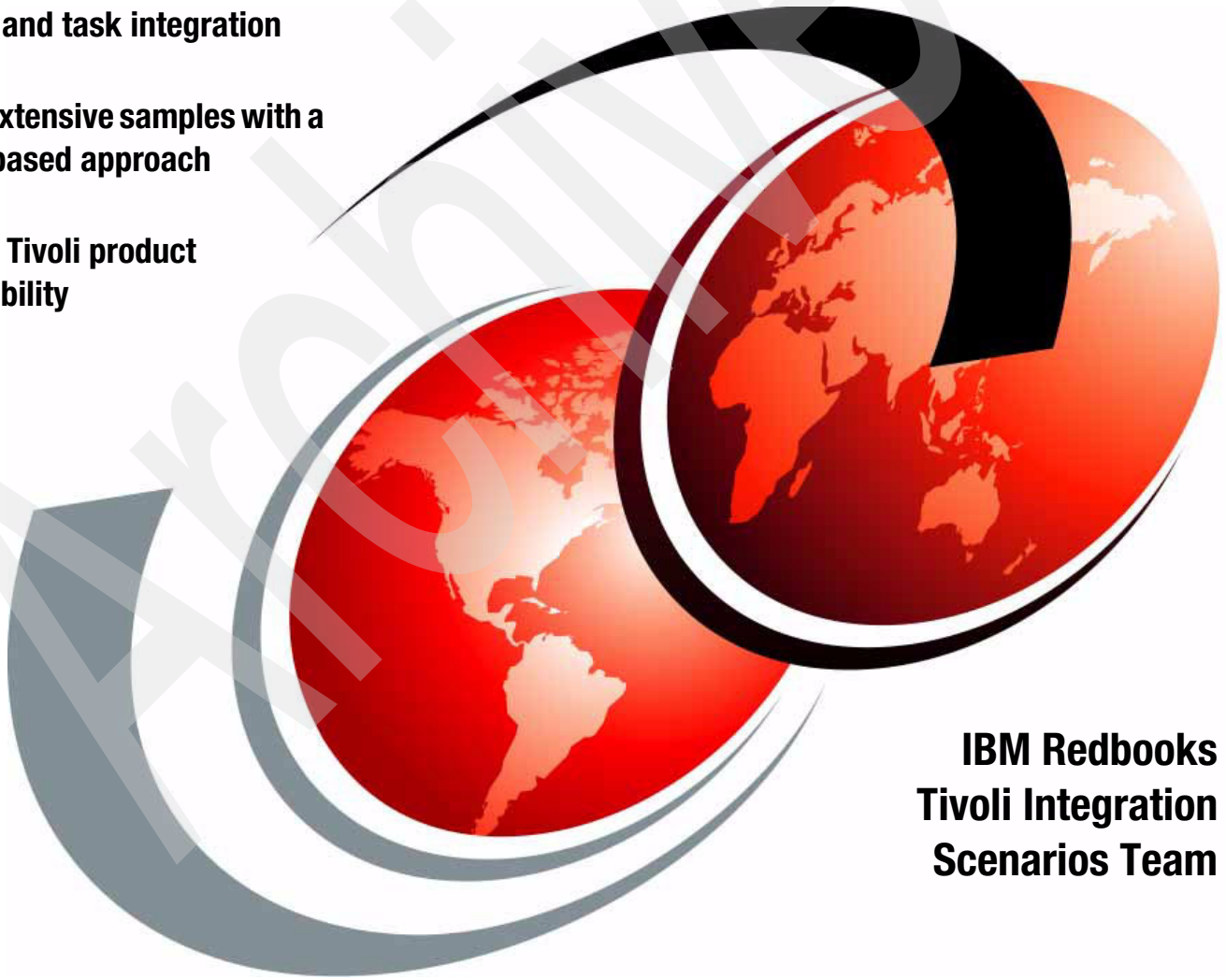
Tivoli software

IBM

Tivoli Integration Scenarios

- Describes security, data, navigation, reporting, and task integration
- Provides extensive samples with a scenario-based approach
- Discusses Tivoli product interoperability



IBM Redbooks
Tivoli Integration Scenarios Team

Archived

ibm.com/redbooks

Redbooks

Archived



International Technical Support Organization

Tivoli Integration Scenarios

January 2011

Archived

SG24-7878-01

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

Archived

Second Edition (January 2011)

This edition applies to the following product versions:

- IBM Tivoli Change and Configuration Management Database V7.2
- IBM Tivoli Asset Management for IT V7.2
- IBM Tivoli Service Request Manager V7.2
- IBM Tivoli Application Dependency Discovery Manager V7.2
- IBM Tivoli Netcool Performance Manager V1.2.1
- IBM Tivoli Service Automation Manager V7.2.1
- IBM Tivoli Provisioning Manager V7.2
- IBM Tivoli Monitoring V6.2.2.1
- IBM Tivoli Netcool/OMNIbus V7.3 and V7.2.1
- IBM Tivoli Business Service Manager V4.2.1
- IBM Tivoli Network Manager for IP V3.9
- IBM Tivoli Netcool/Impact V5.1.1
- IBM Tivoli Netcool/Webtop V2.2
- IBM Tivoli Workload Scheduler V8.5.1
- IBM Tivoli Usage and Accounting Manager V7.1.2
- IBM Tivoli Storage Productivity Center V4.1.1
- IBM Tivoli Identity Manager V5.1
- IBM Tivoli Access Manager for e-business V6.1
- IBM Tivoli Storage Productivity Center V4.1
- IBM Maximo Asset Management for Energy Optimization V7.1
- IBM Tivoli System Automation for Multiplatforms V3.2
- IBM Rational Asset Manager V7.2

© Copyright International Business Machines Corporation 2011. 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 xiii

Tables xxix

Examples xxxi

Notices xxxv
Trademarks xxxvi

Preface xxxvii
The team who wrote this book xxxvii
Now you can become a published author, too! xlvi
Comments welcome xlvi
Stay connected to IBM Redbooks xlvii

Summary of changes xlix
January 2011, Second Edition xlix

Part 1. Introduction 1

Chapter 1. Integration overview 3
1.1 Integration overview 4
1.2 Aspects of integration 5
1.3 Product coverage 6

Chapter 2. Integration scenarios 7
2.1 Overview of the scenarios 8
2.2 Common elements 12
2.3 Lab configuration 12

Part 2. Green IT and Energy Management integration scenarios 17

Chapter 3. Collect, visualize, report on energy and thermal metrics for IT, facilities, property, enterprise assets 19
3.1 Terminology 20
3.2 Scenario overview 20
3.3 Products involved 21
3.3.1 IBM Systems Director Active Energy Manager 21
3.3.2 IBM Tivoli Monitoring for Energy Management 22
3.4 Benefits 24
3.5 Architectural diagram of integration 24
3.6 Implementation steps 25
3.6.1 Installing and configuring the Tivoli Monitoring for Energy Management Active Energy Manager (AEM) agent 25
3.6.2 Installing and configuring Tivoli Monitoring for Energy Management Reporting and Optimization 28
3.6.3 Configuring Tivoli Usage and Accounting Manager for collecting Active Energy Manager information 38
3.7 Scenario walk-through 42
3.7.1 Data Center environment 42

© Copyright IBM Corp. 2011. All rights reserved.

3.7.2 Visualizing and reporting energy and thermal metrics 43
3.7.3 Visualizing and reporting energy and thermal metrics using Tivoli Monitoring for Energy Management Reporting and Optimization 54
3.7.4 Visualizing Tivoli Usage and Accounting Manager reports based on IBM Tivoli Monitoring for Energy Management 59
3.8 Summary 62

Chapter 4. Managing business service with energy and environment 63
4.1 Scenario overview 64
4.2 Products involved 65
4.3 Benefits 66
4.4 Architectural diagram of integration 68
4.5 Implementation steps 69
4.5.1 Prerequisites 70
4.5.2 Discovering IBM Tivoli Energy Management devices in Tivoli Applications Dependency Discovery Manager and creating IdML book 70
4.5.3 Configuring Energy Dashboard at Tivoli Business Manager 73
4.5.4 Importing discovered resources into Tivoli Business Service Manager 81
4.5.5 Integrating Tivoli Usage and Accounting Manager BIRT Reports into Tivoli Common Reporting engine of Tivoli Business Manager 84
4.6 Scenario walk-through 90
4.6.1 Visualizing aggregated power and thermal metrics, and launching to reports . . . 92
4.6.2 Viewing energy metrics and health status of resources 94
4.6.3 Launching to additional managed resource information 97
4.6.4 Launching to Tivoli Usage and Accounting Manager reports from Tivoli Common Reporting 98
4.7 Summary 101

Chapter 5. Green IT scenario for Asset Management 103
5.1 Scenario overview 104
5.2 Products involved 104
5.3 Benefits 105
5.4 High-level implementation steps 106
5.5 Implementing Tivoli Monitoring for Energy Management into Maximo Asset Management for Energy Optimization 106
5.5.1 Configuring IBM Tivoli Integration Composer for data connection 106
5.5.2 Configuring the mapping in Tivoli Monitoring for Energy 109
5.5.3 Configuring Tivoli Maximo Asset Management for Energy Optimization 110
5.5.4 Loading meter information from Tivoli Monitoring for Energy Management into Maximo Asset Management for Energy Optimization 112
5.5.5 Loading temperature metrics data from Tivoli Monitoring for Energy Management into Maximo Asset Management for Energy Optimization 113
5.5.6 Configuring data centers 114
5.5.7 Using data center graphical views 116
5.6 Architectural diagram of integration 118
5.7 Scenario walk-through 120
5.7.1 New assets from strategy to ready 120
5.7.2 Event Management based on Energy Management 122
5.8 Summary 124

Chapter 6. Tivoli integration scenario for Virtual Infrastructure 125
6.1 Scenario overview 126
6.2 Products involved 126
6.3 Benefits 126

6.4 Integration architecture 127
6.5 Implementation steps 128
6.5.1 IBM Tivoli Monitoring for Virtual Servers 128
6.5.2 IBM Tivoli Monitoring security integration 130
6.5.3 Tivoli Netcool/OMNIbus 134
6.6 Scenario walk-through 136
6.6.1 Creating a service in TADDM that contains a Virtual Center application 136
6.6.2 Checking for existence of service in Tivoli Business Service Manager 148
6.6.3 Showing launch to IBM Tivoli Monitoring and Tivoli Application Dependency Discovery Manager (preconfigured functionality) 150
6.6.4 Creating new virtual machine on the Virtual Center 152
6.6.5 Auto-discovering the virtual machine by using TADDM rediscovery 153
6.6.6 Accepting (adding) virtual machine to existing Virtual Center Application 157
6.6.7 Verifying that virtual machine connects to IBM Tivoli Monitoring server 159
6.6.8 Verifying this virtual machine appears in Tivoli Business Service Manager 162
6.7 Summary 165

Part 3. Tivoli Integrated Portal integration scenarios 167

Chapter 7. Operational drill down 169
7.1 Scenario overview 170
7.2 Products involved 170
7.3 Benefits 171
7.4 Scenario setup 171
7.4.1 Installing products into a single Tivoli Integrated Portal environment 172
7.4.2 Creating a custom dashboard view for network management 176
7.4.3 Integrating products to receive transactions data 183
7.4.4 Creating a custom dashboard view for transaction management 185
7.5 Sample walk-through 187
7.5.1 Scenario for transaction management 188
7.5.2 Scenario for network management 194
7.6 Summary 197

Chapter 8. Tivoli reporting integration: IBM Tivoli Monitoring and TADDM 199
8.1 Scenario overview 200
8.2 Products involved 200
8.3 Benefits 201
8.4 Architecture of the integration 202
8.5 Implementation steps 205
8.5.1 Database native client 206
8.5.2 Configuration of the products 209
8.5.3 Databases setup 210
8.5.4 Data sources definition in IBM Tivoli Common Reporting 222
8.5.5 Cross-product model update 229
8.5.6 Package import 235
8.6 Scenario walk-through 237
8.6.1 Integration test 237
8.6.2 Aggregating resources utilization per machine model and manufacturer 241
8.6.3 Aggregating resources utilization per operating system 242
8.7 Online resources 243
8.8 Summary 244

Chapter 9. Tivoli reporting integration: IBM Tivoli Monitoring and IBM Tivoli Service Request Manager 245

Archive

9.1 Scenario overview . . . 246
9.1.1 Locate and prepare data sources . . . 246
9.1.2 Build the framework model . . . 246
9.1.3 Build and publish the content . . . 247
9.1.4 Other ways of creating reports . . . 247
9.2 Products involved . . . 248
9.3 Benefits . . . 248
9.4 Creating a framework model . . . 248
9.4.1 Creating a Database View . . . 253
9.4.2 Creating a Consolidation View . . . 258
9.4.3 Integrating Tivoli Service Request Manager . . . 262
9.4.4 Publishing the framework model package . . . 276
9.5 Creating reports . . . 278
9.5.1 Server Statistics Chart report . . . 279
9.5.2 Server Incident List report . . . 285
9.5.3 Top Servers with Incident List report . . . 286
9.6 Summary . . . 290

Chapter 10. Protecting Tivoli Integrated Portal with Tivoli Access Manager for e-business . . . 291
10.1 Scenario overview and products involved . . . 292
10.2 Benefits . . . 293
10.2.1 Value of Tivoli Access Manager WebSEAL . . . 293
10.2.2 Value of Tivoli Integrated Portal . . . 294
10.2.3 Value to Tivoli Integrated Portal and WebSEAL Integrated Solution . . . 294
10.3 Integrating a Tivoli Integrated Portal environment with IBM Tivoli Access Manager WebSEAL . . . 295
10.3.1 Configuring for integration: Tivoli Integrated Portal . . . 295
10.3.2 Configuring for integration: Tivoli Access Manager WebSEAL . . . 299
10.4 Summary . . . 301

Chapter 11. WebGUI launch to IBM Tivoli Monitoring . . . 303
11.1 Scenario description . . . 304
11.2 Products involved . . . 304
11.3 Benefits . . . 304
11.4 Architectural diagram of the integration . . . 305
11.5 Implementation steps . . . 305
11.5.1 Product installations . . . 305
11.5.2 Event integration . . . 307
11.5.3 Event flow check . . . 308
11.5.4 Implementing launch-in-context integration between user consoles . . . 311
11.5.5 Single sign-on option for the products . . . 318
11.6 Scenario walk-through . . . 319
11.7 Summary . . . 322

Part 4. Cloud computing and Provisioning integration scenarios . . . 323

Chapter 12. Tivoli Service Automation Manager and Cloud Computing . . . 325
12.1 Cloud Computing overview . . . 326
12.2 Products involved . . . 326
12.3 Benefits . . . 327
12.4 Scenario overview . . . 327
12.5 Creating the service definition . . . 328
12.5.1 Using the Tivoli Service Automation Manager REST interface . . . 328

12.5.2 Understanding the request creation object . . . 329
12.5.3 Obtaining the resource pool name . . . 330
12.5.4 Obtaining the images list . . . 332
12.5.5 Creating a new catalog request . . . 334
12.5.6 Updating the request with the required information . . . 345
12.6 Monitoring the business service . . . 351
12.6.1 Selecting monitoring agents . . . 351
12.6.2 Configuring Internet Service Monitoring . . . 351
12.6.3 Configuring Transaction Tracking . . . 358
12.7 Triggering automated service delivery . . . 369
12.7.1 Detecting a service event . . . 369
12.7.2 Triggering the service instance request . . . 375
12.8 Summary . . . 378

Chapter 13. Automated Storage Provisioning . . . 379
13.1 Scenario overview . . . 380
13.2 Products involved . . . 380
13.3 Benefits . . . 380
13.3.1 Benefits of Automated Provisioning . . . 380
13.3.2 Benefits of Storage Management . . . 381
13.3.3 Benefits of an integrated end-to-end strategy . . . 381
13.4 Integration configuration steps . . . 381
13.4.1 Useful resources . . . 382
13.4.2 Integration prerequisites and roles . . . 382
13.4.3 Configuring Tivoli Storage Productivity Center . . . 383
13.4.4 Configuring Tivoli Provisioning Manager security . . . 385
13.4.5 Defining the server in Tivoli Provisioning Manager . . . 387
13.5 Scenario walk-through . . . 390
13.5.1 Solution architecture . . . 390
13.5.2 Environment prior to end-to-end provisioning workflow . . . 391
13.5.3 Triggering the end-to-end provisioning workflow . . . 393
13.5.4 Viewing the environment after the workflow has completed . . . 395
13.6 Summary . . . 398

Chapter 14. Tivoli Provisioning Manager discovery of TADDM discovered CIs . . . 399
14.1 Scenario description . . . 400
14.2 Products involved . . . 400
14.3 Benefits . . . 400
14.4 Implementation steps . . . 400
14.5 Summary . . . 404

Chapter 15. Tivoli Provisioning Manager and Tivoli Identity Manager integration . . . 405
15.1 Scenario introduction . . . 406
15.2 Benefits . . . 406
15.3 Products involved . . . 406
15.4 Architecture of the scenario . . . 406
15.5 Implementation of the scenario . . . 408
15.5.1 Tivoli Provisioning Manager automation package . . . 408
15.5.2 Tivoli Identity Manager solution . . . 409
15.5.3 Installing and configuring the Tivoli Identity Manager solution . . . 410
15.5.4 Considerations . . . 410
15.6 Scenario walk-through . . . 410
15.7 Summary . . . 412

Archived

Part 5. IBM Service Management integration scenarios 413

Chapter 16. Taking automatic actions based on predefined policies 415
16.1 Scenario overview 416
16.2 Products involved 416
16.3 Benefits 417
16.4 Architectural diagram of the integration 417
16.5 Implementation steps 418
16.5.1 Probe for Tivoli EIF 418
16.5.2 Tivoli Event Synchronization 419
16.5.3 Event forwarding to Tivoli Netcool/OMNIbus 425
16.5.4 Creating the mapping table and populating the data in CCMDB 425
16.5.5 Tivoli Netcool/OMNIbus additional fields 445
16.5.6 Configuring Tivoli Service Request Manager to initialize application server . . . 446
16.5.7 Tivoli Netcool/OMNIbus Gateway for Tivoli Service Request Manager 446
16.5.8 Tivoli Netcool/OMNIbus Process Agent control 450
16.5.9 The Tivoli Netcool/Impact project 452
16.5.10 Tivoli Netcool/Impact Enrichment Policy for Incident Ticket 456
16.5.11 CCMDB Web Services 458
16.5.12 Tivoli Netcool/Impact Web Services DSA Policy for RFC 459
16.5.13 Tivoli Netcool/Impact policy for Tivoli Provisioning Manager workflow 470
16.5.14 Tivoli Netcool/Impact event mapping 472
16.5.15 Edit Netcool/OMNIbus event list view 473
16.6 Scenario walk-through 474
16.7 Summary 476

Chapter 17. Asset and CI integration 477
17.1 Differences between assets and CIs 478
17.2 Scenario overview and products involved 479
17.3 Benefits 480
17.3.1 The challenge of controlling assets and CIs 480
17.3.2 Integration advantages 480
17.4 Case study: Fictional IBM RedBooks Company A 481
17.4.1 Linking assets to CIs using GUID 482
17.4.2 Importing CIs to the CCMDB using Tivoli Integration Composer 499
17.4.3 Activating the CI types 505
17.4.4 Setting up the actual CI adapter 506
17.4.5 Verifying that the data has been imported into CCMDB 510
17.5 Adding a Configuration Item field in the Asset application 510
17.6 Using a link rule to link assets to CIs 515
17.6.1 Specifying top-level classes for CIs and for IT assets 515
17.6.2 Specifying reconciliation properties 519
17.6.3 Configuring the reconciliation for linking CIs to assets with a link rule 522
17.6.4 Specifying the CI to reconciliation link relationship 535
17.6.5 Creating the action for the escalation for CI-to-asset link 537
17.6.6 Configuring the escalation to link CIs with IT assets 538
17.7 Summary 543

Chapter 18. Rational Asset Manager and Tivoli Change and Configuration Management Database integration 545
18.1 Rational Asset Manager 546
18.2 Definitive Media Library 546
18.3 Scenario overview 547
18.4 Products involved 548

18.5 Benefits . . . 549
18.6 Implementation . . . 549
18.6.1 Downloading and installing the integration module . . . 549
18.6.2 Creating a discovery library adapter (DLA) to import in the TADDM . . . 558
18.6.3 Loading the DLA output file into the TADDM . . . 564
18.6.4 Mapping actual CIs from TADDM to CCMDB . . . 566
18.6.5 Activating the CI types . . . 573
18.6.6 Setting up the actual CI adapter . . . 574
18.6.7 Verifying that the data has been imported into CCMDB . . . 579
18.6.8 Creating and configuring the integration module in the CCMDB . . . 580
18.6.9 Creating the Definitive Media Library (DML) . . . 587
18.6.10 Synchronizing the asset from CCMDB with Rational Asset Manager . . . 591
18.7 Summary . . . 593

Chapter 19. User and password management with Tivoli Identity Manager and Tivoli Service Request Manager . . . 595
19.1 Scenario overview . . . 596
19.2 Products involved . . . 596
19.3 Benefits . . . 597
19.3.1 Manage Tivoli Service Request Manager users . . . 597
19.3.2 Create Tivoli Service Request Manager service requests . . . 597
19.4 Installation roadmap . . . 598
19.4.1 Installing the integration package . . . 598
19.4.2 Configuring the Tivoli Service Request Manager server . . . 598
19.5 Password change scenario . . . 615
19.6 User management scenario . . . 618
19.7 Summary . . . 622

Chapter 20. Employee onboarding . . . 623
20.1 Scenario overview . . . 624
20.2 Products involved . . . 624
20.3 Benefits . . . 624
20.3.1 Paperless activities . . . 624
20.3.2 Speed up productivity . . . 624
20.3.3 Pattern for new hires . . . 624
20.3.4 Increased sense of competence . . . 625
20.3.5 Integrated platform . . . 625
20.4 Roles . . . 625
20.5 Architectural diagram of integration . . . 626
20.6 Implementation steps . . . 627
20.6.1 Human Resource (HR) Manager . . . 627
20.6.2 People Manager . . . 629
20.6.3 Service Desk Analyst . . . 632
20.6.4 IT Asset Manager . . . 633
20.6.5 Additional integration: Enterprise Asset Manager . . . 637
20.6.6 Additional integration: Security Manager . . . 638
20.6.7 Additional integration: Self-service . . . 638
20.6.8 Additional integration: HR informed that employee onboarding is complete . . . 638
20.7 Scenario walk-through . . . 639
20.8 Summary . . . 640

Chapter 21. TADDM L2 discovery using IBM Tivoli Monitoring agents . . . 641
21.1 Scenario description . . . 642
21.2 Products involved . . . 642

21.3 Architecture . . . 643
21.4 Benefits . . . 643
21.5 Implementation steps . . . 644
21.6 Scenario walk-through . . . 655
21.6.1 Deploy the TADDM discovery bundle using Tivoli Enterprise Portal . . . 655
21.6.2 View TADDM L2 Discovery data . . . 657
21.7 Summary . . . 660

Chapter 22. Managing business services in an integrated environment . . . 661
22.1 Definitions . . . 662
22.1.1 What is a service . . . 662
22.1.2 The service hierarchy . . . 662
22.2 Scenario overview and products involved . . . 663
22.3 Benefits . . . 664
22.4 The Common Data Model . . . 665
22.5 Integration and data transfer methods . . . 666
22.6 Configuring the environment for integration . . . 667
22.6.1 Exporting CIs from TADDM to CCMDB . . . 668
22.6.2 Exporting the service definition from TADDM . . . 681
22.6.3 Launch-in-context integration . . . 687
22.7 Additional integration: Integrating Tivoli System Automation Application Manager with Tivoli Business Service Manager . . . 695
22.8 Summary . . . 697

Chapter 23. CCMDB V7.2.1: New launch-in-context technology . . . 699
23.1 Scenario overview and benefits . . . 700
23.2 Components of the new LIC technology . . . 701
23.3 Naming and Reconciliation Service . . . 702
23.4 Context Menu Services . . . 703
23.5 Deployment and configuration of DIS/CMS . . . 704
23.5.1 Deployment components . . . 704
23.5.2 Options for creating the CMS and DIS tables . . . 704
23.5.3 Deployment scenarios . . . 705
23.6 CMS Registry Loader externals . . . 708
23.6.1 Reasons to use CMS Registry Loader . . . 708
23.6.2 Syntax . . . 708
23.6.3 LIC using TADDM . . . 711
23.7 Summary . . . 715

Part 6. Client integration scenarios . . . 717

Chapter 24. Tivoli Workload Scheduler: Architecture and concepts . . . 719
24.1 Components . . . 720
24.1.1 Engine . . . 720
24.1.2 Database . . . 720
24.1.3 The embedded WebSphere Application Server . . . 720
24.1.4 Event Processor . . . 722
24.1.5 Tivoli Dynamic Workload Console . . . 722
24.1.6 Dynamic Workload Broker Component . . . 723
24.2 Network communication . . . 723
24.2.1 Engine initialization . . . 724
24.2.2 Workstation interprocess communication . . . 725
24.2.3 Job execution . . . 727
24.2.4 Event driven workload automation . . . 728

24.2.5 Dynamic scheduling . . . 731
24.2.6 Store-and-forward . . . 732
24.3 High availability . . . 733
24.3.1 DB2: ACR and HADR . . . 733
24.3.2 Fault tolerance . . . 736
24.3.3 Switch Fault Tolerance . . . 738
24.3.4 Switch Manager . . . 739
24.3.5 Switch Event Processor . . . 740
24.4 Virtualization . . . 741
24.4.1 The $MASTER keyword and unixlocl method . . . 741
24.4.2 The mm resolve master . . . 742
24.4.3 FINAL stream . . . 743

