver.getComputerSystemsStatistics();  
size = stats.size();  
currentRecord = 0;
```

6. Select **fetch** for the method from the data set methods list. Add the following code:

```
if(currentRecord >= size) {  
    return false;  
}  
item = stats.get(currentRecord);  
row["OSRunning"] = item[0];  
row["NumberOfMachines"] = item[1];  
currentRecord++;  
  
return true;
```

7. Save the report.
8. If you use the BIRT 2.2.x All-in-One package, as described in “Setting up the BIRT 2.2.x All-in-One package” on page 297, set up your CMDBDriver Java project in the same workspace as the TADDM Reports BIRT project. If you use the BIRT 2.2.x All-in-One package, you can now perform a quick test to make sure that your data source and data set work properly:
 - Right-click **ComputerSystems** for the data set, and select **Edit**.
 - Select the last option **Preview Results**.
 - Depending on the speed of your TADDM Server, you will see the results preview as shown in Figure 7-12.

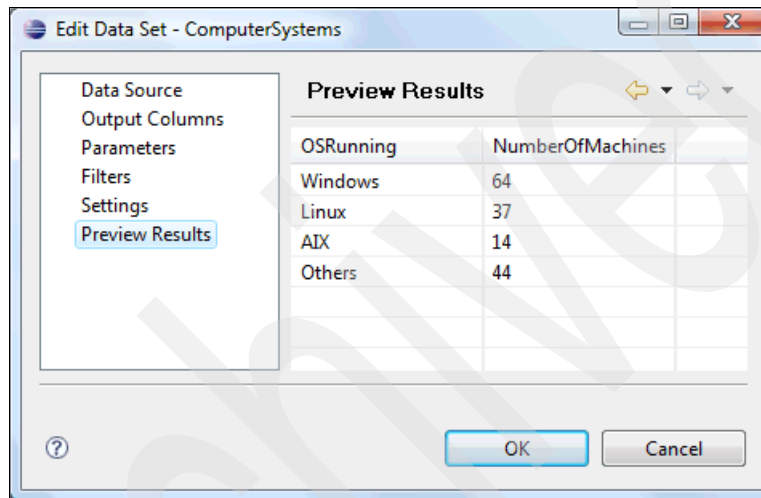


Figure 7-12 Scripted data set results preview

9. Refer to Figure 7-11 on page 309, and select the **Layout** tab on the lower right panel, and then, select **Palette** on the left.
10. Select **Text** from the list under Report Items and drag it to the report layout. The Edit Text Item dialog shows up, as shown in Figure 7-13 on page 311.
11. Change the selection at the top of Figure 7-13 on page 311 from Auto to **HTML**, and then, insert the HTML markup text as shown in Example 7-3 on page 311. This text will become the report title section.

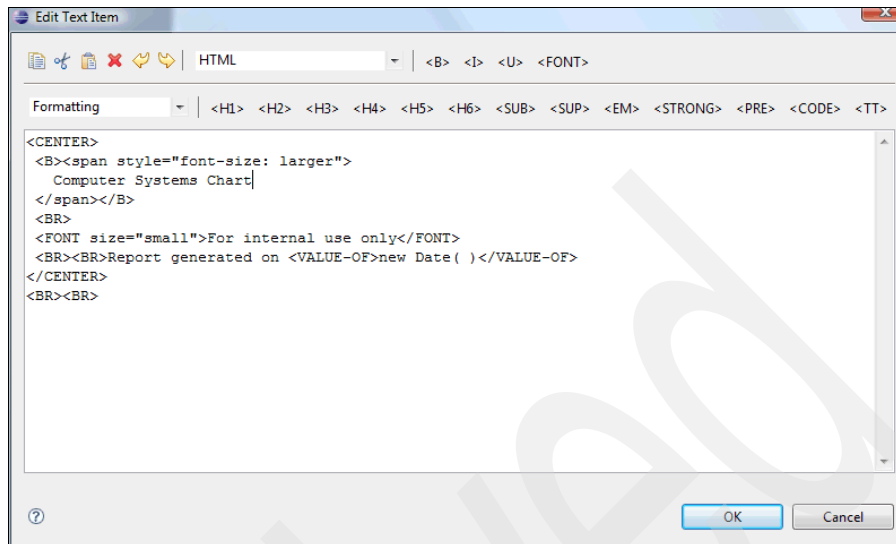


Figure 7-13 Edit Text Item dialog with the changes added

Example 7-3 Sample report title in HTML

```
<CENTER><B><span style="font-size: larger">
Computer Systems List
</B></span><BR>
<FONT size="small">For internal use only</FONT><BR><BR>
Report generated on <VALUE-OF>new Date( )</VALUE-OF>
</CENTER><BR><BR>
```

12. Select **Chart** from the **Palette** tab and drag it to the report layout. The New Chart dialog is displayed as shown in Figure 7-14 on page 312.

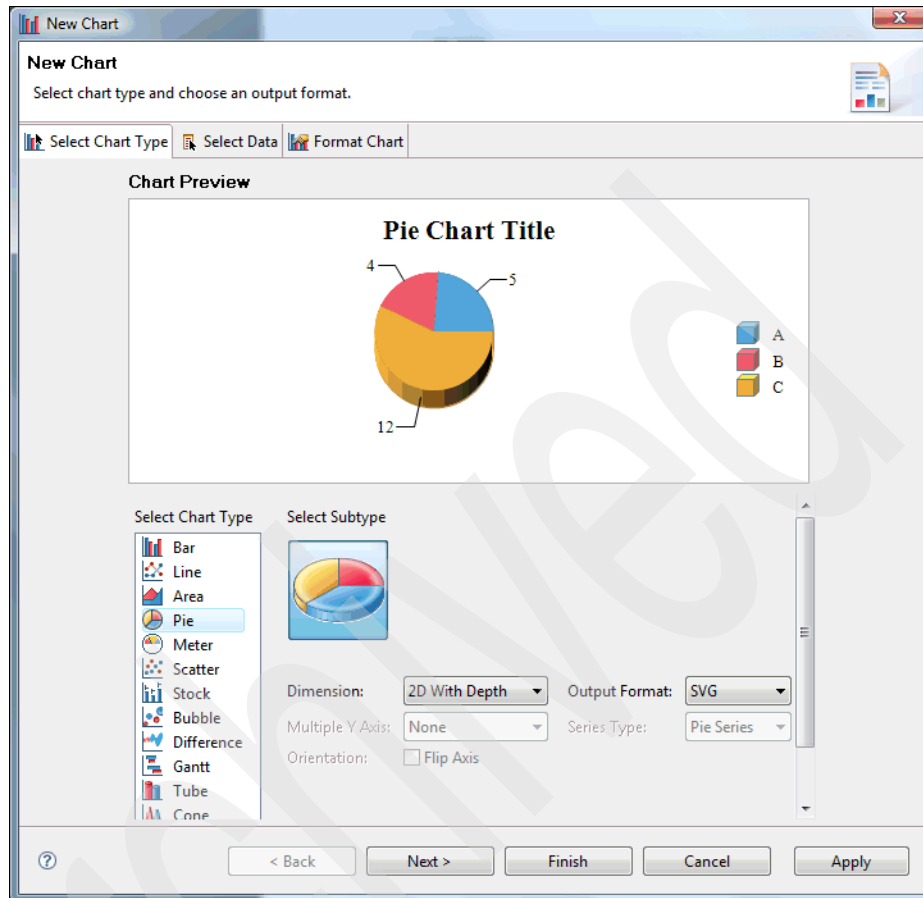


Figure 7-14 New Chart dialog with the Pie type chart selected

13. Select **Pie** from Select Chart Type list, and optionally, choose **2D With Depth** from the Dimension list box.
14. Click the **Select Data** tab. Choose **Use Data Set**, and then, select **ComputerSystems** for the data set, which we created earlier, as shown in Figure 7-7 on page 302, and wait until the data is loaded into the Data Preview panel.
15. Select **OSRunning** and either right-click and select **Use as Category Series** or just simply drag it to Category Definition text box space. The color of the column will change, which indicates that the data has been bound to the chart.

16. Select **NumberOfMachines**, the second column, but drag this column to the Slice Size Definition text box space, or right-click **NumberOfMachines** and select **Plot as Value series**. You see the results in Figure 7-15.

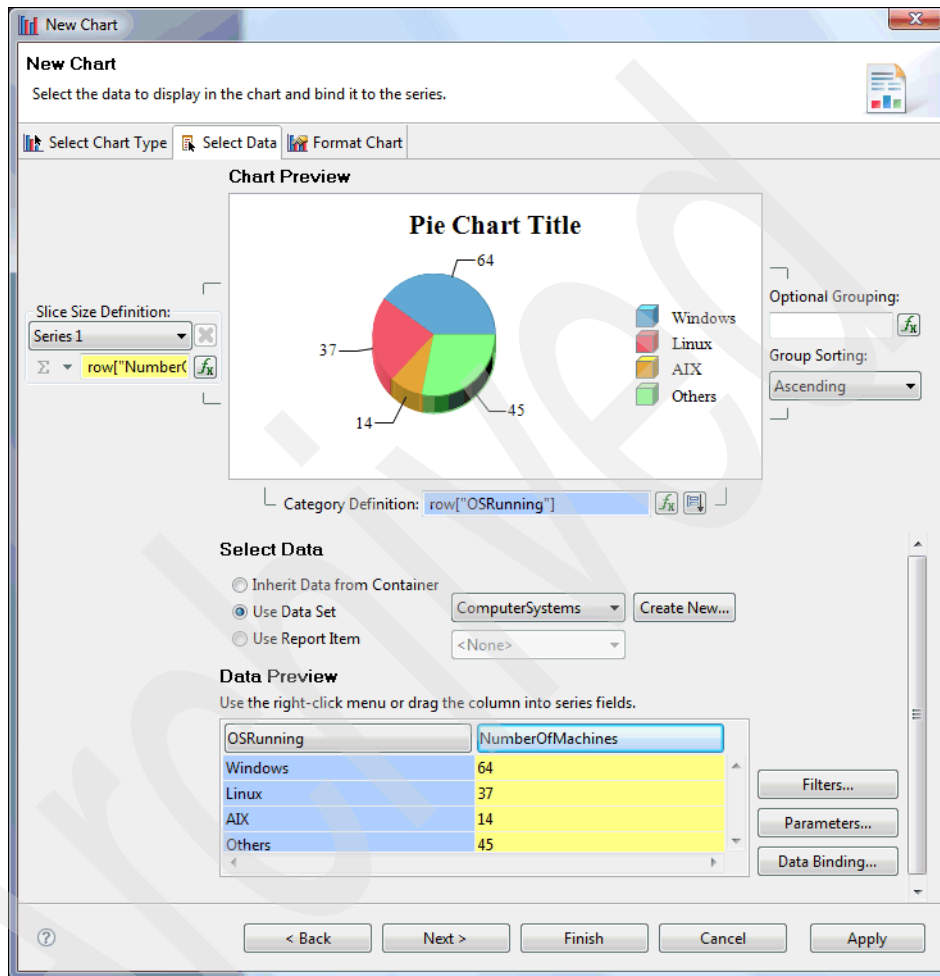


Figure 7-15 Preview of the chart after the data set is bound to it

17. Finally, select **Format Chart** to add a chart title or interactivity behavior. Select **Title** under the Chart Area list and enter a title in Chart Title text box, as shown in Figure 7-16 on page 314.

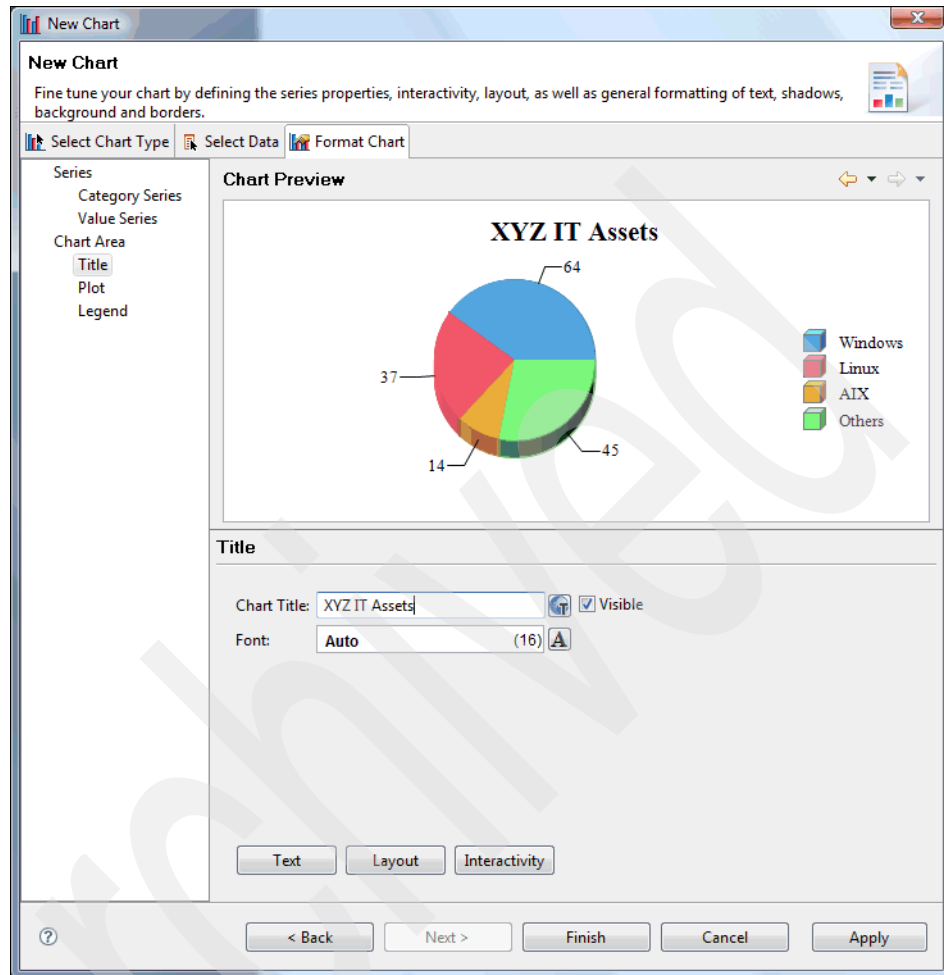


Figure 7-16 Formatting the chart

18. Click **Finish**. Adjust the chart frame to the preferred size. Save the report design. The final layout is displayed in Figure 7-17 on page 315.

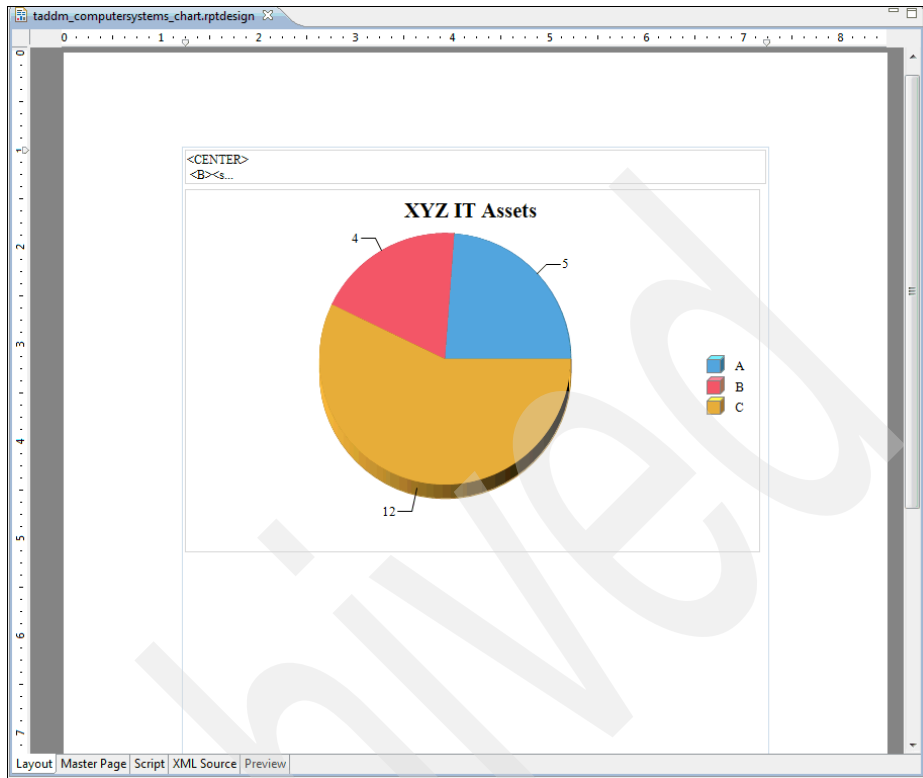


Figure 7-17 The final report layout

Note: You cannot use the Preview tab in the layout panel or actually run the report in Eclipse, because both `api-dep.jar` and one of the Eclipse internal plug-ins define and implement Apache's Log4J interfaces, which causes a class loader exception. This problem does not occur when you run the report outside of the Eclipse environment.

Deploying the report design on the TADDM Server

Assuming that you have deployed BIRT Report Viewer on TADDM Tomcat as explained in 7.2, “Deploying BIRT Report Viewer on TADDM” on page 294, follow these steps to deploy and view your new report:

1. Copy the report design file, `taddm_computersystems_chart.rptdesign`, from the Eclipse BIRT project to your TADDM Server machine under `<COLLATION_HOME>/dist/deploy-tomcat/birt`. If you do not find the `birt` folder, it means that you did not deploy BIRT Report Viewer to TADDM Tomcat.

2. Copy CMDBDriver.class, which you built using Example 7-1 on page 304, to `<COLLATION_HOME>/dist/deploy-tomcat/birt/WEB-INF/classes`. If the classes folder does not exist, create a classes folder.
3. Copy api-client.jar, api-dep.jar, and platform-model.jar from `<COLLATION_HOME>/dist/sdk/lib` to `<COLLATION_HOME>/dist/deploy-tomcat/birt/WEB-INF/lib`.
4. Restart the TADDM Server.
5. Using your browser, navigate to the following URL:
http://<TADDM_HOST>:9430/birt/frameset?__report=taddm_computersystems_chart.rptdesign
6. Wait for the report to be rendered in the browser. The result is displayed in Figure 7-18.

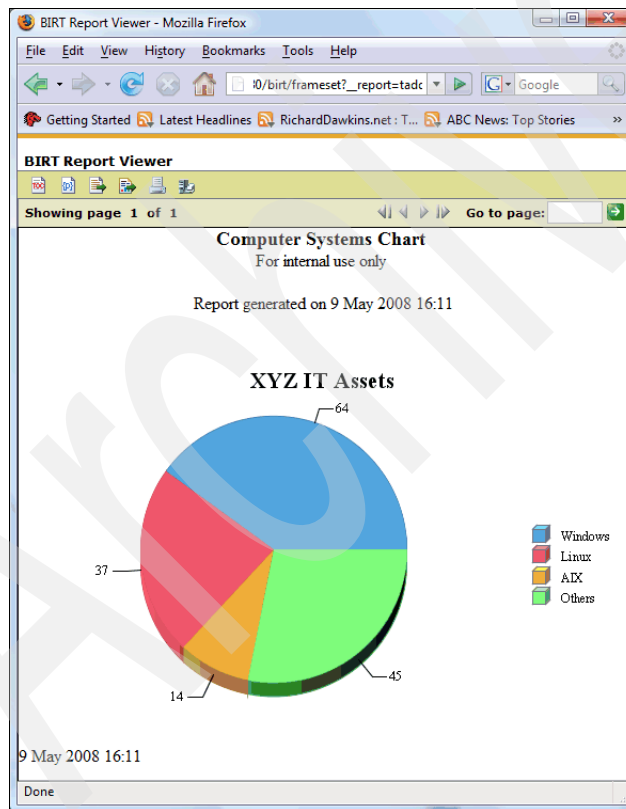


Figure 7-18 TADDM Computer Systems report using BIRT

7.3.2 Designing reports with TADDM Database Views

GlobalCo wants to run a quick software audit on XYZ to verify that all of the software products that are installed on certain Windows machines are licensed.

XYZ kept the Windows software product licenses in an Excel Spreadsheet, which lists the installed software product name, version, vendor, and the machine name on which the product is installed.

In this scenario, we show you how to use BIRT to produce a license cross-check report on a particular Windows machine. We show you how BIRT can access more than one source of information, such as an Excel sheet and TADDM's CMDB data, and then, organize the data on one report.

We use TADDM Database Reporting Views as the CMDB data source in order to show you an alternative way to access TADDM data without using any programming.

Issues with using the scripted data set

The biggest disadvantage of using the scripted data set is the need to use Java/Javascript programming to access data. The alternative is to use Java Database Connectivity (JDBC) Data Source where BIRT accesses specially created CMDB database views directly. TADDM 7.1 ships with a script that creates those views. These views were designed to mimic the information that is displayed in the Details Panel of the Product Console.

Generating TADDM Database views

Complete the following steps to add the database views that you see listed in the steps. Both Oracle and DB2 scripts are included. For details about setting up the views, refer to the Tivoli Application Dependency Discovery Manager Version 7.1 information center, under the SDK Developer's Guide, and then, look for Exploring database reporting views. Here, we explain how to generate the database views for DB2 only.

To add or drop the views for a DB2 database, run the following scripts:

- The following script creates the reporting views:

```
dist/support/bin/make_db2_reporting_views.sh [dbpassword]
```

- The following script deletes the previously created reporting views:

```
dist/support/bin/drop_db2_reporting_views.sh [dbpassword]
```

where:

The dbpassword is optional. The command prompts you for a password if no password is provided on the command line.

- These shell scripts call the following SQL scripts:

```
dist/support/bin/make_db2_reporting_views.sql  
dist/support/bin/drop_db2_reporting_views.sql
```

Creating data sources for the database views and Excel

We start the same way that we did in 7.3.1, “Designing reports with scripted data source” on page 297:

1. Reuse the same project to create a new Report design. We call it `taddm_software_audit.rptdesign`.
2. Create a new data source, and select a JDBC data source, as shown in Figure 7-19, and then, click **Next**.

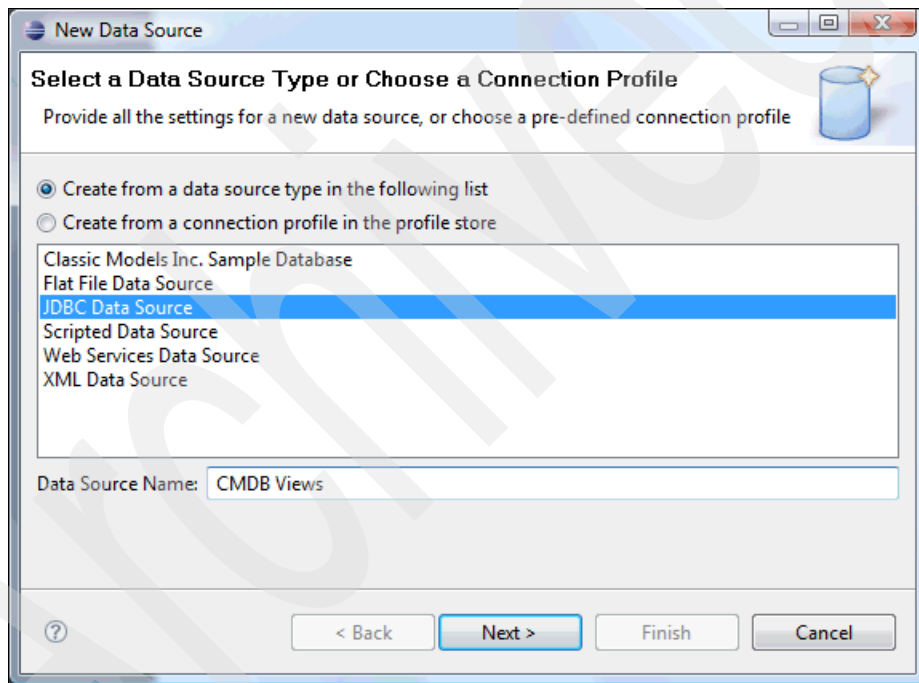
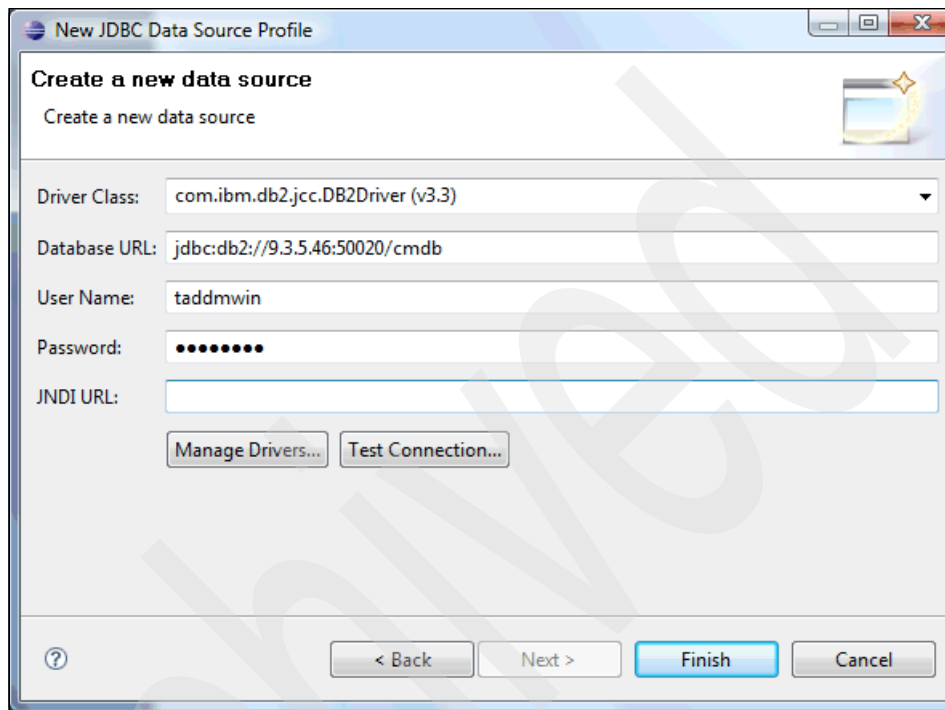


Figure 7-19 New JDBC data source

3. You need to specify the JDBC driver class, as well the Database URL or connection string on Figure 7-20 on page 319. If you use DB2 as the CMDB data set, choose **com.ibm.db2.jcc.DB2Driver** from the list of Driver Classes, and fill the rest of the entries with the database URL and the database username and password. The user that you specify needs to be the CMDB instance owner or a DB user with permission to access the CMDB views. Refer to Figure 7-20 on page 319 for an example. The URL is a JDBC

connection string that has the form `jdbc:<db name>://<host name>:port/<instance name>`. You almost certainly have other values to enter here apart from the Driver Class field.



New JDBC Data Source Profile

Create a new data source

Driver Class: com.ibm.db2.jcc.DB2Driver (v3.3)

Database URL: jdbc:db2://9.3.5.46:50020/cmdmb

User Name: taddmwin

Password:

JNDI URL:

Manage Drivers... Test Connection...

< Back Next > Finish Cancel

Figure 7-20 JDBC connection details for CMDB on DB2

4. If you do not find `com.ibm.db2.jcc.DB2Driver` class in the list, you need to add the DB2 JDBC JARs to the BIRT-managed drivers. Click **Manage Drivers** and add the DB2 JDBC driver as shown in Figure 7-21 on page 320.

Note: The DB2 Universal JDBC driver is a Type 4 driver that can connect directly to a DB2 server. No DB2 client software is required on the platform where the driver is installed. You can find the driver in a pre-existing DB2 installation or download the driver from this Web site:

https://www14.software.ibm.com/webapp/iwm/web/preLogin.do?lang=en_US&source=swg-dm-db2jdbcdriver

The driver files are:

- ▶ db2jcc.jar
- ▶ db2jcc_license_cu.jar

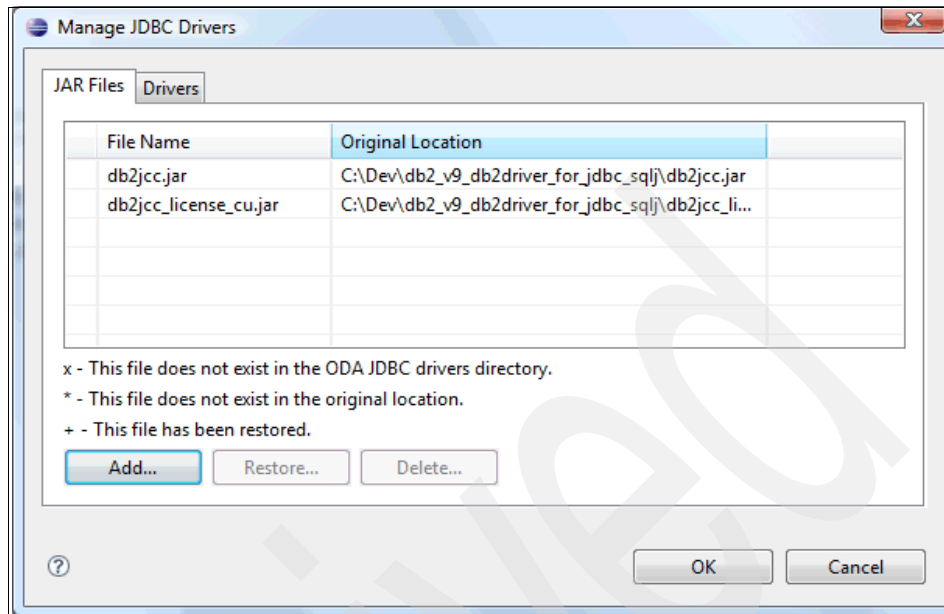
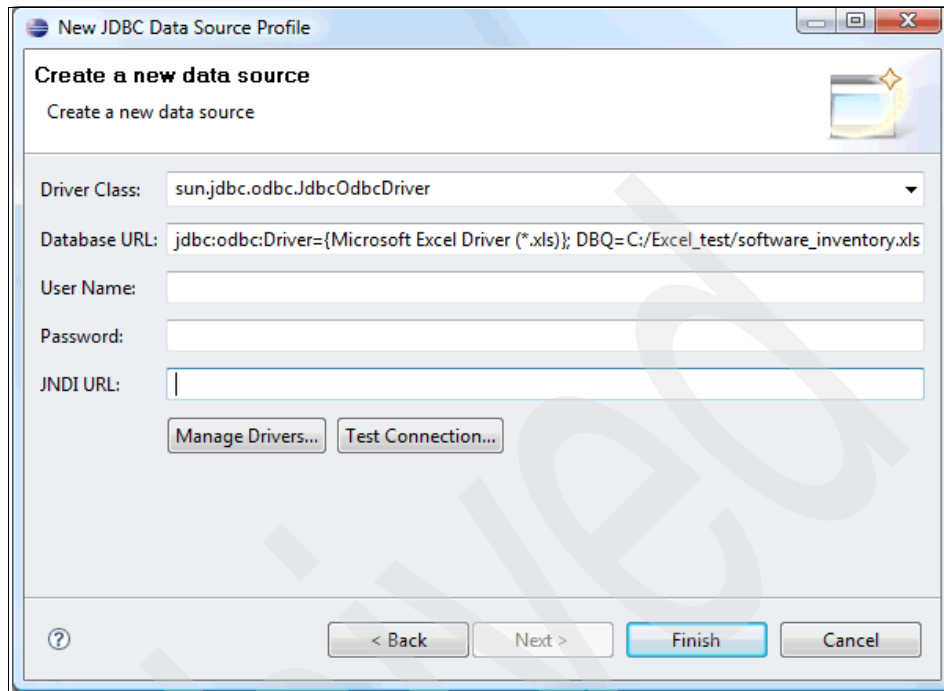


Figure 7-21 Adding DB2 JDBC JAR files to BIRT-managed drivers

5. Click **Test Connection** to make sure that your settings are correct.
6. Create a second report, and call it excel_software_inventory.rptdesign.
7. Create a data source for the Excel sheet, and call it Excel, for example.
8. Use the JDBC data source, and enter the JDBC connection details as shown in Figure 7-22 on page 321. You need to enter the ODBC driver class in the Driver Class field. You do not need to add any JARs.



New JDBC Data Source Profile

Create a new data source

Driver Class: sun.jdbc.odbc.JdbcOdbcDriver

Database URL: jdbc:odbc:Driver={Microsoft Excel Driver (*.xls)}; DBQ=C:/Excel_test/software_inventory.xls

User Name:

Password:

JNDI URL:

Manage Drivers... Test Connection...

< Back Next > Finish Cancel

Figure 7-22 JDBC connection details for Microsoft Excel sheet

9. Test the connection, and then, click **Finish**.

Creating data sets

Follow these steps to create two data sets, one in each of the reports. One data set loads data from the Software Components View and the other data set loads data from Software_inventory.xls.

The inventory sheet name is “Products”, and it contains the information that is shown in Figure 7-23.

	A	B	C	D	E
1	Product Name	Product Version	License	Manufacturer	Host Name
2	Symantec AntiVirus	9.0.310	1234-1234-1234-1234	Symantec	southend
3	Tivoli Monitoring	6.2.0	5555-3333-5566-8888	IBM	TORONTO
4	Photoshop Pro	8.0.12	5678-5678-5678-5678	Adobe	WACO
5	FrameMaker	7.2.1	8765-8765-8765-9876	Adobe	COPENHAGEN
6	Tivoli Continues Data Protection	3.1.0.41	9876-5432-1234-5678	IBM	TORONTO

Figure 7-23 Software_inventory.xls sample content

To create two data sets:

1. In `taddm_software_audit.rptdesign`, create a new data set and specify `CMDB_Views` as the data source and `SQL Select Query` as the data set type. Click **Next**.
2. Filter out unwanted schema by choosing your CMDB instance schema and selecting **View** from the Type list box, and then, click **Apply Filter**.
3. In the SQL edit box, enter the asterisk character (*) to the right of “select”, and then, set the cursor on the space next to “from”.
4. From the list of views on the left under Available Items, select **CDT_SOFTWARE_COMPONENT_VIEW** as shown in Figure 7-24.

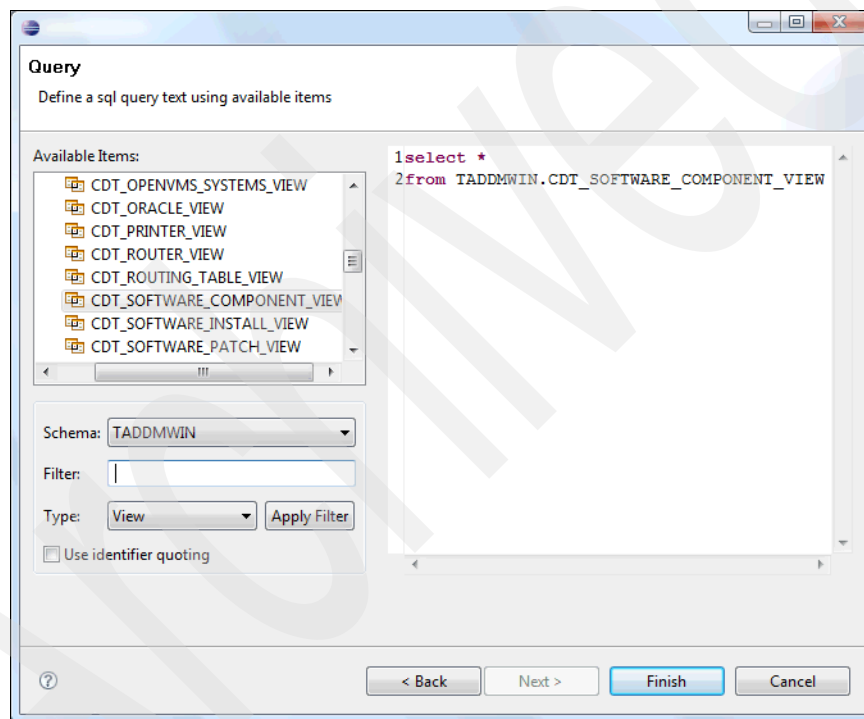


Figure 7-24 Setting up the software component query

5. Click **Finish**. Then, right-click the data set that you have created, and select **Edit** → **Preview Results** to check that you can load the data correctly.
6. In `excel_software_inventory.rptdesign`, create a second data set for the Excel sheet, selecting **Excel** as the data source and **SQL Select Query** as the type. Call it `Software_Lic`, for example.
7. In Figure 7-25, enter the following SQL statement in the SQL edit box:

```
select *  
from [Products$]  
where Products is the name of the sheet.
```

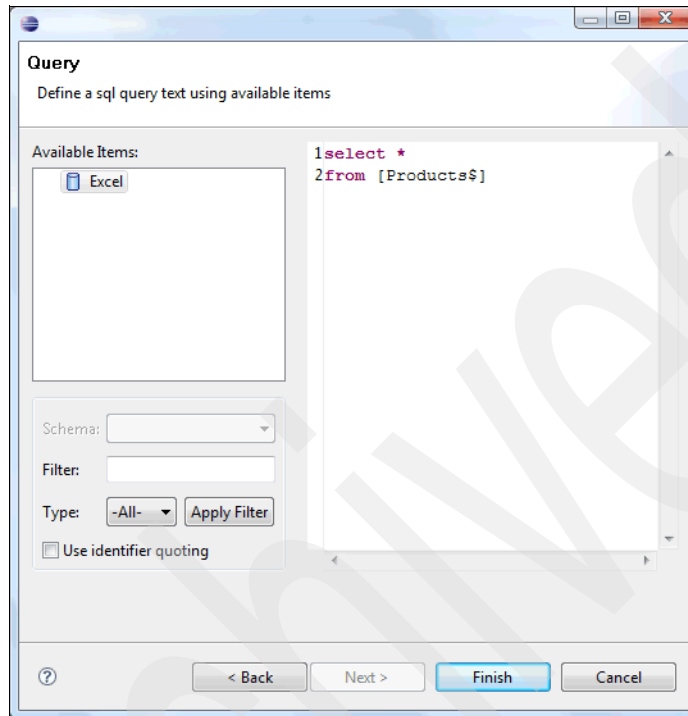


Figure 7-25 Query definition of Excel data set

8. Click **Finish**. Then, check **Preview Results** to see that the data is loaded correctly. Figure 7-26 on page 324 shows the preview results of our Excel sheet.

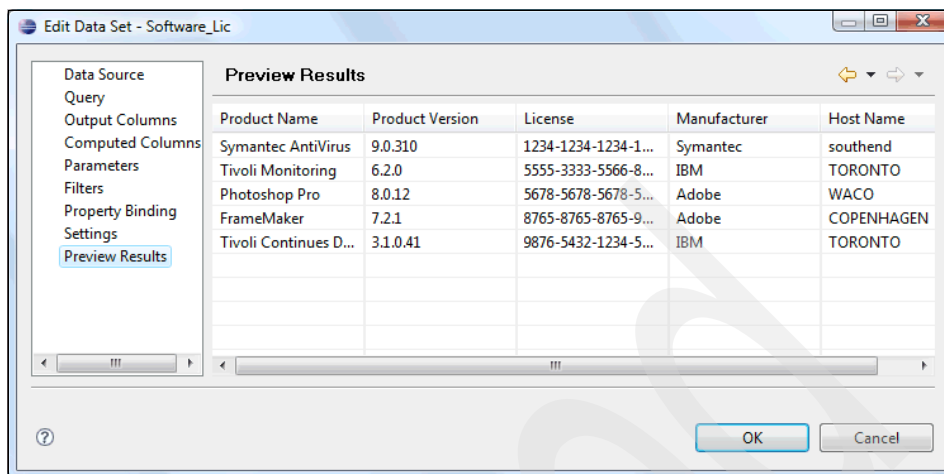


Figure 7-26 Preview results of the Software_Lic data set

Designing the reports

Follow these steps to design the reports:

1. While in the `taddm_software_audit.rptdesign` layout view, choose the **Palette** tab on the left panel. The palette displays all of the elements that you can place in a report.
2. Drag a Table element from the palette, and drop it in the report in the layout editor. The Insert Table dialog prompts you to specify the number of columns and the number of detail rows to create for the table. The dialog also prompts you to select a data set to bind with the table.
3. On the Insert Table dialog, specify the following values, as shown in Figure 7-27.

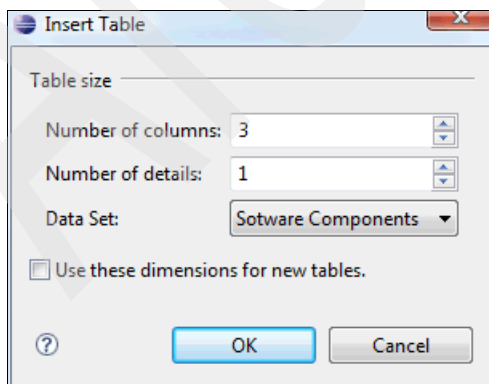


Figure 7-27 Insert Table dialog

4. Choose **OK**. A table with three columns and one detail row appears in the layout editor. Now, you are ready to insert the data into the table.
5. Choose **Data Explorer**. Expand **Data Sets** → **Software Components**.
6. Drag **FQDN**, **PRODUCT_NAME**, and **PRODUCT_VERSION** one by one from Data Explorer, and drop them in the first, second, and third cells in the table's detail row.
7. In the layout editor, the table cells contain data elements. Appearing above these data elements are the label elements that the layout editor automatically added to the header row. These labels display the field names as static text and serve as the column headings. You can change these labels to more elegant heading names and format them if you prefer.
8. Group the data in the table under FQDN by selecting the table, right-clicking, and selecting **Insert Group**. Edit the entries as shown in Figure 7-28.

New Group

Group Details

Name: FQDN Grouping

Group On: FQDN

Interval: No interval

Range:

Use fixed base value for interval: ☐

Hide Detail: ☐

Sort Direction: ☒ Ascending ☐ Descending

Page break: Before: Auto After: Auto Inside: Auto

Repeat Header: ☒

Bookmark:

Table of Contents

Item Expression: row["FQDN"]

Style:

Filters and Sorting

Filters **Sorting**

Filter by: Add... Remove Edit... Up Down

Expression	Operator	Value 1	Value 2

OK Cancel

Figure 7-28 Add grouping feature to the table

9. Select **FQDN** (the data element, not the grouping element), and from **Property Editor - Data** in the panel beneath the layout, choose the **Properties** tab, and then, select **Visibility**.
10. Select the **Hide Element** check box so that the host name is not repeated with every row.
11. Optionally, add a title to the report by dragging a Text element from the palette. You can use a variation of Example 7-3 on page 311.
12. The final design is shown in Figure 7-29.

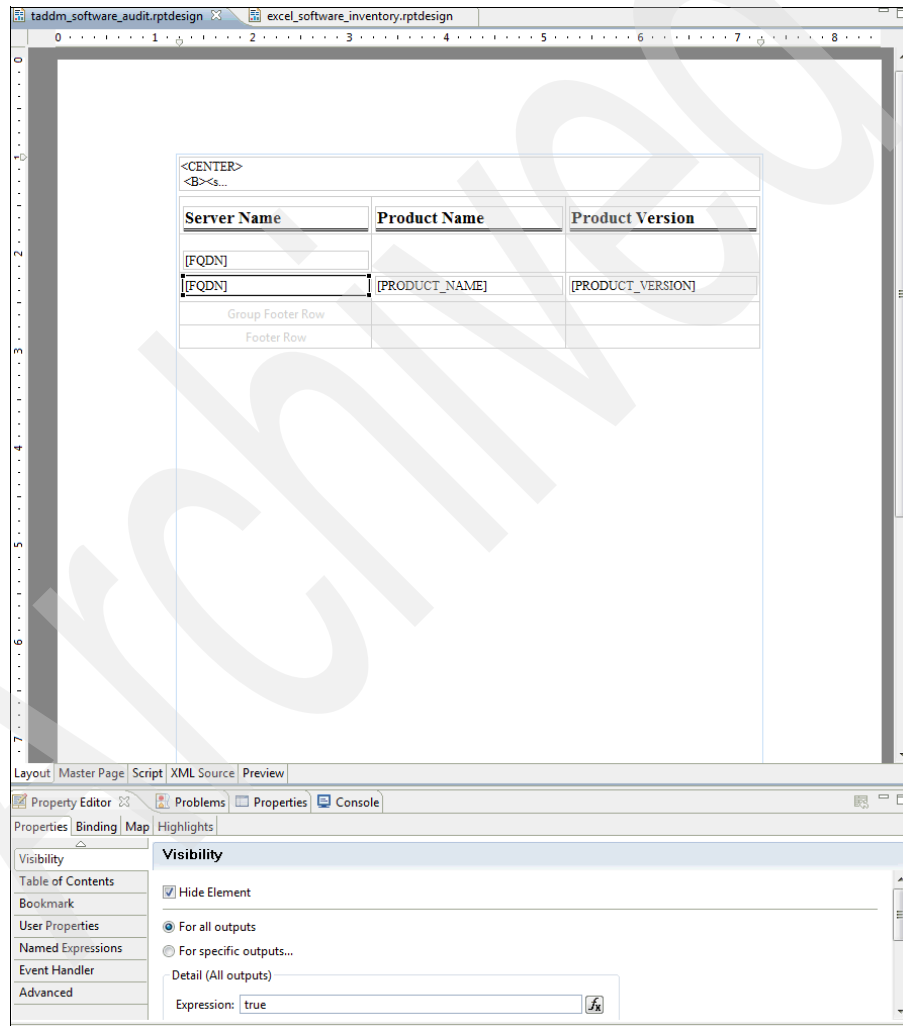


Figure 7-29 The final design of taddm_software_audit.rptdesign

13. In the `excel_software_inventory.rptdesign` layout, drag a table element with 3x1 size.
 14. Choose **Data Explorer** → **Data Sets** → **Software_Lic**.
 15. Drag the **Product Name**, **Product Version**, and **License** data elements, one by one, from Data Explorer, and drop them in the first, second, and third cells in the table's detail row.
 16. In Data Explorer, right-click **Report Parameters** and select **New Parameter**.
 17. In the New Parameter dialog, enter the name of the parameter as `hostname`.
 18. In the same way, create a second parameter and call it `productname`.
 19. Optionally, add a title to the report, using a variation of Example 7-3 on page 311, and format the header row to your preference.
- The final report design is shown in Figure 7-30.

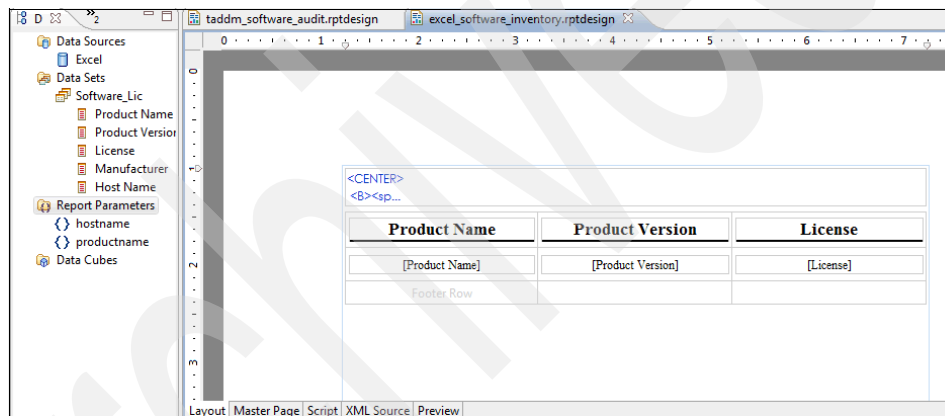


Figure 7-30 The final design of `excel_software_inventory.rptdesign`

Creating a link between the reports

The reporting scenario that we need to implement has the following flow:

1. The user runs the `taddm_software_audit` report.
2. While browsing through the report, the user finds a particular software component that was discovered by TADDM as being installed on a particular host.
3. The user wants to make sure that this software component is licensed to run on this specific host.
4. The user clicks the software component. A new window appears that shows the `excel_software_inventory` report with only the software products that are licensed to run on the host in question.

5. If the software component was licensed for this host and is included in the Excel sheet (software_inventory.xls), it is highlighted in the new window.
6. If no highlighted item is shown in the new window, it means that either the discovered software component does not need a license or it is possibly an illegal installation.

Follow these steps to link the two reports so that you can implement this scenario:

1. In the taddm_software_audit report layout, select the data element, **PRODUCT_NAME**.
2. Select the **Property Editor - Data** tab, **Properties**, and **Hyperlink**. The Hyperlink properties appear, as shown in Figure 7-31.

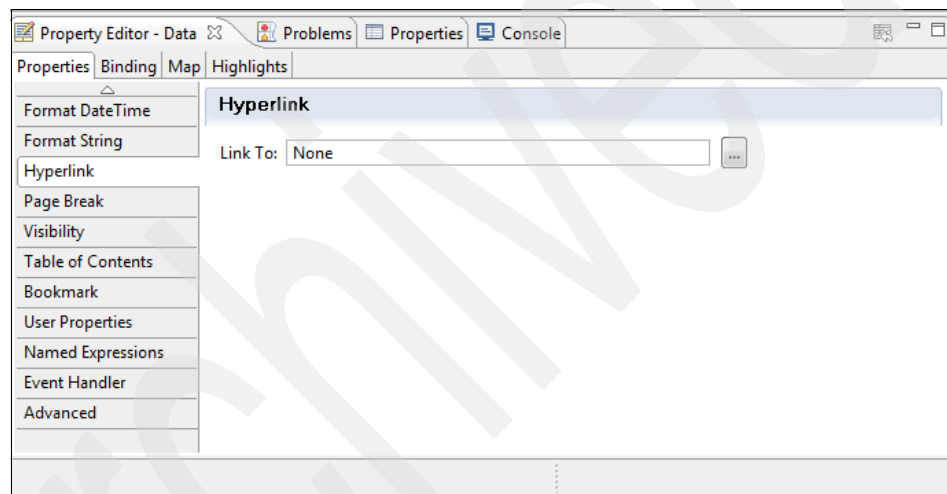


Figure 7-31 Choosing the Hyperlink option

3. Choose ... (the ellipsis).
4. On the Hyperlink Options panel, complete the following tasks:
 - a. For Select Hyperlink Type, choose **Drill-through**.
 - b. In Step 1: Select a target report, select **Report Design**, then, choose **Browse For File**, and navigate to excel_software_inventory.rptdesign.
 - c. Because the target report has parameters, Report Parameters is enabled.
 - d. Select the field beneath the Parameters column heading.
 - e. Select **hostname** from the drop-down list box in this field.
 - f. Select the field beneath the Values column heading, and type this expression in the Values field for the hostname parameter : row["FQDN"].

- g. Type this expression in the Values field for the productname parameter:
row["PRODUCT_NAME"].
- h. For Step 4: Show Target report in, select **New Window**.
- i. The Hyperlink Options panel is shown in Figure 7-32. Click **OK**.

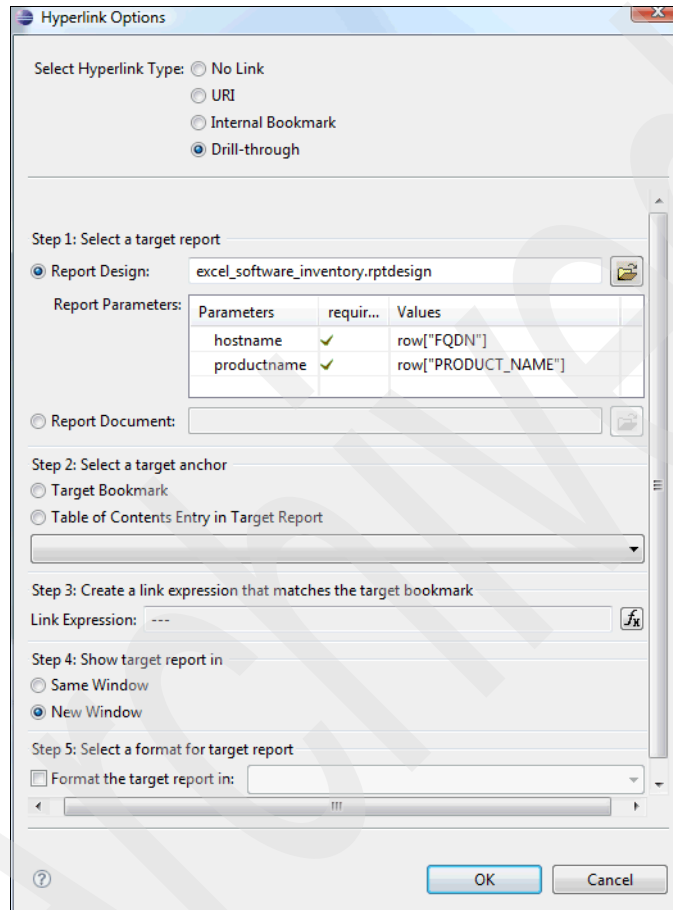


Figure 7-32 The Hyperlink Options choices

- j. In the excel_software_inventory report layout, select the data table.
- k. In the Property Editor tab, select the **Filters** tab.
- l. Click **Add**.
- m. On the New Filter Condition panel, enter the following expression:

```
params["hostname"].toUpperCase().indexOf(row["Host  
Name"].toUpperCase(), 0)
```

- n. Change the operator to **Greater than or Equal**. Then, enter 0 in the value text box as shown in Figure 7-33, and then, click **OK**.

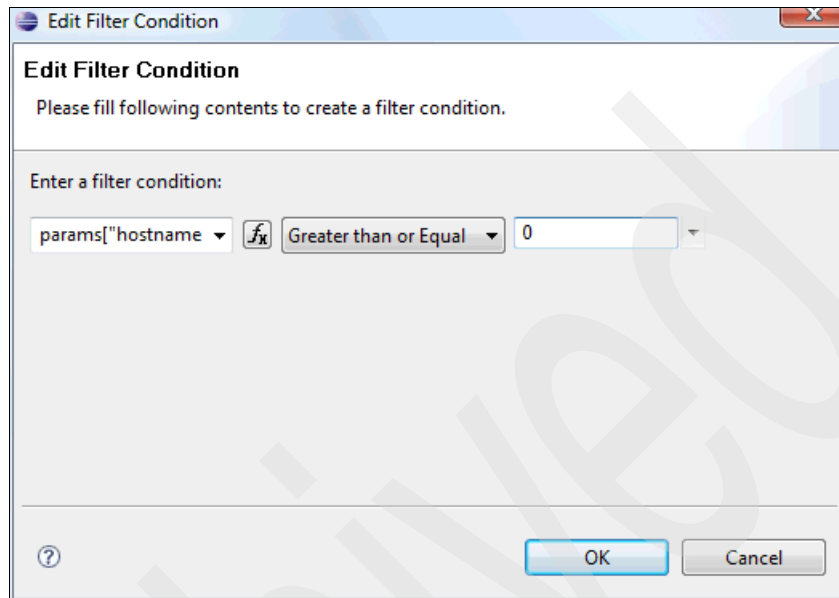


Figure 7-33 excel_software_audit table filter options

- o. Select the table details row.
- p. In the Property Editor, select the **Highlights** tab.
- q. Click **Add**.
- r. On the New Highlight panel, enter the following expression:
`params["productname"].toUpperCase().indexOf(row["Product Name"].toUpperCase(), 0)`
- s. Change the operator to **Greater than or Equal**. Then, enter 0 in the value textbox. Choose the type of highlight that you want to see when a software component has been found in the software inventory sheet, as shown in Figure 7-34 on page 331. Then, click **OK**.

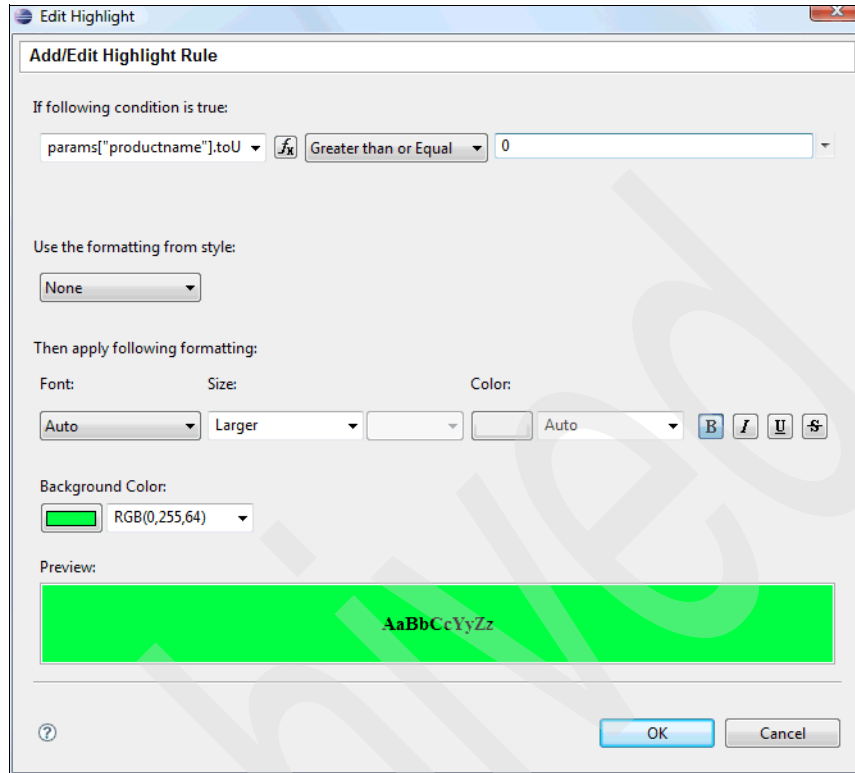


Figure 7-34 The Highlight options for excel_software_inventory row

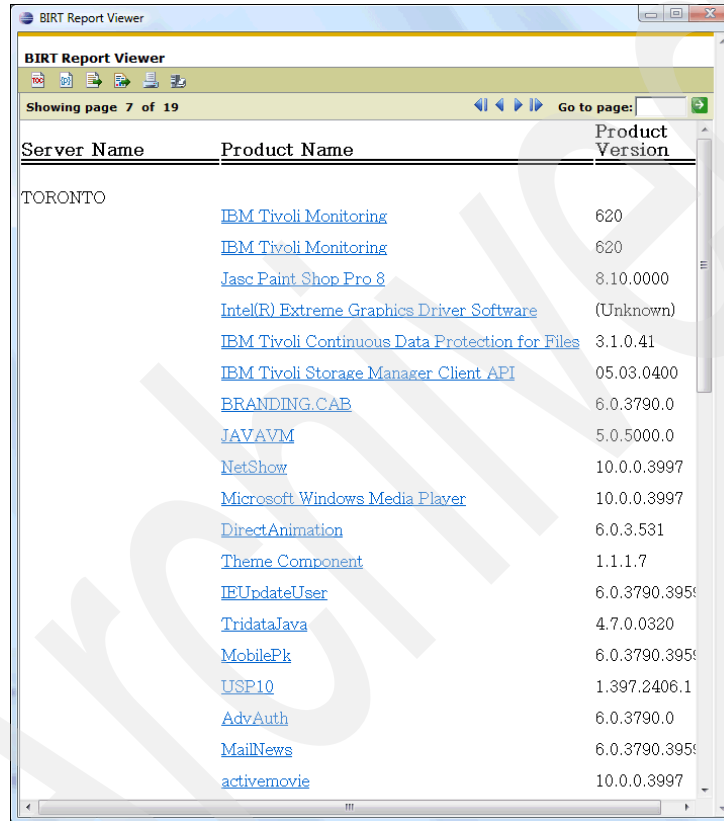
- t. In order to see the host name in the title of the excel_software_inventory report, type the Example 7-4 code in the title Text element. Make sure that you select **HTML** from the drop-down list.

Example 7-4 Sample title text box content for excel_software_inventory report