Chapter 25. Tivoli System Automation for Multiplatforms: Concepts . . . 745
25.1 Overview . . . 746
25.2 Resources . . . 746
25.2.1 Persistent resource attributes . . . 746
25.2.2 Dynamic resource attributes . . . 746
25.3 Resource managers . . . 747
25.4 Resource groups . . . 748
25.4.1 Persistent resource group attributes . . . 748
25.4.2 Dynamic resource group attributes . . . 749
25.4.3 Managed resources . . . 749
25.5 Managed relations . . . 750
25.5.1 Start and stop dependencies . . . 750
25.5.2 Location dependencies . . . 751
25.6 Equivalencies . . . 752
25.6.1 Example . . . 752
25.6.2 Shadow resources and shadow equivalencies . . . 753
25.7 Quorum . . . 755
25.7.1 Tie breaker . . . 755
25.7.2 Critical resource protection . . . 756
25.8 Behavior patterns . . . 757
25.8.1 MonitorCommand . . . 757
25.8.2 StartCommand . . . 760
25.8.3 StopCommand . . . 761

Chapter 26. Implementing high availability for the Tivoli Workload Scheduler master domain manager . . . 763
26.1 Overview of the setup . . . 764
26.2 Benefits . . . 764
26.3 Installing DB2 . . . 764
26.3.1 Creating DB2 instance owner and file systems . . . 764
26.3.2 Installing and upgrading DB2 . . . 766
26.3.3 Creating DB2 instances . . . 766
26.3.4 Enabling TCP/IP communication . . . 766
26.4 Installing Tivoli Workload Scheduler . . . 767
26.4.1 Creating Tivoli Workload Scheduler user and file systems . . . 767
26.4.2 Installing the Master Domain Manager . . . 768
26.4.3 Configuring HADR and ACR . . . 771
26.4.4 Installing the Backup Master Domain Manager . . . 773
26.5 Installing and configuring Tivoli System Automation for Multiplatforms . . . 774
26.6 Integrating DB2 HADR and Tivoli System Automation for Multiplatforms . . . 775
26.7 Integrating Tivoli Workload Scheduler and Tivoli System Automation for Multiplatforms

Archived

for high availability 778
26.7.1 The embedded WebSphere Application Server 778
26.7.2 Engine 780
26.7.3 Event Processor 785
26.7.4 Tivoli Dynamic Workload Console 786
26.7.5 Dynamic Workload Broker Component 788
26.8 Testing the policies 790
26.8.1 Plan extension 790
26.8.2 Corrupt Symphony / message files 791
26.8.3 The embedded WebSphere Application Server crash situation 792
26.8.4 Node failure 792

Chapter 27. Implementing high availability for Tivoli Workload Scheduler dynamic workload 793
27.1 Overview 794
27.2 Setup 794
27.3 Benefits 795
27.4 Installing Tivoli Workload Scheduler 796
27.5 Installing and configuring Tivoli System Automation for Multiplatforms 796
27.6 Virtualizing dynamic workload 797
27.7 Testing the policies 801

Appendix A. Additional material 803
Locating the web material 803
Using the web material 804
System requirements for downloading the web material 804
How to use the web material 804

Related publications 805
IBM Redbooks 805
Online resources 805
How to get Redbooks 806
Help from IBM 807

Index 809

Archive

Figures

2-1 Lab environment for the Green IT and Energy Management integration, and IBM Service Management integration scenarios . . . 13
2-2 Lab environment for the Cloud Computing and Provisioning integration scenarios . . . 14
2-3 Lab environment for the Tivoli Integrated Portal integration scenarios . . . 15
3-1 Architectural overview scenario 1 . . . 25
3-2 AEM Agent Configuration . . . 26
3-3 AEM plug-in installer location . . . 27
3-4 AEM plug-in installer wizard . . . 27
3-5 AEM Agent in Tivoli Enterprise Portal client . . . 28
3-6 Reporting and Optimization TCR and TIP configuration settings . . . 29
3-7 Reporting and Optimization Tivoli Data Warehouse configuration settings . . . 30
3-8 Verifying the installation of Reporting and Optimization . . . 31
3-9 Summarization and pruning settings for AEM attribute group . . . 32
3-10 Historical collection settings for AEM attribute group . . . 33
3-11 Reporting and Optimization Preferences . . . 34
3-12 Refreshing the Reporting and Optimization data . . . 34
3-13 Data refresh successful . . . 35
3-14 Configuring Server Pools . . . 36
3-15 Energy Rate Plans . . . 36
3-16 Adding an energy rate plan . . . 37
3-17 Data Analyzer Settings . . . 37
3-18 Data Center Settings . . . 38
3-19 Tivoli Usage and Accounting Manager tasks list . . . 39
3-20 Tivoli Usage and Accounting Manager Data Source creation . . . 40
3-21 Tivoli Data Warehouse general properties . . . 40
3-22 JobRunner log files . . . 41
3-23 Data Center setup: Green IT scenario 1 . . . 42
3-24 Default Active Energy Manager workspace . . . 43
3-25 Create new navigator . . . 44
3-26 Create new navigator view . . . 45
3-27 Creating a new navigator item . . . 45
3-28 Creating a new navigator item . . . 46
3-29 Opening the new navigator view . . . 46
3-30 Opening the new navigator item . . . 47
3-31 Splitting a view . . . 47
3-32 Adding a plot chart view . . . 48
3-33 Assign query dialog . . . 48
3-34 View properties . . . 48
3-35 Query specification . . . 49
3-36 Defining query filters . . . 50
3-37 Filter specification . . . 50
3-38 View style properties . . . 51
3-39 CPU Plot chart completed . . . 52
3-40 Selecting an Active Energy Manager query . . . 52
3-41 Total power and thermal data of the data center . . . 53
3-42 Blade Centers power and thermal data . . . 53
3-43 Reporting and Optimization Report sets . . . 54
3-44 Reporting and Optimization Data Center Power Usage report . . . 55

© Copyright IBM Corp. 2011. All rights reserved.

3-45 Reporting and Optimization Report Parameters . . . 56
3-46 Reporting and Optimization Data Center Infrastructure Efficiency report . . . 56
3-47 DCIE and PUE relative scale . . . 57
3-48 Reporting and Optimization Standby and Power-Down Recommendations for Server Pool report . . . 58
3-49 Reporting and Optimization Data Center Power Usage Comparison . . . 59
3-50 Login to Tivoli Usage and Accounting Manager Reporting Server . . . 60
3-51 Tivoli Usage and Accounting Manager Report types . . . 60
3-52 Tivoli Usage and Accounting Manager report . . . 61
3-53 Quick summary . . . 62
4-1 Information sources for BSM dashboard with energy and environment management . 65
4-2 Where does all energy go? . . . 67
4-3 Central role of IT in the energy management for the business . . . 68
4-4 High level architecture for the scenario . . . 69
4-5 Path for Tivoli Common Reporting embedded into Tivoli Business Service Manager . 70
4-6 Data integration process between Tivoli Monitoring Services and TADDM . . . 71
4-7 Topology and details for IT resources in TADDM . . . 72
4-8 Data source connection . . . 73
4-9 Energy templates in Tivoli Business Service Manager service model . . . 75
4-10 Basic service tree for Energy Dashboard . . . 76
4-11 Data fetchers for Energy Dashboard . . . 76
4-12 Service tree template for Energy Dashboard . . . 77
4-13 Energy dashboard page structure . . . 78
4-14 Charting portlet selection . . . 78
4-15 Tivoli Charts selection . . . 79
4-16 Tivoli Monitoring for energy management web service . . . 79
4-17 Group selection of Tivoli Monitoring for Energy Management attributes . . . 80
4-18 Chart type selection . . . 80
4-19 Energy management dashboard . . . 81
4-20 BIRT Report project . . . 84
4-21 Importing from file . . . 85
4-22 Selecting the .zip file . . . 86
4-23 BIRT designer navigator structure . . . 87
4-24 Data source information . . . 88
4-25 Export reports . . . 89
4-26 Import report package in Tivoli Common Reporting . . . 89
4-27 Tivoli Usage and Accounting Manager reports into Tivoli Common Reporting . . . 90
4-28 Using energy metrics in business service model . . . 91
4-29 Energy Dashboard . . . 91
4-30 Charting portlets . . . 92
4-31 Tivoli Monitoring for Energy Management report launching from Tivoli Business Service Manager charts . . . 93
4-32 DCIE trend report . . . 94
4-33 Service tree for Energy Dashboard . . . 95
4-34 Temperature highlighted because warning threshold is reached . . . 95
4-35 Temperature and power demand thresholds defined per template . . . 96
4-36 Threshold definition for service instances . . . 96
4-37 Launching to Tivoli Enterprise Portal . . . 97
4-38 Tivoli Enterprise Portal . . . 97
4-39 Launching to Tivoli Applications Dependency Discovery Manager . . . 98
4-40 TADDM configuration item details page . . . 98
4-41 TADDM changes to a configuration item . . . 98
4-42 Tivoli Usage and Accounting Manager reports . . . 99

4-43 Tivoli Usage and Accounting Manager Top accounts charges report 99
4-44 Tivoli Usage and Accounting Manager Invoice report 100
4-45 Quick summary 101
5-1 Asset Management today 104
5-2 Asset Management improvements 105
5-3 Simplified integration diagram 106
5-4 Tivoli Integration Composer 107
5-5 Tivoli Integration Composer: Define a new data source 108
5-6 Tivoli Integration Composer: Define source data 108
5-7 Tivoli Integration Composer: Define target data 109
5-8 Tivoli Integration Composer: Mapping process 110
5-9 Maximo: Reconciliation and promotion process 111
5-10 Maximo: Cron Task configuration 112
5-11 Maximo: Cron task configuration for metric data 113
5-12 Data Center product overview 118
5-13 Component overview: Maximo for Energy Optimization 119
5-14 Maximo heat map: data center overview 120
5-15 Maximo Asset Management for Energy Optimization: heat map 121
5-16 Asset relocation 121
5-17 Maximo heat map drill down 122
5-18 BIRT report: power demand by asset 123
5-19 Quick summary 124
6-1 Architecture overview 127
6-2 Example of IBM Tivoli Monitoring for Virtual Server agent 129
6-3 Security architecture 130
6-4 Filter configuration 131
6-5 LDAP with only wasadmin user 132
6-6 Data integration architecture 133
6-7 Start Product Console 137
6-8 TADDM Product Console login window 138
6-9 Overview window of TADDM Product Console 139
6-10 Business Applications window (under Topology) 140
6-11 Create Business Application Wizard - Business Application name 141
6-12 Select the Business Application components 142
6-13 Optional details for the Business Application 143
6-14 Vmware Infrastructure Business Application visible under Topology 144
6-15 Create Business Service Wizard starting window 144
6-16 Select the components of the Business Service 145
6-17 Optional detail for the Business Service 146
6-18 VMware Infrastructure Service visible under Topology 147
6-19 VMware infrastructure application topology 148
6-20 Default window 149
6-21 Imported Business Service View 149
6-22 Service Viewer 150
6-23 Launch-in-context 150
6-24 IBM Tivoli Monitoring workspace 151
6-25 Details View of a server 152
6-26 Change History View 152
6-27 Create virtual machines 153
6-28 Starting the TADDM discovery process 154
6-29 Selecting the scope and profile for the discovery 155
6-30 Discovery in progress 155
6-31 New Virtual Machine is discovered by TADDM 156

6-32 Change events shown in Active Event List 156
6-33 Add the newly discovered virtual machine into the VMware Infrastructure Business Application 157
6-34 Newly discovered Virtual Machine is being added to the Business Application 158
6-35 Business Application Topology reflects the newly discovered VM 159
6-36 Workspace showing new machine 160
6-37 Workspace showing default agents 161
6-38 Service Component Registry 162
6-39 Launch-in-context data 163
6-40 Updated Service Tree 164
6-41 Quick summary 165
7-1 Scenario overview 170
7-2 IBM Tivoli Netcool/OMNIbus core 173
7-3 WebGUI default installation 174
7-4 Components installed 175
7-5 New template 176
7-6 Incoming status rule 177
7-7 RouterService 178
7-8 Identification fields tab 179
7-9 Service instance 180
7-10 New Page 180
7-11 Select portlets 181
7-12 NodeClickedOn source 182
7-13 Portlet target 182
7-14 Context tab 183
7-15 General settings 183
7-16 Selecting portlets 185
7-17 TBSM-ITCAM tab 186
7-18 Tivoli Charts 186
7-19 Tivoli Charts 187
7-20 CAM_Performance_Trend chart 187
7-21 Scenario for transaction management 188
7-22 Right click the failed node 189
7-23 Affecting events view 189
7-24 Response Time Critical event 190
7-25 Application Management console 191
7-26 Application Management console 191
7-27 WebSphere Application Server custom link 192
7-28 WebSphere Application Server Resources console 192
7-29 Thread pools 193
7-30 Identify the root cause of the issue 193
7-31 Scenario for network management 194
7-32 Structure View 195
7-33 Diagnostic Tools 196
7-34 Quick summary 197
8-1 Integration architecture 202
8-2 Sample cross-product report 203
8-3 Cross-product model fragment 204
8-4 Database views and tables behind the sample report 205
8-5 Sample DB2 client command-line window 208
8-6 Sample DB2 client connection to Tivoli Data Warehouse database and resources script execution 211
8-7 Sample MANAGEDSYSTEM table contents 213

8-8 Sample DB2 client connection and IBM_TRAM.db2 script execution 214
8-9 IBM_TRAM schema tables 216
8-10 Common data population utility 217
8-11 Sample population tool execution 222
8-12 IBM Tivoli Common Reporting for Asset and Performance Management navigation pane 224
8-13 Switching between Cognos applications 224
8-14 Data sources administration portlet 225
8-15 Enter data source name 225
8-16 Specify connection type 226
8-17 Enter database alias from the DB2 client catalog 226
8-18 Enter database credentials 226
8-19 Testing the connection 227
8-20 Successful connection test 228
8-21 Required data sources 228
8-22 Framework Manager destination directory 230
8-23 Opening Cognos model project 231
8-24 First login window in Framework Manager 232
8-25 Second login window in Framework Manager 232
8-26 Namespaces and query subjects which are available in the model 233
8-27 Data Sources model node 233
8-28 Schema name property 234
8-29 Model publication wizard 234
8-30 Model publication summary 235
8-31 Importing report package 236
8-32 Package entry selection 236
8-33 Tivoli Reporting Package - cross-product model 237
8-34 Launching Query Studio for cross-product package 238
8-35 Cross-product model in Query Studio 239
8-36 Sample cross-product report 240
8-37 Cross-product query details 241
8-38 CPU average utilization for particular manufacturers and machine models 242
8-39 CPU utilization for particular operating systems 243
8-40 Quick summary 244
9-1 Framework Manager: TCR authentication 248
9-2 Framework Manager: Select data source 249
9-3 Framework Manager: Select TADDM project file 249
9-4 Framework Manager: Select objects 250
9-5 Framework Manager: Main windows 251
9-6 Framework Manager: TDW data source configuration 251
9-7 TCR: Data source connections 252
9-8 Framework Manager: IBM_TRAM data source configuration 252
9-9 Framework Manager: Create Namespace 254
9-10 Framework Manager: Define Query Set 254
9-11 Framework Manager: Windows-Linux CPU Hourly Query Set 255
9-12 Framework Manager: Windows-Linux nameset and query sets 255
9-13 Framework Manager: Create relationship with Time Dimension 256
9-14 Framework Manager: Time Dimension Relationship definition 257
9-15 Framework Manager: ManagedSystem relationship 258
9-16 Framework Manager: Query Subject Definition - CPU Hourly 259
9-17 Framework Manager: Consolidation View Query Subjects 260
9-18 Framework Manager: Create shortcut 261
9-19 Framework Manager: Windows-Linux shortcut 261

9-20 Framework Manager: Renamed model and package 261
9-21 Framework Manager: Create Tivoli Service Request Manager Namespace 262
9-22 Framework Manager: Run Metadata Wizard 263
9-23 Framework Manager: Tivoli Common Reporting data sources 263
9-24 Framework Manager: New data source wizard 264
9-25 Framework Manager: Specify name and description for data source 264
9-26 Framework Manager: Select type for data source 265
9-27 Framework Manager: DB2 as data source 265
9-28 Framework Manager: Specify credentials for data source 266
9-29 Framework Manager Metadata wizard: CI 267
9-30 Framework Manager Metadata wizard: INCIDENT 268
9-31 Framework Manager: Metadata wizard Import 269
9-32 Framework Manager: Testing a query subject 270
9-33 Framework Manager: Test Result INCIDENT 270
9-34 Framework Manager: Create relationship between CI and INCIDENT 271
9-35 Framework Manager: Relationship Definition CI and INCIDENT 272
9-36 Framework Manager: Testing a relationship 273
9-37 Framework Manager: Test Result CI and INCIDENT 273
9-38 Framework Manager: Create relationship between ManagedSystem and CI 274
9-39 Framework Manager: Relationship Definition Managed System and CI 274
9-40 Framework Manager: Test results CI and ManagedSystem 275
9-41 Framework Manager: Context Explorer CI 275
9-42 Framework Manager: Test result CI and ManagedSystem 276
9-43 Framework Manager: Tivoli Service Request Manager Create Shortcut 276
9-44 Framework Manager: Tivoli Service Request Manager Incident Model shortcut . . . 277
9-45 Framework Manager: Publish Package wizard 277
9-46 Framework Manager: Verify Model warning messages 277
9-47 Tivoli Integrated Portal: Select Common Reporting 278
9-48 TCR: Launching Report Studio 278
9-49 TCR Report Studio: Select Reporting Package 279
9-50 TCR Report Studio: Create a new report 279
9-51 TCR Report Studio: Insert Chart 280
9-52 TCR Report Studio: Insert CPU metrics 280
9-53 TCR Report Studio: CPU chart 281
9-54 TCR Report Studio: Y1 Axis properties 282
9-55 TCR Report Studio: Add Filter 282
9-56 TCR Report Studio: System Name filter 283
9-57 TCR Report Studio: Server Statistics Chart prompts 284
9-58 TCR Report Studio: Block object 284
9-59 TCR Report Studio: Server Incident List unsorted 285
9-60 TCR Report Studio: Drill through parameters Incident List 286
9-61 TCR Report Studio: Parameters for Incident List report 287
9-62 Top Server Incidents List 287
9-63 Server Incident List 288
9-64 Hourly CPU metrics 288
9-65 Hourly memory metrics 289
9-66 Hourly disk metrics 289
9-67 Quick summary 290
10-1 Tivoli Integrated Portal and WebSEAL integration flow 292
10-2 Enabling Trust Association 295
10-3 Interceptors 296
10-4 Custom Properties 297
10-5 Define the properties 297

10-6 Custom property definition . . . 298
10-7 WebSEAL . . . 299
10-8 Quick summary . . . 301
11-1 Overview of IBM Tivoli Netcool/OMNIbus Web GUI integration with Tivoli Enterprise Portal . . . 305
11-2 Events listed in Tivoli Enterprise Portal console of IBM Tivoli Monitoring . . . 309
11-3 Events from IBM Tivoli Monitoring are listed in Tivoli Netcool/OMNIbus Web GUI . 310
11-4 Information Details of an event in Web GUI . . . 311
11-5 Tool Creation page at WebGUI console . . . 312
11-6 Adding new Script Tool for launch-in-context integration of the console . . . 313
11-7 Script to open Tivoli Enterprise Portal console from WebGUI of Tivoli Netcool/OMNIbus . . . 315
11-8 Menu Configuration page of Web GUI . . . 316
11-9 Update alerts menu to add new tool as new option . . . 317
11-10 Adding new tool Open_TEP to Alerts menu . . . 318
11-11 Using created tool to open Tivoli Enterprise Portal console for selected event . . . 319
11-12 Workspace in Tivoli Enterprise Portal for system related to event in IBM Tivoli Netcool/OMNIbus . . . 320
11-13 Log in to Tivoli Enterprise Portal console if SSO is not enabled . . . 321
11-14 Quick summary . . . 322
12-1 Tivoli Service Automation Manager object structures . . . 329
12-2 Source Objects definition . . . 329
12-3 List of attributes of the PMSCMR object . . . 330
12-4 List of available Web Services . . . 330
12-5 Method used to query the available resource pools . . . 331
12-6 List of Catalog requests . . . 343
12-7 Details of the request . . . 344
12-8 Items added in the request . . . 344
12-9 Required parameters for the offering . . . 345
12-10 List of Catalog Requests . . . 349
12-11 The list of parameters updated . . . 350
12-12 Example of Service Request in progress . . . 350
12-13 Accessing the ISM Configuration in the Tivoli Enterprise Portal . . . 351
12-14 Creating a new Internet Service Monitoring profile . . . 352
12-15 Adding a new HTTP monitoring element to the ISM profile . . . 352
12-16 HTTP Element Configuration for Internet Service Monitoring . . . 353
12-17 Basic configuration details for HTTP element . . . 354
12-18 Advanced configuration details for HTTP element . . . 354
12-19 Service level classification configuration details for HTTP element . . . 355
12-20 Distributing single profile . . . 356
12-21 Distribution of profiles to a selected Internet Service Monitoring agent . . . 357
12-22 Accessing the Application Management Configuration Editor . . . 358
12-23 Application Management Configuration Editor window . . . 359
12-24 Editor window with the Default profile selected . . . 360
12-25 Transactions associated with Default profile . . . 360
12-26 Create New Application . . . 361
12-27 Create New Transaction . . . 361
12-28 New transaction details for IBM HTTP Server transactions . . . 362
12-29 Filter configuration for WebSphere plug-in transactions . . . 363
12-30 Transactions configured for TradeApplication . . . 363
12-31 Transaction reporting configuration for WebSphere Plugin URLs . . . 364
12-32 Create new profile . . . 365
12-33 New profile details . . . 365

12-34 Add transaction to Transaction Tracking profile 366
12-35 Transaction selection for addition to a profile 367
12-36 Application Management Configuration Editor profile distribution. 367
12-37 Basic EIF configuration for situation 370
12-38 EIF Slot Customization 370
12-39 Select Event Class dialog for Situation EIF Slot Customization 371
12-40 Map all attributes. 371
12-41 Event class selection for situation Slow_Transactions 372
12-42 Edit procedure. 376
12-43 Quick summary 378
13-1 Cost of disks has decrease; quantity of disks needed has increased. 381
13-2 Log in to Tivoli Integrated Portal 383
13-3 Ensure single sign-on is enabled 384
13-4 Export Keys. 385
13-5 Verifying the Realm name. 386
13-6 Importing the Tivoli Storage Productivity Center Security Keys 387
13-7 Running the TPC_DefineTPCServer.wkf provisioning workflow. 388
13-8 Running Discovery of Tivoli Storage Productivity Center 389
13-9 Physical solution architecture 390
13-10 Logical solution design 391
13-11 Tivoli Storage Productivity Center Validation: No disk mapped to hosts 392
13-12 Check of the AIX system shows no SAN Volumes are available 392
13-13 External application starting the end-to-end provisioning workflow 393
13-14 Tivoli Provisioning Manager workflow triggered 394
13-15 Workflow status details 394
13-16 Disk is now allocated to system 395
13-17 AIX system has SAN-attached disk through automated provisioning processes . . 396
13-18 Validating that the Trade application has been built successfully. 397
13-19 Quick summary 398
14-1 Tivoli Application Dependency Discovery Manager Discovery Configuration. 401
14-2 Configuring the discovery 401
14-3 Submitting discovery. 403
14-4 Discovery in progress 403
14-5 Quick summary 404
15-1 Tivoli Identity Manager Tivoli Provisioning Manager Integration architecture. 407
15-2 Integration logic. 407
15-3 Integration workflow 411
15-4 Running the workflow manually 411
15-5 Quick summary 412
16-1 Architectural diagram of data integration 417
16-2 Welcome window 420
16-3 License Agreement 421
16-4 Directory Name 421
16-5 Configuration 1 422
16-6 Configuration 2 422
16-7 Tivoli Enterprise Monitoring Server Configuration. 423
16-8 Set Schedule Window1 426
16-9 Change Window Calendars 426
16-10 Create Configuration Item. 428
16-11 Create IMPACTMAP Object 429
16-12 ChangeClassification Attribute 430
16-13 incidentClassification Attribute 431
16-14 Impact Attribute. 432

16-15 Urgency Attribute 433
16-16 Ownergroup Attribute 434
16-17 Classificationid Attribute 435
16-18 Turning Admin Mode ON 436
16-19 Turning Admin Mode off 437
16-20 Create the Impactmap application 437
16-21 Drag and Drop Classification ID 438
16-22 Application Designer Control Palette 438
16-23 Edit Properties Menu 439
16-24 Set classificationid attribute. 439
16-25 IMPACTMAP. 440
16-26 New Action 441
16-27 Escalations 443
16-28 Response Plans 444
16-29 Set Workflow Action 445
16-30 CHANGERP Workflow. 445
16-31 Creating a tool in nco_config 448
16-32 Creating a trigger in nco_config 449
16-33 New Project. 452
16-34 Data source settings for MAXDB 453
16-35 Connection test. 453
16-36 Data source settings for NCOMS 454
16-37 Data type example for CLASSANCESTOR table 455
16-38 Introduction 460
16-39 WSDL file and Client Stub 461
16-40 Web Service name, port, and method 462
16-41 Web Service method parameters 463
16-42 Web Service method parameters 464
16-43 Web Service method parameters 465
16-44 Web Service End Point 466
16-45 Summary and Finish 467
16-46 Event Mapping 472
16-47 The nco_event default view 473
16-48 Open Situations in Tivoli Enterprise Portal 474
16-49 Processed Events in Tivoli Netcool/OMNIbus. 474
16-50 Automatic Incident Ticket 475
16-51 Automatic Request for Change. 475
16-52 Acknowledged Situations in Tivoli Enterprise Portal. 476
16-53 Quick summary 476
17-1 Life cycle of an asset and of a CI 479
17-2 Integration overview 480
17-3 IBM RedBooks Company A topology 482
17-4 Accessing Link Rules application 483
17-5 New Link Rule. 484
17-6 Link Rules window 484
17-7 Link Clauses 485
17-8 Data Set 1 Configuration. 485
17-9 Link Clauses configuration 486
17-10 Adding a new condition. 487
17-11 Saving the Link Rule 487
17-12 Reconciliation Task application. 488
17-13 New Reconciliation Task. 488
17-14 Setting the Data Set 489

17-15 Selecting a Link Rule 489
17-16 Set the Schedule 490
17-17 Checking the Cron Task Instance 491
17-18 Reload Request window 491
17-19 Click OK to Reload the Request 492
17-20 Acessing the Database Configuration application 492
17-21 Object list 493
17-22 Database Configuration Relationships window 493
17-23 Where Clause in Database Configuration 493
17-24 Actions application 494
17-25 Actions application 495
17-26 Selecting Parameter / Attribute 496
17-27 Escalation Application 497
17-28 Configuring the Escalation 498
17-29 Escalation Points window 498
17-30 Actions tab 499
17-31 Activate the Escalation 499
17-32 Tivoli Integration Composer menu 500
17-33 New data source 500
17-34 New data source 501
17-35 New data source 502
17-36 New mapping 502
17-37 Test succeeded 503
17-38 Test Connection 504
17-39 CI Types 505
17-40 CI Types 506
17-41 Mapping 506
17-42 New Mapping 507
17-43 Source data source 507
17-44 Target data source 508
17-45 Column expressions 509
17-46 Actual Configuration Items 510
17-47 Acessing the Application Designer application 511
17-48 Asset application 511
17-49 Acessing the Control Palette 512
17-50 Drag Multipart Textbox 512
17-51 Check the properties from the Multipart Textbox 513
17-52 Multipart Textbox properties 513
17-53 Save Application Definition 514
17-54 Configuration Item field created 514
17-55 Organizations Application 516
17-56 System Settings 516
17-57 System Settings window 517
17-58 IT Asset Top-Level Classification 518
17-59 Authorized CI Top-Level Class 518
17-60 Authorized CI Top-Level Classification 519
17-61 System Properties application 519
17-62 Global Properties 520
17-63 Global Value 520
17-64 Live Refresh 521
17-65 Live Refresh window 521
17-66 Check current value 522
17-67 Link Rules application 523

Archive

17-68 Click New Link Rule 523
17-69 Link Rules 524
17-70 Setting the Type of Reconciliation 524
17-71 Select CI 525
17-72 Click New Row in Link Rules application 526
17-73 Link Clauses 527
17-74 Click Select Value 527
17-75 Select value for the Data Set 2 Class 528
17-76 Select Value 528
17-77 Select Value 529
17-78 Saving the Link Rule created 529
17-79 Reconciliation Tasks application 530
17-80 New Reconciliation Task 530
17-81 Reconciliation Task name 531
17-82 Check the Type of Reconciliation topic 531
17-83 Select Link Rule 532
17-84 Set a Schedule 532
17-85 Reconciliation Task cron task 533
17-86 Cron Task Instance 533
17-87 Reloading the request 534
17-88 Click OK 534
17-89 Database Configuration application 535
17-90 Select CI 535
17-91 Save the Object 536
17-92 Actions application 537
17-93 New action 537
17-94 Actions application 538
17-95 Parameter / Attribute field from the Action 538
17-96 Acessing Escalations application 539
17-97 Select CI 540
17-98 Condition of the Escalation 540
17-99 Setting the Schedule 541
17-100 Check the Repeat 541
17-101 Click Save 542
17-102 Activate the Escalation 542
17-103 Quick summary 543
18-1 Definitive Media Library 547
18-2 Scenario architecture 547
18-3 RAM Home window 549
18-4 RAM Extensions 550
18-5 The .zip files for integrating with CCMDB 550
18-6 Language window 552
18-7 Introduction window 552
18-8 Choose PSI Package 553
18-9 Validating Package window 553
18-10 Package Validation Results 554
18-11 Software license agreement 554
18-12 Adding features 555
18-13 Login information 556
18-14 Pre-Install Summary 556
18-15 Package deployment results 557
18-16 Maximo System Information 557
18-17 Administration button 558

18-18 Rational community 559
18-19 Connections tab 559
18-20 Create a CCMDB connection 560
18-21 CCMDB Connection Properties 561
18-22 Assets tab 561
18-23 Asset State 562
18-24 View Search Shortcut 562
18-25 Synchronization schedule of CCMDB Connection 563
18-26 Accessing MSS from TADDM 564
18-27 MSS list 565
18-28 MSS list 565
18-29 Configuration Item List 565
18-30 New Data Source on Tivoli Integration Composer 566
18-31 New Data Source 566
18-32 Define the Data Source name 567
18-33 Data Source connection information 568
18-34 Test connection 568
18-35 Creating a new Mapping 569
18-36 New mapping 569
18-37 Test the connection 570
18-38 Connection Information 571
18-39 Accessing the CI Types application 573
18-40 Changing status 574
18-41 Creating a new mapping 574
18-42 Mapping name 575
18-43 Connection information 575
18-44 Connection information 576
18-45 Expressions 577
18-46 Actual Configuration Items 579
18-47 Imported assets 579
18-48 Web Services Library 580
18-49 Create a Web Service from Object Structure 580
18-50 Web services 581
18-51 Select Records 582
18-52 Deploy Web Service 582
18-53 Opening Integration Modules application 583
18-54 IMPORTSOFTWAREIMAGE operation is added 583
18-55 Select the OMP 584
18-56 Go To End Points 584
18-57 Select RAMHANDLER 585
18-58 Extensions page from RAM 585
18-59 Extensions page 585
18-60 End Points page 586
18-61 Rational Asset Manager repository administrator 586
18-62 Create the DML 587
18-63 Owner field 587
18-64 Details section 587
18-65 Classify window 588
18-66 Scope section 588
18-67 Rational Asset Manager OMP 589
18-68 Select from OMP 589
18-69 List of the imported software images 590
18-70 RAMCCMDB community 591

18-71 Change and Configuration Management Database Connections 591
18-72 Synchronize 592
18-73 Quick summary 593
19-1 Architecture of the integration 596
19-2 System Properties 599
19-3 Live Refresh button 600
19-4 Create WS from Object Structure 600
19-5 Create Web Service from an Object Structure Service Definition window 601
19-6 Web Services Library 601
19-7 Users menu 602
19-8 Enable Login Tracking 602
19-9 buildmaximoear.cmd 604
19-10 buildmaximoear.cmd 604
19-11 Enterprise Applications window 605
19-12 Remote File System 605
19-13 Browse to the location of the maximo.ear 606
19-14 elect installation options window 607
19-15 Map modules to servers window 608
19-16 Summary page 609
19-17 Save to Master Configuration 610
19-18 Classpath entry 611
19-19 Define Operation 613
19-20 maximoserviceprofile.jar 614
19-21 Create Service window 614
19-22 Tivoli Identity Manager Self Service Console 615
19-23 Change Password option 616
19-24 Change Password window 617
19-25 Request Submitted 617
19-26 View My Requests 618
19-27 Service Requests window 618
19-28 Tivoli Identity Manager Self Service console 619
19-29 Request Account page 620
19-30 SRMTIMIntegration service t 620
19-31 Request Account window 621
19-32 View My Requests 621
19-33 Quick summary 622
20-1 Roles 625
20-2 Architectural diagram of integration 626
20-3 HR record creation 627
20-4 Cron Task Parameters 628
20-5 Offering Catalog link on Startcenter 629
20-6 Create Account window 630
20-7 Order submitted 630
20-8 Change and reset password 631
20-9 Requesting a notebook 631
20-10 Notebook offering 632
20-11 New requests are available 632
20-12 Detailed view of the catalog order 633
20-13 Work order 633
20-14 Job plan 634
20-15 Checking Material (hardware and software) availability 634
20-16 Assign license to the hardware 635
20-17 Licenses application 636

20-18 Asset application 636
20-19 User association with the notebook 637
20-20 Quick summary 640
21-1 Implementation architecture 643
21-2 TADDM Login Console 650
21-3 Click Discovery Profiles 650
21-4 Create a new discovery profile 651
21-5 Create New Profile 651
21-6 New ITM Scope Sensor 652
21-7 View ITM Scope Sensor created in the Discovery Profiles View 653
21-8 Create a TADDM Scope set containing Tivoli Monitoring Portal Server details 653
21-9 Create an Access List 654
21-10 Add managed system 655
21-11 Select the monitoring agent bundle 656
21-12 Tivoli Monitoring Deployment Status by Product Workspace 656
21-13 Run Discovery 657
21-14 Discovery in progress 657
21-15 TADDM Discovered CI 658
21-16 TADDM L1 discovery data for a sample server 658
21-17 TADDM L2 discovery data using Tivoli Monitoring Agents for the sample server . 659
21-18 Quick summary 660
22-1 TADDM business service supported by a hierarchy of IT services 663
22-2 Functions and relations between tools 664
22-3 Primary source for service definition; how service is monitored and controlled 665
22-4 Integration tools and methods 666
22-5 Tivoli Integration Composer main menu 669
22-6 Defining a new data schema 669
22-7 Connection information for the new data source 670
22-8 Importing the data schema 670
22-9 The .schm file 670
22-10 The data schema analysis 671
22-11 Final analysis report 671
22-12 Configuring the TADDM data source 672
22-13 Source and target mapping 672
22-14 Importing the mapping 673
22-15 Mapping file between TADDM and CCMDB 673
22-16 Database connection information 674
22-17 Importing the data schema 674
22-18 Data schema file 675
22-19 The data schema analysis 675
22-20 TADDM connection parameters 676
22-21 Source and target for the data mapping 676
22-22 Importing the mapping between TADDM and CCMDB 676
22-23 The mapping file import 677
22-24 Directory contents 677
22-25 Running the commandLine.bat without parameters 678
22-26 The CI types list 678
22-27 The integration composer directory 679
22-28 The completion of the mapping between TADDM and CCMDB 680
22-29 Initial installation window 681
22-30 The selection of the XML toolkit data source 682
22-31 TADDM connection configuration 683
22-32 TADDM database configurations 684

22-33 XML toolkit directory and TIP host name . . . 685
22-34 The DL_TADDM_HostName parameter . . . 686
22-35 The service hierarchy directory . . . 686
22-36 Directions for the launch-in-context functionality . . . 687
22-37 The launch menu in Tivoli Business Service Manager . . . 691
22-38 The launch-in-context configuration . . . 692
22-39 Configuration Item application signature options . . . 693
22-40 Select Action menu . . . 694
22-41 The signature option security configuration . . . 695
22-42 Quick summary . . . 697
23-1 Navigation across products . . . 701
23-2 LIC components . . . 702
23-3 Naming . . . 702
23-4 Reconciliation . . . 703
23-5 Details Menu of the Actual Configuration Item . . . 712
23-6 TADDM authentication window . . . 713
23-7 Change history . . . 713
23-8 View Details panel . . . 714
23-9 Quick summary . . . 715
24-1 Engine initialization . . . 725
24-2 Workstation interprocess communication . . . 726
24-3 Job execution details . . . 727
24-4 Event Driven Workload Automation: deployment process . . . 728
24-5 Tivoli Workload Scheduler Object Monitor . . . 730
24-6 File Monitor . . . 730
24-7 Dynamic scheduling . . . 731
24-8 Store-and-forward Event Driven Workload Automation . . . 732
24-9 Store-and-forward Tivoli Workload Scheduler . . . 733
24-10 DB2 ACR . . . 734
24-11 DB2 HADR synchronization modes . . . 735
24-12 Fault tolerance I . . . 737
24-13 Fault tolerance II . . . 738
24-14 Switch Fault Tolerance . . . 739
24-15 Switch Event Processor . . . 741
24-16 unixlocl access method . . . 742
25-1 Resource managers . . . 748
25-2 Managed Resources . . . 750
25-3 StartAfter relationship . . . 751
25-4 Collocated / Affinity relationship . . . 752
25-5 Automation problem: incorrect solution . . . 753
25-6 Automation problem: correct solution . . . 754
25-7 Quorum . . . 755
25-8 Network tie breaker . . . 756
25-9 Behavior patterns . . . 757
25-10 StartCommand state diagram . . . 761
25-11 StopCommand state diagram . . . 762
26-1 DB2 file system structure . . . 765
26-2 Tivoli Workload Scheduler file system structure . . . 767
26-3 Tivoli Workload Scheduler setup, 1 . . . 768
26-4 Tivoli Workload Scheduler setup, 2 . . . 769
26-5 Tivoli Workload Scheduler setup, 3 . . . 769
26-6 Tivoli Workload Scheduler setup, 4 . . . 769
26-7 Tivoli Workload Scheduler setup, 5 . . . 770

26-8 Tivoli Workload Scheduler setup, 6 770
26-9 Tivoli Workload Scheduler setup, 7 773
26-10 Tivoli Workload Scheduler setup, 8 774
26-11 DB2 and embedded WebSphere Application Server setup 780
26-12 FTA and MDM setup 785
26-13 Event Processor setup 786
26-14 Dynamic Workload Broker Component dependency 789
26-15 Dynamic Workload Broker Component connection in Tivoli Dynamic Workload Console 790
27-1 Job execution: physical view 794
27-2 Job execution: logical view 794
27-3 Tivoli Workload Automation instances 795
27-4 Split-brain 795
27-5 Logical resource dependency 800

Archived

Tables

2-1 Green IT and Energy Management scenarios covered in this book 9
2-2 Tivoli Integrated Portal scenarios covered in this book 9
2-3 Cloud Computing and Provisioning scenarios covered in this book 10
2-4 IBM Service Management scenarios covered in this book 10
2-5 Scenarios implemented by IBM clients covered in this book 11
3-1 Green IT terminology 20
4-1 Common data model classes to Tivoli Business Service Manager templates 83
8-1 Windows agent columns mapping on ComputerSystem table 218
8-2 Linux agent columns mapping on ComputerSystem table 219
8-3 UNIX agent columns mapping on ComputerSystem table 219
8-4 TADDM database columns mapping on ComputerSystem table 220
12-1 Object Parameters 335
12-2 information about the attributes 347
13-1 Workflow parameters 388
16-1 Overview of event mapping 416
16-2 Configuration Item list 428
16-3 Impact mapping 441
16-4 Actions 442
16-5 Data types 455
16-6 Creating the Links 456
16-7 Event mapping settings 472
16-8 Mapping of events 474
19-1 File and directory information 598
20-1 Scenario walk-through 639
21-1 Supported operating systems 644
22-1 Tools and their usage in the context of managing services 663
22-2 Signature options configurations 693
22-3 Select Action configurations 693
23-1 Supported actions 709
23-2 Supported parameters 709
25-1 Dynamic resource attributes 747
25-2 OpState dynamic resource group attribute 749
25-3 OpState change App1: Offline 758
25-4 OpState change App1: Failed Offline 758
25-5 OpState change App2: Offline 759
25-6 OpState change App2: Failed Offline 759
27-1 Files and chapter where used 804

© Copyright IBM Corp. 2011. All rights reserved.

Archived

Examples

5-1 Tivoli Integration Composer: Windows command line execution 111
6-1 Original wimconfig.xml configuration file 132
6-2 New wimconfig.xml configuration file 132
7-1 Set the environment variables 173
8-1 Creating a new user for database administration 207
8-2 Creating DB2 instance 207
8-3 Adding Tivoli Data Warehouse to the client's catalog 208
8-4 Adding TADDM database to the client's catalog 209
8-5 Establishing connection to Tivoli Data Warehouse database 212
8-6 Creating MANAGEDSYSTEM table 212
8-7 Creating population procedure 212
8-8 MANAGEDSYSTEM table population 213
8-9 Establishing a connection to Tivoli Data Warehouse database 214
8-10 Creating IBM_TRAM schema and tables 215
8-11 Creating population procedure 215
8-12 TIME_DIMENSION table population 216
8-13 Sample JDBC configuration 221
8-14 Executing the Java class manually 222
11-1 Single Tivoli Enterprise Portal Server: TEPS and TEMS on the same server 313
11-2 Single Tivoli Enterprise Portal Server: TEPS and TEMS on separate servers 314
11-3 Multiple Tivoli Enterprise Portal Server servers 314
12-1 Request XML 331
12-2 Response XML 331
12-3 URL used to obtain the images list 332
12-4 Images list 332
12-5 URL to create the Catalog Request 334
12-6 Catalog request response, in json format 336
12-7 Updating the request with the required information 346
12-8 HTTP response 348
12-9 Execute the commands 373
12-10 Additional set of lines 374
12-11 Basic logic 377
16-1 PROBE_EIF.props example file 419
16-2 NCOMS.props example for PA credentials 423
16-3 errorevent.conf file example for EIF probe 424
16-4 Testing connectivity 424
16-5 Event Integration Facility settings 425
16-6 impactFields.sql file 445
16-7 Running the ImpactFields.sql file 446
16-8 GATE_TSRM.props 447
16-9 The tsrm.map example file 449
16-10 The nco_pa.conf example file 450
16-11 TSRM_Incident policy 456
16-12 Changes in wsdl and xsd files 459
16-13 Beginning of policy 468
16-14 End of policy 468
16-15 CCMDB_RFC policy 468
16-16 The soapcli.sh file 470

© Copyright IBM Corp. 2011. All rights reserved.

16-17 Workflow policy . . . 471
17-1 Error message . . . 504
17-2 Command to run . . . 509
18-1 Back up maxdb71 . . . 551
18-2 Backup the registry . . . 551
18-3 solutionInstallerGU . . . 551
18-4 loadidml -f . . . 564
18-5 Run commandline.bat . . . 571
18-6 Run commandline . . . 577
19-1 Modify login.jsp . . . 603
19-2 The enRole.properties file . . . 612
19-3 Add this line to the scriptframework.properties file . . . 612
19-4 Text to be added . . . 612
21-1 Log in to Tivoli Enterprise Monitoring Server . . . 645
21-2 Add TADDM bundle . . . 646
21-3 Distribute the TADDM support bundle . . . 648
21-4 Check deployment status . . . 649
22-1 canvasDynamicSubMenuActions.xml . . . 688
22-2 Add those lines . . . 689
22-3 Updated file . . . 689
22-4 Add these lines . . . 689
22-5 Section to look for . . . 690
22-6 Add these lines . . . 690
24-1 Embedded WebSphere Application Server connection properties in localopts . . . 721
24-2 The embedded WebSphere Application Server and Event Processor state . . . 721
24-3 Virtual Master . . . 743
24-4 FINAL jobs . . . 743
24-5 FINAL stream . . . 744
25-1 Quorum State . . . 755
25-2 Critical resource protection . . . 756
25-3 RetryCount tunable . . . 760
26-1 DB2 instance owner and file system creation . . . 765
26-2 Instance owner and file system creation . . . 766
26-3 DB2 instance creation . . . 766
26-4 DB2 instance level verification . . . 766
26-5 Enabling TCP/IP communication . . . 767
26-6 Verifying TCP/IP communication . . . 767
26-7 Tivoli Workload Scheduler user and file system creation . . . 768
26-8 Stopping embedded WebSphere Application Server instance on twspoc1 . . . 770
26-9 Updating location of log files and archived log files . . . 771
26-10 Enabling LOGINDEXBUILD . . . 771
26-11 Backing up DB2 database . . . 771
26-12 Restoring DB2 database . . . 771
26-13 Configuring HADR on twspoc2 . . . 771
26-14 Configuring ACR on twspoc2 . . . 772
26-15 Starting HADR on twspoc2 . . . 772
26-16 Configuring HADR on twspoc1 . . . 772
26-17 Configuring ACR on twspoc1 . . . 772
26-18 Starting HADR on twspoc1 . . . 772
26-19 Verifying HADR status . . . 773
26-20 Performing HADR takeover on twspoc2 . . . 773
26-21 Stopping embedded WebSphere Application Server instance on twspoc2 . . . 774
26-22 Performing HADR takeover on twspoc1 . . . 774

26-23 Tivoli System Automation for Multiplatforms installation 774
26-24 Installing HADR scripts for Tivoli System Automation for Multiplatforms 775
26-25 Creating and starting the Tivoli System Automation for Multiplatforms cluster 775
26-26 Verifying the state of the Tivoli System Automation for Multiplatforms cluster 775
26-27 Creating network tie breaker for the cluster 775
26-28 XML input file for db2haicu 776
26-29 Verifying HADR standby node 776
26-30 The db2haicu setup 776
26-31 Verifying HADR primary node 776
26-32 The db2haicu setup 777
26-33 Output of lssam command 777
26-34 Updating DataSourceProperties, 1 778
26-35 Updating DataSourceProperties, 2 778
26-36 Creating resources and resource groups for the eWAS instances 779
26-37 Creating managed relations for the eWAS instances 779
26-38 Creating shadow resources and shadow equivalency for the eWAS instances . . . 779
26-39 Starting eWAS resource groups 779
26-40 Output of lssam command 779
26-41 Enabling Switch Fault Tolerance 780
26-42 Creating the virtual master workstation 780
26-43 Creating CPU definition for the backup master domain manager 781
26-44 Adding the FINAL stream to the database 781
26-45 Updating JnextPlan script 782
26-46 Setting the limit for the workstations 782
26-47 Creating resources and resource groups for the engine processes 782
26-48 Creating managed relations for the engine processes 783
26-49 Creating shadow resources and shadow equivalency for mailman processes 783
26-50 Starting resource groups for engine processes 783
26-51 Creating resources and resource groups for the master domain manager 783
26-52 Creating virtual IP address for the master domain manager 783
26-53 Creating managed relations for the master domain manager 784
26-54 Creating least-privilege resource 784
26-55 Starting resource group for the master domain manager 784
26-56 lssam: Tivoli Workload Scheduler engine 784
26-57 Creating floating resource and resource group for the Event Processor 785
26-58 Starting resource group for the Event Processor 785
26-59 The lssam command output 786
26-60 Creating resources for the Tivoli Dynamic Workload Console 787
26-61 Creating managed relations for the Tivoli Dynamic Workload Console 787
26-62 Starting resource group for the Tivoli Dynamic Workload Console 787
26-63 lssam: Tivoli Dynamic Workload Console 787
26-64 BrokerWorkstation.properties on twspoc1 788
26-65 BrokerWorkstation.properties on twspoc2 788
26-66 Defining the broker workstation 788
26-67 Creating resources for the Dynamic Workload Broker Component 789
26-68 Starting resource group for the Dynamic Workload Broker Component 789
26-69 lssam: Dynamic Workload Broker 789
26-70 lssam: JnextPlan, 1 790
26-71 lssam: JnextPlan, 2 791
26-72 Filling the Tivoli Workload Scheduler file system 791
26-73 lssam: Corrupt Symphony/message files 791
26-74 lssam: eWAS crash 792
27-1 Installing and upgrading the Tivoli Workload Scheduler agent 796

27-2 Shutting down Tivoli Workload Scheduler agent 796
27-3 Tivoli System Automation for Multiplatforms installation 796
27-4 Creating and starting the Tivoli System Automation for Multiplatforms cluster 797
27-5 Verifying the state of the Tivoli System Automation for Multiplatforms cluster 797
27-6 Creating network tie breaker for the cluster 797
27-7 Updating CLI connection properties 797
27-8 Raising TCP connection timeout interval 798
27-9 Creating Dynamic Workload Broker Component logical resource 798
27-10 Creating generic JSDL job definition 798
27-11 Creating Tivoli Workload Scheduler pool workstation 799
27-12 Creating resources for the Dynamic Workload Broker Component clients 799
27-13 Starting resource groups for Dynamic Workload Broker Component clients 799
27-14 Creating shadow resources for Dynamic Workload Broker Component clients . . . 799
27-15 Creating resource for Dynamic Workload Broker Component logical resource . . . 800
27-16 Starting the resource group for the Dynamic Workload Broker Component logical resource 800
27-17 lssam: Dynamic Workload Broker Component clients 800
27-18 HA job definition 801
27-19 HA job run 801
27-20 lssam: failure of fta1 801
27-21 HA job run 802

Archive

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.

© Copyright IBM Corp. 2011. All rights reserved.

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®
BladeCenter®
Cognos®
DB2 Universal Database™
DB2®
Dynamic Infrastructure®
Global Business Services®
IBM Systems Director Active Energy Manager™
IBM®
Lotus Notes®
Lotus®
Maximo®
Netcool®
NetView®
Notes®
Passport Advantage®
POWER6®
POWER®
Proviso®
Rational®
Redbooks®
Redpaper™
Redbooks (logo) ®
Service Request Manager®
Symphony™
System x®
System z®
Tivoli Enterprise Console®
Tivoli®
TotalStorage®
WebSphere®
z/OS®
z/VM®

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.

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.

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.

The following company name appearing in this publication is fictitious:

Fictional IBM RedBooks Company A (IBM Redbooks Company A)

This name is used for instructional purposes only.