```
<CENTER>
<B><span style="font-size: larger">
  <VALUE-OF>params["hostname"]</VALUE-OF> Software Products Licenses
</span></B>
<BR>
<FONT size="small">For internal use only</FONT>
</CENTER>
<BR><BR>
```

Now, we are ready to test this scenario:

1. Either copy both reports to TADDM Tomcat under birt as explained in “Deploying the report design on the TADDM Server” on page 315, or just use Eclipse BIRT designer to test.
2. Using Eclipse, right-click **taddm_software_audit** report in the Navigator panel.
3. Choose **Report** → **Run Report**. Figure 7-35 is displayed.



Server Name	Product Name	Product Version
TORONTO	IBM Tivoli Monitoring	620
	IBM Tivoli Monitoring	620
	Jasc Paint Shop Pro 8	8.10.0000
	Intel(R) Extreme Graphics Driver Software	(Unknown)
	IBM Tivoli Continuous Data Protection for Files	3.1.0.41
	IBM Tivoli Storage Manager Client API	05.03.0400
	BRANDING.CAB	6.0.3790.0
	JAVAVM	5.0.5000.0
	NetShow	10.0.0.3997
	Microsoft Windows Media Player	10.0.0.3997
	DirectAnimation	6.0.3.531
	Theme Component	1.1.1.7
	IEUpdateUser	6.0.3790.3950
	TridataJava	4.7.0.0320
	MobilePk	6.0.3790.3950
	USP10	1.397.2406.1
	AdvAuth	6.0.3790.0
	MailNews	6.0.3790.3950
	activemovie	10.0.0.3997

Figure 7-35 TADDM Software Components Report

4. Notice that the Product Name items are hot links. If we click **IBM Tivoli Monitoring**, for example, a new window pops up as shown in Figure 7-36 on page 333.

The screenshot shows the BIRT Report Viewer window. The title bar says "BIRT Report Viewer". The report content area has a header "TORONTO Software Products Licenses" with the subtitle "For internal use only". Below the header is a table with three columns: "Product Name", "Product Version", and "License". The first row, "Tivoli Monitoring", is highlighted in green. The second row, "Tivoli Continues Data Protection", is not highlighted. The timestamp "14/05/2008 9:11 PM" is displayed at the bottom left of the report area.

Product Name	Product Version	License
Tivoli Monitoring	6.2.0	5555-3333-5566-8888
Tivoli Continues Data Protection	3.1.0.41	9876-5432-1234-5678

14/05/2008 9:11 PM

Figure 7-36 Software Inventory report with a highlighted item

5. The Tivoli Monitoring item is highlighted, which indicates that this software component is licensed on the host name named TORONTO.
6. If we click **Jasc Paint Shop Pro 8** in the first report window, the second report window is displayed, but no product is highlighted, which indicates that no license was found for Paint Shop Pro, as shown in Figure 7-37.

The screenshot shows the BIRT Report Viewer window. The title bar says "BIRT Report Viewer". The report content area has a header "TORONTO Software Products Licenses" with the subtitle "For internal use only". Below the header is a table with three columns: "Product Name", "Product Version", and "License". The first row, "Tivoli Monitoring", is not highlighted. The second row, "Tivoli Continues Data Protection", is not highlighted. The timestamp "14/05/2008 9:13 PM" is displayed at the bottom left of the report area.

Product Name	Product Version	License
Tivoli Monitoring	6.2.0	5555-3333-5566-8888
Tivoli Continues Data Protection	3.1.0.41	9876-5432-1234-5678

14/05/2008 9:13 PM

Figure 7-37 Software Inventory report with no highlighted item

7.4 Disaster recovery and validation

In this scenario, we demonstrate how to use the TADDM versioning feature and comparison report to recover from a major system outage and reinstate the service level to where it was before the outage.

7.4.1 Versions

Versions are snapshots of the current infrastructure. They are read-only views of the entire topology.

One of the best practices in IT System management is to create a snapshot of the infrastructure Configuration Items (CIs) before any major IT change. This snapshot will help to restore the environment to its pre-change state if the change is problematic and causes a major reduction in the level of service or even a complete loss of the level of service.

The way to create a snapshot in TADDM is by creating a version before any major change, such as a new IT asset, is added into your current IT environment or a new application is deployed from the development and testing environment to production. You can then use these versions in the “Comparison report”, which compares CIs of the same type between versions.

Version operations

In this section, we discuss the operations that you can perform with versions.

Creating versions

From the Product Console, go to the **Discovery** → **Versions** view, and select **Create**. Enter a version name, and click **OK**.

Note:

- ▶ We suggest that you have naming rules for version names to avoid possible confusion later; for example, use the date as part of the version name. A version ID is generated in TADDM whenever a new version is created. The current version has an ID value of 0.
- ▶ The discovered data for the new version is in the CMDB database and is stored under the archive user schema, which is the archive user specified during the TADDM installation.

Viewing version data

Go to the Versions view, select a version, and click **View**. The view opens in a new window, and the version is shown in the title bar of the new window as shown in Figure 7-38.

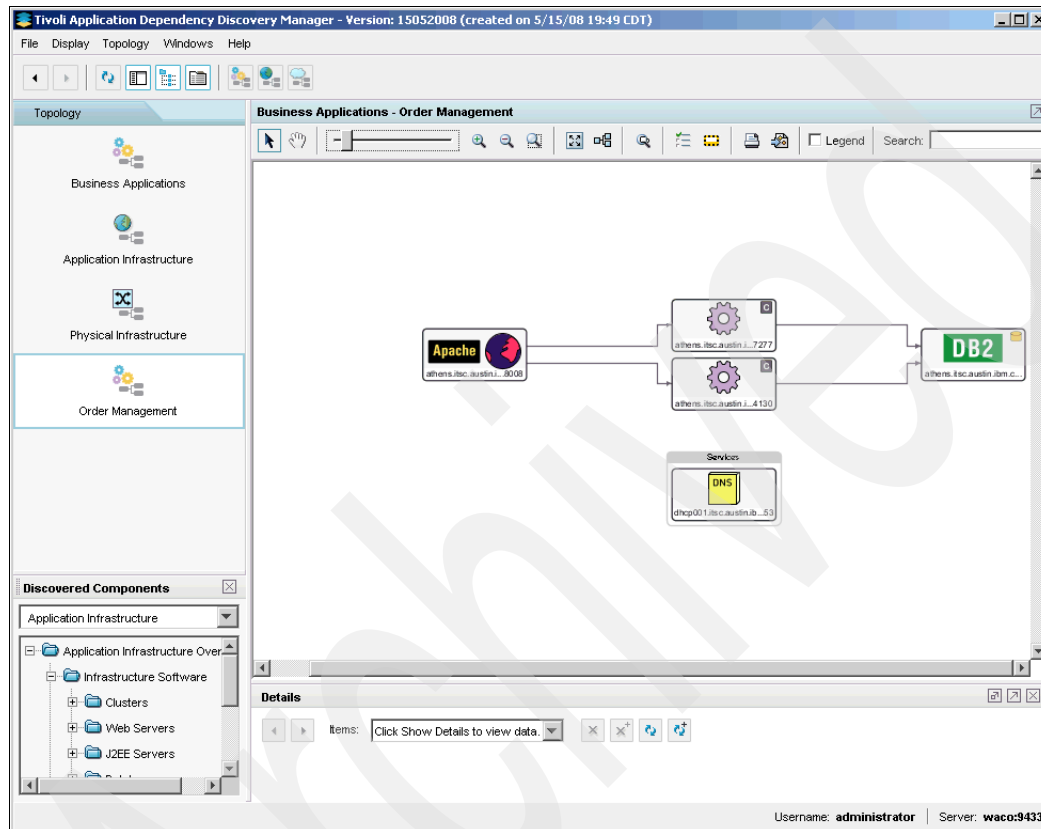


Figure 7-38 Version view

Now, let us assume that XYZ had a major hardware problem, and they had to reinstall all of their business applications from the beginning on new hardware. The XYZ IT team wants to configure the new system with the same settings that were used before the outage.

Assuming that a snapshot of the infrastructure was taken using the TADDM versioning capability, the XYZ IT team can easily configure their new system to the original settings by following these steps:

1. Run a Level 3 discovery on the new environment. This step discovers the IT environment as installed with the configurations set to the default values.

- The XYZ IT team runs the TADDM Comparison report, which guides them through the process of reconfiguring the environment to the state where it was before the hardware problem. To run the report on the component that needs to be reconfigured, right-click the component in the Topology View and choose **Compare Across Versions**, as shown in the Figure 7-39.

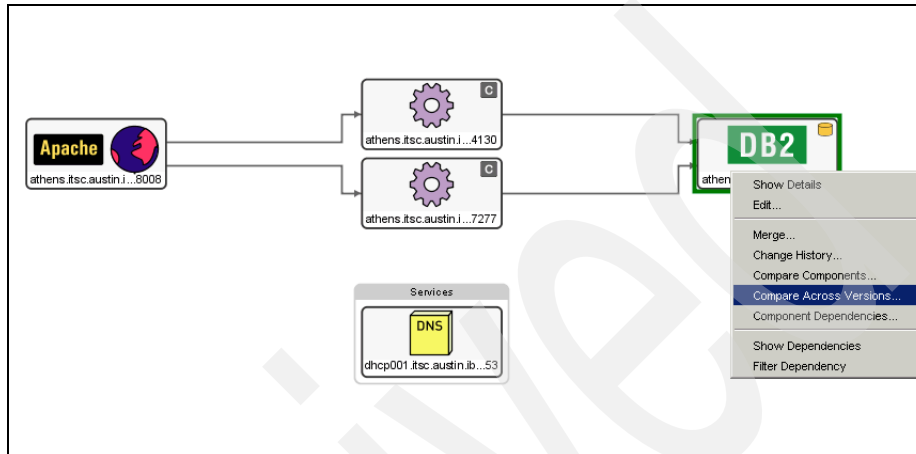


Figure 7-39 Selecting the Comparison report from the Topology View

- In the Included Components box, keep only the current version and the version number against which you want to compare, and set that version as key, as shown in Figure 7-40.

Included Components	
Name	Version
athens.itsc.austin.ibm.com:DB2	Current
athens.itsc.austin.ibm.com...	16052008 (5/16/08 16:41 CDT)

Figure 7-40 Selecting the versions for the comparison report

4. Click **Run Report**, and the result is shown in Figure 7-41.

Component Comparison: Results		
	athens.itsc.austin.ibm.com:DB2 - Versio...	athens.itsc.austin.ibm.com:DB2 - Version:0
DBM Config Values		
NUM_POOLAGENTS		
Value	100(calculated)	200(calculated)
INTRA_PARALLEL		
Value	YES	NO
MAXAGENTS		
Value	200	400

Figure 7-41 DB2 Comparison report example

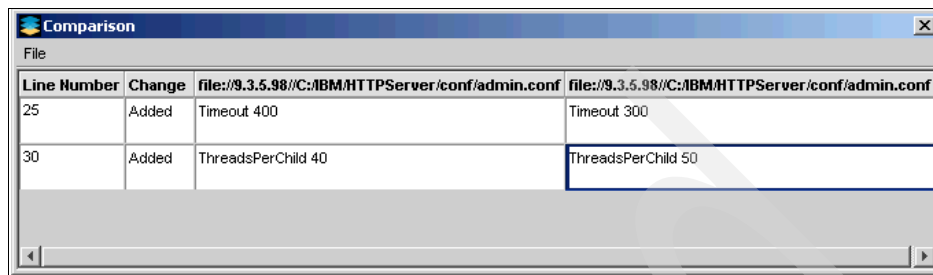
The values in black belong to the version that was selected as key. The values highlighted in red belong to the current configuration. The IT team now can tell which configuration changes they need to make in DB2 to make the configuration the same as it was before the outage.

The IT team can run this comparison report on other components of the business application, such as the Apache server, which is shown in Figure 7-39 on page 336. Figure 7-42 shows the results.

Component Comparison: Results		
	athens.itsc.austin.ibm.com:8008 - Version:2	athens.itsc.austin.ibm.com:8008 - Version:0
Config Contents		
/admin.conf		
Content		
Checksum	PRTGk3fUnT4vXnHmihmhQ==	arfyvylatEnx4f1fFm8N30g==
Last Modified	1210974441921	1210978180562

Figure 7-42 Apache Server Comparison report example

The report in Figure 7-42 on page 337 shows that the content of admin.conf differs between the versions. If you then click the linked entry that is highlighted in blue, you get the reason behind the difference, as shown in Figure 7-43.



Line Number	Change	file://9.3.5.98/C:/IBM/HTTPServer/conf/admin.conf	file://9.3.5.98/C:/IBM/HTTPServer/conf/admin.conf
25	Added	Timeout 400	Timeout 300
30	Added	ThreadsPerChild 40	ThreadsPerChild 50

Figure 7-43 Apache Comparison report details

7.5 Root cause analysis with tracking changes

In this scenario, we illustrate how to use the Application Topology Views, as well as change history reports, to identify the root cause of the service level degradation.

Suppose that the IT team of XYZ has noticed a reduction in the Order Management application performance. The IT team can use the TADDM Change History report feature to identify what was changed in the environment that might have caused the performance degradation. To generate a change history of a business application, follow these steps:

1. Run a discovery on the machine or machines that host the business application in question in order to capture the latest configuration settings.
2. From the Business Application Topology View, right-click the Order Management application, and choose **Change History**.
3. On the Change History panel, verify that the Date Range includes the expected time frame during which the performance degradation must have started to occur, as shown in Figure 7-44 on page 339.

Change History

Date Range
Set Date by: ☒ Relative Timeframe ☐ Absolute Timeframe

From: Weeks

Include Components
Select By:

Components
Component Type:

Available Components

Name

Add >>
<< Remove

Included Components

Name	Version
Order Management	Current

Run Report

Figure 7-44 Change History settings

4. Click **Run Report**. The result is shown in Figure 7-45.

Change History: Results

Component	Type	Change	Date	Attribute	Old Value	New Value	Id
Order Management	Application	Created	5/16/08 16:22 CDT				11C4B4C71C26...
Order Management	Application	Created	5/16/08 16:22 CDT				11C4B4C71C26...
Order Management	Application	Updated	5/16/08 18:25 CDT				11C4B4C71C26...
athens.itsc.austin.ibm.com:DB2	Db2Instance	Updated	5/16/08 18:25 CDT				CE875C215E4F...
DIAGLEVEL	Db2InstanceConfig/value	Updated	5/16/08 18:25 CDT	value	3	4	BED04E612DBA...

Figure 7-45 Change History report for Order Management business application

The report shows that the DB2 diagnostic error capture level increased on the 16th of May from level 3 (all errors and warnings) to level 4 (all errors, warnings, and informational messages). This increase accounts for the performance degradation, because to the host machine hard disk became busier with logging extra informational data.



Part 4

Performance and Troubleshooting Considerations

In this part we focus on performance and troubleshooting considerations for Tivoli Application Dependency Discovery Manager.

Performance considerations

Most complex software packages have a number of parameters that affect their performance, and IBM Tivoli Application Discovery and Dependency Manager (TADDM) is no exception. The default settings that TADDM ships with were found to generally perform well. However, every environment is different, so some adjustment of these settings might be needed to deliver optimal discovery performance.

In this chapter, we discuss:

- ▶ “Performance improvements in TADDM V7.1” on page 344
- ▶ “Discovery tuning” on page 344
- ▶ “Tuning storage performance” on page 350
- ▶ “Caching user interface views” on page 351
- ▶ “Database considerations” on page 355
- ▶ “Java Virtual Machine settings” on page 361
- ▶ “Log settings for production” on page 363
- ▶ “Maintenance” on page 363

8.1 Performance improvements in TADDM V7.1

As clients' discovered environments got larger, the amount of data in client databases increased, and what clients wanted to do with this information became more complicated, performance necessarily became a focus in TADDM 7.1. Table 8-1 shows the performance increases reported by product management for TADDM 7.1 over TADDM 5.1.3.

Table 8-1 Improvements over TADDM 5.1.3

Percentage improved	Function
13%	Discovery overall
60%	User interface
30%	WebSphere sensor
89%	DB2 sensor
93%	Bulk load
40%	eCMDB sync

Of course, performance is specific to the hardware, operating system, size of your environment, and database configuration, so specific environments can differ.

One of the biggest performance enhancements is “*view caching*”, which we discuss in 8.2, “Discovery tuning” on page 344.

Performance continues to be an area of improvement for the 7.1 release of TADDM. For example, with 7.1 Interim Fix 4, business application topology builds (the slowest topology build in 7.1) went from 38+ hours to 3 minutes in one particular client environment.

8.2 Discovery tuning

The TADDM GUI provides information that is useful for monitoring discovery performance. Figure 8-1 on page 345 shows the Discovery Overview window.

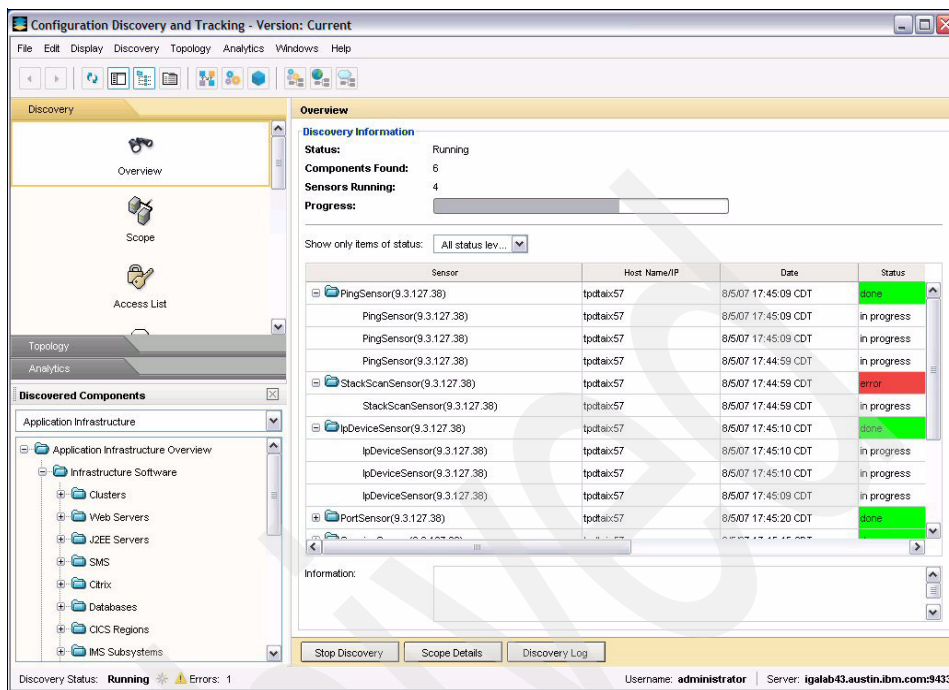


Figure 8-1 Discovery status

The sensors that are displayed on this window will have one of three states: *done*, *error*, and *in progress*, as shown in the Status column of Figure 8-1. The sensors in the done or error states are no longer being processed. To monitor performance, it is only useful to look at the in progress sensors.

The in progress sensor is in one of three stages of execution. These stages are shown in the Description column of the panel. You can group the sensors on the display by these stages, by clicking the **Description** column heading.

The first stage is *started*. For a sensor, the started stage is in the process of discovering one or more configuration items (CIs). Refer to Figure 8-2.

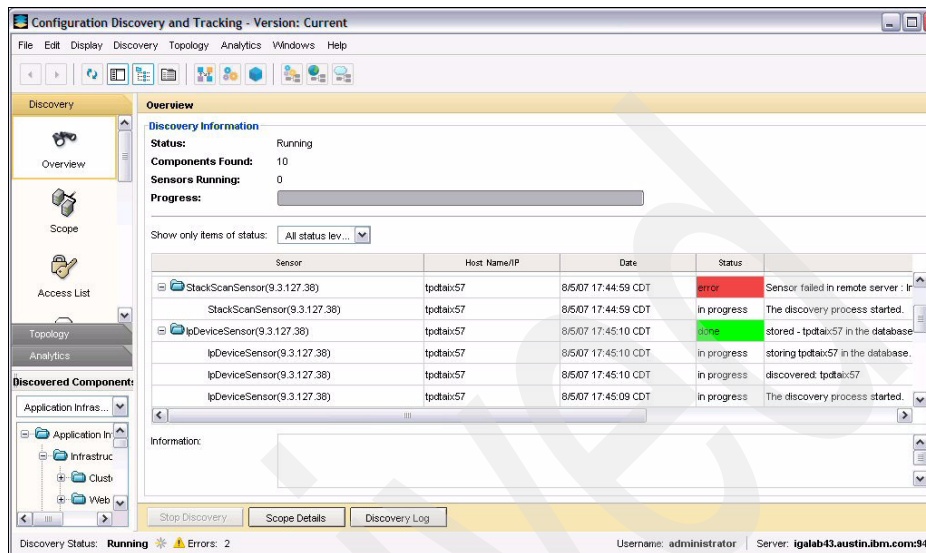


Figure 8-2 Discovery in progress

The second stage is *discovered*. For a sensor, the discovered stage has finished discovering one or more CIs, but it is still waiting for its results to be saved in the datastore. Refer to Figure 8-3 on page 347.

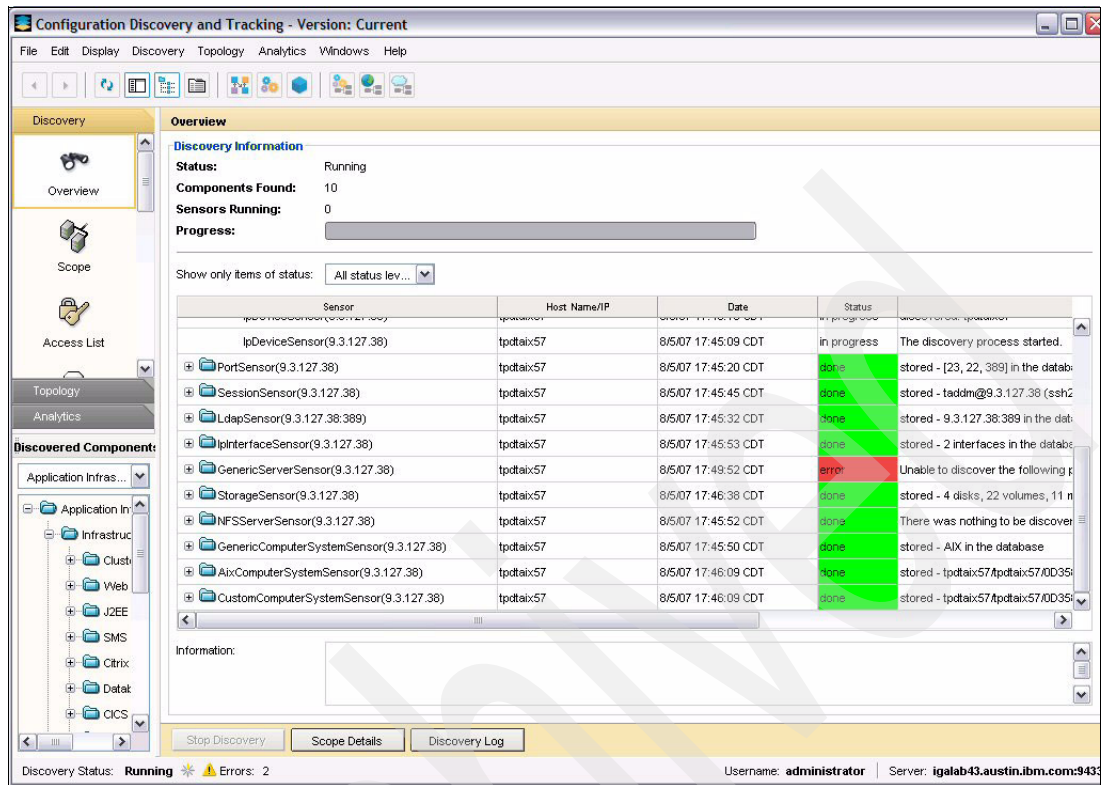


Figure 8-3 Discovery done

The final stage of an in progress sensor is *storing*. As the name implies, sensors in this stage are having their results persisted in the database. Refer to Figure 8-4 on page 348.

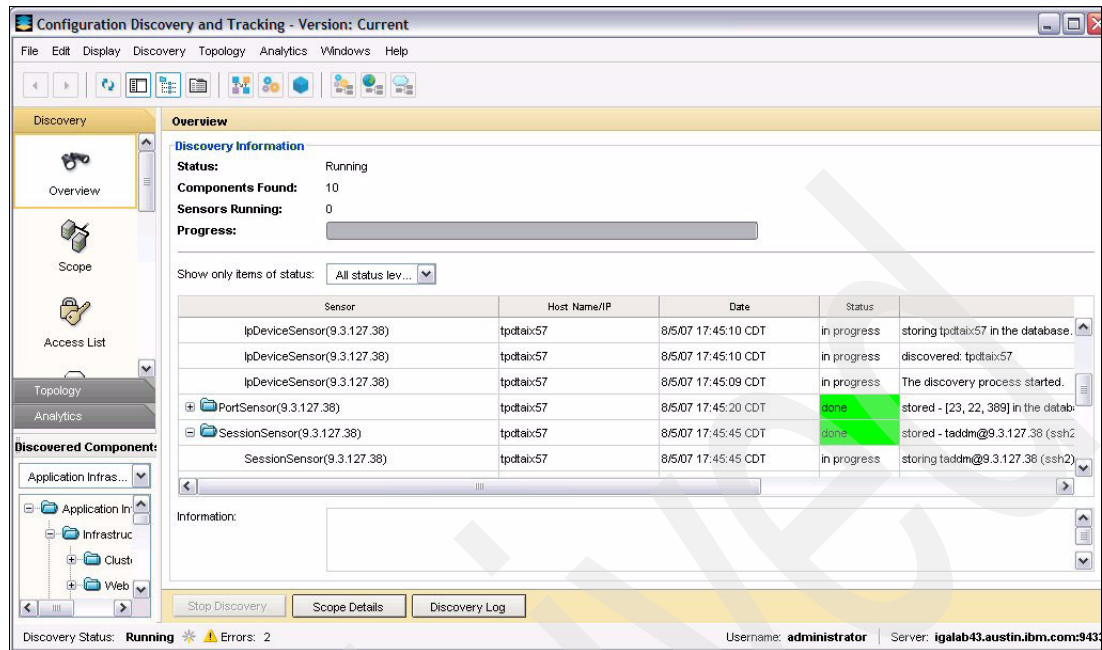


Figure 8-4 In progress status

By observing a discovery run and comparing the number of in progress sensors that are in the started stage to the number of in progress sensors in the discovered or storing stages, you can make an assessment about whether attribute discovery is faster or slower than attribute storage for a particular environment.

The attribute discovery rate is the area with the most potential for tuning. Most of the TADDM modifiable parameters are contained in the collation.properties file:

```
<TADDM_install_dir>/dist/etc/collation.properties
```

This collation.properties file is a Java properties file with a list of keys and value pairs separated by an equal (=) sign. In this file, the property with the most impact on performance is the number of discovery worker threads:

```
# Max number of discovery worker threads
com.collation.discover.dwcount=32
```


Important: The previous recommendation for the `dwcount` setting was 16, but as hardware performance increases over time, the number of discovery worker threads that hardware can support has also increased. In general, if you have CPU cycles available during discoveries, you can increase this setting. (You will also want to up the `topopumpcount` setting.)

Provided the server has sufficient spare capacity, you can increase this setting, which allows more sensors to run in parallel. Two similar properties are:

```
# Max number of seed creators
com.collation.discover.sccount=2
# Max Number of Agent selectors
com.collation.discover.ascount=1
```

However, do *not* change these values. In most cases, to discover the attributes of a CI, a sensor requires a Secure Shell (SSH) or Windows Management Interface (WMI) session with the host computer where the CI resides. To improve performance by reducing the session creation overhead, these sessions are pooled and cached. The default pool sizes are sufficient in most cases. However, when they are not large enough, they can limit the discovery rate. To monitor for this condition, change the following property in the `collation.properties` file from false to true:

```
com.collation.platform.session.ExtraDebugging=false
```

As with all changes to the `collation.properties` file, you must restart the server for the change to take effect. After this change takes place and a discovery is run, you can search the `DiscoverManager` logs for session pool wait time issues. The text to search for in the logs is seconds for pool lock. Example 8-1 is an example of performance degradation that is caused by session pool contention from the `DiscoverManager.log` file.

Example 8-1 Performance degradation

```
2006-08-04 16:11:50,733 DiscoverManager [DiscoverWorker-34]
WindowsComputerSystemAgent(192.168.16.181) INFO
session.SessionClientPool - Session client [3x
ssh2:/admlxz@151.179.84.85]#9612508 waited 158.682 seconds for pool
lock
```

If the log shows excessive waiting for a session from the pool, you can increase the pool size. There are two ways to increase the pool size. First, you can globally increase the per host session pool by changing the following property in the `collation.properties` file:

```
com.collation.platform.session.PoolSize=3
```

It is, however, unlikely that there is contention for sessions for all of the hosts or even most of the hosts in the environment. The contention is probably restricted to a smaller number of larger hosts for which a great many sensors run. TADDM has a concept of a *Scoped Property*, which means that many of the properties in the `collation.properties` file can have one value for general targets and another value for specific targets. You can create a scoped property by appending an IP address or a scope name to the property name, similar to the following example:

```
com.collation.platform.session.PoolSize.10.10.250.1=20
```

In this case, for all hosts, other than 10.10.250.1, the `PoolSize` is 3, but for 10.10.250.1, the `PoolSize` is 20. By looking at the log messages, such as the log message in Example 8-1 on page 349, it is easy to determine for which hosts the default session `PoolSize` is insufficient and make the appropriate changes to the `collation.properties` file.

A related setting is the Gateway pool size. It sets the number of sessions allowed between the server and the Windows Gateway. The property is:

```
com.collation.platform.session.GatewayPoolSize=10
```

In environments that consist mainly of Windows computer systems, adjust this property upwards to be equal to the number of Discover worker threads.

8.3 Tuning storage performance

The second major area for tuning is storage. Storage of the discovery results can be the discovery performance bottleneck if the number of sensors in the storing state hovers around the value of the property:

```
com.collation.discover.observer.topopumpcount
```

This property is the number of parallel storage threads, which is one of the main settings for controlling discovery storage performance; however, you must adjust it carefully. Setting it too high can lead to contention, if more than one thread is trying to update the same object at the same time. Setting this property too high can also increase contention to the point that overall throughput is decreased. Contention is more likely to occur when a small discovery scope is used or when a large number of the discovered CIs are from a single server. If either of these scenarios is commonplace in the environment of a given TADDM Server, set this property to a small number. We recommend even setting this property as low as one.

In general, however, `topopumpcount` needs to be less than the `dwcount` (refer to 8.2, “Discovery tuning” on page 344), because storage tends to be faster than

discovery. Experiment with settings between 1/2 and 2/3 of the dwcount setting to find the optimal setting for your environment.

Note: There is a property in collation.properties that was previously used for integration with other products:

```
com.collation.topomgr.generateExplicitRelationship=true
```

This property is no longer needed by any third-party integrations and needs to be set to false. Leaving this setting at true has a significant adverse affect on performance without any benefit.

8.4 Caching user interface views

View caching was introduced simultaneously in TADDM 7.1 and TADDM 5.1 in order to reduce the load time of commonly used but not regularly updated user interface views. TADDM can be configured to use either a disk cache or an in-memory cache; however, we recommend that you use the disk cache so that you can reuse cached views even after restarting TADDM.

8.4.1 Understanding caching

There are two types of views that TADDM caches: tree views and graph views. *Tree views* are the expanding navigation list that shows up on the bottom left side of the TADDM UI. *Graph views* are the Topology Views that show the objects in graphical format.

Both views can be cached so that the time that it takes to load is substantially reduced. Caching the graph views shows a bigger impact in your user interface usability, because topology graphs take substantially longer to build than the tree views. You configure TADDM to just cache specific types of each of these views.

File structure of the TADDM cache

Views are cached in the TADDM home directory under dist/var/viewmanager. Views and view data are cached as “.ser” files. The ser suffix stands for “serialized object” and is a serialized Java object. The viewmanager directory has a list of the views in the cache (views_in_cache.ser) and a view mapping file (object_view_mappings.ser).

Under the viewmanager directory are two additional directories: graph and tree. Each of these directories contains the actual serialized views that are cached, as shown in Example 8-2 on page 352.

Example 8-2 Directories

```
bash-3.00$ /opt/cmdb/dist/var/viewmgr/graph

bash-3.00$ ls -l
total 928
-rw-r----- 1 cdtadm staff 14951 Feb 20 11:16
12FD02EF766535D78073D671AEB5DF1C_Application Physical Infrastructure
Topology_0.ser
-rw-r----- 1 cdtadm staff 14386 Feb 20 11:18
19A3822D0A40393EA777B1FA159DDB26_Application Software Infrastructure
Topology_0.ser
-rw-r----- 1 cdtadm staff 12858 Feb 20 11:16
24BF9BEAD9FA3DAE93F2C5D14FDBE51E_Application Software Infrastructure
Topology_0.ser
-rw-r----- 1 cdtadm staff 16523 Feb 20 11:18
5A32F36C0BFF3E3284A461CBB79281E8_Application Software Infrastructure
Topology_0.ser
-rw-r----- 1 cdtadm staff 20551 Feb 20 11:16
66F23215BF5B3B1F91F33C102097D858_Application Physical Infrastructure
Topology_0.ser
-rw-r----- 1 cdtadm staff 14957 Feb 20 11:19
66F23215BF5B3B1F91F33C102097D858_Application Software Infrastructure
Topology_0.ser
-rw-r----- 1 cdtadm staff 17120 Feb 20 11:16
6B0C7F20895630A98108568F191CAD4D_Application Physical Infrastructure
Topology_0.ser
-rw-r----- 1 cdtadm staff 12858 Feb 20 11:16
6B4A4F6600D23C9EABB5416DC86846DB_Application Software Infrastructure
Topology_0.ser
-rw-r----- 1 cdtadm staff 17113 Feb 20 11:16
8BBC44A669153395B04FCA6753BB538E_Application Physical Infrastructure
Topology_0.ser
-rw-r----- 1 cdtadm staff 14489 Feb 20 11:18
917F83EFF70E3A7CA599EBE92A9C26EF_Application Software Infrastructure
Topology_0.ser
-rw-r----- 1 cdtadm staff 17098 Feb 20 11:16
B366A37A1B95373CABDC552FC469A20F_Application Physical Infrastructure
Topology_0.ser
-rw-r----- 1 cdtadm staff 30360 Feb 20 11:17
BADCD710E7F633FA834A073F40BCF381_Application Physical Infrastructure
Topology_0.ser
-rw-r----- 1 cdtadm staff 17311 Feb 20 11:19
C4D9D5A852873E008879FF3213C8852B_Application Software Infrastructure
Topology_0.ser
```

```
-rw-r----- 1 cdtadm staff 18500 Feb 20 11:16
DC524696FD393DAFA1CB3581968E4284_Application Physical Infrastructure
Topology_0.ser
-rw-r----- 1 cdtadm staff 20552 Feb 20 11:16
EFF653E260553CA6AD05E3B9E1CD22D9_Application Physical Infrastructure
Topology_0.ser
-rw-r----- 1 cdtadm staff 14957 Feb 20 11:18
EFF653E260553CA6AD05E3B9E1CD22D9_Application Software Infrastructure
Topology_0.ser
-rw-r----- 1 cdtadm staff 142559 Feb 20 11:15
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF_Business Application Topology_0.ser
-rw-r----- 1 cdtadm staff 11290 Feb 20 11:16
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF_Physical Infrastructure Topology_0.ser
```

When to enable caching

View caching is already enabled in the TADDM product. View caching needs to remain enabled in all production environments. Do not disable view caching.

8.4.2 Configuring caching

The default settings are shown in Example 8-3. The first three settings, which are in your `etc/collation.properties` file, enable view caching by default. You must change the fourth setting from “true” to “false”.

Example 8-3 Base recommended view cache settings

```
com.collation.view.cache.disk=true
com.collation.view.cache.optimization.enabled=true
com.collation.view.cache.disk.path=var/viewmgr
com.collation.view.prebuildcache.graph.appinfrastructure.enabled=false
```

The rest of the `com.collation.view.prebuildcache.graph*` settings can be true.

Note: The following setting ships set to false, because there was a bug that was introduced with TADDM V7.1 that caused the business application views to be rebuilt too often for the cache to be of benefit:

```
com.collation.view.prebuildcache.graph.appinfrastructure.enabled=false
```

As for the location of the cache, these settings are important:

```
com.collation.view.cache.disk=true
```

Store the cache on the disk (as opposed to storing it in memory):

```
com.collation.view.cache.disk.path=var/viewmgr
```

Without a leading “/”, the location of the cache files is under the \$COLLATION_HOME/dist directory. With a leading “/”, the location of the cache files can be anywhere on your filesystem.

Note: If views already exist in the cache, they will continue to be built even if you turn the properties setting, which controls that view to false, in order to avoid “out of sync” views. If you want to turn off a cache setting that was previously on, clear out the cache after you modify the collation.properties setting.

Reducing the amount of logging by View Manager

In addition to enabling the view cache, you also need to reduce the amount of log messages for the View Manager.

Add the following line to your collation.properties file:

```
com.collation.log.level.com.collation.proxy.viewmgr.ViewManagerUtil=INFO
```

This line limits the number of messages that are printed by TADDM in the ViewManager log.

After enabling caching, you will not see an immediate improvement, because it is a cache. Therefore, you will see an improvement over time as you navigate the TADDM user interface and the cache is populated with your regularly used views.

8.4.3 Maintaining the cache

If you have too many views cached in TADDM, the time that it takes to update all of them might be longer than the time gained from having them. A good general rule is that if you have more than 20 files in the view cache, you need to clean out the cache.

To clean out the cache, delete the viewmanager directory, the graph and tree subdirectories, and all of their contents.

Note that you cannot selectively remove files from the cache, because the views_in_cache.ser file contains a list of the views that are expected to be available in the cache.

We also recommend that you review the readme files for any new interim fixes or fix packs, because the view caching technology might change, and you want to adjust your settings accordingly.

8.5 Database considerations

Next, we discuss a few database-related considerations.

8.5.1 Database indexes

There are a number of database indexes that you need to apply manually at the end of an installation. In many environments, the clients have never applied the database indexes. If you are experiencing performance problems or have any doubt whether these indexes were applied, you can reapply them.

Back up the TADDM Database before completing this step. From a command prompt, use one of the following procedures to run the commands:

- For Linux, Solaris, AIX, and Linux on System z operating systems, run:

```
cd <dist>/bin dbupdate.sh 71Indexes.sql > 71Indexes.out
```

If the database is DB2, also run:

```
cd <dist>/bin dbupdate.sh 71Db2Indexes.sql > 71Db2Indexes.out
```

- For Windows operating systems, run:

```
cd <dist>\bin dbupdate.bat 71Indexes.sql > 71Indexes.out
```

If the database is DB2, also run:

```
cd <dist>/bin dbupdate.bat 71Db2Indexes.sql > 71Db2Indexes.out
```

On Oracle systems, you can safely ignore the following error, because the error message indicates that the index exists:

```
ORA-01408: Such column list already indexed
```

On DB2 systems, you can safely ignore the following error, because the error message indicates that the index exists:

```
SQL0605W The index was not created, because the index <name> already exists with the required description. SQLSTATE=01550 12.
```

8.5.2 Database settings: DB2

Use the scripts in dist/etc/db2 to get the current recommended settings. If possible, use these scripts to create the database or as a reference when you create the database.

If your database was not created using the scripts, we recommend that you review the dist/etc/db2/upd_db_cfg.sql script to make sure that you have all of the appropriate configuration settings. Many of these configuration settings affect performance. The settings can be applied at any time after the installation.

Example 8-4 The upd_db_cfg.sql script

```
UPDATE DATABASE CONFIG FOR CMDB USING
    DBHEAP 1800
    DFT_DEGREE ANY
    LOGBUFSZ 1024
    LOCKLIST 1500
    NUM_IOCLEANERS 9
    NUM_IOSERVERS 17
    LOGFILSIZ 4096
    LOGPRIMARY 6
    LOGSECOND 20
    APP_CTL_HEAP_SZ 1024
    SORTHEAP 1024
    AVG_APPLS 3
;
CONNECT TO CMDB;

ALTER BUFFERPOOL IBMDEFAULTBP
    SIZE 4000
;
ALTER BUFFERPOOL BUF8K
    SIZE 2000
;
ALTER BUFFERPOOL BUF32K
    SIZE 2000
;
```

The script updates the database configuration with the following settings:

- ▶ DBHEAP: Increased support to match other parameter settings
- ▶ DFT_DEGREE: Enables intra-partition parallelism
- ▶ LOGBUFSZ: Memory used as the log buffer (this value varies based on the database size and activity)

- ▶ LOCKLIST: Space used for locking
- ▶ NUM_IOCLEANERS: Used to clean updated pages in the bufferpool
- ▶ NUM_IOSERVERS: Used to handle the I/O requests to the database (this setting is usually a number that is 2 or 3 higher than the number of physical disk drives on your system)
- ▶ LOGFILSIZ: Size of the DB2 log files
- ▶ LOGPRIMARY: Number of log files
- ▶ NEWLOGPATH: The directory where log files will be created (this path must exist)
- ▶ APP_CTL_HEAP_SZ: Application heap size
- ▶ SORTHEAP: Memory used for an individual sort

The script then updates the bufferpool size, which varies based on the size of the database and the amount of physical memory on the database system.

So, for instance, you can run the following commands to increase the size of the transaction logs:

```
db2 connect to <database name>
db2 update db cfg using LOGFILSIZ 4096
db2 update db cfg using LOGPRIMARY 6
db2 update db cfg using LOGSECOND 20
db2 disconnect <database name>
db2 quit
```

You can adjust the other settings in the same manner.

In addition to setting these database parameters, the following DB2 environment setting provides added performance improvements:

```
db2set DB2_EVALUNCOMMITTED=YES
```

There is a caution about this setting. It is an instance-wide setting and alters the way that the concurrent transactions are processed. Although this change is good for TADDM, it is potentially bad for other databases in the instance.

8.5.3 Initial database statistics on DB2

After installing TADDM, you need to run statistics on the schema. You will want to run the `runstats_db2_catalog` script after installing and starting TADDM for the first time. (The schema is not created until the initial start of TADDM.)

The runstats_db2_catalog script is located on the TADDM Server in \$COLLATION_HOME/support/bin/runstats_db2_catalog.sql.

You can use that tool by executing the following commands:

```
su - <db2 instance owner>
db2 connect to <cmdb>
db2 -xtf runstats_db2_catalog.sql
```

8.5.4 Running statistics

We recommend that you update the database statistics whenever there is a significant change to the contents of the database, for example:

- ▶ After a discovery
- ▶ After a bulk load
- ▶ Other activities that change the database significantly

We recommend that you run statistics on a regularly scheduled basis, at least weekly.

The DB2 query optimizer benefits from having up-to-date statistics on the TADDM tables (for example, it helps to estimate how much buffer pool is available at run time). There is a program in the <TADDM_install_dir>/dist/support/bin called gen_db_stats.jy. This program outputs the database commands for either Oracle or DB2 to update the statistics on the TADDM tables. The following syntax, where <tmpdir> is a directory where you can create this file, shows how you can use the program:

```
cd <TADDM_install_dir>/dist/support/bin
./gen_db_stats.jy ><tmpdir>/TADDM_table_stats.sql
```

After running gen_db_stats.jy has completed, copy the file to the database server and run one of the following commands:

```
DB2: db2 -tvf
x0racle: sqlplus
```

8.5.5 Bufferpool

For db2 bufferpool settings, you want to aim for a 90% hit rate.

Before you can establish the correct number for the bufferpool size, you must manage the tablespaces and the tables. You can manage the tablespaces and the tables with the following buffer.sql script.

First, copy the script into /home/<instance_name>:

```
su <instance_name>  
db2 -tf buffer.sql
```

The script in Figure 8-5 creates the buffer.out file. Evaluate this file or ask your technical support for assistance.

```
select substr(a.tbSPACE,1,20) as tbSPACE, a.datatype,  
       substr(b.bpname,1,20) as bpname, b.npages,  
       a.pagesize, b.pagesize  
from sysibm.sysbufferpools b  
full join sysibm.systablespace a  
  on a.bufferpoolid = b.bufferpoolid  
order by 2, a.pagesize  
;  
  
select substr(tbSPACE,1,20) as tbSPACE, count(*)  
from sysibm.systables  
where type = 'T'  
group by tbSPACE  
;
```

Figure 8-5 Script to create the buffer.out file

The output in the buffer.out file looks similar to Figure 8-6 on page 360.

buffer.out - WordPad

File Edit View Insert Format Help

10 record(s) selected.

SYSCATSPACE	A	IBMDEFAULTBP	20000	4096	4096
SYSTOOLSPACE	A	IBMDEFAULTBP	20000	4096	4096
USERSPACE1	A	IBMDEFAULTBP	20000	4096	4096
USERSPACE3	A	BUF8K	6000	8192	8192
USERSPACE2	A	BUF32K	7000	32768	32768
LARGESPACE3	L	BUF8K	6000	8192	8192
LARGESPACE2	L	BUF32K	7000	32768	32768
TEMPSPACE1	T	IBMDEFAULTBP	20000	4096	4096
LARGETEMP3	T	BUF8K	6000	8192	8192
LARGETEMP2	T	BUF32K	7000	32768	32768

10 record(s) selected.

TBSpace 2

SYSCATSPACE	94
SYSTOOLSPACE	3
USERSPACE1	1862
USERSPACE2	26
USERSPACE3	222

5 record(s) selected.

For Help, press F1

Figure 8-6 Sample buffer.out file

For the output in Figure 8-6, we recommend the following settings:

IBMDEFAULTBP SIZE 60000
 BUF8K SIZE 9000
 BUF32K SIZE 3000

Use the following commands to implement the settings:

db2 ALTER BUFFERPOOL IBMDEFAULTBP SIZE 60000
 db2 ALTER BUFFERPOOL BUF8K SIZE 9000
 db2 ALTER BUFFERPOOL BUF32K SIZE 3000

Use the following command to make sure that the settings took effect:

db2 select bpname, npages from syscat.bufferpools

If you need further bufferpool tuning for performance reasons, and if you have already implemented the previous settings, monitor the database to see the hit ratios, and then, adjust accordingly.

8.6 Java Virtual Machine settings

You can modify the arguments with which the TADDM Java Virtual Machines (JVMs) run. Most of the time, the default settings are sufficient.

8.6.1 Modifying the JVM arguments

There are a few additional parameters that you can set that affect performance but are not directly related to discovery. Because TADDM is a Java application, you can modify the various Java Virtual Machine (JVM) parameters from their default values. There are two ways to modify the various Java Virtual Machine (JVM) parameters.

To change an existing JVM option to another value, edit the following file:
<TADDM_install_dir>/dist/deploy-tomcat/ROOT/WEB-INF/cmdm-context.xml

If eCMDB is in use, modify the following file instead:
<TADDM_install_dir> /dist/deploytomcat/ROOT/WEB-INF/ecmdb-context.xml

To edit one of these files to change the settings for one of the TADDM services, first find the service in the file. The following code is an example of the beginning of a service definition in the XML file:

```
<bean id="Discover"
class="com.collation.platform.jini.ServiceLifecycle" initmethod="
start" destroy-method="stop">
<property name="serviceName">
<value>Discover</value>
```

Within the definition, there are elements and attributes that control the JVM arguments, for example:

```
<property name="jvmArgs">
<value>-Xms8M;-Xmx512M;
-Djava.nio.channels.spi.SelectorProvider=
sun.nio.ch.PollSelectorProvider</value>
</property>
```

You can set the JVM arguments as a semi-colon separated list in the <property name="jvmArgs">.

You can also set JVM-wide settings in the `collation.properties` file. Add a property `"com.collation.jvmargs.ibm"` or `"com.collation.jvmargs.sun"` to set a system-wide JVM argument.

8.6.2 Java Max memory

Prior to Version 7.1, we often recommended to clients that they increase the amount of memory that the JVM used as their discoveries or reports caused TADDM to fail with an `OutOfMemory` condition. Increasing the available memory without addressing the root cause of the failure only delayed the problem. In Version 7.1, most of those situations have been rectified, but it is still possible to fail with an `OutOfMemory` condition. If this type of failure happens, we strongly recommend that you gather the `javacore` and `heapdump` files and open a problem management record (PMR) with support. Work with support to identify and fix the root cause before you increase the amount of memory available to the JVM.

After you have gathered the correct problem determination documentation, however, you can increase the amount of memory in the system that is failing.

To confirm that the problem is caused by an `OutOfMemory` error, open the `javacore` files that are created during the time of the Discovery and are located in `dist/external/gigaspaces-4.1/bin`. Find the Error in the top several lines of the file, which might look similar to Example 8-5.

Example 8-5 Javacore

```
NULL
-----
-
OSECTION TITLE subcomponent dump routine
NULL =====
1TISIGINFO Dump Event "systhrow" (00040000) Detail
"java/lang/OutOfMemoryError" received
1TIDATETIME Date: 2007/10/29 at 12:13:43
1TIFILENAME Javacore filename: /tmp/javacore.20071029.121119.577910.txt
Then search in the file for the text 'servicename', which may look like
this:
1CICMDLINE /CDT/CMDB/dist/external/jdk-1.5.0-AIX-powerpc/jre/bin/java
-Dcom.collation.LogFile=/CDT/CMDB/dist/log/topology.log
-Dcom.collation.servicename=Topology
-Dcom.collation.home=/CDT/CMDB/dist
-Djava.security.policy=/CDT/CMDB/dist/etc/policy.all -Dcom
In this example Topology is the servicename, or the process that caused
the OutOfMemory error.
```

You can then find the Topology service in `cmdb-context.xml`, as described in section X, and up the `-Xmx` value in its `jvmArgs` property.