Archived

Preface

This IBM® Redbooks® publication provides a broad view of how Tivoli® system management products work together in several common scenarios. You must achieve seamless integration for operations personnel to work with the solution. This integration is necessary to ensure that the product can be used easily by the users.

Product integration contains multiple dimensions, such as security, navigation, data and task integrations. Within the context of the scenarios in this book, you see examples of these integrations.

The scenarios implemented in this book are largely based on the input from the integration team, and several clients using IBM products. We based these scenarios on common real-life examples that IT operations often have to deal with. Of course, these scenarios are only a small subset of the possible integration scenarios that can be accomplished by the Tivoli products, but they were chosen to be representative of the integration possibilities using the Tivoli products.

We discuss these implementations and benefits that are realized by these integrations, and also provide sample scenarios of how these integrations work.

This book is a reference guide for IT architects and IT specialists working on integrating Tivoli products in real-life environments.

The team who wrote this book

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



Andre Ricardo Cavalcanti de Araujo is a System Management Information Technology Architect working with Tivoli Management Product in Brazil. He has 13 years of experience in servers and systems support. Andre is an IBM Certified Advanced Deployment Professional. He holds a degree in Telecommunication Engineering and has an MBA in Network Computer Management. His areas of expertise include UNIX/Linux® support, Cisco networking, networking security, and infrastructure and networking management.

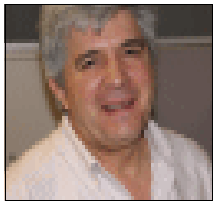


Antonio Orlando Neto is an IT Specialist with IBM in Brazil, supporting Global accounts. He is an IBM Certified Deployment Professional - Tivoli Application Dependency and Discovery Manager V7.1, and ITIL® V2 Certified. He is experienced with several Tivoli tools such as IBM Tivoli Application Dependency Discovery Manager and IBM Tivoli Monitoring. He holds a degree in Social Science from University of Campinas (UNICAMP).

© Copyright IBM Corp. 2011. All rights reserved.



Ayron Dal Pont is an IT Specialist with IBM in Brazil. He holds a degree in Computer Science from UFSC/SC and a Master Business in Information Technology degree from UniAnchieta/SP. He has over 10 years of experience in IT, of which five years are at IBM working with Tivoli products. His areas of expertise include performance management, event management, asset management, and development.



Bill J Layne works for IBM Software Group, Tivoli Division, RTP Lab and currently provides L2 support for the Netcool® suite of products, with specific focus on Tivoli Business Service Manager and Tivoli Integrated Portal. He has a BS Software Engineering degree from Michigan State University. He has a broad range of development experience in the Tivoli and SWG product set and technologies including z/OS®, SNMP and Tivoli Integrated Portal./WebSphere®. IBM Certified Deployment Professional for Tivoli Business Service Manager and ITIL Certified.



Bruno Caiado Paranhos Carneiro is a Service Management Consultant for IBM Software Group. During his career with IBM, he worked as a Process Architect for Global Technology Services, developing service-level management and IT governance projects for worldwide clients. He currently develops sales and implementation strategies for Service Management projects in the IBM Software Group.



Bruno Nogueira Costa is a Staff Software Engineer for Test area in the Tivoli brand Software Group. In charge of Integrations Tests for the SRM product, he previously worked with products of Tivoli EAM Industry Solutions. He has a degree from UNESP/Fatec-SP in Data Processing and postgraduate degree in Software Engineering at UNICAMP. He has nine years of experience in IT industry, with more than six at IBM.



Carlos Alexandre Salgado Penante is an IT Specialist with IBM in Brazil, working as a Tivoli Management Event Analyst. He is IBM Certified on Tivoli Enterprise Console® and has more than 10 years of experience working with many Tivoli products. He graduated in Electronic Engineering and Economics, and has a postgraduate degree in Economic Engineering.



Carlos Felipe Franca da Fonseca is a Certified IT Specialist with IBM in Brazil. He graduated in Information Technology and has a postgraduate degree in Management of Information Systems, he has over 10 years experience in IT and holds 35 certifications, nine of them on IBM products and solutions. His areas of expertise include operating systems, database management systems, virtualization technologies, scripting techniques, IBM System x® and IBM Systems Director.



Carolina Escribano Sanchez is a Tivoli Specialist working for IBM in Spain. She has extensive experience on Tivoli Business Service Manager, Tivoli CCMDB, Tivoli Monitoring, ITCAM, Tivoli Network Manager, Netcool/OMNIbus, Netcool/Impact, Netcool/Reporter and Netcool/Probes, and Netcool/Webtop products. Carolina has a degree in Chemical Engineering Biotechnology Speciality from the Universidad Complutense de Madrid.



Datta S Khot is a Development Lead for Tivoli Provisioning Manager for Cloud team and works at IBM India Software Lab in Pune. Previously he has worked on Tivoli Service Automation Manager product and Virtual Storage Cloud offering development. He has over a decade of experience in Network Attached Storage, Virtualization, High Availability, Live Migration, Java™ Virtual Machine, compilers, networking, security, and identity management technologies. He has designed and developed scalable, reliable, and efficient software for multithreaded and distributed systems on Linux, Windows®, and Solaris platforms. He holds a Master of Technology degree in Computer Science and Engineering from IIT Bombay, India.



David Kwock is a Service Management Solution Architect in the U.S. He has ten years of experience in the IT Service Management field. He has worked at IBM for three year. His areas of expertise include service management, service-oriented architecture, cloud computing and dynamic infrastructure. He has written extensively about IT Service Management and service-oriented architecture including an IBM Redpaper™ publication about Tivoli Provisioning Manager for OS deployment in a retail environment. David has also been a speaker at IBM Executive Summits and Gartner conferences on the topics of IBM Service Management and service-oriented architecture.

Archived



David Stephenson is a native of Sydney, Australia, and for the last ten years has worked for IBM Global Technology Services in distributed Systems Management roles. David specializes in Event Management and his depth of recent experiences includes diverse roles encompassing test management, database administration and design, network management, software and license management, financial modelling, and use of IBM Tivoli Monitoring Version 6. David has co-authored many IBM Redbooks publications about Systems Management and holds a Masters of Commerce degree with an Advance Specialization in Information Systems and Management from the University of New South Wales.

David's recent publications are *IT Asset Management Processes using Tivoli Asset Management for IT*, SG24-7601 and *Deployment Guide Series: Tivoli IT Asset Management Portfolio*, SG24-7602.



Diogo Tadeu Silva de Araujo is a consultant for automation solutions in the IBM Software Group. In his career at IBM, he acted in the integration of several automation solutions, such as: service level management, impact analysis, asset management for IT, monitoring business applications and solutions based on ITIL, besides dealing with the creation of integration solutions for large architectures. He currently works in the technical services team in architecture and implementation of solutions for Service Management and Automation Solutions in IBM Software Group.



Douglas Barranqueiros Gomes has been providing deployment and support on Tivoli tools (IBM Tivoli Configuration and Change Management Database, IBM Tivoli Application Dependency Discovery Manager, IBM Tivoli Business Service Manager and IBM Tivoli Monitoring) in the Enterprise Automation for Architecture and Tools Team with IBM in Brazil. He holds a degree in Computer Science from Carioca University in City of Rio de Janeiro/RJ, Brazil and Master Business Administrator (MBA) in Business Management from Getulio Vargas Foundation (FGV) in City of Campinas/SP, Brazil.

Archive



Eravimangalath P Naveen is an IT Infrastructure Architect at the IBM India Software Lab in Bangalore with more than 10 years of experience. His areas of expertise include various Tivoli products such as IBM Tivoli Monitoring, IBM Tivoli Application Dependency Discovery Manager (TADDM), IBM Tivoli Identity Manager, IBM Tivoli Storage Manager. He also has extensive experience with IBM AIX®, Linux, z/VM® and System z® hardware, VMware, Storage Systems, Networking and Virtualization for all IBM Server Systems. He has been a presenter at ITSO workshop events related to Linux and IBM System z, and also contributed to multiple IBM Redbooks on the same topic.



Krishnan Venkitasubramanian is a Systems Management Consultant with Tivoli Lab Services team in IBM India Software Labs. He holds a Bachelor of Engineering degree in Computer Science from Anna University, India. He has a customer-facing role and is involved with the design and implementation of various Tivoli products, including Tivoli Service Management and Security products. He is involved in both pre-sales and post sales activities of these products.



Leucir Marin Junior has several years in Software Development and currently works on the Tivoli Service Management Excellence Center (SMEC) team, supporting sales in one-of-a-kind projects. He spent the last three years as a member of Tivoli Service Request Manager® development team, specifically leading integrations initiatives with IBM products, such as OMNIbus, Tivoli Enterprise Console, Infoman, NetView® and third-party products such as Peregrine, Remedy, and SAP. Leucir is a Project Management Professional (PMP) and has a Bachelor degree in Computer Sciences and an MBA degree.



Harshini Mariswamy is Staff Software Engineer working in India Software Labs as a Level 3 Support Engineer for Tivoli Integrated Portal. She has been working on this team since 2009. Prior to this, she was working for an SOA-based Asset Development project in Global Business Services®. She holds a Bachelor degree of Engineering in Information Technology from Bangalore University.



Johan Raeymaeckers is an IT Consultant working for JorSy Systems Management in Belgium. He has 11 years of Enterprise Systems Management experience. He is a Certified Tivoli Consultant and Instructor. He worked on various large-scale Tivoli projects and is involved in Tivoli course development for IBM.



Marcel Dubiella Ferreira de Oliveira is a Systems Specialist from the Maximo® support team. He currently provides support for Brazil and Latin America and has been with IBM support team since 2006. He has four IBM certifications and holds a degree in Analysis and Systems Development.



Marcel Laurenz is a Senior IT Consultant and Director of LAURENZ-IT, based in northern Germany. Marcel has more than 15 years of experience in leading international teams and projects, specifically in areas of operations and service management. He is a certified IBM Lotus®, WebSphere, Tivoli, Microsoft®, and ITIL Expert, and holds a degree in Sport Science and Economy. He is an entrepreneur, building high-performance teams.



Marianne Hardh is a Senior IT Specialist from Sweden. She joined Tivoli in 1999 as a consultant working with Tivoli Service Desk products, and later added Tivoli Monitoring and Tivoli Netcool to her portfolio. Previous work has included programming, database design and administration, and several years of consulting in service and asset management products. Her areas of interest are web management, database management, and database warehousing.



Mario Schuerewegen is a Technical Pre-sales Specialist based in Belgium and specializes in Netcool products. He has 10 years of experience in the network and event management field. His areas of expertise include Cisco networking, SNMP, and network management.



Miguel Sanders has been working as an AIX/System p L3 Support Engineer at ArcelorMittal for the past four years. Miguel has been part of the team that designed and deployed the Tivoli Workload Scheduler architecture for ArcelorMittal in Western Europe. Being an AIX Support Engineer, he has gained extensive knowledge about Reliable Scalable Cluster Technology (RSCT) and Tivoli System Automation for Multiplatforms, which resulted in an update of the automation policies presented in this book. Miguel holds a Master's degree in computer engineering.

Archived



Murat Kipel is an IT Specialist at IBM Global Services in Istanbul, Turkey. He has been working with IBM since 2005. His areas of expertise are IBM Tivoli Service management products including CCMDB, Tivoli Service Request Manager, Maximo for Service Providers and Tivoli Application Dependency and Discovery Manage (TADDM). He is an IBM Certified Deployment Professional - Tivoli Application Dependency and Discovery Manager V7.1., and IBM Certified Deployment Professional - Tivoli Change and Configuration Management Database V7.1.1.



Rishikesh Kumar is a Systems Software Engineer in the IBM India Software Lab. He has been part of the Tivoli Security product development team for one year. In his current role, Rishikesh is part of the Tivoli Identity Manager development team based in Pune, India. Rishikesh is author of several technical articles. He is involved in providing and implementing solutions for Dynamic Infrastructure® Lab Project. Rishikesh holds a B.Tech degree in Computer Science from IIT Kharagpur, India.



Neetha Mangalore is an IT Specialist at the IBM India Software Lab and has over three years of experience. She supports the IBM Tivoli Monitoring implementation at IBM India Software Lab and her areas of expertise include Tivoli Monitoring, Tivoli Integrated Portal, BIRT and Netcool OMNIBus. She holds a Master of Science degree in Electronics.



Ozgun Odabasi is an IT Specialist working for Systems Management Infrastructure team in Brno, IBM Delivery Center Central Europe. Before joining the team in 2009, he worked for two years in the IBM Turkey Strategic Outsourcing department. Ozgun started his ITSM career with Tivoli Monitoring and later added Tivoli Netcool products to his portfolio. He is an IBM Certified Deployment Professional - Tivoli Monitoring V6.2. He also holds Bachelor of Science degree in Computer Engineering from Galatasaray University, Istanbul.



Pankaj Pandey works as IT Specialist at the IBM India Software Lab. He has more than give years of IT experience and has a postgraduate degree in Computer Science. He is experienced on Linux and UNIX®, hardware, networking, and TADDM.



Rafael Risso Moleiro is a Tivoli Specialist with IBM in Brazil and has more than 10 years of experience. He graduated in Computer Science in Sao Paulo and has much experience in Tivoli System Management, Event Management, and Security. He has worked at IBM for four years. His primary area of expertise is Cloud Computing, working with Tivoli Service Automation Manager and Tivoli Provisioning Manager.



Sabri Kizanlik is an IT Specialist working for IBM Global Services for three years. He graduated with a Computer Engineering degree from of Bosphorus University in Istanbul at 2007 and then joined IBM Turkey Global Technology Services. He has experience on Tivoli Netcool/OMNIbus, Tivoli Netcool, Webtop, Tivoli Netcool Probes, Tivoli Netcool Impact, Tivoli Netcool Network Manager and Tivoli Netcool Proviso®, and more. He has been a part of several system monitoring projects in Turkey to implement and configure these products at customer sites.



Sandy Cash is a Certified IT Architect with IBM Global Services, specializing in Operations and Infrastructure. He works primarily in the fields of IT Service Management and Systems Management and has extensive experience in Event Management, Change Management, Service Execution and Orchestration, and Service Request Management. He has been with IBM since 1999.



Soumyadeep Paul is an IT Specialist at IBM India Software Lab in Bangalore, India. He has been working with IBM since December 2007. His areas of expertise include automated Linux provisioning, deployment of Tivoli products such as Tivoli Provisioning Manager, Tivoli Identity Manager IBM Tivoli Monitoring, and more. He holds a Bachelor of Technology degree in Computer Science and Engineering**.**



Tomasz Sekman is a staff Software Engineer in IBM Krakow Software Laboratory, Poland. He has eight years of experience in software development, many of which are with IBM. He has a degree in Computer Science from AGH University of Science and Technology, Krakow, Poland. He is currently leading enablement and L3 support with the Tivoli Common Reporting team. His areas of expertise include Java technologies, databases and business intelligence platforms (BIRT and Cognos®).



Vasfi Gucer is a Project Leader at the International Technical Support Organization, Austin Center. He has been with the ITSO since January 1999. He has more than 12 years of experience in the areas of systems management, networking hardware, and software on mainframe and distributed platforms. He has worked on various IBM Tivoli client projects as a Systems Architect in the U.S. He writes extensively and teaches IBM classes worldwide on Tivoli software. Vasfi is also an IBM Certified Senior IT Specialist, PMP and ITIL Expert.



Yatresh K Agarwal is a Software Specialist at India Software Labs. He holds a Bachelor of Technology degree in Computer science from NITK Suratkal, India. He is working for Tivoli Lab Services team on Security and Service management products. He is ITIL certified with expertise in service management and ITIL processess. He has implemented Tivoli products for various customers in India. He is involved in both post sales and pre-sales of Tivoli products.

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

Tamikia Barrow, David Bennin, Emma Jacobs, Diane Sherman, Margaret Ticknor, Wade Wallace
International Technical Support Organization

Naeem Altaf, Janet S. Andersen, Cesar Araujo, John Arwe, Pandian Athirajan, Ron B Baker, Debbie Bandera, Chris Barlock, Russ Blaisdell, Greg Bowman, Payal Chakravarty, Sudhakar V Chellam, Carole Corley, Thomas Daniel, Budi Darmawan, Jay Davis, Pam Denny, Scott Dickerson, Ann Funai, Jim Fletcher, Byron C. Gehman, Warren Gill, Pedro Guerra, Benjamin Harmon, Jeff Van Heuklon, Praveen Hirsave, Rachel Jarvie, Conrad Johnson, Michael Jones, George Inness, Michael Kaczmarski, Eswara Kosaraju, Dan Krissell, Jennifer Kuvlesky, Steven Larcombe, Trevor Livingston, Dick Locke, Hari Madduri, Mike Mallo, Edson Manoel, Frank Marullo, Denis McFadden, James Moore, Ken Parzygnat, Rajeev Puri, P G Ramachandran, Sudhindra K Rao, Daniel Rhames, Wayne Riley, Dave Royek, Mike Ruggiero, Louis Salcedo, Ramin Sanaie, Tom Sarasin, Beth Sarnie, Jerry Saulman, Joe Saunders, Chris Schaubach, David Schmidt, Randy Scott, Sandra Tipton, Dat Truong, Mark Vanderwiel, Weibo Yuan, Stephanie L Walter, William L Weddendorf, Jedd Weise, Corey Williams, Bruce Willman, Aya H Zaghloul
IBM USA

Alexander Aguina, Carlos Wagner Dias, Luis Carlos Fonseca Dias, Rodrigo Giometti, Julio Madeira, Reynaldo Mincov Jr, Fernando Moraes, Cristina Zabeu
IBM Brazil

Harald Daur, Dietmar Noll, Birgit Nuechter, Georg Ochs, Werner Staub
IBM Germany

Wendy Berman, Pablo Caceres, Paul Chen, Rick Chen, Andreea Jurj, Erin Heximer, Steve Roberts, Joe Wigglesworth, Alice Yeung
IBM Canada

Stefano Borghetti, Xavier Giannakopoulos, Pietro Iannucci
IBM Italy

Artur Gruszecki, Franciszek Kulig
IBM Poland

Jonathan Baggott, Olivier Bonnet, Stuart Cook, Don Wildman
IBM UK

Nilesh S Ghorpade
IBM India

Mark Amstrong
IBM Ireland

Now you can become a published author, too!

Here's an opportunity to spotlight your skills, grow your career, and become a published author - all at the same time! Join an ITSO residency project and help write a book in your area of expertise, while honing your experience using leading-edge technologies. Your efforts will help to increase product acceptance and customer satisfaction, as you expand your network of technical contacts and relationships. Residencies run from two to six weeks in length, and you can participate either in person or as a remote resident working from your home base.

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 Redbooks form found at:

  `ibm.com/redbooks`

- Send your comments in an email 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

Archived

Stay connected to IBM Redbooks

- Find us on Facebook:

  http://www.facebook.com/IBMRedbooks

- Follow us on twitter:

  http://twitter.com/ibmredbooks

- Look for us on LinkedIn:

  http://www.linkedin.com/groups?home=&gid=2130806

- Explore new Redbooks publications, residencies, and workshops with the IBM Redbooks weekly newsletter:

  https://www.redbooks.ibm.com/Redbooks.nsf/subscribe?OpenForm

- Stay current on recent Redbooks publications with RSS Feeds:

  http://www.redbooks.ibm.com/rss.html

Archived

Archived

Summary of changes

This section describes the technical changes made in this edition of the book and in previous editions. This edition may also include minor corrections and editorial changes that are not identified.

January 2011, Second Edition

This revision reflects the addition, deletion, or modification of new and changed information described below.

New information

- A new chapter, Chapter 26, “Implementing high availability for the Tivoli Workload Scheduler master domain manager” on page 763 has been added that walks you through the steps of implementing high availability for Tivoli Workload Scheduler master domain manager using Tivoli System Automation for Multiplatforms.

Archived

© Copyright IBM Corp. 2011. All rights reserved.

Archived

Part 1

Introduction

In this part, we introduce the scenarios that are covered in this book. We also describe the lab environment to implement these products.

This part contains the following chapters:

- Chapter 1, “Integration overview” on page 3
- Chapter 2, “Integration scenarios” on page 7

Archived

© Copyright IBM Corp. 2011. All rights reserved.

Archived

1

Integration overview

This chapter provides an overview of the integration of Tivoli products.

This chapter contains the following topics:

- 1.1, “Integration overview” on page 4
- 1.2, “Aspects of integration” on page 5
- 1.3, “Product coverage” on page 6

Archived

© Copyright IBM Corp. 2011. All rights reserved.

1.1 Integration overview

When using unintegrated point products from multiple vendors together, you might have the following concerns:

- Users must manually switch between product consoles, translating and transferring context and data, which is slow and error prone.
- Users must build, apply, and maintain their own integration capabilities.
- Users must learn multiple disparate user interface paradigms.
- Credentials for the same users must be maintained in multiple registries.
- A user must log in separately to each product console.
- Data from multiple product databases must be extracted and combined manually to produce useful reports.
- Data is modeled differently and uses separate identifiers for the same artifacts, so it cannot be combined easily.
- Multiple logs in disparate formats and locations must be scrutinized to diagnose problems.

Based on these concerns, Tivoli development started an integration initiative that provides a guideline about how to converge the products to a common set of rules to allow the products to work together. IBM is implementing this initiative with each product release to enhance the overall integration and all these issues are being targeted by the integration initiatives.

Integration initiatives are as follows:

- Security integration initiatives

  Security integration enables Tivoli products to integrate security aspects, such as authentication and single sign-on, shared user registry support, centralized user account management, consistent authorization, audit log consolidation, and compliance reporting.

- Navigation integration initiatives

  Navigation initiatives allow seamless user interface transition between various Tivoli products when the context is needed. This seamless integration involves integrated user interface and launch in context abilities.

- Data integration initiatives

  Data Integration provides for the communication of information across the product set. This includes the sharing of information about the managed environment and the management environment. Information about the managed environment includes elements such as computers, operating systems, applications, network, storage and other devices. Information about the management environment includes elements such as agents and server components.

- Functional integration initiatives

  The functional integration initiative provides guidance for developing integration modules (IMs) that can be used to automate the execution of selected process steps. An integration module is a piece of code specifically written to enable two products to be integrated to provide a specific function to a customer. IMs are programmatic interactions, not user interface transitions; however, in many cases, navigation and functional integration techniques such as launch-in-context (LIC) are used together to streamline a user's work across multiple products by automating certain tasks and increasing the effectiveness and productivity of the user for manual (or partially automated) tasks.

Archived

- Manageability integration initiatives

  Manageability integration provides the data and interfaces that enable the assessment of system health in both operational and support environments (from availability, performance, and problem determination perspectives). One component of this integration is the Log Analyzer, which is used by clients to help identify the failing product component in the event of a problem.

- Reporting integration

  Reporting integration provides centralized management reporting across various Tivoli products. This reporting integration is realized by using Tivoli Common Reporting.

- Agent management

  Agent management allows self-monitoring of various Tivoli products using IBM Tivoli Monitoring agents.

In this book, we provide a guide for integrating Tivoli technologies based on these initiatives. We present this information in the context of a number of integration scenarios. Scenarios are collections of work items that typically depict how an operations group uses a set of products to achieve a solution. We derived the scenarios from common real-world examples.

Important: We describe only the current implementation of the products. Future product versions and releases are expected to contain additional integration features to allow more seamless coordination between products.

1.2 Aspects of integration

Several aspects of integration and several approaches to achieve integration are available. We describe integration from the operator's perspective. IBM provides multifaceted and deep integration among its products to provide a seamless experience for users and enable the automation of processes:

- Coordinating security
- Navigation
- Establishing single sign-on and single sign-off
- Synchronizing data
- Centralizing the management of resources
- Managing the management system
- Invoking tasks across products

Within each aspect are separate levels of integration. For example, one aspect is *navigation*, which is the ability to move seamlessly between views provided by multiple related products. One level of navigation integration is "*launch*, where one product console can be launched from another. A deeper level of navigation integration is "*launch-in-context*, where the launched console starts in the same context that the user had in the launching console. A user might be looking at an event about a problem with a computer system and launch in context to another product console. When it comes up, it displays further information about that computer system. A deeper level of navigation integration is *shared console*, such as Tivoli Integrated Portal. The same console has panels with information from multiple products. When the user changes contexts in one panel, the other panels switch to the same context.

1.3 Product coverage

The following products address several of the common scenarios that we have identified:

- IBM Tivoli Change and Configuration Management Database
- IBM Tivoli Release Process Manager
- Tivoli Applications Dependency Discovery Manager
- IBM Tivoli Provisioning Manager
- Tivoli Business Service Manager
- IBM Tivoli Netcool/OMNIbus
- IBM Tivoli Netcool/Impact
- IBM Tivoli Service Automation Manager
- IBM Tivoli Monitoring
- IBM Tivoli Monitoring for Energy Management
- IBM Tivoli Monitoring for Virtual Servers
- IBM Tivoli Netcool Performance Manager
- IBM Tivoli Network Manager IP Edition
- IBM Tivoli Composite Application Manager for Transactions
- IBM Tivoli Service Request Manager
- IBM Tivoli Workload Scheduler
- IBM Tivoli Usage and Accounting Manager
- IBM Tivoli Storage Productivity Center
- IBM Maximo Asset Management for Energy Optimization
- IBM Maximo Asset Management
- IBM Tivoli Asset Management for IT
- IBM Tivoli Identity Manager
- IBM Tivoli Access Manager for e-business
- IBM Rational® Asset Manager
- IBM Systems Director

Archived

2

Integration scenarios

This chapter introduces the integration scenarios that we implemented in our environment.

This chapter contains the following topics:

- 2.1, “Overview of the scenarios” on page 8
- 2.2, “Common elements” on page 12
- 2.3, “Lab configuration” on page 12

Archived

© Copyright IBM Corp. 2011. All rights reserved.

2.1 Overview of the scenarios

The integration scenarios are usage patterns of Tivoli products. Tivoli development uses these usage patterns to drive product development. Development identifies and classifies these scenarios so that each scenario can be documented and planned into the product life cycle.

Several scenarios have been identified by the integration team as commonly implemented in client environments. These identified scenarios are useful for product development and direction, because they provide guidance about how the products are used. The scenarios that are implemented in this book are largely based on the input from the integration team, and several clients using IBM products. Of course, these scenarios are only a small subset of the possible integration scenarios that can be accomplished by the Tivoli products, but they were chosen to be a representative of the integration possibilities using the Tivoli products.

The scenarios involve multiple products and encompass various management domains. From availability to provisioning, from security to monitoring, development defines and analyzes scenarios to ensure that integration can be performed and documented to be reused for future implementations.

The tables in this chapter list the scenarios that are covered in this book. Note that these integration scenarios are broadly categorized in five categories:

- Green IT and Energy Management (see Table 2-1 on page 9)
- Tivoli Integrated Portal (see Table 2-2 on page 9)
- Cloud Computing and Provisioning (see Table 2-3 on page 10)
- IBM Service Management (see Table 2-4 on page 10)
- Client implementations (see Table 2-5 on page 11)

This categorization is not absolute, many scenarios can be included in more than one category.

The tables also provide a brief explanation of each scenario. More information about the benefits of these integration scenarios and how these scenarios can be implemented in your environment are in the individual scenario chapters. At the end of each scenario chapter is “Quick Summary” figure that summarizes the scenario description, potential benefits, products involved, and high level steps to implement the scenario.

Archive

Table 2-1 Green IT and Energy Management scenarios covered in this book

Green IT and Energy Management scenarios
Chapter 3, “Collect, visualize, report on energy and thermal metrics for IT, facilities, property, enterprise assets” on page 19. This scenario focusses on collecting data that is relevant for optimizing the energy usage of a data center. The scenario shows how to collect thermal and energy metrics and the ways to look at both the real-time and historical data.
Chapter 4, “Managing business service with energy and environment” on page 63. This scenario provides a unified dashboard that can help you make the correct decisions in terms of saving costs to your company, put the focus on business value services, and manage them through a holistic view of all business value services components, including energy and environmental information.
Chapter 5, “Green IT scenario for Asset Management” on page 103. The chapter guides you through the required steps to configure Maximo Asset Management for Energy Optimization and IBM Tivoli Monitoring for Energy Management to obtain asset and sensor information. This information is used to build graphical views of the data center layout and heat map. You can use these views to monitor the conditions in your data center and to identify potential problems.

Table 2-2 Tivoli Integrated Portal scenarios covered in this book

Tivoli Integrated Portal scenarios
Chapter 6, “Tivoli integration scenario for Virtual Infrastructure” on page 125. This chapter describes the use of Tivoli products to create a fully automated business service. It provides the means for the IT department to have a faster deployment process for new virtual servers and their monitoring.
Chapter 7, “Operational drill down” on page 169. This chapter shows how various Tivoli products can be seamlessly integrated through Tivoli Integrated Portal. We provide several scenarios that show how an operator can use the Tivoli Integrated Portal console to monitor business systems and be able to drill down to solve potential problems.
Chapter 8, “Tivoli reporting integration: IBM Tivoli Monitoring and TADDM” on page 199. This chapter describes integrating IBM Tivoli Common Reporting for Asset and Performance Management (with embedded IBM Cognos 8 technology) with IBM Tivoli Monitoring and IBM Tivoli Application Dependency Discovery Manager (TADDM) and shows you how to create reports using data collected by different Tivoli products. Having cross-product data virtually integrated and having access to them from one location, you can find answers to additional questions from IT management area and make more confident business decisions.
Chapter 9, “Tivoli reporting integration: IBM Tivoli Monitoring and IBM Tivoli Service Request Manager” on page 245. This chapter expands on Chapter 8, “Tivoli reporting integration: IBM Tivoli Monitoring and TADDM” on page 199 and describes how you can create reports integrating data from IBM Tivoli Monitoring and IBM Tivoli Service Request Manager. The purpose of the chapter is to give you the knowledge of the tools available so you can get started with using IBM Tivoli Common Reporting for Asset and Performance Management.
Chapter 10, “Protecting Tivoli Integrated Portal with Tivoli Access Manager for e-business” on page 291. This chapter discusses the value of leveraging a high-performance, multi-threaded reverse proxy to front-end the Tivoli Integrated Portal Interface. This integrated solution provides an authentication and authorization framework to allow controlled access to the Tivoli Integrated Portal. This solution is extremely valuable in cases where access to the Tivoli Integrated Portal is required outside of an organization over the Internet.

Archived

Table 2-3 Cloud Computing and Provisioning scenarios covered in this book

Cloud Computing and Provisioning scenarios
Chapter 12, “Tivoli Service Automation Manager and Cloud Computing” on page 325. This chapter describes how various Tivoli products can be integrated with Tivoli Service Automation Manager to support the delivery of Cloud Computing solutions, resulting in benefits such as better response time to resolve a performance issue and provisioning of new virtual servers without any human interaction based on defined policies.
Chapter 13, “Automated Storage Provisioning” on page 379. This chapter discusses the value of integrating an automated storage provisioning solution as part of an end-to-end service management solution. It describes how you can achieve automated storage provisioning as part of an end-to-end application provisioning solution by leveraging Tivoli Provisioning Manager to deploy the different components of an application and Tivoli Storage Productivity Center to deploy and manage the storage needs for a given business service as part of an integrated workflow.
Chapter 14, “Tivoli Provisioning Manager discovery of TADDM discovered CIs” on page 399. This chapter describes how to use TADDM discovered CIs with Tivoli Provisioning Manager discovery. It describes how Tivoli Provisioning Manager can discover software and hardware components using TADDM and how to manage discovered resources from Tivoli Provisioning Manager.
Chapter 15, “Tivoli Provisioning Manager and Tivoli Identity Manager integration” on page 405. Tivoli Provisioning Manager is used for provisioning machines with the necessary operating system and required software. Tivoli Identity Manager is used for managing the identities and the accounts on the machines. To enable Tivoli Identity Manager to manage the user accounts on the Tivoli Provisioning Manager provisioned machines, the provisioned machines must be registered in Tivoli Identity Manager. The integration scenario described in this chapter addresses this use case.

Table 2-4 IBM Service Management scenarios covered in this book

IBM Service Management scenarios
Chapter 16, “Taking automatic actions based on predefined policies” on page 415. This chapter discusses an integration scenario for taking automatic actions based on predefined policies, when a problem is detected. This integration scenario is based on ITIL best practices, and covers the most commonly used solutions in this type of integration.
Chapter 17, “Asset and CI integration” on page 477. This chapter describes the steps to integrate assets and configuration items (CIs) in your environment. This approach can result in better control of your assets and CIs and increase productivity of your operators and help desk personnel.
Chapter 18, “Rational Asset Manager and Tivoli Change and Configuration Management Database integration” on page 545. This chapter covers the integration of IBM Rational Asset Manager and IBM Tivoli Change and Configuration Management Database (CCMDB). With this integration, you can automatically create assets in Rational Asset Manager from existing CIs in the CCMDB. You can also use Rational Asset Manager as a definitive software library (DSL) for storing software images.
Chapter 19, “User and password management with Tivoli Identity Manager and Tivoli Service Request Manager” on page 595. This chapter describes the integration between Tivoli Identity Manager and Tivoli Service Request Manager. The integration enables the management of Tivoli Service Request Manager users through Tivoli Identity Manager and the ability to create Tivoli Service Request Manager service requests when any passwords are changed through Tivoli Identity Manager.

IBM Service Management scenarios
Chapter 20, “Employee onboarding” on page 623. This chapter describes how Tivoli products can help during the process of receiving a new employee in a company. It demonstrates how an integrated set of Tivoli products can fulfill various services such as security, provisioning, asset management and self-service in a smooth way.
Chapter 21, “TADDM L2 discovery using IBM Tivoli Monitoring agents” on page 641. This chapter describes the use of IBM Tivoli Monitoring agents to discover components in the IT Infrastructure using TADDM. It describes how the IBM Tivoli Monitoring sensor of TADDM discovers configuration items in the IBM Tivoli Monitoring environment with very little configuration overhead.
Chapter 22, “Managing business services in an integrated environment” on page 661. This chapter demonstrates how an integrated environment implemented with TADDM, CCMDB and Tivoli Business Service Manager can improve the efficiency and effectiveness of a service operation. By using this integration you can automate the following tasks: ► Defining the service and its supporting infrastructure, through the discovery of its components. ► Managing service outages through a proper visibility of the service operational status. ► Managing service changes through the control of service configuration. In this chapter there is also an additional section on Tivoli System Automation Application Manager and Tivoli Business Service Manager integration.
Chapter 23, “CCMDB V7.2.1: New launch-in-context technology” on page 699. This chapter describes the Launch In Context (LIC) technology that is available with CCMDB V7.2.1. The LIC technology allows integration of different Tivoli products, increasing operational productivity and reducing cost of ownership of these products. Although this feature is powerful with the current implementation, the products must create product-to-product agreement for launching URLs, and therefore with every change or new release, LIC must be synchronized and revisited. In addition, the launches must be manually configured. CCMDB V7.2.1 introduces an improved LIC technology that is described in this chapter.

Table 2-5 Scenarios implemented by IBM clients covered in this book

Scenarios implemented by IBM clients
Chapter 26, “Implementing high availability for the Tivoli Workload Scheduler master domain manager” on page 763. This chapter walks you through the steps of implementing high availability for master domain controller workload using Tivoli System Automation for Multiplatforms.
Chapter 27, “Implementing high availability for Tivoli Workload Scheduler dynamic workload” on page 793. This chapter walks you through the steps of implementing high availability for dynamic agent workload using Tivoli System Automation for Multiplatforms.

Archived

Each scenario was implemented in the ITSO lab environment that is shown in Figure 2-1 on page 13 and, in general, presented in the following format:

- Scenario overview: Brief description of what this integration is about
- Products involved: Required and optional products
- Benefits of the scenario: Benefits of the scenario, and answers to questions (for example why would clients want to make these integration connections and what is the value behind the integration patterns and scenarios)
- Architectural diagram of integration: Architectural diagram showing the flow of interactions
- Implementation steps: Integration steps for integrating the products
- Scenario walk-through: A sample demonstration of the scenario
- Summary: Summary of benefits and implementation (quick summary figure)

Note: In this book, we do not cover product installation or product features, only the steps required for these integration scenarios. If you are not familiar with a product that is part of the integration scenarios, see the Tivoli Documentation Central website:

https://www.ibm.com/developerworks/wikis/display/tivolidoccentral/Home

2.2 Common elements

From the scenarios that are identified in the tables in 2.1, “Overview of the scenarios” on page 8, common elements of integration exist that are typical for many scenarios. These common elements appear in several scenarios:

- Single sign-on (SSO) requirement on various products. The SSO requirement is common for providing a usable and seamless experience for various products, either through a Java interface or a web interface.
- Resource data in providing the context information for launch in context customization. The common resource data flows include feeding Tivoli Applications Dependency Discovery Manager using Discovery Library and then extracting the data to the necessary applications in a common data format.
- Event data integration in which events can be forwarded to existing systems for further application. The additional facilities for events include the business system view and problem ticket creation.
- Development of a specialized outbound link to quickly jump across various products, while aware of the context of the original invocation.

2.3 Lab configuration

Figure 2-1 on page 13 depicts our lab configuration for the both the Green IT and Energy Management integration and IBM Service Management integration scenarios.

Archived

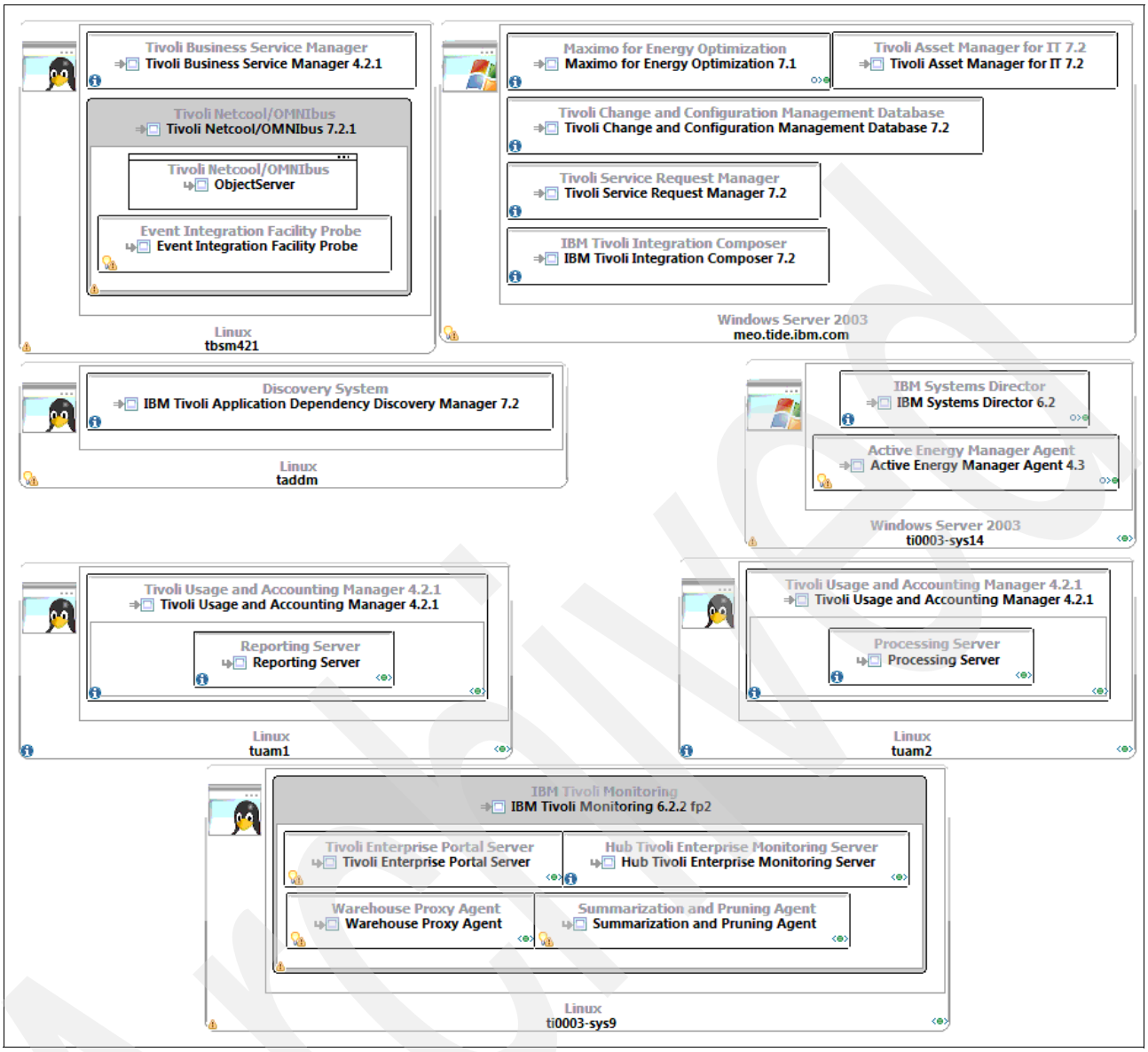


Figure 2-1 Lab environment for the Green IT and Energy Management integration, and IBM Service Management integration scenarios

Austin DEMOcentral data center: In Part 2, “Green IT and Energy Management integration scenarios” on page 17, in addition to the lab environment, we provide several screen captures from an IBM data center in Austin, TX: Austin DEMOcentral data center. This information can provide you with a more realistic picture of the capabilities of the IBM Energy Management products.

Austin DEMOcentral is one of the innovative data centers in IBM where Tivoli products for Energy Management are used to implement a “Green IT” environment.

Figure 2-2 shows our lab configuration for the Cloud Computing and Provision integration scenarios.

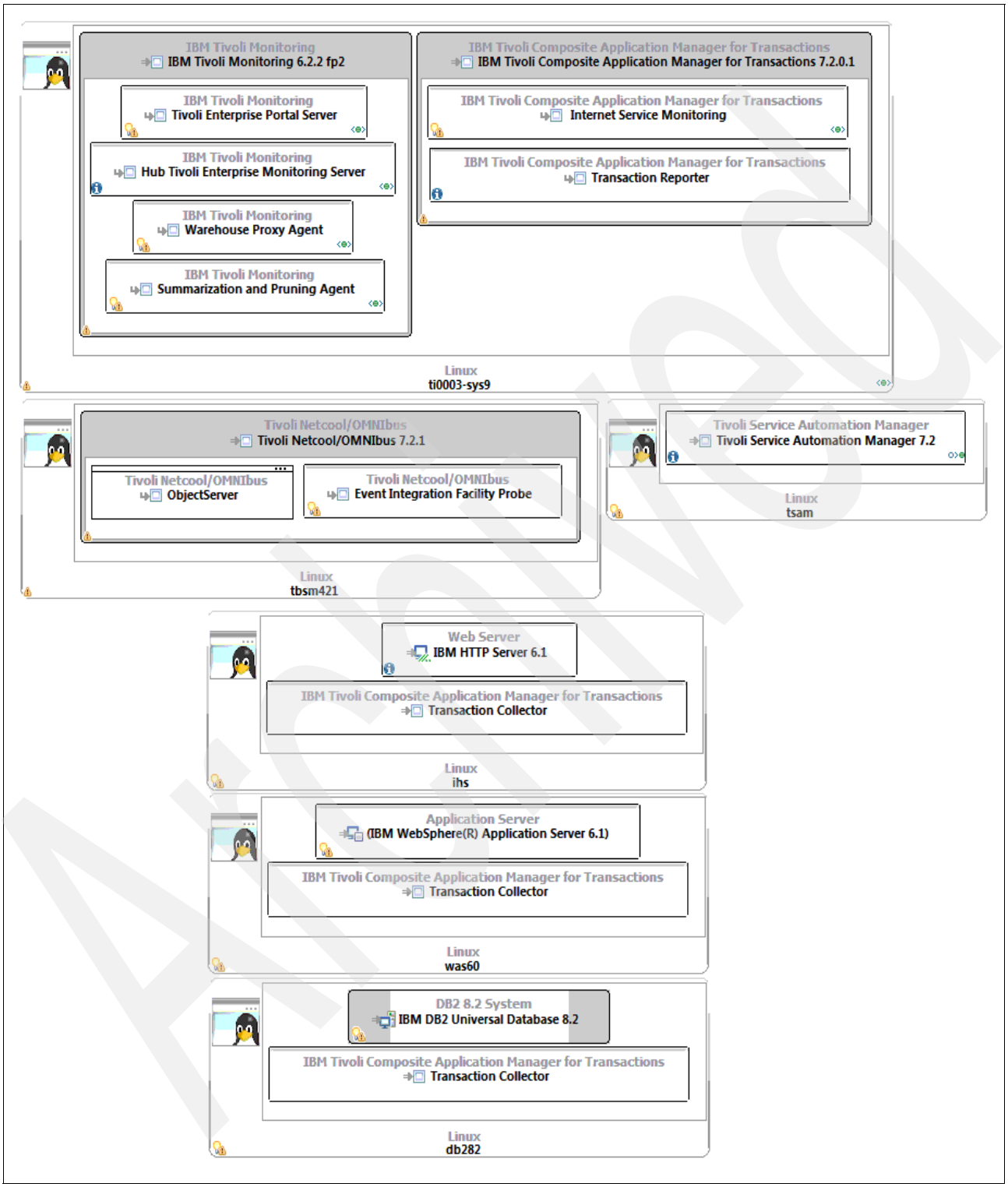


Figure 2-2 Lab environment for the Cloud Computing and Provisioning integration scenarios

Archive

Figure 2-3 shows our lab configuration for the Tivoli Integrated Portal integration scenarios.

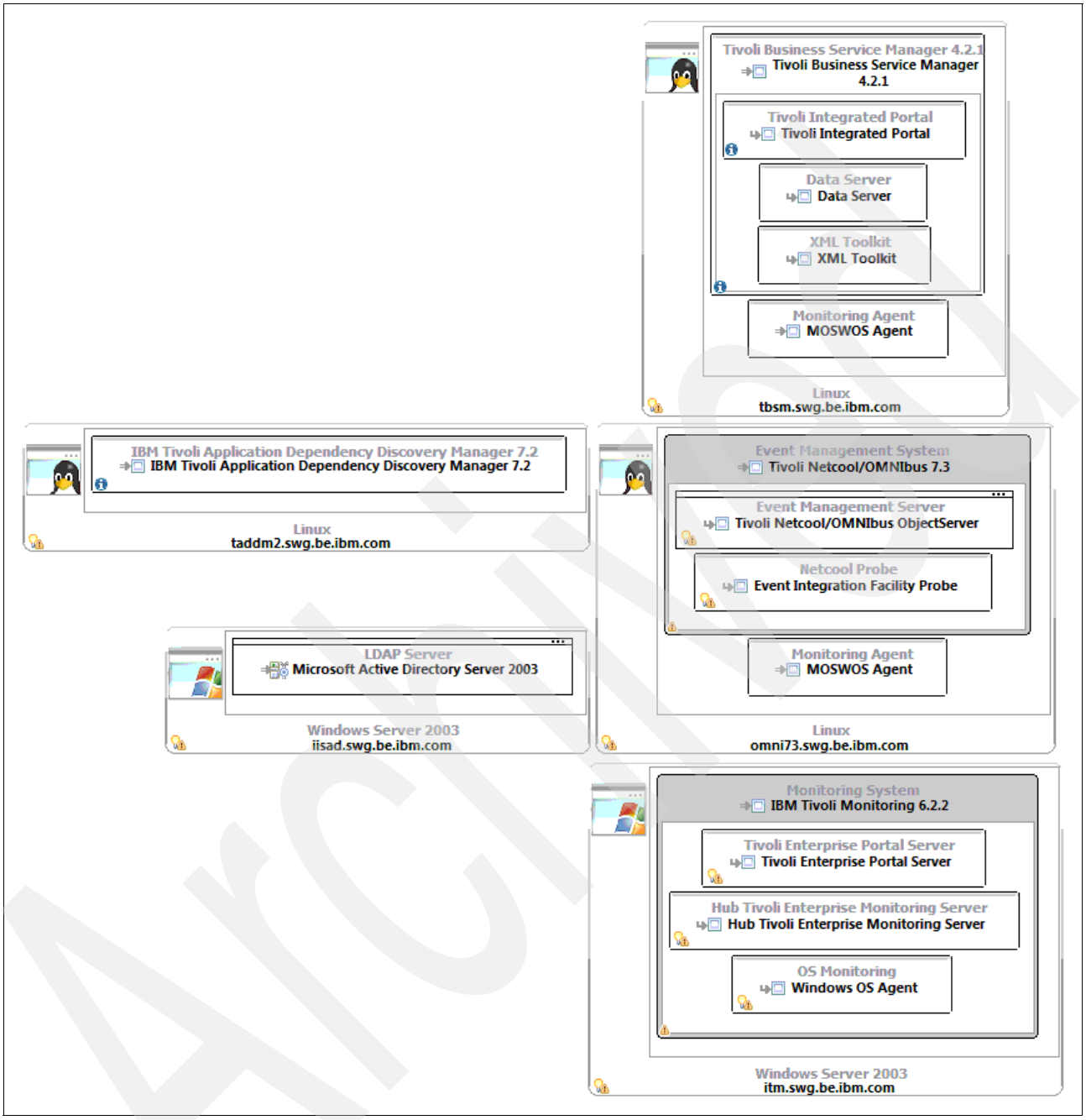


Figure 2-3 Lab environment for the Tivoli Integrated Portal integration scenarios

Note: Although for the most of the scenarios we used the lab environments described, for several scenarios, we used a separate environment.

Archived

Part 2

Green IT and Energy Management integration scenarios

In this part, we cover the Green IT and Energy Management integration scenarios.

This part contains the following chapters:

- Chapter 3, “Collect, visualize, report on energy and thermal metrics for IT, facilities, property, enterprise assets” on page 19
- Chapter 4, “Managing business service with energy and environment” on page 63
- Chapter 5, “Green IT scenario for Asset Management” on page 103
- Chapter 6, “Tivoli integration scenario for Virtual Infrastructure” on page 125

© Copyright IBM Corp. 2011. All rights reserved.

Archived

3

Collect, visualize, report on energy and thermal metrics for IT, facilities, property, enterprise assets

This chapter describes a scenario where various Tivoli products are used to collect and report on energy and thermal metrics for IT, facilities/property and enterprise assets.

This chapter contains the following topics:

- 3.1, “Terminology” on page 20
- 3.2, “Scenario overview” on page 20
- 3.3, “Products involved” on page 21
- 3.4, “Benefits” on page 24
- 3.5, “Architectural diagram of integration” on page 24
- 3.6, “Implementation steps” on page 25
- 3.7, “Scenario walk-through” on page 42
- 3.8, “Summary” on page 62

© Copyright IBM Corp. 2011. All rights reserved.

3.1 Terminology

Before introducing the first Green IT scenario we first introduce important terminology that will be frequently used throughout the Green IT scenarios. Table 3-1 introduces the most important power, thermic and industry terms.