```
<property name="jvmArgs">
```

```
<value>-Xms768M;-Xmx1512M;-Dsun.rmi.transport.tcp.handshakeTimeout=0</value>
```

8.6.3 Java garbage collection

The Solaris JVM benefits from additional garbage collection settings.

Add the following settings for the `com.collation.jvmargs.sun` property in `collation.properties`:

```
XX:+UseConcMarkSweepGC -XX:+HeapDumpOnOutOfMemoryError
```

8.7 Log settings for production

Excessive logging can affect performance as well. We recommend that you use `DEBUG` level logging (refer to 9.2, “Installation logs” on page 369) while you are deploying and configuring your environment; however, you probably want to reduce the amount of logging when your environment is configured and stable.

We recommend a log level of `INFO`:

```
com.collation.log.level=INFO
```

8.8 Maintenance

There are several tasks that you either must or can perform on a regular basis to keep your TADDM environment running well.

8.8.1 Clearing out unknown servers

When Tivoli Application Dependency Discovery Manager discovers servers, sometimes there are running processes discovered that cannot be classified. These processes are stored as unknown server processes in the Configuration Management Database. After many discoveries are run, this list of unknown server processes begins to get large. While creating custom server templates, it is helpful to purge this list before running a discovery to aid in identifying the target processes that are being used to create custom server templates.

Additionally, when discovering multiple times with the wrong credentials, the unknown server report will fill up with these machines. With the right credentials, the server will no longer be unknown. Thus, the reason to purge the unknown server report to start again.

Suppose that the client has an Oracle instance called SAMPLE running on port xxxx with PID=yyyy. The first time that the client ran discovery (the client did not have the right credentials), SAMPLE was discovered as an unknown server with PID=yyyy. The second time that the client ran discovery (again, the client did not have the right credentials), PID was changed to zzzz. Therefore, SAMPLE was discovered again as a different unknown server with PID=zzzz. The third time, the client provided the right credentials, and therefore, SAMPLE was finally discovered as an Oracle instance and included in the list of Oracle instances.

As a consequence of this process, the database contains three instances of the same object. In reality, you might have tens of instances for each object in the unknown servers list, which makes the unknown server feature useless.

If a PID changes between discovery runs, the old runtime process remains in the Unknown Server report.

Clearing unknown servers with a restart

The easiest way to clear the unknown servers requires that you restart TADDM. You can clear the unknown servers list occasionally by setting "com.collation.topobuilder.purgeunknownservers=true" and discovering a single IP, such as the TADDM Server or the Windows gateway. It does not matter what the target of this discovery is, all unknown servers are purged.

Clearing unknown servers without a restart

If you want to clear the unknown servers without a restart, there is a utility available in the Tivoli Open Process Automation Library (OPAL) that can clear the unknown servers while the product is running. The utility is called the Purge Unknown Server Process utility and is available at the following URL:

<http://catalog.lotus.com/wps/portal/topal/details?catalog.label=1TW10CC1M>

After the utility is downloaded and made executable, you can run the utility with the administrator user name and password to purge your unknown servers.

8.8.2 Finding and applying fixes and updates

In this section, we discuss finding and applying TADDM fixes.

Determining your current product level

Fix pack levels are updates in the collation.properties file in the “com.collation.version” property.

Interim fix markers are labeled with the “efix” prefix. Look for the efix.* files in the dist/etc directory. When you unzip the efix packs so that they place all of the files in the correct directories, the fix will place a file in the /etc directory with a name to match the efix pack.

For example, assuming that you are using all of the default directories, if you apply efix.CMDB.5.1.0.20060815.zip, it will place a file named efix.20060815 in the /opt/IBM/CCMDB/CMDB/dist/etc directory.

The other efix packs also create efix.* files in the dist/etc directory.

Note that this procedure does not apply to testfix files. You must document the testfix file that you apply yourself, because there is no versioning on those files. We suggest that you maintain the file name, date stamp, and file size in a spreadsheet.

Determining the maintenance to apply

In general, interim fixes for the most recent version of TADDM are released once a month and fix packs are released once a quarter. You can find the most recent interim fixes and fix packs in the Downloads section of the TADDM support site:

<http://www-306.ibm.com/software/sysmgmt/products/support/IBMTivoliApplicationDependencyDiscoveryManager.html>

In general, you want to apply the most recent fix pack and interim fix to the fix pack.

Installing Interim fixes and fix packs

Fix packs are full product updates. They are allowed to change the database schema and add additional features. Before applying a fix pack, you need to back up your database and your TADDM installation.

Interim fixes are unzipped on top of existing install directions. You want to back up the entire dist directory before installing them.

Troubleshooting

This chapter provides a detailed look at troubleshooting IBM Tivoli Application Discovery and Dependency Manager (TADDM). Most of the troubleshooting focuses on issues that you might encounter during your initial discovery runs and helps you identify the cause of failures in sensors. We spend a considerable amount of time discussing Windows discovery, which is a cause of confusion and issues in many environments.

In this chapter, we discuss:

- ▶ “Log files” on page 368
- ▶ “Installation logs” on page 369
- ▶ “Problem determination tools” on page 369
- ▶ “Log and Trace Analyzer” on page 382
- ▶ “Specific scenarios” on page 386”

Note: The official support site is updated regularly with authorized program analysis report (APAR) descriptions, fixes, and technotes. If you run into a problem that is not covered by this chapter, we recommend that you search the support site:

<http://www-306.ibm.com/software/sysmgmt/products/support/IBMTivoliApplicationDependencyDiscoveryManager.html>

9.1 Log files

The most useful troubleshooting tool in TADDM is the log files. TADDM logs are based on an industry standard log4j format. Each separate Java virtual machine, as well as certain components, creates its own log files. Log files can be configured for varying amounts of data.

The logs are found in the dist/log directory in a TADDM directory. Here are the most important log files:

- ▶ control.log: Lists the activity of the bootstrap of the initial services
- ▶ services/DiscoverManager.log: Lists the activity of all of the sensors. If you do not have split sensor logging enabled, you can use this log to troubleshoot sensor-specific discovery problems.
- ▶ local-anchor*.log: Lists the activity of Weblogic and WebSphere discoveries
- ▶ error.log: Lists errors from all subsystems
- ▶ tomcat.log: Lists the activity of the major server and the startup of all services
- ▶ services/TopologyManager.log: Interface between all other components and the datastore
- ▶ cdm.logL Web portal logs are located here
- ▶ discover.log: Contains messages from the Discover jini service
- ▶ events-core.log: Contains messages from the events core jini service
- ▶ discover-admin.log: Contains messages from the DiscoverAdmin jini service
- ▶ proxy.log: Contains messages from the Proxy jini service
- ▶ topology.log: Contains messages from the topology jini service
- ▶ login.log: User login audit trail
- ▶ l2.log: Contains messages for the Layer 2
- ▶ Services/ApiServer.log: XML, Java, and Enterprise JavaBeans (EJB) interfaces to the configuration management database (CMDB) are processed here
- ▶ services/ChangeManager.log: ChangeManager works with StateManager to process change events after discovery completes
- ▶ services/ReportsServer.log: Handles reports tasks
- ▶ services/MonitorStateManager.log: Processes discovery and change events
- ▶ services/ClientProxy.log: Start here for GUI issues. The GUI talks to the client proxy exclusively

- ▶ services/ViewManager.log: ViewManager builds the configuration item (CI) navigation trees
- ▶ services/ProcessFlowManager.log: Event processing engine for Discovery
- ▶ services/DiscoverObserver.log: Moves completed workitems from DiscoverManager to TopologyManager

You can turn on debug level logging by setting the following property in `$COLLATION_HOME/etc/collation.properties`:

```
com.collation.log.level=DEBUG
```

This setting causes log files to grow quickly. Log files are rotated when they grow to a specified size. You can control the size of each file with the following property:

```
com.collation.log.filesize=20MB
```

You can control the number of rotated files with the following property:

```
com.collation.log.filecount=3
```

9.2 Installation logs

The installation logs are kept in entirely separate directories from the product's runtime logs:

- ▶ `$COLLATION_HOME/./installLogs`
- ▶ `$COLLATION_HOME/./cdb_install*`
- ▶ `$COLLATION_HOME/./installCDT.stderr`
- ▶ `$COLLATION_HOME/./installCDT.stdout`
- ▶ `/root/InstallShield/Universal/common/Gen2/_vpddb/vpd.script`
- ▶ `/root/InstallShield/Universal/common/Gen2/_vpddb/vpd.properties`

9.3 Problem determination tools

In the following section, we provide details about the tools that are available for testing TADDM. TADDM testing includes testing of protocols, such as Secure Shell (SSH) and Windows Management Interface (WMI), from the server (and optionally, through the Windows gateway) and also running commands remotely on the target machines.

The importance of using these test tools is that these commands use the access profiles that are defined in TADDM to access the remote system. In addition to testing the communications infrastructure (including the TADDM gateway for certain protocols), using these test tools verifies that the user ID and passwords are correctly configured.

Navigate to the /opt/IBM/cmd/db/dist/support/bin directory. Next, we identify a few of the tools found in this directory and discuss the command syntax to use.

The following tools are provided with TADDM on an as-is basis. The purpose of these tools is to help test and debug problems with the Configuration Discovery and Tracking setup and use.

The testing tools that are located in the /opt/IBM/cmd/db/dist/support/bin directory are:

testhang.jy	Helps determine if hanging sessions can be cancelled.
testjdbc.jy	Tests database access through JDBC.
testos.jy	Returns all of the operating system details from a specific system.
testping.jy	Returns the number of active IP interfaces within a specified scope.
testportmap.jy	Provides the entire list of active ports on a host.
testportscan.jy	Determines to which session ports a target system is listening and, thereby, which protocols are supported.
testprimaryip.jy	Returns the primary IP address of a multi-homed host.
testsnmp.jy	Verifies the read community name that is used to access the Simple Network Management Protocol (SNMP) Management Information Base (MIB) at a host.
testssh.py	Executes a command on a host using SSH.
testwmi.jy	Verifies that WMI is installed on a system.
wmiexec.jy	Executes a command on a Windows-based system.

9.3.1 testhang.jy

Use the testhang.jy script to verify that hanging sessions with a UNIX host are successfully backed out, as shown in Example 9-1.

Usage: testhang.jy <ip>

This script requires valid Access List entries, anchor hosts, and Windows gateways to access the target device.

Example 9-1 Using testhang.jy

```
[root@taddm02 bin]# ./testhang.jy 9.3.5.212
Testing ...:9.3.5.212
...
2006-06-27 17:54:55,370 [main] INFO session.SessionFactory -
getNewSession(9.3.5.212) portList=null
2006-06-27 17:55:05,484 [main] INFO util.PortScanner - PortScanner: scan for
9.3.5.212 complete; returning: [22]
2006-06-27 17:55:10,488 [main] INFO session.Ssh2SessionClient - 9.3.5.212:
SSH version=[SSH-1.99-OpenSSH_3.9p1] reuse=true
2006-06-27 17:57:10,663 [main] ERROR session.SshSessionClient - readAsString:
IOException: InputStreamPipe closed after 0 bytes
2006-06-27 17:57:10,663 [main] WARN session.SshSessionClient - Command [cat]
failed in session ssh2:/root@9.3.5.212: timed out a
fter 120.003 seconds
Execute Failed
2006-06-27 17:57:10,892 [main] INFO session.Ssh2SessionClient - 9.3.5.212:
SSH version=[SSH-1.99-OpenSSH_3.9p1] reuse=true
Result is :anaconda-ks.cfg
anchor-setup.sh
coll
coll4.0
Desktop
dlmgr_.pro
install.log
install.log.syslog
InstallShield
Result is :total 116
-rw-r--r-- 1 root root 1216 Apr 25 12:07 anaconda-ks.cfg
-rw-r--r-- 1 root root 2308 Jun 20 12:05 anchor-setup.sh
drwxr-xr-x 3 root root 4096 May 2 14:25 coll
drwx----- 8 root root 4096 Jun 20 12:05 coll4.0
drwxr-xr-x 3 root root 4096 Apr 27 10:11 Desktop
-rw-r--r-- 1 root root 15 Apr 25 16:23 dlmgr_.pro
-rw-r--r-- 1 root root 56590 Apr 25 12:07 install.log
-rw-r--r-- 1 root root 9506 Apr 25 12:07 install.log.syslog
drwxr-xr-x 3 root root 4096 May 2 13:07 InstallShield
```

9.3.2 testjdbc.jy

Use the testjdbc.jy script to verify JDBC connectivity to a database.

Usage: testjdbc.jy {d[b2]|o[oracle]|s[ysbase]} {ip|host} port user
password {oracle SID|sysbase db|db2 db}

Example 9-2 shows a successful attempt to connect to the IBM Tivoli Intelligent Orchestrator (TIO) database (tiodb). This database is owned by the instance that is listening on port 50000 on the tioserver system (9.3.5.216) and using the credentials of the tioadmin user (tioadmin/smartway).

Example 9-2 Using testjdbc.jy to connect to tiodb

```
[root@taddm02 bin]# ./testjdbc.jy db2 9.3.5.216 50000 tioadmin smartway tiodb
Testing ...:db2 9.3.5.216 50000 tioadmin smartway
optional SID/db:tiodb
Establishing DB2 connection
Connection string: jdbc:db2://9.3.5.216:50000/tiodb
Connection successful.
Version: 8020400
```

9.3.3 testssh.py

Use the testssh.py utility to verify SSH access to a system.

The testssh.py script uses the Access List credentials to access a system in your infrastructure and execute a command specified on invocation.

Usage: testssh.py <ip> <command>

To get the universal time and date settings for a system, you can use:

```
[root@taddm02 bin]# ./testssh.py 9.3.5.216 "date -u"
Testing ...:9.3.5.216 date -u
...
Result is :Sat Jun 17 01:01:06 UTC 2006
```

Note: If the command that is passed to testssh.py is made up of more than one word, enclose the entire command in double quotation marks.

9.3.4 testos.jy

Use the testos.jy script to verify that Configuration Discovery and Tracking can access operating system information about a system that is identified by the IP address, as shown in Example 9-3.

The testos.jy program is invoked using the IP address as the only argument.

Usage: testos.jy <ip>

The testos.jy script requires valid Access List entries, anchor hosts, and Windows gateways to access the target device.

Example 9-3 shows how to use the testos.jy script properly.

Example 9-3 Using testos.jy

```
[root@taddm02 bin]# ./testos.jy 9.3.5.212 ls
Testing os layer on ip 9.3.5.212
...
2006-06-28 08:53:12,741 [main] INFO session.SessionFactory -
getNewSession(9.3.5.212) portList=null
2006-06-28 08:53:22,847 [main] INFO util.PortScanner - PortScanner: scan for
9.3.5.212 complete; returning: [22]
2006-06-28 08:53:27,248 [main] INFO session.Ssh2SessionClient - 9.3.5.212:
SSH version=[SSH-1.99-OpenSSH_3.9p1] reuse=true
[testos] computer system: ... before
2006-06-28 08:53:27,999 [main] INFO ip.SystemSigner - System signature set
to: 9.3.5.212(0011250D31BA)
[testos] computer system:
{signature=9.3.5.212(0011250D31BA);type=ComputerSystem;systemId=7f0100;OSRunnin
g=interface com.collation
.platform.model.topology.sys.linux.Linux;fqdn=taddm01;name=taddm01}
[testos] opsys: ... before
[testos] opsys: {osId=1;OSName=Linux;name=Linux;parent=interface
com.collation.platform.model.topology.sys.linux.LinuxUnitaryCompu
terSystem}
[testos] command: kill: kill
[testos] command: port map: lsof -nP -i | awk '{print $2, $8, $9}' | sort -k 2
| uniq -f 1
[testos] command: ps users: ps auxw
[testos] hostid: 7f0100
[testos] hostname: taddm01
[testos] architecture: i686

[testos] kernel architecture: i686
[testos] kernel version: 2.6.9-34.ELsmp
[testos] pid to runtime process map: {}
[testos] computer model: i686
```

```

[testos] computer system:
{signature=9.3.5.212(0011250D31BA);type=ComputerSystem;systemId=7f0100;OSRunnin
g=interface com.collation
.platform.model.topology.sys.linux.Linux;fqdn=taddm01;name=taddm01}
[testos] CPU speed: 2993.323

[testos] CPU type: Intel(R) Pentium(R) 4
[testos] Number of CPUs: 2

2006-06-28 08:53:28,936 [SSH2TransportRX] WARN collation.stderr - mount
cIntudp_create: RPC: Program not registered
[testos] exported NFS file systems:
[testos] network interfaces: [9.3.5.212]
[testos] ip forwarding?: 0
[testos] ip interfaces: array([ipNetwork=interface
com.collation.platform.model.topology.net.IpNetwork;L2Interface=interface com.
collation.platform.model.topology.net.L2Interface;ipAddress=interface
com.collation.platform.model.topology.net.IpAddress}, {ipNet
work=interface
com.collation.platform.model.topology.net.IpNetwork;L2Interface=interface
com.collation.platform.model.topology.net
.L2Interface;ipAddress=interface
com.collation.platform.model.topology.net.IpAddress}],
com.collation.platform.model.topology.net.
IpInterface)
[testos] manufacturer: Red Hat
[testos] memory size: 2017

[testos] nmap: /opt/IBM/cmdb/dist/external/nmap/Linux/nmap
[testos] os name: Linux
[testos] os path:
PATH=$PATH:/usr/local/bin:/bin:/usr/bin:/usr/sbin:/sbin:/usr/X11R6/bin;
[testos] os type: 2
[testos] os version: Red Hat Enterprise Linux ES release 4 (Nahant Update 3)
except...
[testos] computer system: ... before
[testos] computer system:
{signature=9.3.5.212(0011250D31BA);type=ComputerSystem;systemId=7f0100;OSRunnin
g=interface com.collation
.platform.model.topology.sys.linux.Linux;fqdn=taddm01;name=taddm01}
[testos] opsys: ... before
[testos] opsys: {osId=1;OSName=Linux;name=Linux;parent=interface
com.collation.platform.model.topology.sys.linux.LinuxUnitaryCompu
terSystem}
[testos] command: kill: kill
[testos] command: port map: lsof -nP -i | awk '{print $2, $8, $9}' | sort -k 2
| uniq -f 1
[testos] command: ps users: ps auxw
[testos] hostid: 7f0100

```

```

[testos] hostname: taddm01
[testos] architecture: i686

[testos] kernel architecture: i686
[testos] kernel version: 2.6.9-34.ELsmp
[testos] pid to runtime process map: {}
[testos] computer model: i686

[testos] computer system:
{signature=9.3.5.212(0011250D31BA);type=ComputerSystem;systemId=7f0100;OSRunnin
g=interface com.collation
.platform.model.topology.sys.linux.Linux;fqdn=taddm01;name=taddm01}
[testos] CPU speed: 2993.323

[testos] CPU type: Intel(R) Pentium(R) 4
[testos] Number of CPUs: 2

[testos] exported NFS file systems:
[testos] network interfaces: [9.3.5.212]
[testos] ip forwarding?: 0
[testos] ip interfaces: array([{ipNetwork=interface
com.collation.platform.model.topology.net.IpNetwork;L2Interface=interface com.
collation.platform.model.topology.net.L2Interface;ipAddress=interface
com.collation.platform.model.topology.net.IpAddress}, {ipNet
work=interface
com.collation.platform.model.topology.net.IpNetwork;L2Interface=interface
com.collation.platform.model.topology.net
.L2Interface;ipAddress=interface
com.collation.platform.model.topology.net.IpAddress}],
com.collation.platform.model.topology.net.
IpInterface)
[testos] manufacturer: Red Hat
[testos] memory size: 2017

[testos] nmap: /opt/IBM/cmd/db/dist/external/nmap/Linux/nmap
[testos] os name: Linux
[testos] os path:
PATH=$PATH:/usr/local/bin:/bin:/usr/bin:/usr/sbin:/sbin:/usr/X11R6/bin;
[testos] os type: 2
[testos] os version: Red Hat Enterprise Linux ES release 4 (Nahant Update 3)

```

9.3.5 testping.jy

The testping.jy script gets the number of IP interfaces that are currently active in a specific scope. The scope might identify a single IP device, a subnet, or a range.

You invoke the testping.jy program by using the scope type and IP addresses or subnet and netmask information as arguments.

Usage: testping.jy -t type <address info>

```
-t ip <ipaddress>
-t net <network> <netmask>
-t range <startip> <endip>
```

To get the number of active IP interfaces on specific hosts, use the ip <ipaddress> invocation:

```
[root@taddm bin]# ./testping.jy -t ip 9.3.5.216
Ping type is: ip
responder count: 1
```

To see how many IP interfaces are responding from a specific subnet, use the net <subnet> <netmask> invocation:

```
[root@taddm bin]# ./testping.jy -t net 9.3.5.0 255.255.255.0
Ping type is: net
responder count: 74
```

To get the number of active IP interfaces within a range, you can use the range <startip> <endip> notation:

```
[root@taddm bin]# ./testping.jy -t range 9.3.5.1 9.3.5.100
Ping type is: range
responder count: 45
```

9.3.6 testportmap.jy

Use the testportmap.jy script to see which ports are active on a device.

Usage: testportmap.jy <ip>

The testportmap.jy script requires valid Access List entries, anchor hosts, and Windows gateways to access the target device. Example 9-4 shows you how to use the testportmap.jy script properly.

Example 9-4 Using testportmap.jy

```
[root@taddm02 bin]# ./testportmap.jy 9.3.5.225
Testing portmap on ip 9.3.5.225
...
2006-06-27 16:38:31,867 [main] INFO session.SessionFactory -
getNewSession(9.3.5.225) portList=null
2006-06-27 16:38:41,972 [main] INFO util.PortScanner - PortScanner: scan for
9.3.5.225 complete; returning: [22]
```

```

2006-06-27 16:38:46,295 [main] INFO session.Ssh2SessionClient - 9.3.5.225:
SSH version=[SSH-1.99-OpenSSH_3.9p1] reuse=true
Result is:
2540 *:111
2540 *:111 (LISTEN)
2783 *:113 (LISTEN)
2802 127.0.0.1:25 (LISTEN)
20827 127.0.0.1:32774->127.0.0.1:32774
24470 127.0.0.1:631 (LISTEN)
20827 *:1920 (LISTEN)
2768 *:22 (LISTEN)
2560 *:32768
2560 *:32769 (LISTEN)
20827 *:32836 (LISTEN)
20827 *:32837 (LISTEN)
20827 *:3661 (LISTEN)
29192 *:37646 (LISTEN)
3873 *:523
3873 *:523 (LISTEN)
20827 *:6014 (LISTEN)
2560 *:616
24470 *:631
2476 *:68
8405 9.3.5.225:22->9.3.4.165:35843 (ESTABLISHED)
20827 9.3.5.225:32838->9.3.4.174:1918 (ESTABLISHED)
29192 9.3.5.225:32977->9.3.5.212:37645 (ESTABLISHED)
PID NODE NAME

```

9.3.7 testportscan.jy

Use the testportscan.jy tool to verify if the device is listening on any of the ports listed in Table 9-1 to determine which protocol to use to access the system.

Table 9-1 Ports and related protocol

Port	Protocol
22	sshd
23	telnet
135	DCOM

Usage: testportscan.jy <ip> [<wait-time> <attempts>]

Example 9-5 shows how to determine on which ports the host on the IP address 9.3.5.174 listens: 22, 23, or 135. Between each of the four attempts, the script waits two seconds.

Example 9-5 Using testportscan.jy

```
[root@taddm02 bin]# ./testportscan.jy 9.3.5.174 2 4
Testing portscan on ip 9.3.5.174 with 4 attempts and a waittime of 2
seconds.
2006-06-27 17:20:41,371 [main] INFO util.PortScanner - PortScanner:
scan for 9.3.5.174 complete; returning: [23, 22]
2006-06-27 17:20:49,394 [main] INFO util.PortScanner - PortScanner:
scan for 9.3.5.174 complete; returning: [23, 22]
2006-06-27 17:20:57,424 [main] INFO util.PortScanner - PortScanner:
scan for 9.3.5.174 complete; returning: [23, 22]
2006-06-27 17:21:05,441 [main] INFO util.PortScanner - PortScanner:
scan for 9.3.5.174 complete; returning: [23, 22]
[root@taddm02 bin]# ./testportscan.jy 9.3.4.174 1 4
Testing portscan on ip 9.3.4.174 with 4 attempts and a waittime of 1
seconds.
```

Important error information: If you receive an error similar to this error message:

```
Traceback (innermost last):
  File "./testportscan.jy", line 41, in ?
TypeError: com.collation.platform.util.PortScanner(): 1st arg can't
be coerced to String
```

Change the line in the script that reads:

```
scanner = PortScanner(ia, waittime, numAttempts)

to

scanner = PortScanner(ip, waittime, numAttempts)
```

You might experience a situation where the script never ends. To correct this situation, use the script in Example 9-6 on page 379, which shows a working version of the testportscan.jy script. In the code that is listed in Example 9-6 on page 379, we highlighted, in **bold**, the changes compared to the version delivered with the generally available (GA) code of IBM Tivoli Change and Configuration Management Database Configuration Discovery and Tracking V1.1.

```
#!/usr/bin/env ./jython_coll
## $Id: testportscan.jy,v 1.3 2005/03/22 22:15:24 jbarrera Exp $
## Copyright 2001-2005 Collation Inc. All Rights Reserved
## This software is the proprietary information of Collation Inc.
# Use is subject to license terms.
#
import sys
import string
import java

from java.lang import *
from java.net import InetAddress
from java.util import Collections

from com.collation.platform.util import PortScanner

try:
    ip = sys.argv[1]
    try:
        arg2 = sys.argv[2]
    except:
        arg2 = "10"
    try:
        arg3 = sys.argv[3]
    except:
        arg3 = "3"
    waittime = 1000 * string.atoi(arg2)
    numAttempts = string.atoi(arg3)
    print "Testing portscan on ip", ip, "with", numAttempts,
    "attempts and a waittime of", arg2, "seconds."
except:
    print "Usage: testportscan <ip> <waittime-in-seconds>
    <num_attempts>"
    System.exit(0)

oldlist = None
ia = InetAddress.getByName(ip)

i = numAttempts;
while i >= 1:
    i = i - 1;
    scanner = PortScanner(ip, waittime, numAttempts)
    ports = [ 22, 23, 135 ]
```

```
list = scanner.scanPorts(ports)
Collections.sort(list)
if oldlist != None and not list.equals(oldlist):
    print "*** lists are different!"
    print "old: ", oldlist
    print "new: ", list
    break
oldlist = list
System.exit(0)
```

9.3.8 testprimaryip.jy

Use the testprimaryip.jy script to get the primary IP address of a multi-homed device, as shown in Example 9-7.

Usage: testprimaryip.jy <ip>

The testprimaryip.jy script requires valid Access List entries, anchor hosts, and Windows gateways to access the target device.

Example 9-7 Using testprimaryip.jy

```
[root@taddm02 bin]# ./testprimaryip.jy 9.3.5.252
Testing : 9.3.5.252
Primary IP : 9.3.5.252
```

9.3.9 testsnmp.jy

Use the testsnmp.jy script to verify the community name that is used to access the SNMP MIB at a target system, as shown in Example 9-8.

Usage: testsnmp.jy -h <host> -c <community>

Example 9-8 Using testsnmp.jy

```
[root@taddm02 bin]# ./testsnmp.jy -h 9.3.5.216 -c public
Testing 9.3.5.216 with community public
GET of .1.3.6.1.2.1.1.5.0 returned tioserver.tivdemo.com
```

9.3.10 testwmi.jy

Use the testwmi.jy script to verify the WMI connectivity between a Windows gateway and a Windows target, as shown in Example 9-9.

The testwmi.jy script uses the Access List credentials to access the Windows-based system, so it fails unless proper Access List entries to connect to both the Windows gateway (using SSH) and the target system do not exist.

Usage: testwmi.jy <ip>

Example 9-9 Using testwmi.jy

```
[root@taddm02 bin]# ./testwmi.jy 9.3.4.174
Testing WMI on host 9.3.4.174
...
2006-06-27 16:28:47,785 [main] INFO session.SessionFactory -
getNewSession(9.3.4.174) portList=null
2006-06-27 16:28:57,888 [main] INFO util.PortScanner - PortScanner: scan for
9.3.4.174 complete; returning: [135]
2006-06-27 16:29:06,875 [main] INFO util.PortScanner - PortScanner: scan for
9.3.5.252 complete; returning: [22]
2006-06-27 16:29:07,868 [main] INFO session.Ssh2SessionClient - 9.3.5.252: SSH
version=[SSH-2.0-1.75 sshlib: WinSSHD 4.13] reuse
=true
2006-06-27 16:29:08,526 [main] INFO session.UnscopedGateway -
Gateway.prepare(9.3.5.252): first attempt
2006-06-27 16:29:11,043 [main] INFO session.UnscopedGateway -
Gateway.prepare(9.3.5.252): succeeded!
2006-06-27 16:29:11,531 [main] INFO session.AbstractWindowsSessionClient -
GetDesiredVersion returns version 20060607 after 0 se
conds.
2006-06-27 16:29:12,093 [main] INFO session.AbstractWindowsSessionClient -
GetVersion returns version 20060607 after 0 seconds.
2006-06-27 16:29:12,719 [main] INFO session.AbstractWindowsSessionClient -
GetVersion returns version 20060607 after 0 seconds.
```

9.3.11 wmiexec.jy

Use the wmiexec.jy script to execute a command on a Windows-based target system using the WMI protocol.

The wmiexec.jy script uses the Access List credentials to access the Windows-based system, so it fails unless proper Access List entries to connect to both the Windows Gateway (using SSH) and the target system do not exist.

Usage: wmiexec.jy <ip> <command>

Example 9-10 shows how we execute the **hostname** command on a remote system (9.3.5.174) through a Windows gateway (9.3.5.252).

Example 9-10 Using wmiexec.jy