Table 3-1 Green IT terminology

Term	Description
PDU	Power distribution unit: A device that distributes power to a data center.
IPDU	Intelligent PDU: PDU that is capable of energy metering at the outlet ports
UPS	Uninterruptible power supply: A device that provides immediate emergency power, usually by means of batteries, if power fails.
CRAC	Computer room air conditioner: Monitors and controls a computer room's temperature, humidity, and air flow.
Chiller	Provides chilled water used to cool and dehumidify air.
Power saver mode	Provides a way to save power by dropping the voltage and frequency a fixed percentage. This percentage is predetermined to be within a safe operating limit and is not user configurable. Under current implementation this percentage is a 14% frequency drop. Power saver mode is available on POWER6® Blades and 9406/9117-MMA rack systems.
Power capping	Enforces a user-specified limit on power usage. Setting a power capping value ensures that system power consumption stays at or below the value that is defined by the setting. Power capping is available only on supported IBM Servers and IBM Blade Servers.
Carbon footprint	The footprint is the total set of greenhouse gases (GHG) emissions caused by an organization, event or product. For simplicity of reporting, it is often expressed in terms of the amount of carbon dioxide (CO2). For our scenarios, carbon footprint is used in the context of the data center: the total emission of CO2 of the data center is expressed in kg CO2.
DCiE	Data Center infrastructure Efficiency: is a performance metric for calculating the energy efficiency of a data center. DCIE is the percentage value derived, by dividing information technology equipment power by total facility power. For example, if a data center has a total power consumption of 8000kW, and 6000kW is being used for all IT equipment, the DCIE equals 75%.
PUE	Power Usage Effectiveness: Inverse of DCIE. PUE is the percentage value derived by dividing total facility power by IT facility power.
kWh	kilo watt hour: The amount of energy delivered in one hour at a power level of 1000W (measurement unit for selling electricity).

3.2 Scenario overview

This first Green IT scenario focusses on collecting data that is relevant for optimizing the energy usage of a data center. To take appropriate actions for optimizing the energy consumption of a data center, we must first accurately collect thermal and energy metrics. The scenario also focusses on the ways to look at both the real-time and historical data.

The power and thermal data for the IT and facility/property assets is collected and analyzed using the various components of the IBM Tivoli Monitoring for Energy Management bundle. Tivoli Monitoring for Energy Management collects its data from several sources:

- IT components (Blade Centers, Rack Servers)
- Facilities/property and enterprise assets (PDU)
- Uninterruptible power supplies (UPS)
- Computer room air conditioners (CRAC)

The collected data is visualized and reported on in various ways. The Tivoli Enterprise Portal is used to visualize the collected thermal and energy data. By using workspaces that are provided in IBM Tivoli Monitoring for Energy Management, data can be combined with existing performance monitoring data that is provided by the Tivoli Enterprise Portal. This way offers administrators an integrated view of real-time power, temperature and application and system performance data in their data centers.

The second usage of the thermal and energy data involves the Reporting and Optimization feature of the Tivoli Monitoring for Energy Management product. The Reporting and Optimization feature reports on the usage and estimated cost of energy that is used in the data center. This scenario shows examples of reports for energy management costs in a data center, itemized by resource, such as servers, network devices, cooling equipment and other facilities in the data center. The scenario also demonstrates the modeling and recommendation engine of the Reporting and Optimization feature, which allows for the optimization of data centers based on power usage and associated costs.

Finally, thermal and energy data can be used by Tivoli Usage and Accounting to generate more complex reports for allocate costs to clients, departments, accounts, servers, and more.

3.3 Products involved

The products and product components involved in this scenario are as follows:

- IBM Tivoli Monitoring V6.2.2
- IBM Systems Director V6.2
- IBM Systems Director Active Energy Manager™ V4.3
- IBM Tivoli Monitoring for Energy Management V6.2.1
- IBM Tivoli Common Reporting V1.2
- IBM Tivoli Integrated Portal V1.1.1
- IBM Tivoli Usage and Accounting Manager V7.1.2

3.3.1 IBM Systems Director Active Energy Manager

IBM Systems Director Active Energy Manager provides you with resources to monitor and manage power consumption in a data center, toward a green infrastructure. You use it to control energy efficiency on both IBM hardware products and third-party products. By implementing Active Energy Manager with properly configured compatible products, you are also able to determine suitable power allocation, predict energy consumption, and establish limits for power usage.

3.3.2 IBM Tivoli Monitoring for Energy Management

IBM Tivoli Monitoring for Energy Management provides management, monitoring, reporting, and recommendations to help optimize energy resources for efficient data center operation.

The Tivoli Monitoring for Energy Management software is consists of several separate component products. Tivoli Monitoring for Energy Management bundle includes the following existing software products:

- IBM Tivoli Monitoring
- IBM Systems Director
- IBM Systems Director Active Energy Manager

In addition to these existing products that are part of the bundle, the Tivoli Monitoring for Energy Management offering also includes a number of specific “energy” monitoring agents:

- Active Energy Manager agent
- Eaton Power Xpert agent
- Johnson Controls Metasys agent
- Siemens APOGEE agent
- APC InfraStruXure agent

Also included in addition to these monitoring agents is the Tivoli Monitoring for Energy Management Reporting and Optimization feature. This feature reports on the usage and estimated cost of energy used in the data center.

The next sections briefly describe the Tivoli Monitoring for Energy Management agents and the Reporting and Optimization feature.

Active Energy Manager Agent

The Active Energy Manager Agent (AEM Agent) is an IBM Tivoli Monitoring Agent that provides the integration between Tivoli Monitoring and IBM Systems Director Active Energy Manager. It is installed on the IBM Systems Director server where it collects the data from the IBM Systems Director Active Energy Manager. This way the AEM agent, indirectly, collects data from the following devices:

- IBM BladeCenter®
- Racks
- Rack Servers
- System z10s
- PDUs
- UPS units
- Sensors
- CRACs

Eaton Power Xpert agent

The IBM Tivoli Monitoring for Energy Management: Eaton Power Xpert Agent is used to monitor Eaton Power Xpert Devices. The agent uses the Simple Network Management Protocol (SNMP) to remotely collect metrics from the following devices:

- PDUs through the PXGX 1000 gateway card
- UPS units through the PXGX 2000 interface
- 4000, 6000 and 8000 Meters

Archived

Johnson Controls Metasys agent

The Johnson Controls Metasys agent uses web services to remotely collect metrics from the Johnson Controls Metasys Building Automation and Control system. This way the agent can collect data from the following devices:

- CRACs
- Chillers
- PDUs
- UPS units
- Generators
- Sensor devices

Siemens APOGEE agent

The Siemens APOGEE agent uses SOAP to remotely collect data from the Siemens APOGEE Building Automation and Control System. The APOGEE agent collects data for the following APOGEE devices:

- Air Handle Units
- Chiller Plants
- Chillers
- CRACs
- Meters
- Generators
- PDUs
- Sensors
- UPS units

APC InfraStruXure agent

The APC InfraStruXure agent uses SNMP to remotely collect metrics from the following APC devices:

- UPS units
- PDUs
- Cooling devices
- Environmental monitoring devices
- Rack Air Removal Units

Note: The Tivoli Monitoring for Energy Management provides monitoring of a wide range of IT and facilities devices. If your data center hosts devices that are not supported by any of these agents, you can use the IBM Tivoli Agent Builder to design and package a custom agent to track and monitor power and thermal data. Creating a custom agent and integrating it with Tivoli Monitoring for Energy Management is described in *IBM Tivoli Monitoring for Energy Management Reporting and Optimization User's Guide*, SC27-2418.

Reporting and Optimization feature

The Tivoli Monitoring for Energy Management Reporting and Optimization feature is installed on Tivoli Common Reporting and Tivoli Integrated Portal. It is used to create reports for energy management costs. The Reporting and Optimization component uses the historical data from the Tivoli Data Warehouse, collected by the different Tivoli Monitoring for Energy Management agents.

Archived

Reporting and Optimization uses the Tivoli Integrated Portal for consolidation of its user interface components. The Reporting and Optimization user interface consists of three major components:

- Reports concerning energy, CPU utilization and recommendations
- Energy usage metrics charting portlets
- Report parameter configuration portlets

The Reporting and Optimization reports can be itemized by resource, such as servers, storage, network devices, cooling equipment, and other facilities in the data center. You can analyze power usage and DCiE metrics trends for the data center using Reporting and Optimization charts.

Reporting and Optimization also has a modeling and recommendation engine to help optimize data centers, based on power usage and associated costs. Using these results, you can perform “what if” calculations to determine power savings and cost savings alternatives.

3.4 Benefits

A first and important step towards a Green IT data center is the accurate collection of power and thermal data for all devices in the data center. Visualizing the collected data provides an insight in the power consumption, energy cost and carbon emission of the components in the data center. It enables us to assess the efficiency of energy consumption in the data center. Even more important is that correctly interpreting the collected data (both real-time and historical) enables us to identify opportunities for saving energy and associated costs.

In this scenario, we first demonstrate the IBM Tivoli Monitoring for Energy Management capabilities to collect, visualize, and report on power, thermal, systems and application data from a wide variety of IT and facilities equipment. We show how Tivoli Monitoring for Energy Management data provides administrators with an unprecedented integrated view of power, temperature, and application performance. With this information, the administrator can react appropriately if problems occur that are related to power or thermal issues.

Next, we demonstrate the capabilities of the Reporting and Optimization feature of IBM Tivoli Monitoring for Energy Management. The feature uses the historical power and CPU data to identify opportunities for saving energy and associated costs by determining which servers or server pools can be put on power savings mode, standby mode, or completely powered off during certain time intervals.

3.5 Architectural diagram of integration

Figure 3-1 on page 25 shows the architectural diagram of this integration scenario. The power and thermal data is collected by the separate Tivoli Monitoring for Energy Management agents from the various IT and infrastructure components. The agents report the data to Tivoli Monitoring where it is stored and summarized in the Tivoli Data Warehouse. The data can be examined using the Tivoli Enterprise Portal client, by looking at the workspaces provided by Tivoli Monitoring for Energy Management or by creating custom workspaces and views.

Archived

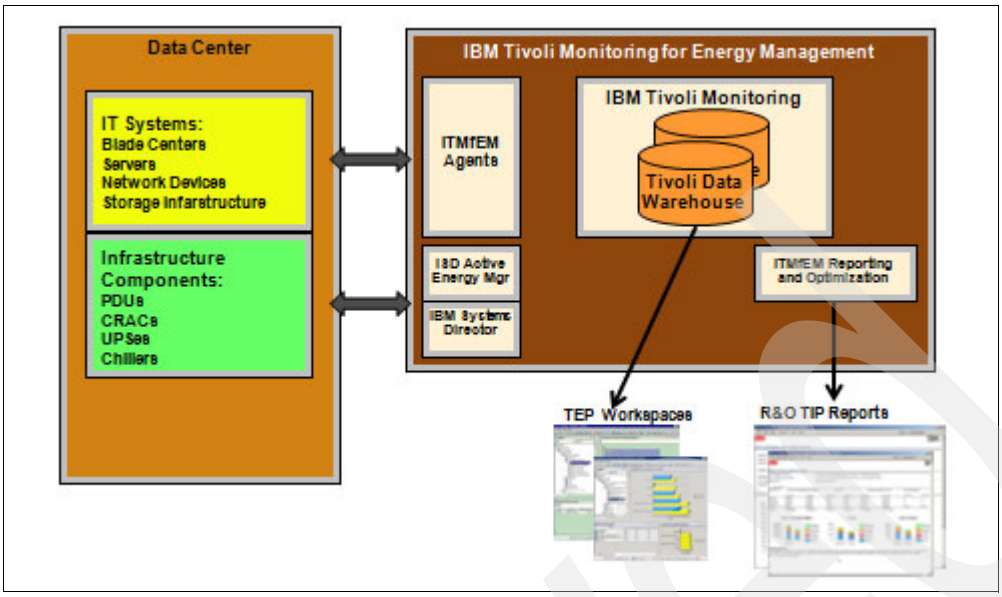


Figure 3-1 Architectural overview scenario 1

The Reporting and Optimization (R&O) feature adds various energy management reports to the Tivoli Integrated Portal (TIP). For example reports that provide insight into the total energy consumption, carbon footprint and energy efficiency of a data center. Also reports are available to identify opportunities for saving energy and associated costs by determining which servers or server pools can be put on power savings mode, standby mode, or completely powered off during certain time intervals.

3.6 Implementation steps

This section describes how to integrate the Tivoli products that are used in this scenario. It does not cover the complete installation of the products, only the installation steps that are relevant for the integration of the products are shown.

First, we show how to configure the Tivoli Monitoring for Energy Management Active Energy Manager (AEM) agent. The AEM agent provides the integration between IBM Tivoli Monitoring and IBM Systems Director Active Energy Manager. Then, we show how to install and configure the Tivoli Monitoring for Energy Management Reporting and Optimization feature, which provides the integration between Tivoli Monitoring for Energy Management and the Tivoli Integrated Portal.

3.6.1 Installing and configuring the Tivoli Monitoring for Energy Management Active Energy Manager (AEM) agent

The AEM agent actually consists of two parts: an Tivoli Monitoring AEM agent and an IBM Systems Director plug-in. Both components must be installed on the IBM Systems Director server.

Note: We assume that you have installed the Active Energy Manager on the IBM Systems Director server. This needs to be installed before the AEM agent plug-in can be installed.

Tivoli Monitoring AEM agent is installed in the same way as any other Tivoli Monitoring agent. Detailed installation instructions can be found in the *IBM Tivoli Monitoring: Installation and Setup Guide*, GC32-9407. During the installation of the Tivoli Monitoring AEM agent you are presented with a configuration window as shown in Figure 3-2.

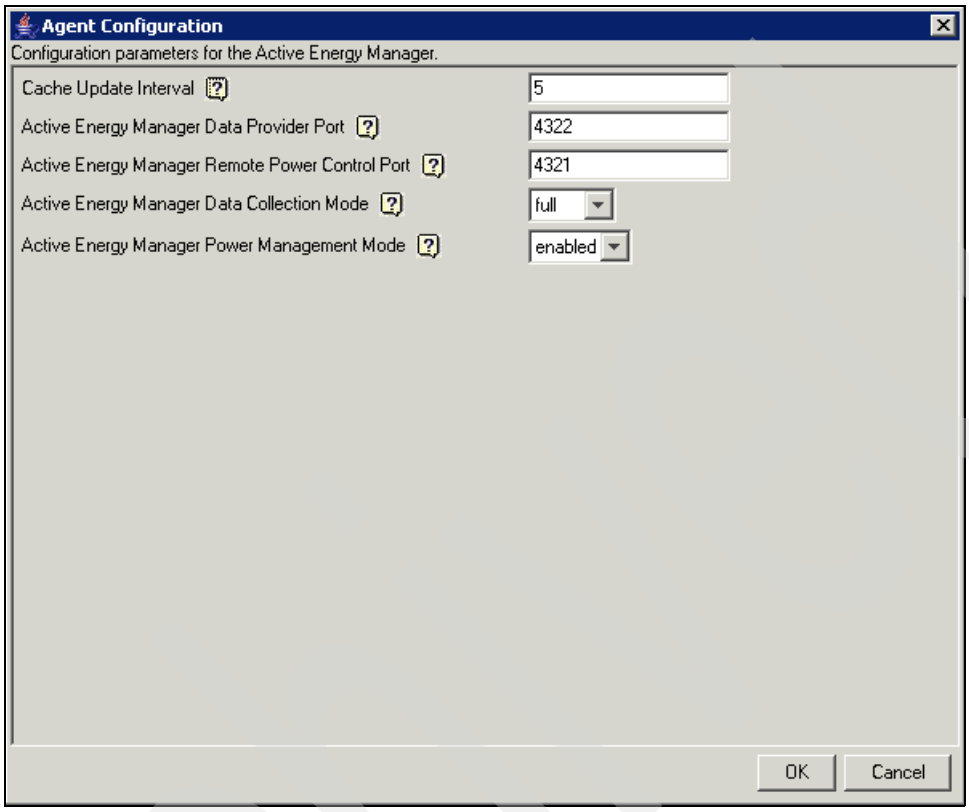


Figure 3-2 AEM Agent Configuration

The Active Energy Manager Agent configuration consists of the following settings:

- Cache Update Interval: The interval in minutes at which the AEM agent retrieves the data from the IBM Systems Director AEM plug-in. The minimum value is 1 minute, the default value is 5 minutes.
- Active Energy Manager Data Provider Port: The TCP/IP port to which the AEM agent connects to retrieve data from the IBM Systems Director AEM-plug-in. The default value is 4322.
- Active Energy Manager Remote Power Control Port: The TCP/IP port to which the AEM agent connects for power control operations to the IBM Systems Director AEM-plug-in. The default value is 4321.
- Active Energy Manager Data Collection Mode: The mode in which data is collected from IBM Systems Director AEM. Possible values are full or limited. If full is selected, all AEM data is collected. If limited is selected, only CPU, power, and temperature data are collected.
- Active Energy Manager Power Management Mode: The mode in which power management is handled. Possible values are enabled or disabled. When enabled, a power cap or power saver mode can be set on supported systems using the Tivoli Monitoring Tivoli Enterprise Portal client. If disabled is selected, the power management modes are not available from the Tivoli Monitoring Tivoli Enterprise Portal client.

After installing and configuring the Tivoli Monitoring AEM agent, the AEM plug-in must be installed on the IBM Systems Director server. This plug-in is used to provide the data from IBM Systems Director Active Energy Manager to the Tivoli Monitoring AEM agent.

The installation executable for the plug-in is located on the Tivoli Monitoring for Energy Management installation media as shown on Figure 3-3

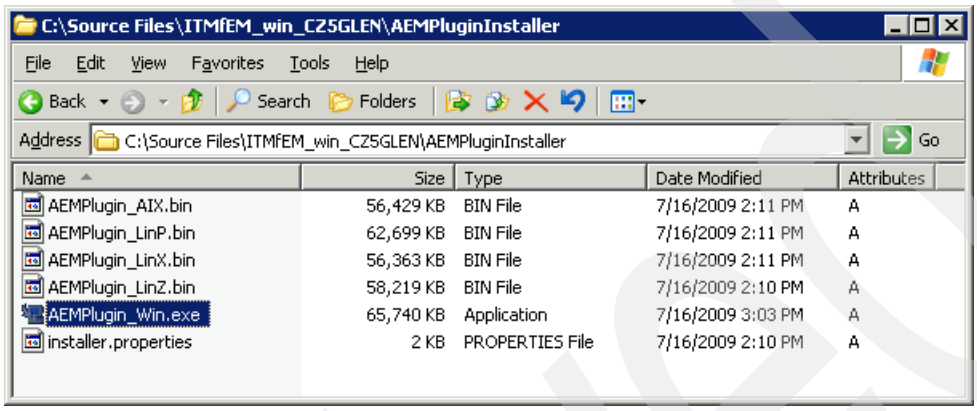


Figure 3-3 AEM plug-in installer location

The installation is started by launching the executable for the corresponding platform. The installation wizard takes you through a number of installation steps, which require no specific configuration. The installation wizard is shown in Figure 3-4.

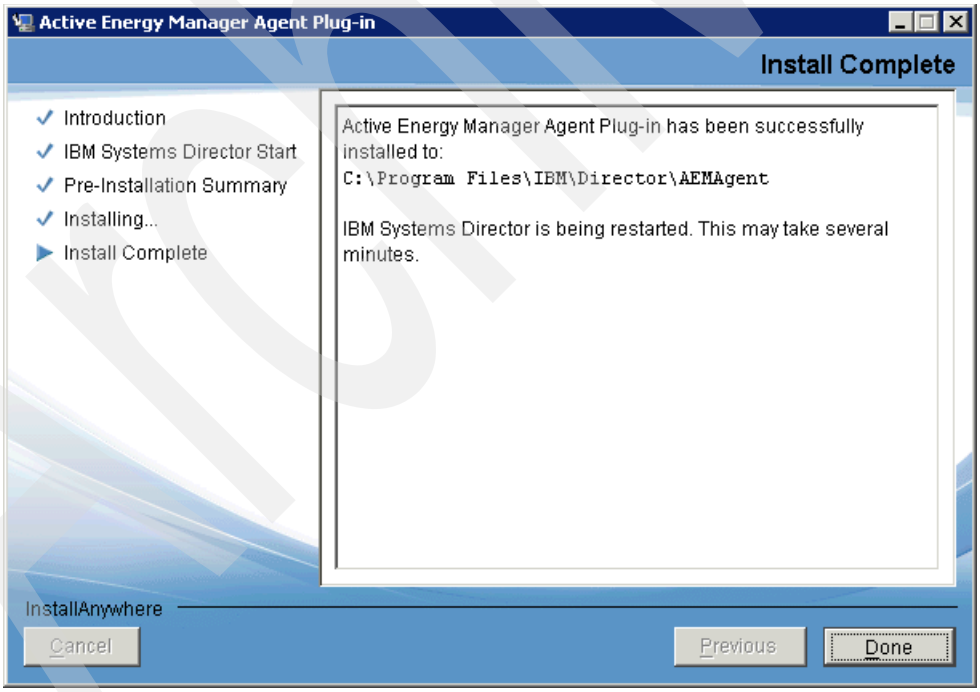


Figure 3-4 AEM plug-in installer wizard

After successful installation of the AEM-plug-in the AEM agent should correctly appear in the Tivoli Enterprise Portal client. The Agent Status workspace should correctly display the various AEM managed objects as shown in Figure 3-5 on page 28. The active objects that are shown in this workspace will depend on the devices available on the IBM Systems Director server.

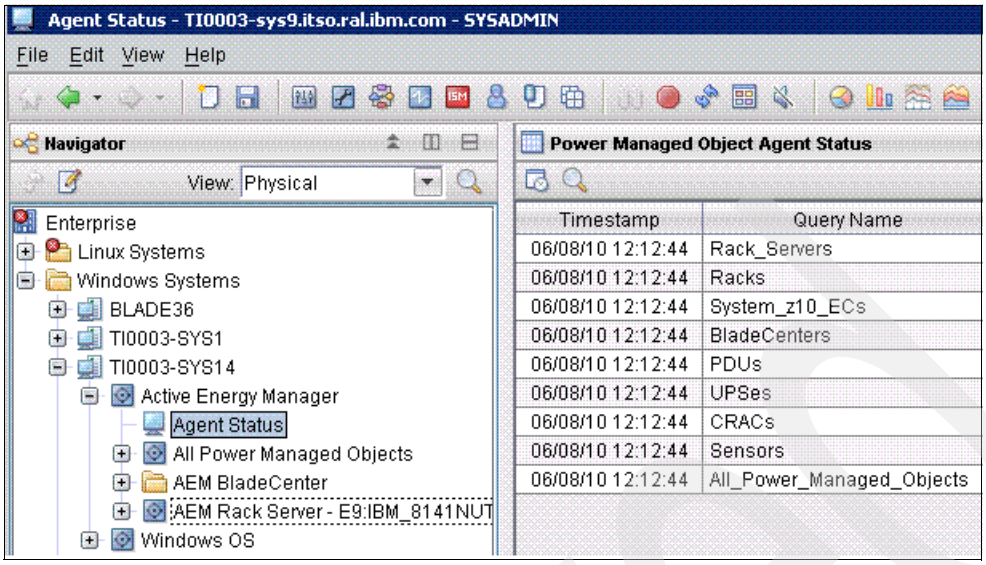


Figure 3-5 AEM Agent in Tivoli Enterprise Portal client

Important: When installing the Tivoli Monitoring for Energy Management AEM plug-in version 6.2.1 with IBM Systems Directory 6.2, the plug-in does not work correctly. Although the installation of the plug-in is successful, the agent status workspace in the Tivoli Enterprise Portal shows an error for the object status. The `MANIFEST.MF` file included with the AEM plug-in is not compatible with IBM Systems Director 6.2. For more information see the Technote:

`http://www.ibm.com/support/docview.wss?uid=swg21432288`

3.6.2 Installing and configuring Tivoli Monitoring for Energy Management Reporting and Optimization

Tivoli Monitoring for Energy Management Reporting and Optimization needs to be installed on top of Tivoli Common Reporting and Tivoli Integrated Portal. As a pre-requisite the Tivoli Integrated Portal must be configured to enable *base charting*. If base charting support is not enabled, Tivoli Integrated Portal must be reinstalled.

The installation images for Reporting and Optimization consist of an executable that needs to be launched on the Tivoli Integrated Portal server. For example, on a Windows Tivoli Integrated Portal server the ITMFEMRO_Windows.exe is launched to install the Reporting and Optimization feature. The executable launches an installation wizard that guides the user through the installation and configuration.

The wizard prompts for the Tivoli Common Reporting (TCR) and Tivoli Integrated Portal (TIP) configuration settings as shown in Figure 3-6 on page 29.