```
[root@taddm02 bin]# ./wmiexec.jy 9.3.4.174 "hostname"
Testing ...:9.3.4.174 hostname
...
2006-06-27 16:26:31,528 [main] INFO util.PortScanner - PortScanner: scan for 9.3.5.252
complete; returning: [22]
2006-06-27 16:26:32,692 [main] INFO session.Ssh2SessionClient - 9.3.5.252: SSH
version=[SSH-2.0-1.75 sshlib: WinSSHd 4.13] reuse
=true
2006-06-27 16:26:33,320 [main] INFO session.UnscopedGateway - Gateway.prepare(9.3.5.252):
first attempt
2006-06-27 16:26:35,729 [main] INFO session.UnscopedGateway - Gateway.prepare(9.3.5.252):
succeeded!
2006-06-27 16:26:36,216 [main] INFO session.AbstractWindowsSessionClient - GetDesiredVersion
returns version 20060607 after 0 seconds.
2006-06-27 16:26:36,734 [main] INFO session.AbstractWindowsSessionClient - GetVersion returns
version 20060607 after 0 seconds.
2006-06-27 16:26:37,390 [main] INFO session.AbstractWindowsSessionClient - GetVersion returns
version 20060607 after 0 seconds.
Result is :taddm99
```

9.4 Log and Trace Analyzer

With the Log and Trace Analyzer, you can gather system and performance data from local and remote systems. You can use this data for problem determination in case a less than optimal system event occurs.

You can use the Log and Trace Analyzer to create resource sets. *Resource sets* are sets of definitions, which contain the path locations of the logs that you need to examine and the levels of information that the logs contain. You can keep customized definitions to reuse. The definitions provide the same set of instructions about where to find a log and what kind of information to gather from the log, which saves time during subsequent log imports.

The Log and Trace Analyzer also makes it possible for you to download and store symptom database catalogs to your local system. These catalogs provide detailed diagnostic solutions to a variety of scenarios, which can provide direction to your troubleshooting tasks.

Downloading the Log and Trace Analyzer

To download the Log and Trace Analyzer, complete the following steps:

1. If you do not have the IBM Support Assistant (ISA) installed, go to the ISA Web site to download the software. Instructions for downloading and installing the ISA are on the ISA Web site:

<http://www.ibm.com/software/support/isa/>

2. Using the ISA built-in Updater component, download and install the ISA plug-in for IBM Tivoli Application Dependency Discovery Manager 7.1 from the IBM Web site. Instructions for downloading and installing the ISA plug-in are on the ISA Web site:

<http://www.ibm.com/software/support/isa/>

3. Using the ISA built-in Updater component, download and install the plug-in for the Log and Trace Analyzer from the IBM Web site. The Log and Trace Analyzer plug-in is included in the list of plug-ins for Common Component Tools:

<http://www.ibm.com/software/support/isa/>

4. After the installation of the Log and Trace Analyzer is complete, start the ISA.
5. From the list of tasks, click **Tools**.
6. From the list of products, click **IBM TADDM 7.1**.
7. From the list of tools for IBM TADDM 7.1, click **Log and Trace Analyzer**. The Log and Trace Analyzer starts working.

Importing TADDM log files to the Log and Trace Analyzer

To import the TADDM log files to the Log and Trace Analyzer, complete the following steps:

1. Copy the relevant log files from the TADDM Servers to the system where you installed the IBM Support Assistant workbench. Put the log files for each server in a unique directory, for example, c:\TADDM\logs\serverxxxx\...
2. Import the TADDM log files. The Log and Trace Analyzer organizes related log files into *log sets*. You can use log sets to import and analyze a set of related log files. This facility is used to organize and import your TADDM log files. Log set definitions provide information to the Log and Trace Analyzer specifying where log and trace data reside and describing what type of data to gather from local and remote systems. The Log and Trace Analyzer allows you to import predefined log sets that contain the necessary path information required for retrieving log files on demand. There are two ways to import the log files:
 - a. Create the initial TADDM log set:

- i. Click **File** → **Import Log File**.
 - ii. Create a new log set.
 - iii. Type the name for the log set. For example, you can type the following text: TADDM Log files for server xxxx.
 - iv. Click **Add**.
 - v. In the Name Filter window, to limit the list of log files to the TADDM log files, type *Discovery*.
 - vi. Select the type of log file that you are adding to the log set.
 - vii. Type the name of the log file on your local system. Ensure that the type of log file matches the log file that you specified.
 - viii. Enter the correct version of the TADDM product that corresponds to the log file. Refer to the Log and Trace Analyzer online help for additional options.
 - ix. To add the log file to the log set, click **OK**.
 - x. For every log file that you want to include in the log set, repeat steps i through ix. The first time that you create the log set, you will save time later by including every log file that you want to include in the log set.
- b. Reuse an existing TADDM log set:
 - i. Select **File** → **Import Log File**.
 - ii. Select an existing Log Set Definition from the drop-down list of defined log sets.
 - iii. If necessary, change the contents of the log set definition. You can add, edit, or remove from the list of log files in the log set.
 - c. To indicate that the file needs to be imported to the log set, select the check box next to the log file.
 - d. To import the log files, click **Finish**.

You can create and reuse as many log sets as you need. For example, when importing log files from multiple servers, you need more than one log set.

Analyzing TADDM log files with the Log and Trace Analyzer

Using the Log and Trace Analyzer, you can correlate multiple TADDM log files into a single view. The TADDM log files can be combined in a single view, which is ordered by time stamp, to correlate the operation of the TADDM components. There are two ways to correlate log files:

- Simple: To correlate all imported log files, complete the following steps:
 - a. In the Log and Trace Analyzer navigation tree view, right-click **Logs**.

- b. Click **View All Logs**.
- ▶ **Advanced:** To correlate a set of log files by creating a custom correlation, complete the following steps:
 - a. In the Log and Trace Analyzer navigation tree view, right-click **Correlations**.
 - b. Click **New** → **Log Correlation**.
 - c. In the window that is displayed, type the name for the correlation.
 - d. Add the log files that you want to include for the correlation.
 - e. Click **Finish**.
 - f. Refresh the navigation tree view.
 - g. In the navigation tree view, right-click the correlation name that you typed and click **Open With** → **Log View**.

After you create a view of the logs, you can organize the log data to isolate problems. The following list identifies several of the ways that you can organize the data:

- ▶ **Sort log records:** For example, you can sort by time, component, and server name.
- ▶ **Highlight log records:** For example, you can highlight all error events in red or show all events from a specific component in blue. Highlighting is similar to filtering, but instead of eliminating data from a view, you can highlight the relevant information within the full list of events.
- ▶ **Filtering log records:** You can narrow the scope of a problem and the data shown based on filter criteria. Examples of filter criteria include time stamps, severity, component, and server.
- ▶ **Finding log records:** You can search for specific information in a log file. For example, you can search to see the events that are related to interaction with a specific server or user.

For more information about how to organize the data, in the Log and Trace Analyzer online help, search for the “Analyzing log files” topic. “Filtering, Sorting, Finding, and Highlighting” is a subheading in this topic.

In addition, there are other topics in the online help that you might find useful:

- ▶ When trying to correlate log files from multiple servers, the time clocks on those servers can be out-of-sync. This synchronization problem might be something simple, such as different time zones, or more subtle, such as a clock being a few milliseconds off from another server’s clock. The Log and Trace Analyzer imbeds a function to synchronize the time between multiple log files by allowing you to adjust the time stamps in a log file. For more

information, refer to the topic “Synchronizing time of log records for distributed applications” in the Log and Trace Analyzer online help.

- ▶ You can use symptom catalogs to quickly recognize known problems. The Log and Trace Analyzer provides a log analysis capability that allows it to recognize known problems that are defined in a knowledge database, called the “*symptom catalog*”. IBM provides a symptom catalog for known problems with several products, including TADDM. It also provides a way for you to capture and define your own symptom information. For more information, refer to the topic titled “Synchronizing time of log records for distributed applications” in the Log and Trace Analyzer online help.

9.5 Specific scenarios

Next, we describe several common scenarios when working with TADDM, related questions, and possible explanations for why the scenarios occur.

9.5.1 Common problems

In this section, we discuss common problems and possible reasons behind these problems.

I have a software process that was not discovered, and I want to know why:

- ▶ Discovery of certain applications requires an entry in the Access List. You can search for the specific software on the support site for details.
- ▶ Only processes with TCP connections are discovered. Other processes are discarded. Run the `lsnf` command to verify if the process in which you are interested has a TCP connection.

I know that two software processes talk to each other through TCP, but this dependency is not listed or shown:

- ▶ Connections between software servers are detected only if the TCP connection is established at the time that the discovery is run.
- ▶ The connection might be discovered eventually after a series of discovery runs, or a manual connection can be created.

Why are the connections between software processes on the same machine not shown:

- ▶ By default, servers with listening ports on loopback interfaces are suppressed.

- ▶ The relationships between software processes are controlled by the `com.collation.platform.os.ignoreLoopbackProcesses=true` property in the `collation.properties` file.
- ▶ The connections might not show with tunnelled connections.

Why is the Environment section of the Runtime tab for an application empty:

- ▶ The **ps** command on Linux needs to be `setuid`, or `sudo` access must be granted to the Collation Service Account.
- ▶ The system-V **ps** command on Hewlett-Packard UNIX (HP-UX) is unable to show Environment variables.
- ▶ The user must be the root user on HP-UX and Linux. Use the **sudo** command.

9.5.2 Troubleshooting problems with sensors

In this section, we provide best practices for troubleshooting problems with sensors.

Suggestions for searching logs

In this section, we provide suggestions for viewing and browsing log files in order to find the most pertinent information:

- ▶ Use **less**, **grep**, and **vi** for searching logs in UNIX.
- ▶ If you install Cygwin, you can use these commands on Windows.
- ▶ Start at the end of the file and search backwards.
- ▶ Search for ERR and look for stack traces that are generated when an exception happens.
- ▶ Filter the DiscoverManager log by using the following methods:
 - Make a subset of the log by using the **grep** command for the sensor in which you are interested.
 - If the result is still too verbose, use **Target** or **Thread** parameters for additional filtering.
 - If you are looking at an entire log, start by finding the Target/Sensor combination in which you are interested, such as `IpDeviceSensor-9.3.5.184`. Follow its execution with repeated **Find-next** searches on the Thread ID, such as `DiscoverWorker-10`.
 - If you are searching a filtered log and find something interesting, note the time stamp, for example, `2007-08-29 21:42:16,747`. Then, look in the full log for the lines surrounding that time stamp.

Understanding DiscoverManager log messages

Figure 9-1 shows the parts of a DiscoverManager log message.

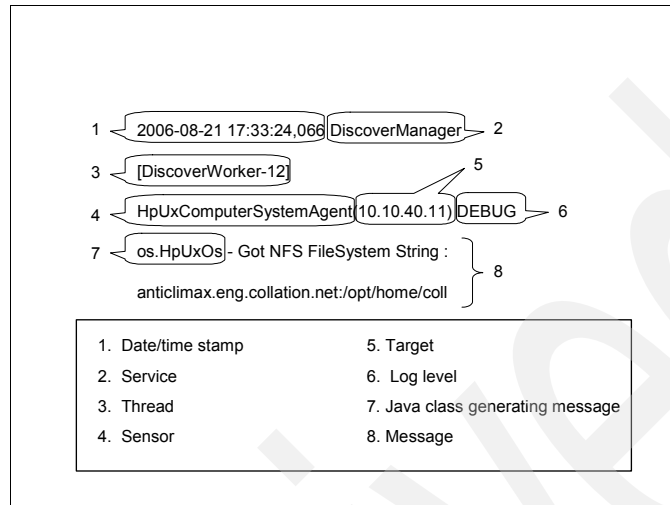


Figure 9-1 Components of a log message

Troubleshooting specific sensors

There are many sensors that ship with TADDM. Many of them have troubleshooting information that is available in a document called *Best Practices for using TADDM sensors*. In this section, we list the sensors that are included in the white paper and where to find that information. The *Best Practices for using TADDM Sensors* white paper is available at:

<http://www-1.ibm.com/support/docview.wss?uid=swg27010399>

These are the sensors that are included in the white paper:

- ▶ Apache
- ▶ Checkpoint
- ▶ CheckpointSNMPAgent
- ▶ Citrix
- ▶ DB2 database
- ▶ Internet Information Services (IIS)
- ▶ i5/OS®
- ▶ JBoss server
- ▶ MQServer
- ▶ Oracle database
- ▶ Oracle Application Server
- ▶ SAP – CCMS and SLDSNMPMib2
- ▶ SQL Server database

- ▶ StackScan
- ▶ VMware ESX
- ▶ Weblogic Application Server
- ▶ WebSphere Application Server
- ▶ Windows OS

9.5.3 Storage errors in sensors

The last activity that happens within a sensor is the storage of discovered information in the database. That activity can generate a class of errors called *storage exceptions*, which appear in the sensor status as “A storage error occurred” and often appear in the logs with the following information in the error context: `com.collation.discover.observer.StorageException`.

There are two causes of storage errors. The first cause is a storage error due to a `MissingKeyException`. The second cause is due to resource contention in the database.

MissingKeyExceptions

`MissingKeyExceptions` occur because there is a piece of missing data. The Globally Unique Identifier (GUID) that is being inserted for the object does not match; we cannot match the data that is being stored with the data that is already in the database; or a similar problem exists. In other words, it is a data problem.

There are also cases with an error similar to “storing 1 server bound to unknown SAP”. This error can occur if the server is bound to an IP address that is different from the IP address that was used for the discovery scope. This condition can happen in servers with multiple IP addresses or interfaces where another IP address was used for discovery. The solution is to use the IP address to which the server is bound for discovery.

Resource contention in the database

A `StorageError` without the `MissingKeyException` in the error.log is a database deadlock. A *database deadlock* means that the sensors were trying to grab the same set of database locks, each sensor grabbed several of the database locks, and then, neither sensor was able to continue because they were each waiting for locks that the other sensor held. When discovering only one or two hosts with parallel storage turned on, deadlocks are not completely avoidable. However, deadlocks are a temporary condition. When multiple hosts are discovered, this deadlock happens less frequently.

To fix this problem, you need to either set the storage thread count to a lower value, or you need to increase the storage retry count in the `collation.properties` file.

A higher threadcount (or storage threads) increases throughput with the increased likelihood of contention.

One method to work around this deadlock is to lower the number of storage threads by changing the `com.collation.discover.osbserver.topopumpcount` from 4 to 1 when performing a discovery on only one WebSphere cell. This change eliminates parallel database inserts.

The other alternative is to raise the number of storage attempts. The more topopumps (or storage threads), the more chance for contention and the greater need for a higher storage attempt count.

Change the following line in the `collation.properties` file from 3 attempts to 6:

Number of times TopoPump will try to store a result:

```
com.collation.discover.observer.topopumpstorageattempts=3
```

to:

```
com.collation.discover.observer.topopumpstorageattempts=6
```

9.5.4 Application programming interfaces (APIs)

The `api.sh` command often does not provide error messages on failure. There are two logs that can help you in this scenario.

ApiServer.log

If a specific API query fails, but most queries work, you can examine the `ApiServer.log`. For SQL-style search queries, such as Model Query Language (MQL), you can search for the query itself in the log file, as illustrated in Example 9-11.

Example 9-11 Finding a MQL query in the `ApiServer.log`

```
2007-10-30 14:48:33,154 ApiServer [RMI TCP
Connection(136)-10.199.2.114] DEBUG
server.DataWorker - find, sessionId: 937921212956061062, query: select
* from UserData where objRef=='39FC5CABAA00364E97DBDBCE8DE8F5D3', jdoQ:
null, jdoVar: nul
l, mss: null, permissions: null
```

After you have found the query, search downward in the log file for the string `ERROR`. The cause of the failure is usually logged as an `ERROR` message. In Example 9-12, we learn that “The delete failed” due to a `NullPointerException`, which means that the system was unable to find what it was looking for. The cause, in this case, is likely that the GUID (the long string `39FC5CABAA00364E97DBDBCE8DE8F5D3`) either does not exist, has a typographical error, or was formatted incorrectly.

Example 9-12 The ERROR message in ApiServer.log

```
2007-10-30 14:48:33,258 ApiServer [RMI TCP
Connection(136)-10.199.2.114] ERROR
server.DataWorker - The delete failed.
java.lang.NullPointerException
    at
com.ibm.cdb.api.server.DataWorker.deleteById(DataWorker.java:1939)
    at
com.ibm.cdb.api.server.ApiServerBean.deleteById(ApiServerBean.java:207)
    at
com.collation.proxy.api.server.ApiServer.deleteById(ApiServer.java:815)
```

ClientProxy.log

All external calls to TADDM, which includes calls through the `api.sh` script or Java-based API calls, come through the `ClientProxy` service. If there is a problem with this service, you find information in the `dist/log/services/ClientProxy.log` file.

9.5.5 Troubleshooting Windows discoveries

Next, we discuss considerations for Windows discoveries.

Windows discovery can be challenging

Windows security is, understandably, an obscure and challenging technical area, mastered by Windows administrators, different in every environment, changing with every service pack, and designed to keep malicious hackers from discovering information about the server. TADDM is a system management tool designed to discover information about servers in your environment. There is an inherent conflict, and it is difficult to overcome.

TADDM is tested with full local administrator user access on each server to be discovered. Every client has a different security policy and removes rights from the service account that TADDM is allowed to use to discover. Often, the TADDM administrator and the Windows administrators are not in the same organization, building, or city, so overcoming this situation becomes an organizational, as well as a technical, challenge.

Considering this challenge, we now discuss how to troubleshoot problems with Windows discovery.

testwmi command

First, try to run the wmitest tool on the TADDM Server to get an idea of what TADDM struggles with when discovering a given server. Perform the following steps and look at the output to get an idea of the cause of the problem:

1. Log on to the IBM Tivoli Change and Configuration Management Database (CCMDB)-server using **putty** and log on with the cdtadmin user. Navigate to the /opt/IBM/cmdm/dist/support/bin directory and issue the following command:

```
#./testwmi.py 10.176.55.17
```

2. Carefully scan the output to see if it includes a clue to what the actual problem is with the given target. If no obvious problem is reported, continue with the tests in the following steps.

Verifying WMI on the target machine

Perform the following steps to verify that WMI is on the target machine:

1. Run the built-in WMI test tool, WMI Tester, on the Windows servers, by opening a DOS window and running the command **wbemtest**. The following GUI is displayed (Figure 9-2 on page 393).

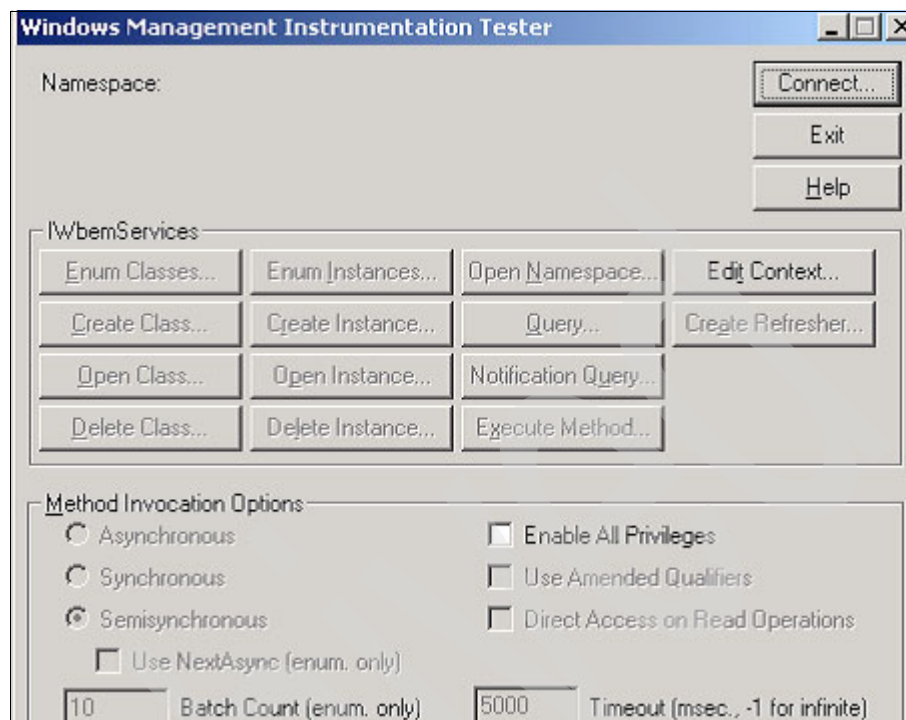


Figure 9-2 WMI Tester

If the WMI Tester GUI is not displayed, there is a basic problem with WMI on the machine, or the taddm user does not have the necessary rights to run WMI. This situation requires investigation on the target machine to get WMI working, or alternatively, try to log on with another user (Administrator) to check if there is a difference.

2. Click **Connect**, fill in the correct namespace (root\cimv2), and click **Connect** (Figure 9-3 on page 394).

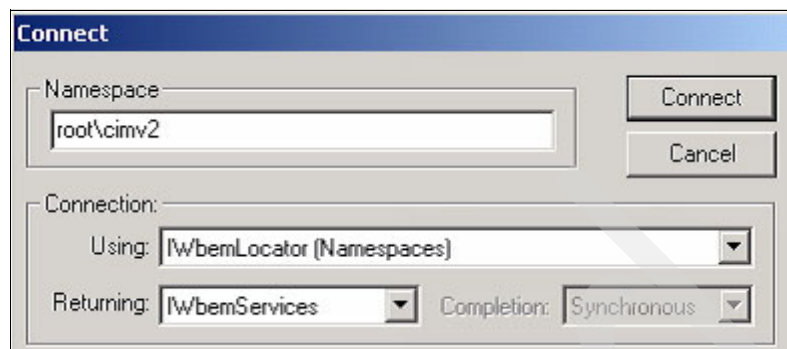


Figure 9-3 Namespace

The GUI will be redisplayed, now with all buttons active (Figure 9-4).

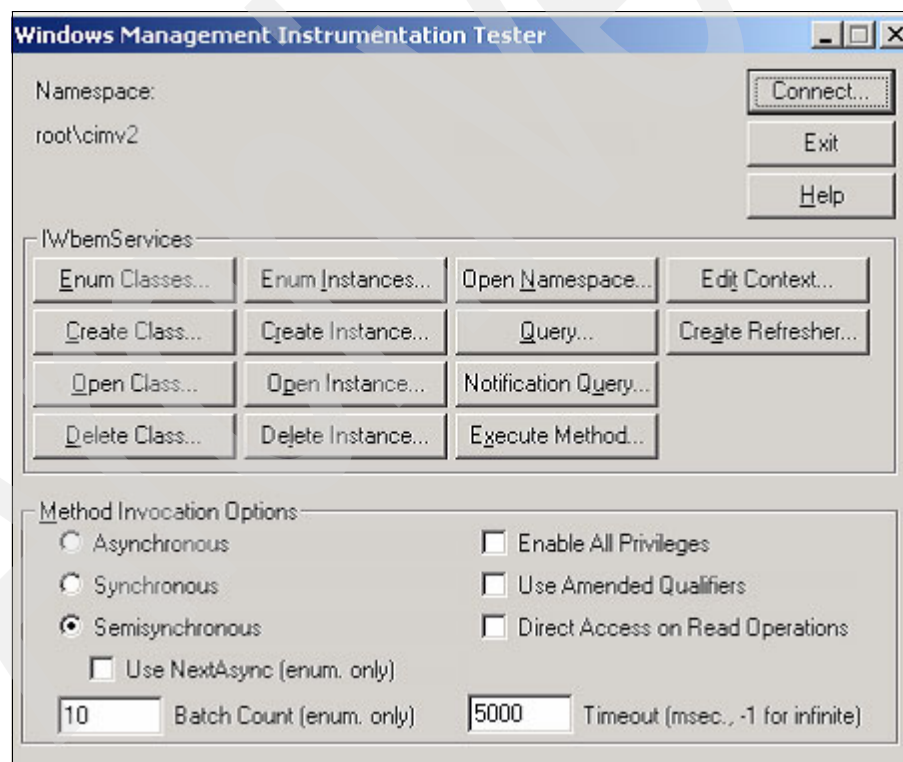


Figure 9-4 WMI Tester

This test verifies that WMI is working locally on the target.

Note: Scriptomatic is an alternative tool to use for WMI testing. Scriptomatic is a compiled HTML document that is used to create scripts, which in turn are used to retrieve WMI data - either locally or remotely. You can download Scriptomatic (including the documentation) from this Microsoft Web site:

<http://www.microsoft.com/downloads/details.aspx?familyid=09DFC342-648B-4119-B7EB-783B0F7D1178&displaylang=en>

You can also query these additional classes that are used by TADDM:

- ▶ Win32_Process
- ▶ Win32_OperatingSystem
- ▶ Win32_WMISetting
- ▶ Win32_ComputerSystem

Verify that these classes can be queried using Scriptomatic locally on the target system and remotely from the gateway.

WMI functionality with WMI DIAG

WMI DIAG is a tool that is provided by Microsoft that can verify that WMI is configured and accessible. WMI DIAG is available at the following Web site:

<http://www.microsoft.com/downloads/details.aspx?familyid=d7ba3cd6-18d1-4d05-b11e-4c64192ae97d&displaylang=en>

Follow the instructions to install and run the utility. Verify that WMI is working correctly.

Verify WMI from the gateway to the target

The next step is to try WMI connectivity between the TADDM gateway and the target. Use the same procedure that was just outlined to start the **wbemtest** tool:

1. Log on remotely with the **taddm** user to the TADDM gateway using Remote Desktop or Virtual Center. Open a DOS window, and issue the **wbemtest** command.
2. Log on locally to the WMI namespace to verify the basic functionality of WMI on the gateway by clicking **Connect** and filling in the correct namespace (**root\cimv2**). Click **Connect** as shown in Figure 9-5 on page 396.

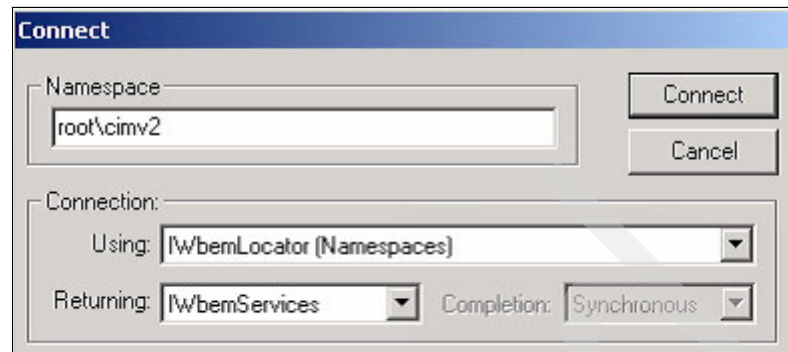


Figure 9-5 Connect

WMI is working locally on the TADDM gateway if the GUI is redisplayed with all buttons active.

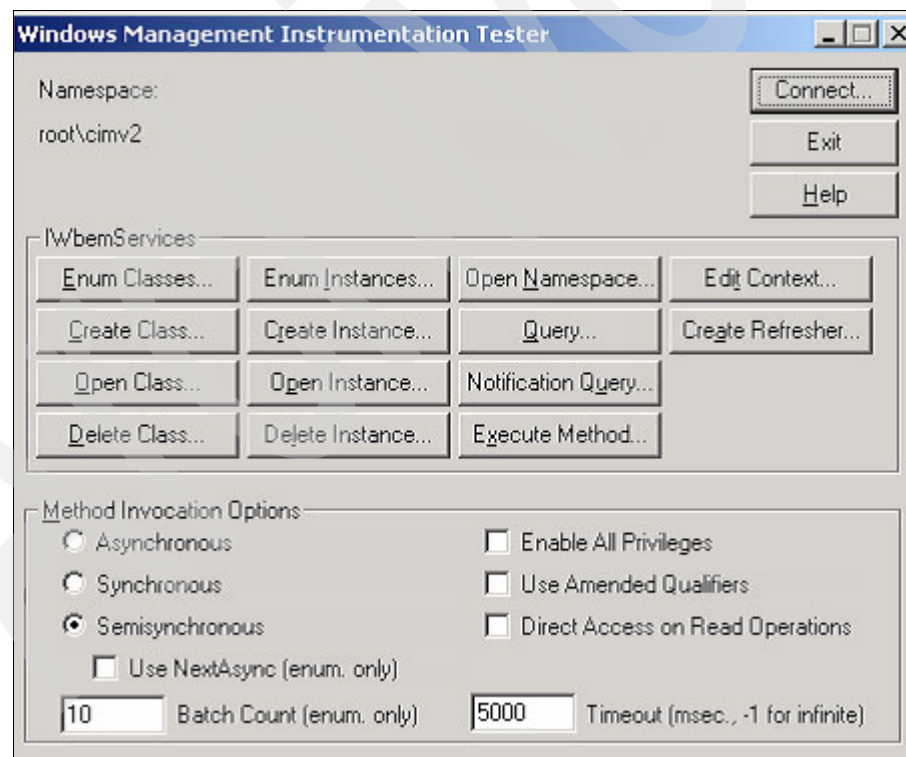


Figure 9-6 WMI Tester

WMI connectivity between the gateway and the target

Next, verify the WMI connectivity between the TADDM gateway and the target.

Click **Connect** and fill in the correct namespace (`\\servername\root\cimv2`), including the name or IP address of the target machine. Click **Connect**.

As in the previous example, the GUI is redisplayed with all buttons active, and the namespace that is shown in the upper left corner includes the name or IP address of the target machine (Figure 9-7).

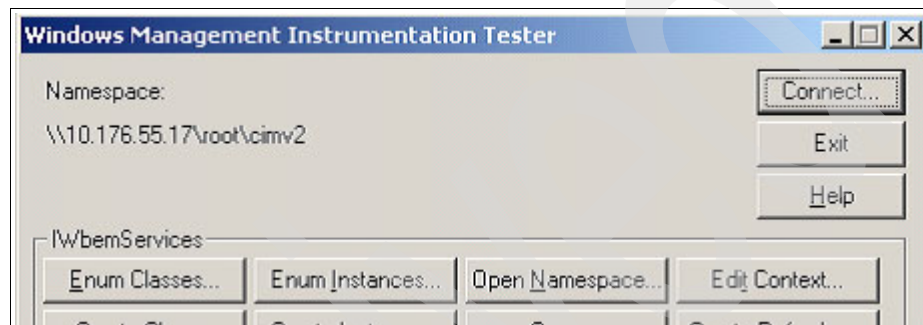


Figure 9-7 WMI Tester

This verifies that the correct WMI connectivity exists between the TADDM gateway and the target.

Using testwmi again

The next step to perform is to verify the WMI functionality from the TADDM Server through the TADDM gateway to the target. This WMI functionality includes SSH communications between the TADDM Server and the TADDM gateway, as well as WMI connectivity between the TADDM gateway and the target. In addition, both WMI and SSH credentials as defined in TADDM are used for connectivity.

Navigate to the `/opt/IBM/cmdm/dist/support/bin` directory. Issue the following command to check for WMI functionality:

```
#./testwmi.py <target ip>
```

Closely verify the output for problems related to passwords, access rights, or problems related to connectivity (that is, Remote Procedure Call (RPC) problems).

Testing specific queries

TaddmTool is the script deployed to Windows gateways for discovery. Most of the data that is gathered during Windows discovery is from queries that are initiated by TaddmTool. You can run a specific TaddmTool query (perhaps a query that is failing in the log files) to identify if a specific command is not working. You can find the **TaddmTool.exe** in C:\WINDOWS\Temp\taddm.xxxxx on the Windows Gateway server.

The syntax to run a command is:

TaddTool.exe arguments command @IP parameters required arguments

Example 9-13 shows the QueryServices command.

Example 9-13 TaddmTool command

```
TaddmTool.exe -DTADDM_ID=12345 -DTADDM_USERNAME=taddm  
-DTADDM_PASSWORD=password123 -DTADDM_INTERACTIVE=yes QueryServices  
@9.43.73.81 servicename
```

The **TaddmTool.exe** arguments are:

- ▶ -DTADDM_ID=<id# from /WINDOWS/temp/taddm.xxxxx>
- ▶ -DTADDM_USERNAME=<id on the target>
- ▶ -DTADDM_PASSWORD=<password on target>
- ▶ -DTADDM_INTERACTIVE=yes

If no “@IP” is specified, the command is run on the local server.

You can use any of these commands:

- ▶ AdsiDump
- ▶ AdsiDumpLocal
- ▶ AdsiDumpRemote
- ▶ AdsiEnum
- ▶ AdsiEnumLocal
- ▶ AdsiEnumRemote
- ▶ CheckServices
- ▶ Db2Find
- ▶ Db2FindSchema
- ▶ Db2Svce2Inst
- ▶ GetActiveDirectoryLdapParameters
- ▶ GetActiveDirectoryNamingContexts
- ▶ GetCitrixInformation
- ▶ GetEnvironment
- ▶ GetEtcServices
- ▶ GetFileInfo

- GetFreeDiskSpace
- GetHostId
- GetInstalledSoftware
- GetIpInterfaces
- GetLongPath
- GetNonDefaultServices
- GetNonDefaultServicesWithDescriptions
- GetPortMap
- GetProcessEnvironment
- GetRouteInfo
- GetSecurity
- GetShortPath
- GetSMSInformation
- GetSMSParentChilds
- GetSystemInfo
- GetTaddmToolVersion
- GetWmiClassProperties
- GetWmiClassValues
- Help
- InstallProvider
- Kill
- ListDevices
- ListDNSServers
- ListDrives
- ListIpAddresses
- ListKernelModules
- ListProcesses
- ListShares
- Md5Hash
- NetConnect
- Obscure
- ProbePort
- Ps
- QueryRegistry
- QueryServices
- RestartWmi
- RunCommand
- RunCommandUtf8
- SqlDump
- StartAnchor
- StartWmi
- StopWmi
- TestWmi
- Unobscure
- WinError

9.5.6 Troubleshooting SSH

Test tools on the TADDM Server make it possible to test using the access credentials that are defined in TADDM.

Navigate to the test tool directory and issue the following command:

```
./testssh.py 10.176.52.22 "uname"
```

or

```
./testssh.py 10.176.52.22 "sudo lsof"
```

You can find various commands that are failing in the sensor log files and test them through the command line **ssh** or the **testssh** command.

Environment

Many problems in TADDM are caused by the configuration of the underlying system environment. One example of this is the SSH environment. TADDM SSH connections are made with a Java SSH library, and the behavior in this library is differs slightly from logging in via SSH and sourcing the server account's environment. Therefore, we recommend that you test failing commands using the following syntax from the TADDM Server:

```
ssh username@10.176.52.22 uname
```

Note that the command is on the same line as and is actually an argument to the **ssh** command. In this case, the **uname** command is executed with the same environment as the TADDM SSH connection.

If this type of test fails, try the following command to examine the environment that is being used:

```
ssh username@10.176.52.22 env
```

Rivest, Shamir, and Adelman (RSA) fingerprint

After reprovisioning a server, the RSA fingerprint stored on the TADDM Server will be different from the key that was sent from the server. When starting an SSH session to the remote server from TADDM, you will get a message similar to Figure 9-8 on page 401.

```

-bash-3.00$ ssh taddm@10.176.52.25
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@    WARNING: REMOTE HOST IDENTIFICATION HAS CHANGED!    @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
IT IS POSSIBLE THAT SOMEONE IS DOING SOMETHING NASTY!
Someone could be eavesdropping on you right now (man-in-the-middle attack)
It is also possible that the RSA host key has just been changed.

```

Figure 9-8 TADDM message

This error message indicates that the entry in the known_hosts file has to be updated. To update the entry in the known_hosts file:

1. Log in to the TADDM Server using SSH and start an SSH session to the server in question:
ssh taddm@<host>.
2. Enter the password for the taddm user.
3. Note the error message indicating the file to be updated and the line in the file to be changed (known_hosts and line 67 in the example in Figure 9-9).

```

Add correct host key in /opt/IBM/home/cdtadmin/.ssh/known_hosts to get rid of this message
Offending key in /opt/IBM/home/cdtadmin/.ssh/known_hosts:67

```

Figure 9-9 Error message

4. Open the file known_hosts in vi and navigate to the line containing the key for the server in question (search for the IP address or host name of the server).
5. Press dd to delete the line; press Escape and :wq and Enter to save the file and exit vi.
6. Start an SSH session to the server in question by running:

```
ssh taddm@<host>
```

This command is shown in Figure 9-10.