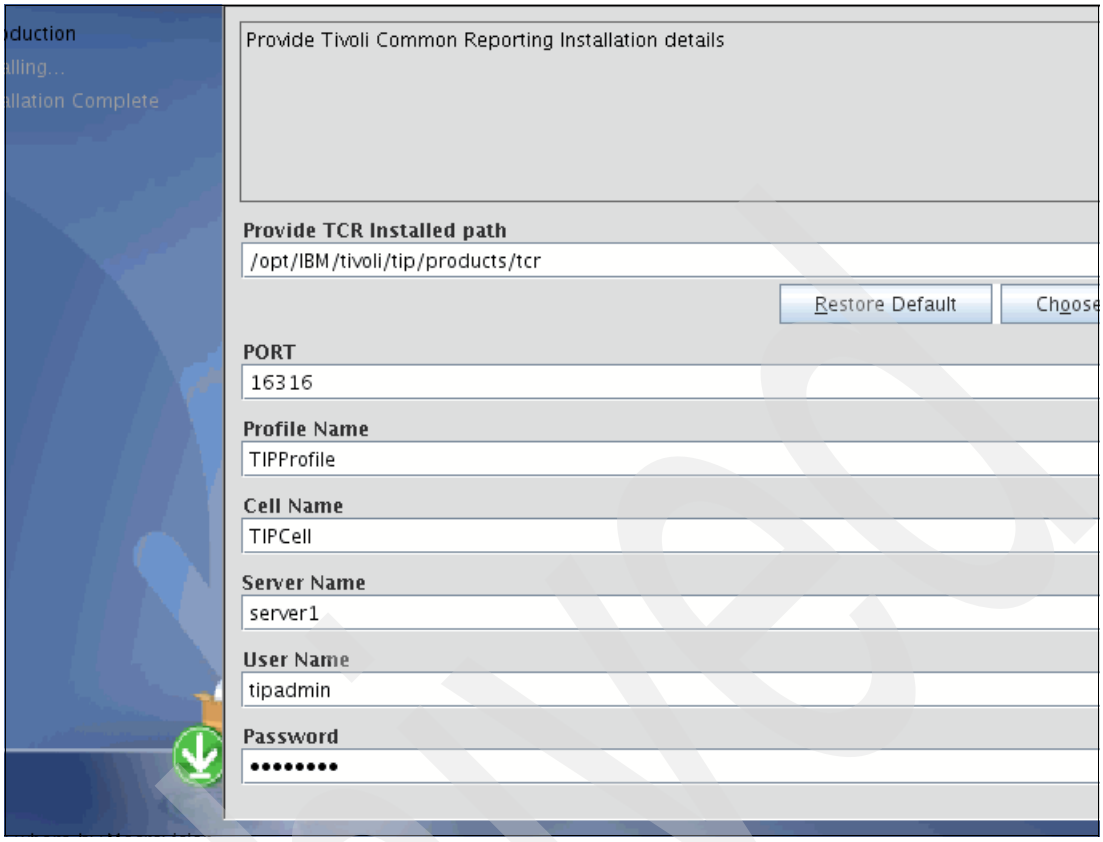


Figure 3-6 Reporting and Optimization TCR and TIP configuration settings

The following configuration settings must be provided:

- Provide TCR Installed path: This is the installation directory for the Tivoli Integrated Portal or Tivoli Common Reporting Server.
- Port: The port number for the Tivoli Integrated Portal Server has a default value of 16316 (secure access). The value for unsecure access is 16315.
- Profile Name: The Tivoli Integrated Portal profile name has a default value of TIPProfile in an embedded WebSphere Application Server.
- Cell Name: The Tivoli Integrated Portal cell name has a default value of TIPCell in an embedded WebSphere Application Server.
- Server Name: The name of the Tivoli Integrated Portal Server has a default value of server1 in an embedded WebSphere Application Server.
- User Name: The Tivoli Integrated Portal administrator user ID has a default value of tipadmin.
- Password: The password for the Tivoli Integrated Portal administrator user ID has no default value.

After providing the Tivoli Common Reporting and Tivoli Integrated Portal configuration settings, the wizard prompts for the Tivoli Data Warehouse configuration settings, as shown in Figure 3-7 on page 30.

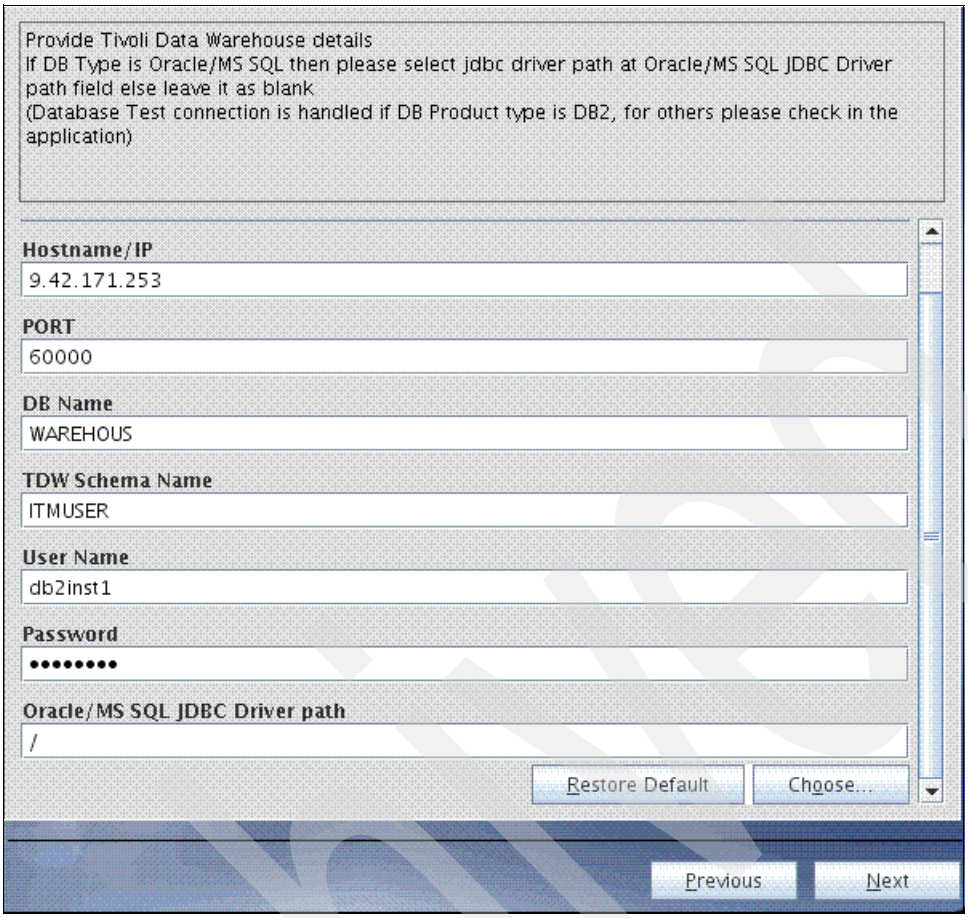


Figure 3-7 Reporting and Optimization Tivoli Data Warehouse configuration settings

The Tivoli Data Warehouse consists of the following configuration settings:

- Database Product type: The database server, such as DB2®, Oracle, or MS SQL Server
- Hostname/IP: The fully qualified host machine name or IP address for the database server
- Port: The number of the port for the database server that has a default value of 50000
- DB Name: The name of the Tivoli Data Warehouse database that has a default value of WAREHOUS
- TDW Schema Name: The database schema name
- User Name: The administrator user ID for the Tivoli Data Warehouse database
- Password: The password for the administrative user
- JDBC Driver Path: If DB product type is Oracle or MS SQL Server, provide the path to the respective JDBC or ODBC driver library.

When the wizard finishes, you can verify the correct installation of Reporting and Optimization by opening the Tivoli Integrated Portal Console. After logging on, you see a new “Energy” item in the explorer on the left side. Additionally when you expand the “Reporting” item there will be a new item in the common reporting section named Tivoli Monitoring for Energy Management. This section contains the various Tivoli Monitoring for Energy Management Reporting and Optimization reports as shown in Figure 3-8 on page 31.

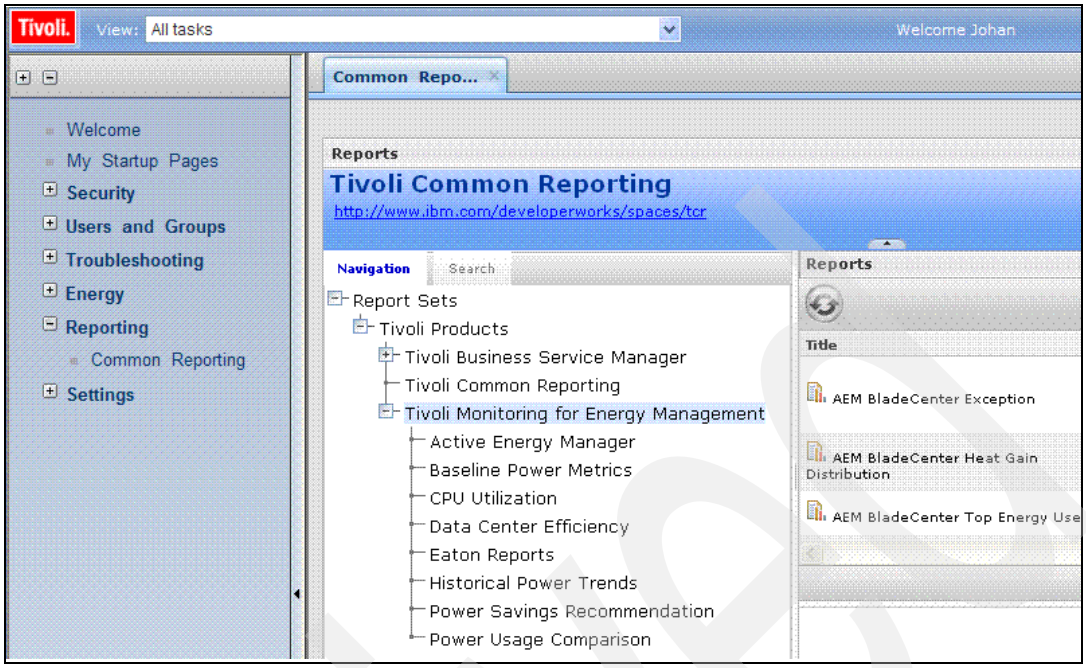


Figure 3-8 Verifying the installation of Reporting and Optimization

The data that is used for these Reporting and Optimization reports is extracted from the Tivoli Data Warehouse. Each report requires specific historical data. To make this historical data available the historical data collection of Tivoli Monitoring must be correctly configured.

Enabling the historical data for a specific attribute group is a two-step procedure. The historical data configuration is done using the Tivoli Enterprise Portal client's History Configuration option. First, specify the summarization and pruning options for the attribute group. Figure 3-9 on page 32 shows the summarization and pruning settings for the attribute group KE9_ALL_BLADECENTERS_POWER_DATA.

Note: E9 is the product code for the IBM Tivoli Monitoring for Active Energy Management Agent.

Archive

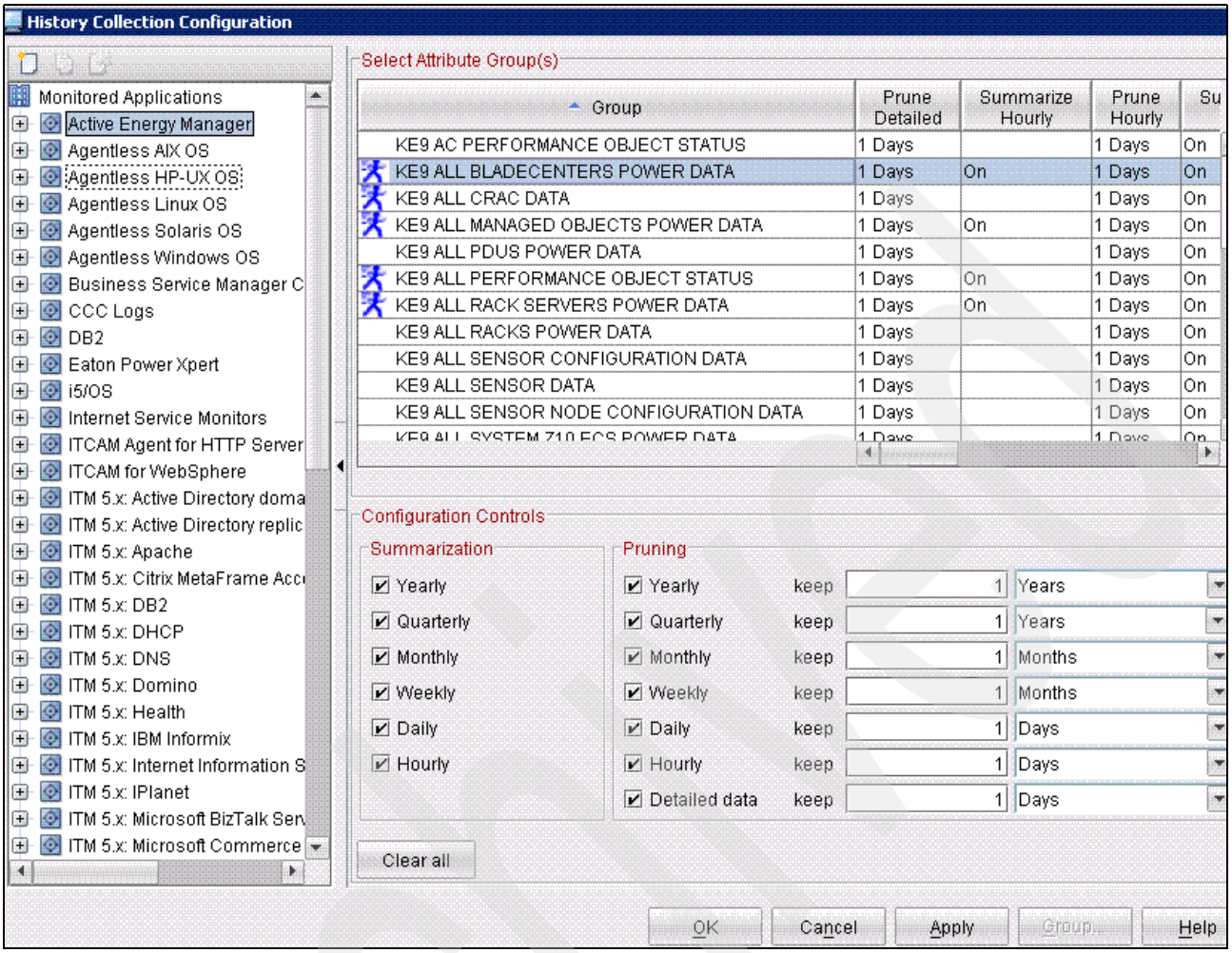


Figure 3-9 Summarization and pruning settings for AEM attribute group

It is important to turn on summarization for all data intervals: yearly, quarterly, monthly, weekly, daily, hourly. Also if you plan to use pruning to save disk space in the database, the minimum periods of time are: 3 years, 8 quarters, 8 months, 14 weeks, 14 days, 72 hours. More aggressive pruning can cause reports to not display data for earlier periods. There is no need to keep “Detailed data” after summarization.

Next, create a collection setting for the attribute group. The collection setting is shown in Figure 3-10 on page 33. The collection settings are as follows:

- Collection Interval: Interval at which the agent collects the attribute group data
- Collection location: Location where the short term historical data is stored, either at the TEMA or the Tivoli Enterprise Monitoring Server.
- Warehouse Interval: Interval at which the data is sent to the warehouse (1 hour, 12 hours or 1 day).

Note: Figure 3-10 on page 33 shows warehouse interval of one day. Set it to 1 hour if you want hourly Reporting and Optimization reports.

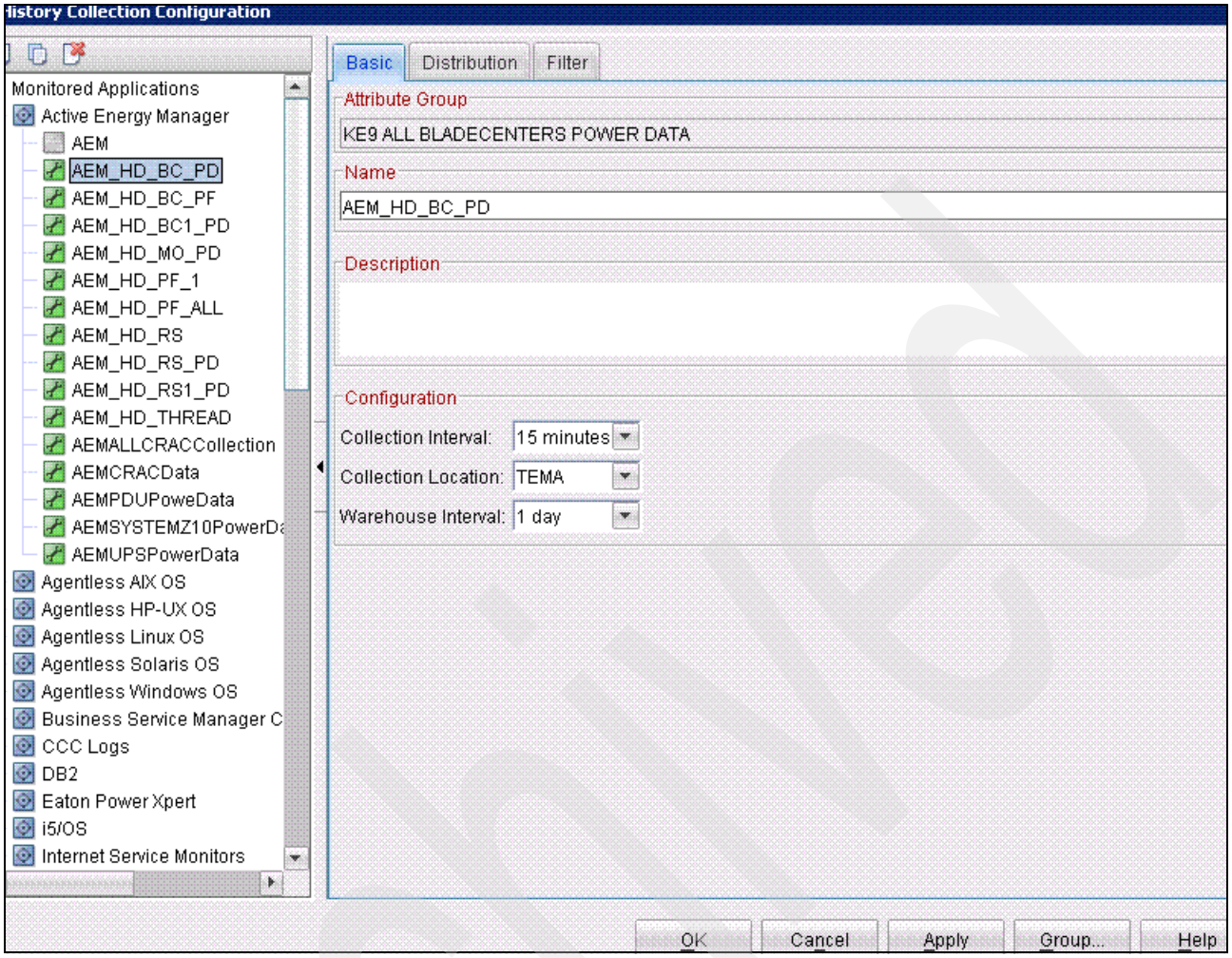


Figure 3-10 Historical collection settings for AEM attribute group

The *IBM Tivoli Monitoring for Energy Management Reporting and Optimization User's Guide*, SC27-2418, contains a complete description of which attribute groups are required for which Reporting and Optimization reports. The reports work only if the correct attribute groups have been configured for historical data collection.

Configuring Tivoli Monitoring for Energy Management Reporting and Optimization Preferences settings

After configuring the Tivoli Monitoring Data Warehousing settings, the Tivoli Monitoring agents start collecting the data for the attribute groups and forward it to the Tivoli Data Warehouse.

Before you can use the reports of Tivoli Monitoring for Energy Management Reporting and Optimization, you must configure the preferences. This step is done from the Tivoli Integrated Portal console by selecting **Energy** → **Optimization for Energy Management** → **Preferences** from the left Explorer window.

Next, perform the following steps:

1. Configure the “Server Settings” by clicking the **Data Refresh Settings** as shown in Figure 3-11.

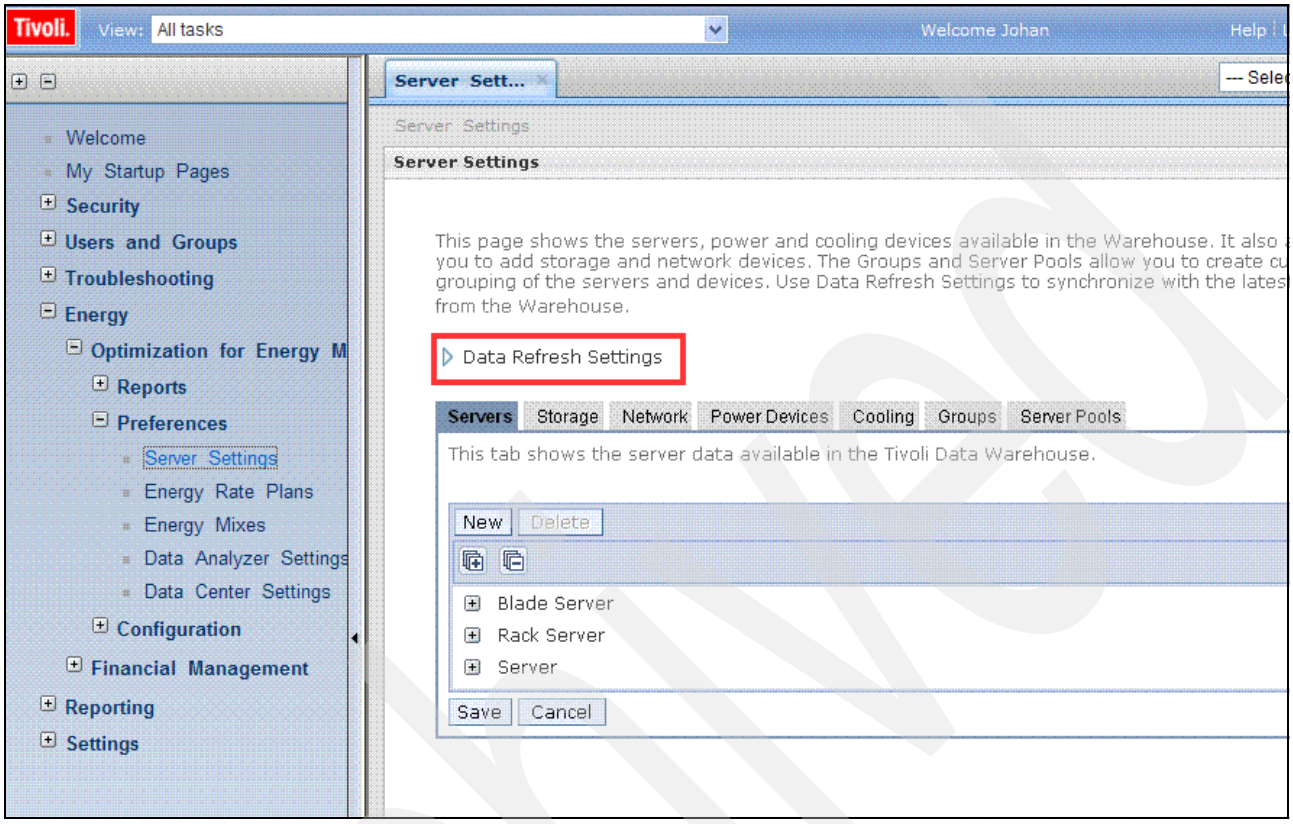


Figure 3-11 Reporting and Optimization Preferences

2. Select the **Refresh** option as shown in Figure 3-12.

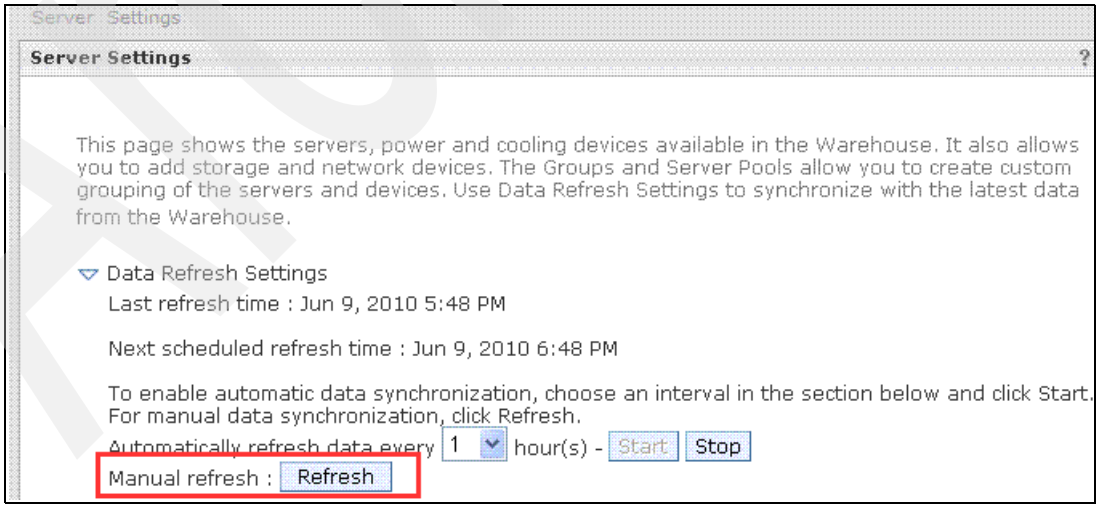


Figure 3-12 Refreshing the Reporting and Optimization data

This step starts the process of loading the most current data from the Tivoli Data Warehouse. If everything is configured correctly, you see the result shown in Figure 3-13.