```

-bash-3.00$ ssh taddm@10.176.52.20
Warning: Permanently added '10.176.52.20' (RSA) to the list of known hosts.

```

Figure 9-10 RSA key added

Note that the RSA key is added to known_hosts for the server.



Part 5

Planning for a Client Engagement

In this part, we focus on service engagement planning for Tivoli Application Dependency Discovery Manager. The target audience of this part is IBM Business Partners and Solution Developers.

Planning for a client engagement

In this appendix, we discuss several areas of IBM Tivoli Application Dependency Discovery Manager (TADDM) that you need to consider during a client engagement. The target audience of this appendix is IBM Business Partners and Solution Developers. The topics that we discuss include:

- ▶ “Services engagement preparation” on page 406
- ▶ “Solution scope and components” on page 407
- ▶ “Services engagement overview” on page 410
- ▶ “Estimating the activities and timings of the engagement” on page 416

Important: The time estimates in this chapter are not representative of all the possible implementation scenarios of a IBM Tivoli Application Dependency Discovery Manager-based solution. Each environment is unique, and the time estimates that we provide must be regarded as general guidelines, not absolute numbers.

Services engagement preparation

This section describes the resources that are available to help you successfully deliver a solution. The end goal of a services engagement can be comprised of all or part of the following items:

- ▶ Describe the IBM Tivoli Application Dependency Discovery Manager V7.1 architecture and components.
- ▶ Plan and design an IBM Tivoli Application Dependency Discovery Manager V7.1 solution that is based on the client requirements and environment.
- ▶ Install and configure prerequisites for IBM Tivoli Application Dependency Discovery Manager V7.1.
- ▶ Install and configure IBM Tivoli Application Dependency Discovery Manager V7.1 infrastructure components and integrated products (IBM Tivoli Enterprise Console®, IBM Tivoli Netcool, IBM configuration management database (CMDB), and so on).
- ▶ Perform performance tuning and problem determination for IBM Tivoli Application Dependency Discovery Manager V7.1.

We discuss these topics in two sections:

- ▶ “Implementation skills” on page 406
- ▶ “Available resources” on page 407

Implementation skills

To successfully develop and deploy a IBM Tivoli Application Dependency Discovery Manager-based solution, you must acquire certain specialized skills. You need these skills to implement and customize the solution:

- ▶ Working knowledge of operating system (OS) administration, networking, and firewall concepts.
- ▶ Basic knowledge of the enterprise discovery monitoring concepts.
- ▶ Basic knowledge of protocols, such as Simple Network Management Protocol (SNMP), Secure Shell (SSH), Java Management Extensions (JMX), and Windows Management Interface (WMI).
- ▶ The exact skills balance that you will need depends on the environment that you intend to build with this technology and also the server platform on which you intend to host the management solution.

Available resources

You need the prerequisite skills that are listed in “Implementation skills” on page 406 to customize or develop the solution. For each of these skills, many resources are available to help you acquire the necessary skill level. Several of the educational resources available are:

- ▶ Further technical information, including trial code, white papers, and support links, can be found at:
<http://www-306.ibm.com/software/tivoli/products/monitor/>
- ▶ Classroom training: IBM PartnerWorld® provides current information about available classes, their dates, locations, and registration. Additionally, check the PartnerEducation Web site, which serves as a single point-of-contact for all IBM Business Partner education and training. Further details can be found at:
<http://www-306.ibm.com/software/tivoli/education/>
- ▶ IBM Technical Education Services (ITES) offers a variety of classes at all knowledge levels to help you achieve any of the offering’s prerequisite skills.
- ▶ IBM Redbooks publications: You can access various practical and architectural information regarding IBM hardware and software platforms from IBM Redbooks publications. PDFs are available for download from the Web site:
<http://ibm.com/redbooks>

Solution scope and components

Define the scope of the solution. The solution can be one of the two types of basic offerings that are described in “Basic solution definition” on page 409, or you can add additional components, as shown in “Advanced solution definition” on page 410.

IBM Tivoli Application Dependency Discovery Manager

Tivoli Application Dependency Discovery Manager provides the necessary visibility required to achieve operational management of infrastructure resources.

Agent-free automatic discovery is employed together with a data center reference model to produce complete cross-tier dependency maps and topological views.

The following list describes the product operations:

- ▶ The agent-free Discovery Engine instructs and coordinates the Discovery Sensors to determine and collect the identity, attributes, and settings of each application, system, and network component.
- ▶ The discovered data is fed to the Data Center Reference Model creating the specific runtime, cross-tier application topologies.
- ▶ The topologies, along with their configuration data, interdependencies, and change history, are stored in the TADDM Database.
- ▶ The Product Console provides analytics and topological views of the TADDM Database.

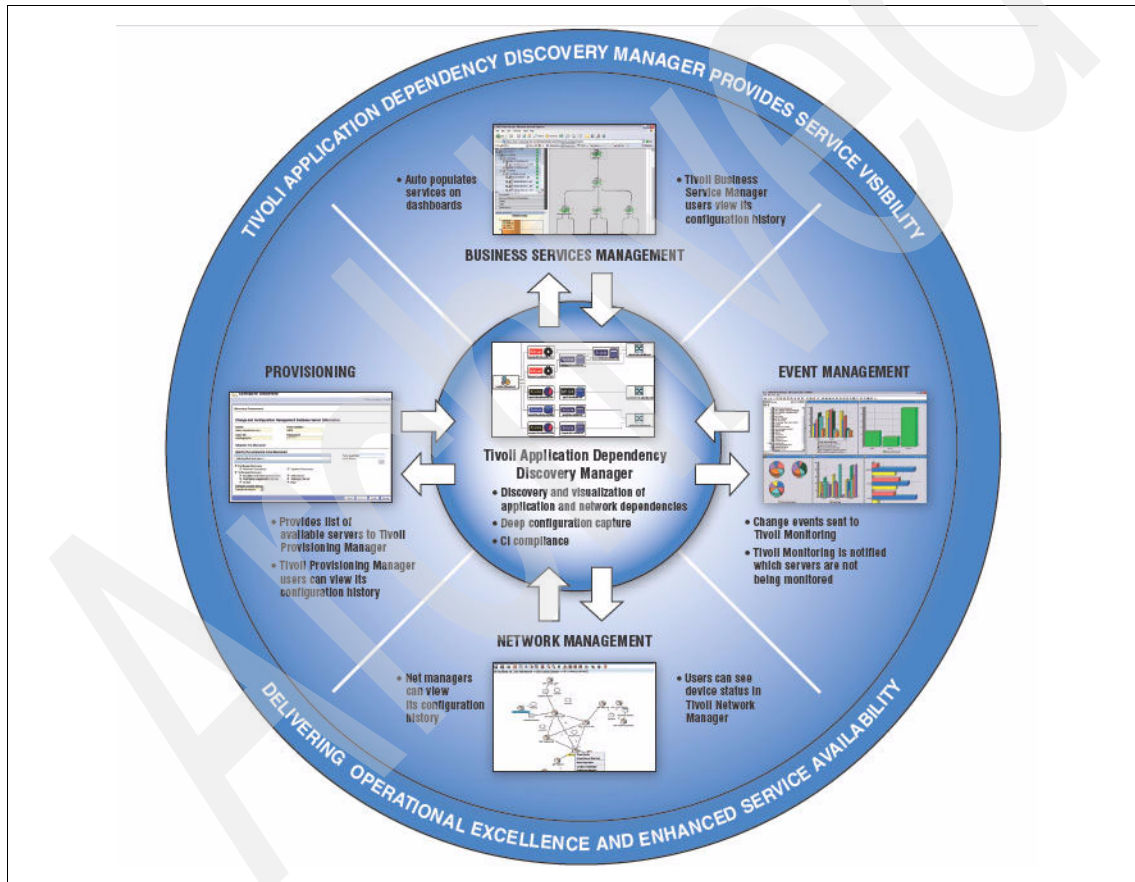


Figure 9-11 Portfolio of products

As shown in Figure 9-11 on page 408, IBM offers a complete portfolio of operational management products that, when used with Tivoli Application Dependency Discovery Manager, form the basis of solutions to enhance:

- ▶ Business systems management:
 - Tivoli Business Service Manager auto-populates services on dashboards.
 - Tivoli Business Service Manager allows users to view service configuration history.
- ▶ Provisioning of software and servers:
 - Provides a list of available servers to Tivoli Provisioning Manager.
 - Tivoli Provisioning Manager users can view its configuration history.
- ▶ Monitoring and event management:
 - Change events are sent to Tivoli Monitoring.
 - Tivoli Monitoring is notified which servers are not being monitored.
- ▶ Network management:
 - Network managers can view its configuration history.
 - Users can see device status in Tivoli Network Manager.

Basic solution definition

IBM Tivoli Application Dependency Discovery Manager delivers automated discovery and configuration tracking capabilities to build application maps that provide real-time visibility into application complexity. These detailed maps include complete data on runtime dependencies, in-depth configuration values, and accurate change histories.

Tivoli Application Dependency Discovery Manager provides the insight that you need to understand complex configurations and how those configurations have changed. Tivoli Application Dependency Discovery Manager is able to proactively notify your event management products when a change has occurred, so that you have the ability to better understand which changes might be the root causes of problems that your business critical systems are experiencing.

Advanced solution definition

TADDM helps you understand configurations, map applications and changes, and address compliance measures. Key highlights are:

- ▶ Visualize interdependencies and relationships between applications, computer systems, and network devices through application mapping and agent-less, credential-free discovery capabilities.
- ▶ Take advantage of integration with other Tivoli operational management products, such as IBM Tivoli Business Systems Manager, IBM Tivoli Provisioning Manager, IBM Tivoli Monitoring Services, IBM Tivoli Composite Application Manager for service-oriented architecture (SOA), and many more.
- ▶ Learn how configuration items (CI) are configured and changing over time by capturing the configuration of each CI, tracking changes to it, and providing analytics to report on the history of the configuration changes.
- ▶ Determine if configurations comply with your policies by comparing discovered configurations to a “reference master” to reveal policy violations.
- ▶ Use industry-standard secure protocols in the discovery process to ensure that sensitive data is accessible only by authorized access.
- ▶ Use configurable discovery profiles and a phased-in discovery process to control where users discover, what they discover, and how deep they go, from lightweight, credential-free discovery up to agent-less deep-dive discovery.

Rapidly deploy and share application maps across operational management products, teams, and processes through open application programming interfaces (APIs), an extensible model, a pluggable sensor framework, and a software development tool.

Services engagement overview

Implementers of a solution routinely rely on their skills and previous experiences as a guide, but there are always issues that might require educated guesswork. The goal of this section is to help you minimize the guesswork that is involved in planning and implementing a solution by providing a framework and time estimates for the major tasks.

A typical services engagement consists of these actions:

- ▶ Build an executive assessment (refer to “Executive Assessment” on page 411).
- ▶ Set up a demonstration system or proof of technology (refer to “Demonstration system setup” on page 412).

- Analyze the solution tasks (refer to “Analyze solution tasks” on page 413).
- Create a contract, commonly known as a *Statement of Work* (refer to “Creating a contract” on page 414).

We describe the representative tasks and the time involved for custom solution execution in the following section. The actual set of tasks to accomplish and the time involved can vary, because each client has a unique set of needs. However, this list can help you understand the implementation details, size the solution more accurately for the client, and ensure a profitable engagement for yourself.

It is important to work with your clients to understand their expectations. After you gather this data, document the tasks, deliverables, and associated costs in a Statement of Work. The Statement of Work acts as your contractual agreement with the client for the duration of the project; therefore, a detailed and well-defined Statement of Work is advantageous both to you and to your client.

A good overall understanding of the solution scope is a crucial prerequisite to successfully developing and implementing it. As a Solution Provider, you must understand what is involved in developing such a solution before you can discuss it with your client and size it for a cost estimate.

Executive Assessment

The Executive Assessment is a service that can be offered to prospective clients as a billable service. It offers a process that is designed to help you evaluate the business needs of a company that is planning to deploy a solution for e-business.

This toolset helps you ask the right people the right questions, so that you get the information that you need to propose the appropriate solution. The complete Executive Assessment process takes approximately 10 to 16 hours. The task breakdown is shown in Table A-1 on page 411.

Table A-1 Solution tasks

Task	Estimated time (hours)
An initial fact-finding meeting, asking questions, and gathering data	3
A review and analysis of competing solutions	2
Preparation of a set of strategic recommendations	1
Creation of a demonstration prototype	3 - 9
A presentation of findings and close for a contract	1

Task	Estimated time (hours)
Total	10-16

This is a business-case assessment, not a technical assessment, so the audience must be business owners, line-of-business executives, marketing and sales managers, and finally, the IT manager. The business owner or line-of-business executive is likely to be the decision maker.

For their initial investment, your clients get these deliverables:

- ▶ A business assessment prepared by a professional (you)
- ▶ A competitive analysis
- ▶ A prototype solution for their review
- ▶ A strategic and tactical proposal for justifying and implementing their solution for e-business

Demonstration system setup

A demonstration system is typically set up in advance to show your clients the attributes of the solution. The demonstration system can typically be set up with a limited number of systems that are separate from the system that will be used by the production system. The demonstration can be virtualized with technologies, such as Zen, VMware, or Microsoft Virtual Server.

A simple demonstration can be performed on one server that has the IBM Tivoli Application Dependency Discovery Manager Server components installed and a selection of IBM Tivoli Application Dependency Discovery Manager Discovery Sensors used.

If needing to Discover specific resources, such as applications or databases, is a key part of the engagement, it is important to get samples of these resources and show how IBM Tivoli Application Dependency Discovery Manager can be used to discover them.

The demonstration system allows your clients to evaluate whether the solution suits their particular needs.

Important: At an early stage in your client engagement, you must establish the key success criteria and areas that will enable “quick wins” for your client’s requirements. Your demonstration can then be built around this success criteria.

The tasks of demonstrating the solution and its time estimates are shown in Table A-2.

Table A-2 Solution demonstration tasks

Task	Estimated time (hours)
Set up hardware.	1 - 2
Install and configure TADDM Server components.	2 - 3
Configure Discovery.	2 - 3
Demonstrate Discovery to client.	4
Total	9 - 12

Analyze solution tasks

After the client agrees to use the solution in their environment, determine the effort that is involved in implementing the solution. Then, collect these estimates of the effort and include them in a contract or Statement of Work.

We discuss these tasks in detail in “Estimating the activities and timings of the engagement” on page 416. These tasks are our suggested tasks and order; you might complete the tasks in a different order, or you might omit or add tasks depending on the environment in which you implement the solution. The amount of skill and experience that you and your team have on the solution and also the access, which is facilitated by your client, to the necessary resources influence the overall solution timing. The assumption of the estimated timings that we present is typically based on the knowledge of:

- ▶ The operating systems
- ▶ The relational database management system (RDBMS) and database configuration and management
- ▶ IBM Tivoli Application Dependency Discovery Manager

Depending on your skills and experience, the estimates presented might be too high or too low. Table A-3 illustrates one method of approximating more realistic time estimates for your efforts that is based on whether you or your team are new to each skill area or can be considered experts. A novice represents someone who completed training in the skill area, but who has no hands-on experience. An expert represents someone who completed training in the skill area and has also implemented IBM Tivoli Application Dependency Discovery Manager projects. You can use the percentages in Table A-3 to adjust your time estimates.

Table A-3 Skill adjustment

Skill	Novice: Increase by	Expert: Reduce by
Experience of the operating system	25%	10%
Experience of RDBMS and database management	10%	10%
In-depth understanding of IBM Tivoli Application Dependency Discovery Manager	40%	20%
In-depth understanding of resource discovery techniques	30%	30%
Experience with Data Federation using IBM WebSphere Federation Server	20%	10%

For the detailed task breakdown, refer to “Estimating the activities and timings of the engagement” on page 416.

Creating a contract

A *contract* or *Statement of Work* is a binding contractual agreement between you and your client that defines the service engagement that you must perform and the result that the client can expect from the engagement. The contract must leave nothing in doubt.

A Statement of Work contains these components:

- ▶ An executive summary of the solution, which is typically a short (less than a page) summary of the solution and its benefit. You must specify any major restriction of the implementation, such as:
 - The solution is only implemented for the finance application servers.
 - The solution will be implemented in phases.

- ▶ A solution description, which contains the major components and solution building blocks that will be implemented. It must cover the conceptual architecture of the solution and the solution scope in general. This description is for technical personnel in order that they understand the implementation scope.
- ▶ Assumptions, which list all of the assumptions that are used to prepare the contract and to provide task estimation. Any deviation from the assumptions that are used will definitely impact the scope of the engagement and must be managed using the change management procedure. Typical changes include cost changes or scope changes.
- ▶ Business partner responsibilities, which list all of the responsibilities or major tasks to be performed by you or your team to implement the solution.
- ▶ Client responsibilities, which lists all of the responsibilities or items that the client must provide for you or your team in order to perform the engagement. If you cannot obtain any item in the client responsibilities, a change management procedure might be invoked.
- ▶ Staffing estimates, which list the estimated personnel that must implement the solution.
- ▶ Project schedule and milestones, which show the major steps, schedule, and achievement calendar that can be used to check the project progress.
- ▶ Testing methodology, which lists the test cases to ensure that the project implementation is successful.
- ▶ Deliverables, which provide tangible items that the client will get at the end of the service engagement, including:
 - Machine installation
 - Documentation
 - Training
- ▶ Completion criteria that list the items that are provided to the client, which indicates that the engagement is successfully completed. For most service engagements, the completion criteria are probably the most delicate items to define. It must have clear targets to which both parties agree, and it must not be too general.

A sample Statement of Work is provided in Appendix B, “Sample Statement of Work for Tivoli Application Dependency Discovery Manager” on page 421.

Estimating the activities and timings of the engagement

The fundamentals of delivering a profitable and successful services engagement include correctly identify the tasks that you must perform and adequately allocating the necessary time to perform them. In this section, we guide you through the tasks that you might need to perform to implement a Tivoli Application Dependency Discovery solution, and we estimate the timing. The estimates rely on basic assumptions, which invalidate the estimates if the items in the following list become a significant requirement for the client.

Perform environmental analysis and plan tasks

In this section, we discuss the tasks for environmental analysis engagement. Table A-4 shows the timing estimate for the major components of the tasks for the environmental analysis service.

Table A-4 Estimated time in hours for identified tasks

Task	Simple application (hours)	Complex application (hours)
Identify business objectives and project sponsor.	2	2
Gather details of Discovery and Application Dependency requirements (estimates are per application component to be discovered).	2	8
Complete design.	5	8
Communicate plan to project sponsor.	2	2
Total hours	11	20

We define a “*simple application*” to be a standard commercial application, such as an RDBMS, Web server, or a Directory Server. We define a “*complex application*” as a customized application that is developed by the client or an application that has many related subcomponents.

To help gather these technical requirements, consider these questions:

- ▶ How many servers and different server platforms are there in the environment?
- ▶ What application components exist in the data center environment?
- ▶ Where are the application components and IT resources geographically located?
- ▶ Are there any firewalls and DMZs in the environment?

Before you can derive the solution for the client scenario, you need to calculate the approximate number of Configuration Items (CIs) and the approximate number of DomainManagers that are required.

Because enterprise data center environments vary dramatically, IBM defines the concept of a *server equivalent (SE)* to normalize and present a standard set of performance and scale metrics. A representative unit of IT infrastructure, a server equivalent, is defined as a computer system with standard configurations for the operating system, network interfaces, and storage interfaces, and it also accounts for installed software, such as a database (DB2), a Web server (Apache or IPlanet), or an application server (WebSphere or WebLogic). An SE also accounts for network, storage, and other subsystems that provide services to the optimal functioning of the server. Each SE consists of 200 CIs.

As defined by the Information Technology Infrastructure Library (ITIL), a Configuration Item (CI) is any component that is under the control of Configuration Management and, therefore, subject to formal Change Control. Each CI in the Configuration Management Database (CMDB) has a persistent object and change history associated with it. Examples of a CI are a computer system, operating system, network interface, and database buffer pool size.

It is difficult to provide a standard number for the number of CIs in an SE. The actual number of CIs in an SE varies depending on the complexity of the infrastructure. For example, a complex database server with a large number of instances, databases, and tablespaces has a larger number of CIs per SE, and the number of CIs affects the overall performance.

For example, a server starts with approximately 500 CIs before discovery. A Red Hat Linux 4.0 operating system that is installed with no applications creates an extra 40 CIs. Installing DB2 Version 8.2 Fix Pack 10 creates approximately 300 CIs. Installing WebSphere Application Server 6.0 creates another 350 CIs.

The distributed architecture of TADDM is designed to scale to millions of CIs. In particular, with a scalable relational data store, such as DB2, TADDM can accommodate a large number of server equivalents.

The key considerations are:

- ▶ Discovery Techniques: Native Discovery, International Development Markup Language (IDML) Book Load, or application programming interface (API)
- ▶ Data loading time
- ▶ Number of servers
- ▶ Complexity of the servers

Plan the solution

Planning the deployment of a TADDM solution includes the subtasks that are described in the following sections.

Gather requirements

At the beginning of your engagement, you need to meet with your clients to understand their proposed objectives and to gather their requirements. First, determine the functional requirements. Functional requirements define the business functions that the monitoring system is going to provide. You determine your requirements by developing a good understanding of the business needs and a good understanding of what you hope to achieve. For example, look at issues, such as business goals, purpose, and usage questions, who the users are, and how they expect to interact. It is important to gather these requirements early and to discover any challenges that might lie ahead while the challenges can still be dealt with easily. After you determine the functional requirements, you can clarify the technical or system requirements.

The technical requirement involves spending time at the client site to determine and understand the available data sources.

Design the solution

Topics that you must address range from scalability, functionality, and the performance of this solution.

Design involves understanding the client's environment, including hardware, software, data volumes, special requirements, and operational procedures. It is necessary to identify and plan for any additional tuning of software that might be required because of the client's environment or special needs. In addition, you need to perform an analysis of the modifications that are made to the scenarios and reports. After you design the proposed solution and review it with your client, you are ready to begin the development of the offering.

Perform a gap analysis

This task might involve performing a gap analysis to give the client an estimate of the development effort that is required to set up the solution. At its core, the analysis seeks to determine what customizable components need to be extended, modified, or created. The number and complexity of customizable components drive the size of the project and the required resources.

After you design the proposed solution and review it with your client, you are ready to proceed.

Implement the solution

You implement the solution by using the tasks that are described in Table A-5 on page 420. Note that here we estimate the times to perform the activity a single time. Remember that the numbers of each item can vary, which reflects on the total time for the project. The number of applications to discover and also the amount of configuration that is required for the TADDM Server and database environment also affect the total time for the project.

Table A-5 Timeline estimates for implementation activities

Task	Estimated time (hours)
Identify servers and configure any firewalls to allow appropriate IBM Tivoli Application Dependency Discovery Manager traffic.	2
Install and configure the OS and install one TADDM Server.	8
Configure TADDM.	3-6
Install and configure Enterprise Configuration Management Database (eCMDB).	3-6
Configure anchors.	2-4
Configure Gateways.	2-4
Configure users and TADDM Security.	2-4
Integrate IBM Tivoli Application Dependency Discovery Manager with Tivoli Business Systems Manager (TBSM), IBM Tivoli Monitoring (ITM), and Tivoli Provisioning Manager (TPM).	2-6
Configure Discovery profiles and run sample discovery.	2-4
Test solution.	14
Deliver education: IBM Tivoli Application Dependency Discovery Manager for Administrators (three days)	28
Document solution.	14
Total	30-64

Close the engagement

When the technical work is complete, and the education is delivered, formally close the engagement with the project sponsor or their deputy. We suggest that you cover the following agenda items during the meeting with the project sponsor:

- ▶ Review of original business objectives.
- ▶ Summarize how the solution meets the defined objectives.
- ▶ Summarize the services delivered.
- ▶ Summarize new capabilities.
- ▶ Summarize other services or products identified during the engagement.
- ▶ Thank the sponsor and close.



Sample Statement of Work for Tivoli Application Dependency Discovery Manager

In this appendix, we provide a skeleton document that you can use for developing your own customized Statement of Work.

Building an auto-discovery and device dependency solution

The content of the Statement of Work includes activities to:

- ▶ Install the IBM Tivoli Application Dependency Discovery Manager (TADDM) component infrastructure

This task typically includes installing the TADDM Server, Enterprise Configuration Management Database (eCMDB) Server, and Configuration Management Database (CMDB).

- ▶ Install appropriate IBM Tivoli Application Dependency Discovery Manager fix pack platforms.
- ▶ Configure Discovery Anchors and Windows Gateways.
- ▶ Configure and run TADDM discoveries.
- ▶ Configure integration points with Operational Management Products.

Building the IBM Tivoli Application Dependency Discovery solution Statement of Work consists of the following sections:

- ▶ “Executive summary” on page 422
- ▶ “Solution description” on page 423
- ▶ “Assumptions” on page 424
- ▶ “Business partner responsibilities” on page 424
- ▶ “Client responsibilities” on page 425
- ▶ “Staffing estimates” on page 425
- ▶ “Testing” on page 425
- ▶ “Deliverables” on page 426
- ▶ “Completion criteria” on page 426

Executive summary

This service provides an auto-discovery solution that delivers automated application dependency mapping and configuration auditing. TADDM provides an unparalleled level of visibility into how the infrastructure actually delivers the applications upon which your business environment is based.

By complimenting application map data with other enterprise application data, such as governance, finance, and so forth, TADDM delivers integrated, meaningful information that allows you to transform your IT service management (ITSM) strategy.

This service builds the infrastructure that is required to support your IT operations personnel to ensure and improve application availability in application environments. TADDM provides the operational staff with a top-down view of applications so that the staff can quickly understand the structure, status, configuration, and change history of their business-critical applications. This top-down view enables immediate isolation of issues in times of performance or availability problems and more effective planning for nondisruptive application change.

After this work is completed, you have the infrastructure necessary to successfully view top-down, cross-tier views of how the IT infrastructure actually delivers applications. Hence, TADDM allows you to:

- ▶ Understand the structure of interdependent and complex applications.
- ▶ Rapidly isolate configuration-related application problems, which reduces troubleshooting time from hours and days to minutes.
- ▶ Better understand the impact of component-level events in order to sort issues based on application and service impact.
- ▶ More effectively plan change so that application upgrades and deployments can occur without disruptions.
- ▶ Create a shared topological definition of applications for use by other management applications, such as service level managers and provisioning tools.

Solution description

This solution builds and deploys an auto-discover solution that allows you to visualize the computing resources in your IT Infrastructure in order to plan for and react to any events that might affect the delivery of critical business services.

At the core of IT Service Management is full visibility into the IT services and the underlying infrastructure that supports these services. After all, if you have to manage the services, you need a good understanding of exactly what data makes up these services.

Although this data exists in several databases (mainly asset and inventory systems and siloed operational management product repositories), this data might not be accurate or comprehensive.

This solution builds a comprehensive repository, which:

- ▶ Can accurately and comprehensively discover all of the infrastructure that supports the application
- ▶ Has built-in automation
- ▶ Automatically discovers all of the cross-tier infrastructure and creates a top-down cross-tier map of the components that deliver the application.

We will install and configure the discovery components to gather data from one or more systems in your heterogeneous operating environment.

Assumptions

These are the assumptions that are made in this Statement of Work:

- ▶ We will have local administrator access on the servers on which the TADDM components will be installed.
- ▶ We will have administrative access to the servers on which the components to be discovered are installed.
- ▶ We will have access to Network Administrators who will be able to configure firewall ports.
- ▶ We will have details of which users need access to the TADDM environment, which must be supplied by the client.

Business partner responsibilities

This service will be provided according to the high standards of *<name of Business Partner>*, an IBM Certified Business Partner.

We will provide:

- ▶ Skilled staff to undertake the defined activities.
- ▶ Documentation of the completed solution.
- ▶ Project management of these activities.

Note: Insert any additional responsibilities here that you will be assuming as part of this project.

Client responsibilities

This section describes the responsibilities that the client has to the Business Partner, for example:

- ▶ Designating a representative who will be the focal point for all communication with the Business Partner relative to this project and who will have the authority to act on the client's behalf in matters regarding this project
- ▶ Designating operations personnel to work with the Business Partner as appropriate
- ▶ Providing all product data in the requested format
- ▶ Providing all data and information required for the implementation
- ▶ Providing a suitable workspace with Internet and telephone access for the services specialists while they are working on client premises
- ▶ Providing user IDs, passwords, and IP addresses as required, enabling the Business Partner to perform the service

Note: Add any client responsibilities that you need to assign in order to complete a successful delivery of your service.

Staffing estimates

The project will be performed with one TADDM specialist who will be on site as required by the project schedule. We will also provide project management services, and we will be on-site at the end of the project for its formal closure. The project is estimated to be performed within <x> working days. This is <x> person days of effort in total.

We expect that we will need a single member of your staff working with us throughout this time, who will also perform any mediation role that is required between any other required technical resources within your computer operation and us. This requirement is for <x> person days in total.

Note: You might want to revise these estimates if you want to provide extra services, such as education.

Testing

The testing of the solution will be done through the use of the infrastructure in order to ensure that the resources are successfully being discovered.

Testing will be completed successfully when:

- ▶ The TADDM Server installation is completed and the components are running.
- ▶ The appropriate discovery scopes have been defined, and resources are initially discovered.
- ▶ The appropriate type of discovery sensors, such as Java Management Extensions (JMX), Windows Management Interface (WMI), and Simple Network Management Protocol (SNMP), are used to obtain application, system, and network components.
- ▶ Changing the properties of an infrastructure have been identified by TADDM, and if there is an available Tivoli Enterprise Console (TEC) or OMNIBus Server, the appropriate events have been sent.

Deliverables

At the end of this engagement, you have:

- ▶ One TADDM environment with all of the required server-side software installed
- ▶ An agreed upon subset of applications and infrastructure components discovered in TADDM
- ▶ Documentation for the TADDM Server configuration and the discovered environment

Completion criteria

The completion criteria for this project is:

- ▶ The successful completion of all the tests
- ▶ The delivery of the solution documentation

Abbreviations and acronyms

API	application programming interface	ITES	IBM Technical Education Services
CCMDB	Change and Configuration Management Database	ITIL	Information Technology Infrastructure Library
CCMS	Computer Center Management System	ITSM	IT service management
CDM	common data model	ITSO	International Technical Support Organization
CI	configuration item	JDBC	Java Database Connectivity
CIM	Common Information Model	JMX	Java Management Extensions
DAS	DB2 Administration Server	JSP	JavaServer Pages
DB2	database	JVM	Java Virtual Machine
DLA	Discovery Library Adapter	LDAP	Lightweight Directory Access Protocol
DLFS	Discovery Library File Store	LOB	line of business
DNS	Domain Name Service	MAC	Media Access Control
eCMDB	Enterprise Configuration Management Database	MSS	Management Software System
ECMDB	Enterprise Domain Manager	NFS	Network File System
EJB	Enterprise JavaBeans	NIC	network interface card
GSKit	Global Security Kit	ODBC	Open Database Connectivity
HIPAA	Health Insurance Portability and Accountability Act	OMPs	Operational Management Products
IBM	International Business Machines Corporation	OPAL	Open Process Automation Library
IdML	Identity Markup Language	PKI	public key infrastructure
IDML	Identification Markup Language	SDK	software development kit
IETF	Internet Engineering Task Force	SLD	System Landscape Directory
ISA	IBM Support Assistant	SNMP	Simple Network Management Protocol
ISM	IBM Service Management	SOX	Sarbanes-Oxley Act
ISST	IBM Software Services for Tivoli	SOW	Statement of Work
IT	information technology	SSH	Secure Shell

TADDM	Tivoli Application Dependency Discovery Manager
TBSM	Tivoli Business Systems Manager
TDS	Tivoli Directory Server
TEC	Tivoli Enterprise Console
TIO	IBM Tivoli Intelligent Orchestrator
UI	user interface
UML	Unified Modeling Language
UUID	Universally Unique Identifier
WMI	Windows Management Interface

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 publications

For information about ordering these publications, refer to “How to get IBM Redbooks publications” on page 430. Note that the documents referenced here might be available in softcopy only:

- ▶ *IBM Tivoli Application Dependency Discovery Manager Capabilities and Best Practices*, SG24-7519

Online resources

These Web sites are also relevant as further information sources:

- ▶ Cygwin Web site:
<http://www.cygwin.com/>
- ▶ Cygwin User Guide:
<http://cygwin.com/cygwin-ug-net/cygwin-ug-net.html>
- ▶ TADDM Wiki site:
<http://www.ibm.com/developerworks/wikis/pages/recentlyupdated.action?key=tivoliaddm>
- ▶ TADDM documentation:
http://publib.boulder.ibm.com/infocenter/tivihelp/v10r1/index.jsp?topic=/com.ibm.taddm.doc_7.1/plugin_files/cmdb_relnotes.html
- ▶ *TADDM Planning and Installation Guide*:
http://publib.boulder.ibm.com/infocenter/tivihelp/v10r1/topic/com.ibm.taddm.doc_7.1/cmdb_install.pdf
- ▶ *TADDM User Guide*:
http://publib.boulder.ibm.com/infocenter/tivihelp/v10r1/topic/com.ibm.taddm.doc_7.1/cmdb_user.pdf

- ▶ TADDM support site:
<http://www-306.ibm.com/software/sysmgmt/products/support/IBMTivoliApplicationDependencyDiscoveryManager.html>
- ▶ Distributed Management Task Force Web site:
<http://www.dmtf.org>
- ▶ IBM Support Assistant (ISA) Web site:
<http://www.ibm.com/software/support/isa/>
- ▶ IBM Systems Journal article about IBM Service Management:
<http://www.research.ibm.com/journal/sj46-3.html>
- ▶ Microsoft WMIDiag tool:
<http://www.microsoft.com/downloads/details.aspx?familyid=d7ba3cd6-18d1-4d05-b11e-4c64192ae97d&displaylang=en>
- ▶ Microsoft Scriptomatic tool:
<http://www.microsoft.com/downloads/details.aspx?familyid=09DFC342-648B-4119-B7EB-783B0F7D1178&displaylang=en>
- ▶ *Best Practices for using TADDM Sensors* white paper:
<http://www-1.ibm.com/support/docview.wss?uid=swg27010399>
- ▶ *TADDM Best Practices for Deployment Planning* document:
<http://www.ibm.com/developerworks/wikis/display/tivoliaddm/Best+Practices+for+Deployment+Planning#BestPracticesforDeploymentPlanning->

How to get IBM Redbooks publications

You can search for, view, or download IBM Redbooks publications, Redpapers, Technotes, draft publications and additional materials, as well as order hardcopy IBM Redbooks publications, at this Web site:

ibm.com/redbooks

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

Archived

Index

A

Access
 Operating system information 373
Access list 22, 179, 371–373, 376, 380–381
Active ports 376
Add Domain 164
add or remove listeners 273
AIX system 82
 Net cost 82
aligned with business needs 5
ALTER BUFFERPOOL 360
Anchor host 75, 175–176, 182, 191, 371, 373, 380
Anchor server 73, 75, 175, 177, 181–182
 Anchor port 177
Apache 81, 388
APAR descriptions 367
applying fixes 364
architecture overview 38

B

BIRT (Business Intelligence Reporting Tool) 293
 All-in-One Package 297
 deploying 294
 designing reports 297
 introducing 294
BIRT Report Viewer 294
bufferpool 357–358
bufferpool tuning 361
bulk load program 246
Business application 62
business owner 412
business requirements 419

C

Case study 80
CCMS (Computer Center Management System) 24
cdm.log 368
Change History Report 28
Change Manager 84
Checkpoint 388
CI calculation 83, 86
Classroom Training 407

ClientProxy.log 391
closing the engagement 421
Collation.properties 348–349, 353, 369
Collation.stderr 374
COLLATION_HOME 157
Collation@ Service Account 387
common data model (CDM) 11
 attributes 11
 classes 11
 data types 11
 models 11
 relationships 11
Common Information Model (CIM) 40
Community name 380
Community string 76
Computer system 175–176, 350, 373–375
ComputerSystem class 13
ComputerSystem sub-model 41
concepts 263
config tool 189
Configuration discovery 175, 182–183, 194
 Secure environments 182
Configuration item (CI) 12, 62, 81–87
Configuration management 36, 82
 Database 36, 65
Configuration Management Database (CMDB) 23
control status 140
control.log 368
create DB2 database users 106
create roles 57
Creating a deployment plan 64
creating data sets 321
Credential-less discovery 75, 209
custom server extensions 238
custom servers 233
CustomServerSensor 206
Cygwin SSH 183

D

data model 11
 logical components 11
 physical components 11
Database server 75–76, 81–82, 86–87

- Database sizing 63
 - deep mode 63
- database sizing considerations 63
- database statistics 357
- Datacenter 70, 80, 85–86
- datastore 346
- Db2 connection 372
- DB2 settings 356
 - APP_CTL_HEAP_SZ 357
 - bufferpool settings 358
 - DBHEAP 356
 - DFT_DEGREE 356
 - LOCKLIST 357
 - LOGBUFSZ 356
 - LOGFILSIZ 357
 - LOGPRIMARY 357
 - NEWLOGPATH 357
 - NUM_IOCLEANERS 357
 - NUM_IOSERVERS 357
 - SORTHEAP 357
- decision maker 412
- deep mode 63
- defined objectives 421
- dependency discovery 207
- dependency types 207
 - Service 208
- deploying anchors 174
- deployment checklist 74
- deployment plan 64
- Deployment planning 80, 88
- deployment planning 80
- discover observer service 202
- discoveries across the firewall 175
- DiscoverManager.log 387
- DiscoverManager.log 349
- discovery 22
- Discovery Library Adapter (DLA) 263
 - book methods 276
 - book properties 276
 - concepts 263
 - configuration parameters 272
 - managing property change listeners 273
 - states 274
 - understanding APIs 270
 - using APIs 271
 - when to use 269
- Discovery Library books 263
- Discovery Library File Store (DLFS) 263
- Discovery Library XML schema specification 263

- Discovery process 175
- discovery process 23
- discovery profiles 208
- discovery scope 22, 211
- discovery sensor sequence 204
- Discovery_Rate 85
- DiscoveryLibraryAdapter class 273
- DMTF (Distributed Management Task Force) 40
- Domain manager 76, 87
- dwcount setting 349, 351

E

- eCMDB 54
- ECMDB server 63, 76, 85, 88
- efix packs 365
- EJBs (Enterprise JavaBeans) 41
- enable SSH communication 182
- engagement activities 417
- engagement timings 417
- Enterprise JavaBeans (EJB) 63
- error.log 368
- Establish a secure (SSL) session 25
- events-core.log 368

F

- File Based User Registry 156
- Firewall 71, 80, 85, 174–175, 182–183
- firewall zone 53, 182
- functional requirements 419

G

- gap analysis 420
- Gartner Group 36
- grep command 387

H

- Hanging session 370
- Hardware requirements 72
- hashmap 243

I

- IBM and ITIL 6
- IBM Certified Business Partner 426
- IBM Common Data Model (CDM) 40
 - configurations 41
 - signature 40
- IBM Common Data Model (CDM) dependencies 41

- containment dependencies 41
 - service dependencies 41
 - transactional dependencies 41
- IBM Redbooks publications 407
- IBM Service Management (ISM) 11
 - architecture 10
 - benefits 7
 - introduction 6
 - overview 8
 - TADDM 11
- IBM Support Assistant (ISA) 383
- IBM Technical Education Services (ITES) 407
- IBM Tivoli Change and Configuration Management Database Configuration Discovery and Tracking v1.1 378
- Identity Markup Language (IdML) 263
- IdML book
 - Loader 84
- idmcert.jar 290
- IIS (Internet Information Services) 388
- implement the solution 420
- improve staff awareness 5
- Info util.port scanner 371, 373, 376, 378, 381–382
- Information Technology Infrastructure Library (ITIL) 4
- installation troubleshooting 193
- installIF.sh 160
- installing
 - Cygwin SSH 183
 - DB2 Enterprise Server 92
 - installation troubleshooting 193
 - Interim Fix 0003 126
 - logs 195
 - TADDM 89
 - TADDM domain server on Linux 126
 - TADDM domain server on Windows 108
- integration overview 266
- Interim fixes 365
- interim fixpacks 365
- Inventory Report 28
- IPlanet 81
- IT Service Management initiatives 17
- IT service management strategy 16
- ITIL critical success factors 4
- ITIL Version 3 4

J

- Java 362

- Java Max memory 362
- Java SSH library 400
- Java Virtual Machine (JVM) 361
- Java Virtual Machines (JVM) 361
- javacore 362
- JavaSpace 202
- JDBC (Java Database Connectivity) 41
- JDBC connection pools 41
- JDBC connectivity 372
- JDBC resources 207
- JMS (Java Message Service) 41
- JMS topic queue 41
- JMX™(Java Management Extensions) 22
- JSPs (JavaServer Pages) 41
- JVM argument 361
- Jython scripts 243

K

- key design decisions 37

L

- l2.log 368
- lab environment 90
- LDAP (Lightweight Directory Access Protocol) 22
- less command 387
- Level 1 discovery 203
- Level 2 and 3 discovery 204
- line-of-business executive 412
- Listening ports 377
- load scopes 212
- loadidml.sh 246
- Log 382
- Log and Trace Analyzer 382
- Log Trace Analyzer 382
 - analyzing TADDM log files 384
 - importing TADDM log files 383
- logical connections 41
- LOGPRIMARY 357

M

- Management Software System (MSS) 266
- managing discoveries 230
- Microsoft Virtual Server 412
- Microsoft Windows Server 2003 70
- middleware 18
- Model Query Language (MQL) 390
- multiple domains 57

N

- Namespace 393
- naming rules 13
 - class 13
 - naming attribute 13
 - priority 13
 - superior class 13
- native discovery 83–84, 87–88
- Net cost 82
- Net cost for WAS 6.0 82
- Network Administrators 426
- Network device 175
- network router 18
- NEWLOGPATH 357

O

- OMNibus Server 428
- oopback interface 386
- open ports 182
- Operating system (OS) 82, 370
- Operating system information
 - Access 373
- Oracle Application Server 388
- Oracle database 388
- Oracle sid 372
- organization culture 5
- OS Name 374–375
- OS Path 374–375
- OS Type 374–375
- OS Version 374–375
- OutOfMemory condition 362
- OutOfMemory error 362

P

- parallel database inserts 390
- performance degradation 349
- performing environmental analysis 417
- PingSensor 79
- plan the solution 419
 - design the solution 419
 - gathering requirements 419
 - perform gap analysis 420
- planning
 - creating a deployment plan 64
 - database sizing considerations 63
 - deployment checklist 74
 - deployment planning case study 80
 - hardware and software 65

- hardware requirements 72
- planning worksheets 76
- server sizing 62
- TADD installation 62
- planning worksheets 76
- Platform release 65
- PoolSize 350
- Ports
 - Active 376
 - Listening 377
- PortScan seed 204
- PortSensor 79
- Post-processing 84
- Primary ip address 380
- prioritization 250
- Process Flow Manager service 202
- project sponsor 421
- proxy.log 368
- ps command 387

R

- reconciliation 12
- Red Hat 4.0 82
- Red Hat Enterprise Linux 5.0 67
- Red Hat ES 3.0 82
- Redbooks Web site 432
 - Contact us xxvi
- Related setting 350
- Remote Desktop 395
- reporting scenarios 293
- runid 209
- Runstat 358

S

- scope 375
- scope name 350
- Scoped Property 350
- Scriptomatic 395
- searching logs 387
- secured environments 24
- sensors 208
 - logging 208
 - setting up discoveries 209
 - understanding 208
- server equivalent (SE) 81
- Session
 - Hanging 370
- session pool wait time 349

- setting the discovery scope 211
- shallow mode 64
- Sizing 62
- Skill adjustment 415
- SLD (System Landscape Directory) 24
- SMNP MIB 380
- SNMP (Simple Network Management Protocol) 22
- SNMP community strings 22, 182
- Solaris 10 SPARC 68
- Solaris 9 SPARC 68
- span CMDB domains 57
- SQL 22
- SQL server database 388
- SQL style search queries 390
- SSH (Secure Shell) 22
- SSH access 175, 183, 372
- SSH client 183, 192
- SSH port 175, 183, 210
- SSH traffic 175
- SSH version 371, 373, 377, 381–382
- ssh-host-config program 189
- sslpasphrase 163
- StackScan sensor 75, 219
- Stackscan sensor 75
- Start Product Console 142
- State Manager 84
- Statement of Work 413, 423, 426
- storage threads 390
- strategic proposal 412
- su command 75
- SUSE Linux Enterprise Server 69
- switch 18
- Sybase db 372
- SYNC_ALL_ATTRS 64

T

- tablespace 358
- tactical proposal 412
- TADDM sample Statement of Work 423
- TADDM server 62, 71–75, 87, 174, 350, 358
 - Communicate 71
 - Data aggregation 85
 - Domain 72
- TADDM services 140
- Testhang 370
- Testjdbc 372
- Testos 373
- Testping 375

- Testportmap 376
- Testportscan 377
- Testprimaryip 380
- Testsnmp 380
- Testssh 372
- Testwmi 381
- threadcount 390
- tioadmin 372
- tiodb database 372
- tioserver 372
- Tivoli Application Dependency Discovery Manager (TADDM) 89, 175, 193–194, 209, 212, 370, 376
 - agent free discovery engine 39
 - analytics 27
 - anchor servers 52
 - API 39
 - APIs 44
 - architectural details 40
 - architecture 35
 - automatic discovery 12
 - capabilities 17
 - application mapping 17
 - data integration and federation 19
 - native discovery 17
 - reconciliation 19
 - synchronization 20
 - cdm.log 368
 - central viewing console 27
 - Change History Report 28
 - change tracking 26
 - component comparison 29
 - database 39
 - discover.log 368
 - discover-admin.log 368
 - Discovery Library Adapter 39
 - Discovery Library Adapter (DLA) 39
 - discovery process 22
 - discovery requirements 22
 - Domain Manager UI 50
 - eCMDB 54
 - entities discovered 20
 - error.log 368
 - events-core.log 368
 - features 25
 - IBM Common Data Model (CDM) 40
 - installation logs 369
 - Inventory Report 28
 - l2.log 368
 - local-anchor*.log 368

- log files 368
 - control.log 368
 - login.log 368
- our lab environment 90
- overview 16
- performance considerations 343
- problem determination tools 369
- problems addressed 16
- proxy.log 368
- reconciliation 12
- secure interface 27
- server 46
- services/ApiServer.log 368
- services/ChangeManager.log 368
- services/ClientProxy.log 368
- services/DiscoverManager.log 368
- services/DiscoverObserver.log 369
- services/MonitorStateManager.log 368
- services/ProcessFlowManager.log 369
- services/ReportsServer.log 368
- services/TopologyManager.log 368
- services/ViewManager.log 369
- sizing 62
- Statement of Work 423
 - assumptions 426
 - business partner responsibilities 426
 - completion criteria 428
 - customer responsibilities 427
 - deliverables 428
 - executive summary 424
 - solution description 425
 - staffing estimates 427
 - testing 427
- tomcat.log 368
- Topology Builder 39
- topology creation 12
- Topology Manager 39
- topology.log 368
- traffic 421
- troubleshooting 367
- user interface 47
- uses 31
- versioning 30
- Windows gateways 52
- Tivoli Application Dependency Discovery Manager (TADDM) implementation
 - available resources 407
 - client engagement 405
 - engagement preparation 406
 - services engagement overview 410
 - skills required 406
 - solution scope and components 407
 - advanced solution definition 410
 - analyze solution tasks 413
 - basic solution 409
 - creating a contract 415
 - demonstration system set up 412
 - estimating timings 417
 - executive assessment 411
 - plan the solution 419
 - Tivoli Enterprise Console (TEC) 428
 - Tivoli Open Process Automation Library (OPAL) 364
 - tomcat.log 368
 - Topology Builder 84, 202
 - Topology Manager 202
 - Topology reconciliation 62
 - topology.log 368
 - topopumpcount setting 349
 - Tracking server 175, 182–183, 194
 - Troubleshooting
 - analyzing TADDM log files 384
 - Log Trace Analyzer 382
 - sensors 387
 - specific scenarios 386

U

- unicastdiscoveryport 163
- Universally Unique Identifier (UUID) 12
- unknown servers 364

V

- View Manager 84
- viewmanager 351
- viewmanager directory 351
- VMWARE 412
- VMware 412
- VMware ESX 389

W

- wbemtest tool 395
- WebLogic 28, 81
- Weblogic Application Server 389
- WebSphere 81
- WebSphere Application Server 63, 389
- Windows gateway 74, 182–183, 371, 373, 376, 380

Discovery software 183
Windows Gateway server 398
Windows hosts 183
Windows security 391
Windows-based system 182
WMI (Windows Management Interface) 22
WMI connectivity 381
WMI Tester 392
Wmiexec 381
wmitest tool 392

X

Xml file 361

Z

Zen 412



Redbooks

IBM Tivoli Application Dependency Discovery Manager V7.1 Deployment Guide

(1.0" spine)
0.875" <-> 1.498"
460 <-> 788 pages



Deployment Guide Series: IBM Tivoli Application Dependency Discovery Manager V7.1

**Learn about TADDM
functions and
architecture**

**Get tips for installing
and using TADDM**

**Customize and tune
TADDM**

In this IBM® Redbooks® publication, we describe the capabilities and ways to use IBM Tivoli® Application Dependency Discovery Manager (TADDM). It is becoming critical for enterprises to track the IT resources in their environments and, more importantly, the dependencies of their business applications on various components. TADDM provides rich capabilities that discover the components of a complex infrastructure and their interdependencies.

In this book, we provide insight into the TADDM V7.1 capabilities and architecture. We include recommended procedures for installing and configuring TADDM, tips and techniques for populating the TADDM Database and customizing its use, and performance considerations.

Finally, we describe the sales engagement planning for TADDM V7.1, including a sample statement of work. The primary audience for this section is IBM Business Partners and pre-sales Systems Engineers working in this area.

This book is a major reference for IT Specialists and IT Architects working in TADDM V7.1 projects.

**INTERNATIONAL
TECHNICAL
SUPPORT
ORGANIZATION**

**BUILDING TECHNICAL
INFORMATION BASED ON
PRACTICAL EXPERIENCE**

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

For more information:
ibm.com/redbooks