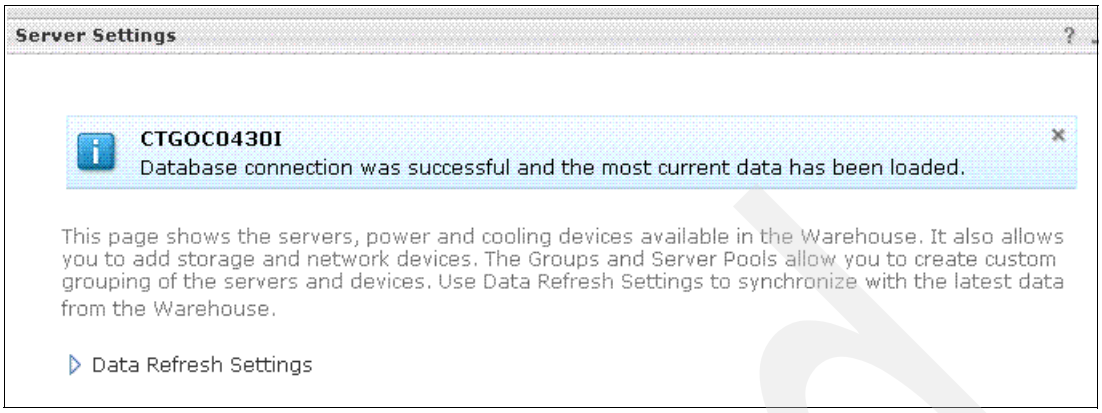


Figure 3-13 Data refresh successful

The tabs are as follows:

- The Server tab shows the Tivoli Monitoring for Energy Management devices for which you are storing data in the Tivoli Data Warehouse. These devices include servers, rack servers and blade servers depending on the configuration of your environment.
- The Storage tab is where you add storage devices such as tape and disk devices.

  In the Network tab, you add network devices such as routers, switches and repeaters.

  Note that the storage and network devices you add here are not added to the Tivoli Data Warehouse database; they are saved in the Reporting and Optimization configuration database.
- The Power Devices and Cooling tabs list the respective devices from the Tivoli Data Warehouse. In the Power Devices tab, you may link devices (such as servers, storage devices) with a power device.
- The Groups tab is where you define a logical collection of servers and devices in the data center and that share common characteristics, for example a group of servers and devices with the same physical location. These groups can be used to filter the results of an Reporting and Optimization report. For example, when running a report to display the power usage of the data center we can provide a group as a parameter to the report so that only the power usage of the devices within that group are shown.
- The Server Pools tab is where you define *server pools*, which are a collection of servers that are designated to run the same workload or application. Server pools are not stored in the Tivoli Data Warehouse, they are stored in the Reporting and Optimization configuration database. Figure 3-14 on page 36 shows an example of server pools.

  Server pools are used in the “Standby and Power-Down Recommendations Server Pool” report which is used to provide power-down and standby recommendations. In this report, Reporting and Optimization analyses the historical CPU data of the servers in a server pool to determine whether there are periods in which a server or servers within the pool can be placed in power save mode or can be powered down.

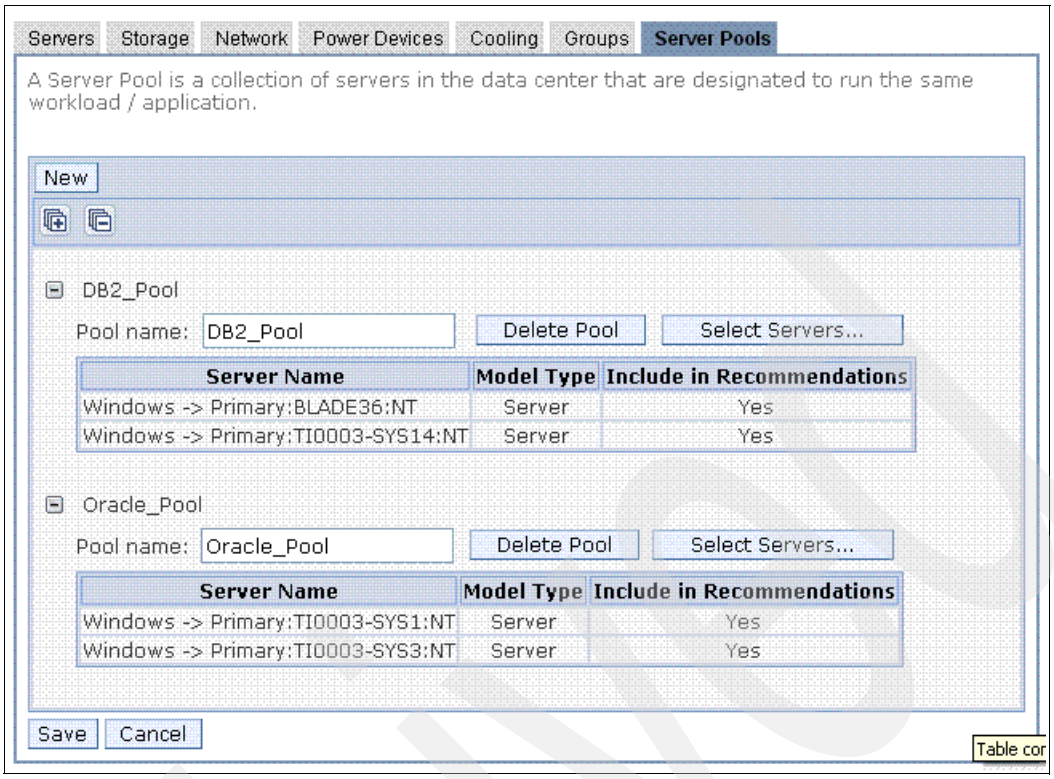


Figure 3-14 Configuring Server Pools

The Energy Rate Plan settings (Figure 3-15) is where you define several energy rate plans that are used for energy cost calculation.

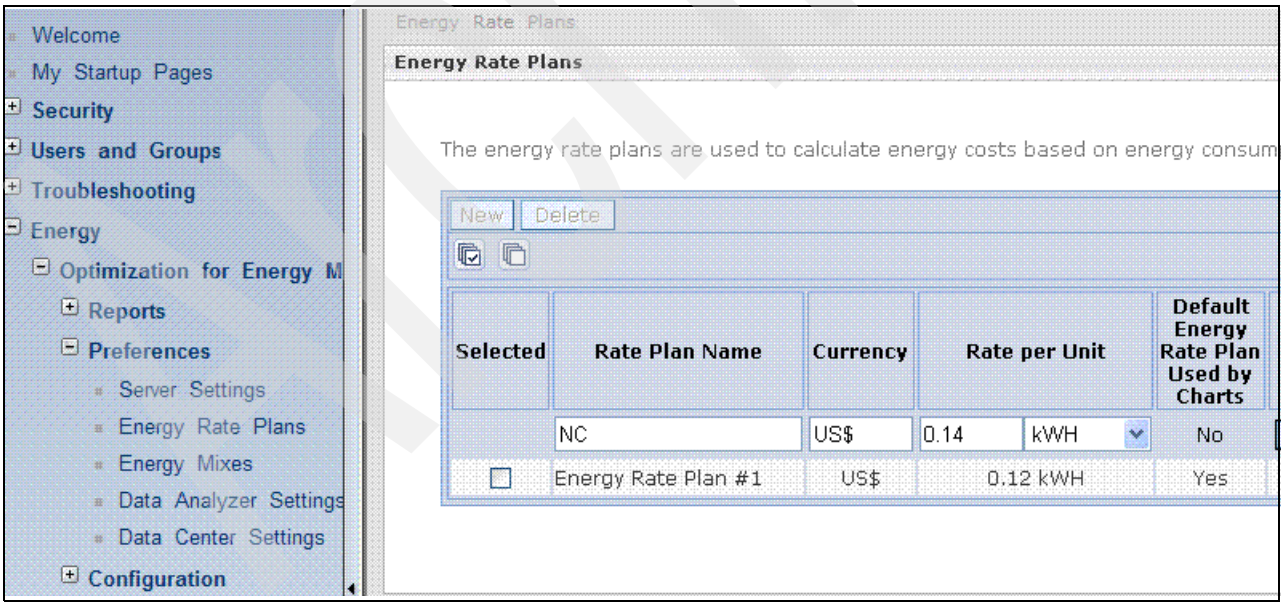


Figure 3-15 Energy Rate Plans

By default, “Energy Rate Plan #1” is defined during installation of the Reporting and Optimization product. You may define additional plans as shown in Figure 3-16 on page 37.

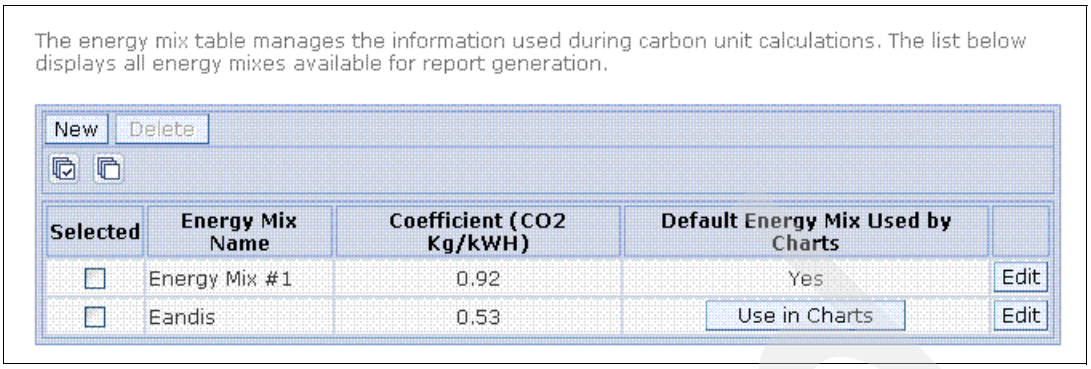


Figure 3-16 Adding an energy rate plan

The Data Analyzer Settings define the scheduling and running time of the Reporting and Optimization data analysis. The data analysis generates the data required by the Power Savings Recommendation report set (Standby mode recommendation, Power-down mode recommendation, and Dynamic power saving mode recommendation). The scheduling options are shown in Figure 3-17.

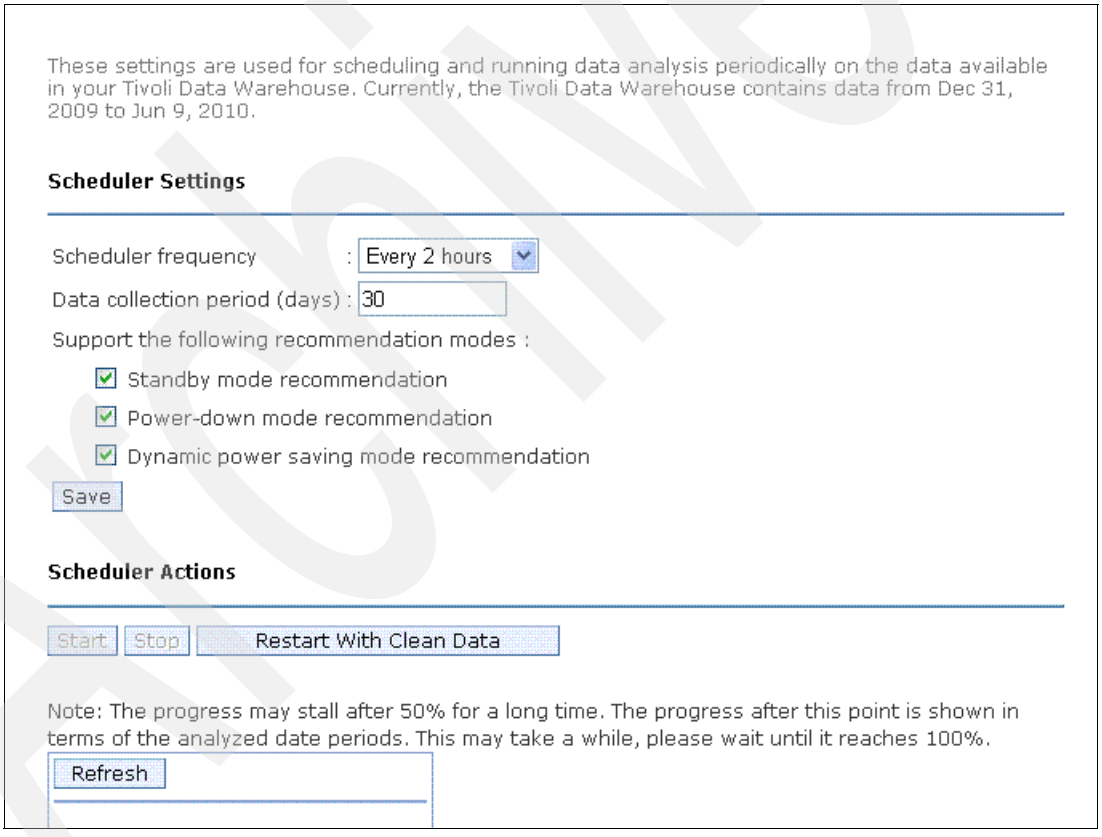


Figure 3-17 Data Analyzer Settings

The Data Center Settings is where you define data center parameters that are not collected by the Tivoli Monitoring for Energy Management monitoring infrastructure. Here you can define the maximum power that is available to the data center and the maximum cooling capacity (Figure 3-18 on page 38). These parameters are used in the Data Center Power Capacity chart and the Data Center Cooling Capacity chart.

Archive

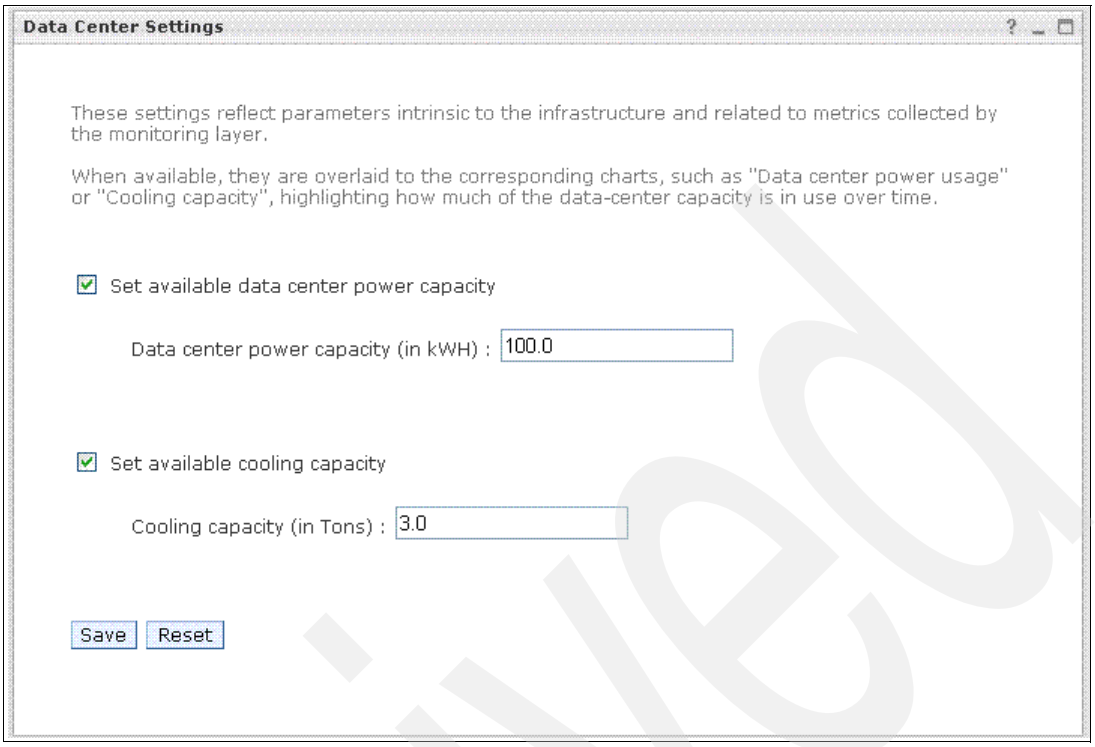


Figure 3-18 Data Center Settings

3.6.3 Configuring Tivoli Usage and Accounting Manager for collecting Active Energy Manager information

Tivoli Usage and Accounting Manager has several embedded data collectors on the product, called *core collectors*. One collector enables Tivoli Usage and Accounting Manager to collect, from IBM Tivoli Monitoring, data that is stored on Tivoli Data Warehouse and treat it to more complex costs analysis.

The following tables in Tivoli Data Warehouse that are created by Tivoli Monitoring for Energy Management are read by the Tivoli Usage and Accounting Manager collector:

- KE9_ALL_RACK_SERVERS_POWER_DATA
- KE9_ALL_BLADECENTERS_POWER_DATA
- KE9_ALL_SYSTEM_Z10_ECS_POWER_DATA
- KE9_ALL_PDUS_POWER_DATA

Configuring the collection of energy metrics and its use on Tivoli Usage and Accounting Manager reports requires the following steps:

1. Access to Tivoli Usage and Accounting Manager engine server as chargeback administrator using a URL similar to the following format:

   `http://IP:11052/ibm/console`

2. Select **Usage and Accounting Manager** → **System Maintenance** → **Data Sources**, as shown in the Figure 3-19 on page 39.

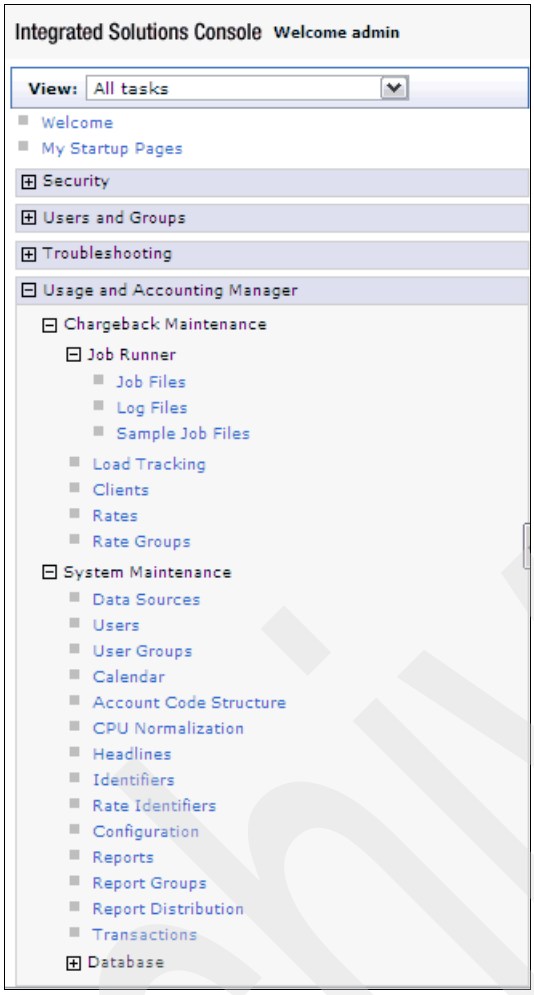


Figure 3-19 Tivoli Usage and Accounting Manager tasks list

3. Create a new collector data source by clicking **Collector-Database** → **New**, as shown in Figure 3-20 on page 40.

Archived

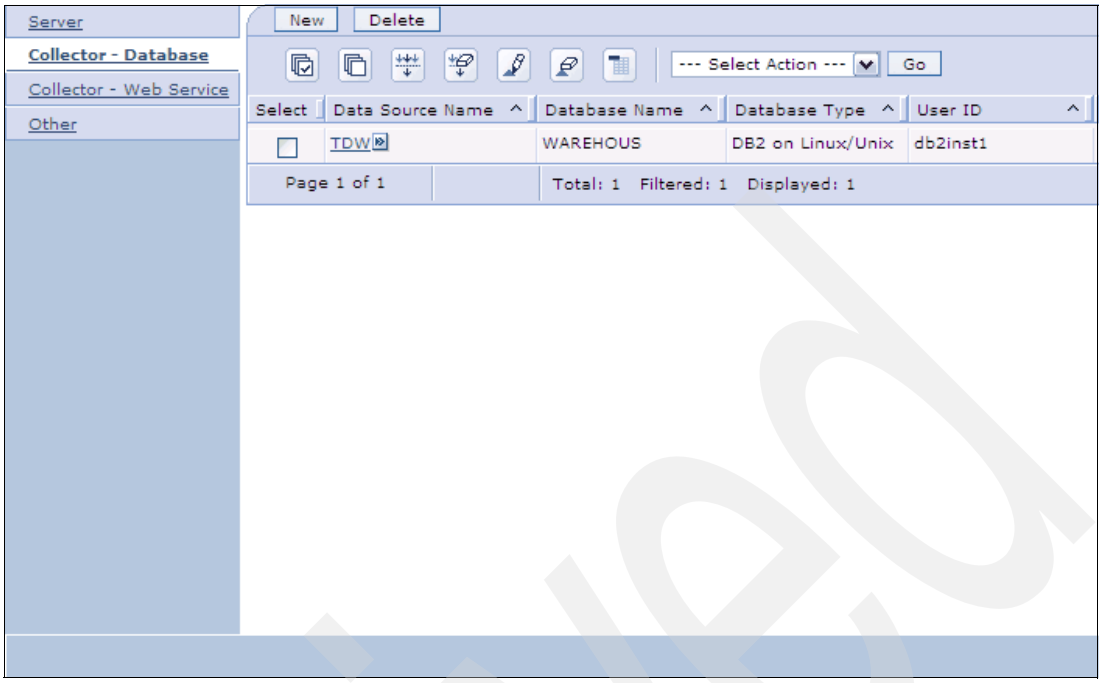


Figure 3-20 Tivoli Usage and Accounting Manager Data Source creation

4. Introduce the correct parameters for the Tivoli Data Warehouse connection, as in Figure 3-21.

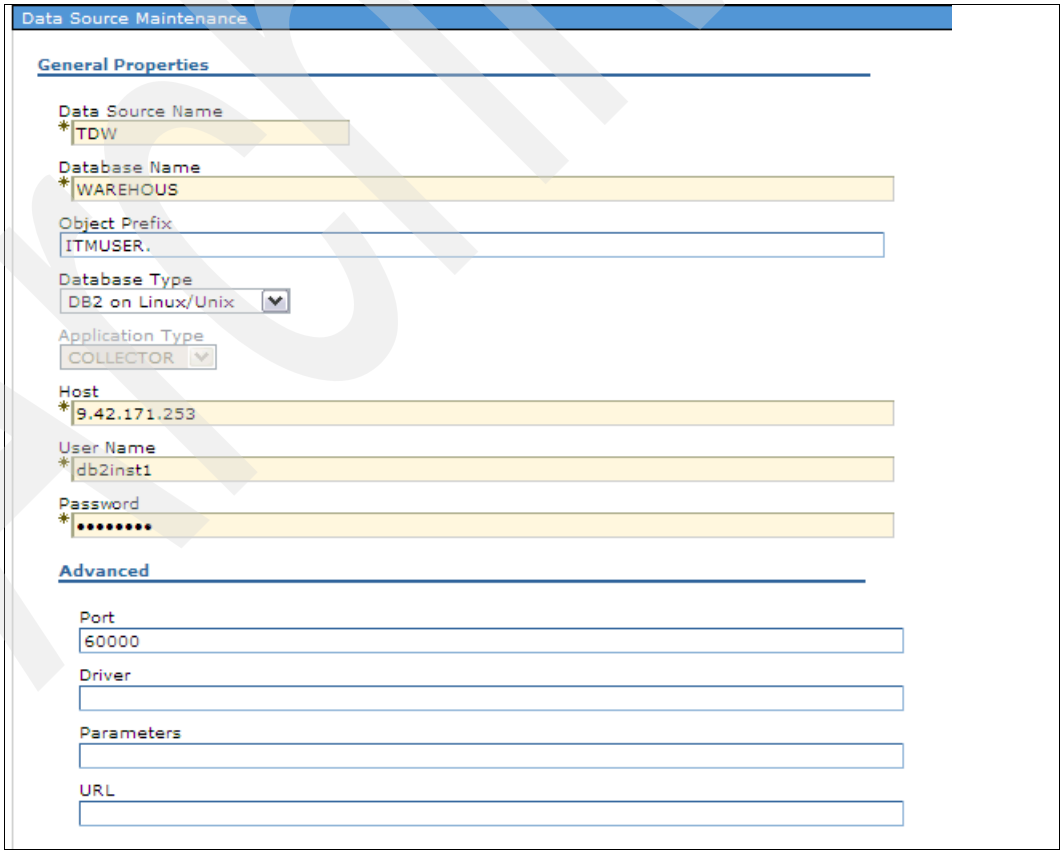


Figure 3-21 Tivoli Data Warehouse general properties

Archived

5. Click **Test Connection** and if it works, click **OK**.
6. Access to Tivoli Usage and Accounting Manager (TUAM) Engine Server as root user.
7. Change to directory:

   `/opt/ibm/tuam/samples/jobfiles`
8. Make a copy of the sample file for the jobrunner to the correct directory and rename it:

   `cp SampleTDW_AEM.xml /opt/ibm/tuam/jobfiles/TDW_AEM.xml`
9. Edit the file `TDW_AEM.xml` and look for the following lines:

   `<Connection dataSourceName="tf620b"/>`
10. Replace these lines with the following line and save the file:

    `<Connection dataSourceName="`*`TDW`*`"/>`

    Where *dataSourceName* must be the same one that is used in Figure 3-21 on page 40.
11. Include a new line in the cron table (crontab) to create a jobrunner task by using the following command:

    `crontab -e`
12. Add a line similar to the following one:

    `15 * * * * (/opt/ibm/tuam/bin/startJobRunner.sh TDW_AEM.xml >> /opt/ibm/tuam/logs/jobrunner/TDW_AEM.log 2>&1)`
13. Access Tivoli Usage and Accounting Manager process engine console and go to **Job Runner** → **Log Files**. See the results for the logs (Figure 3-22).

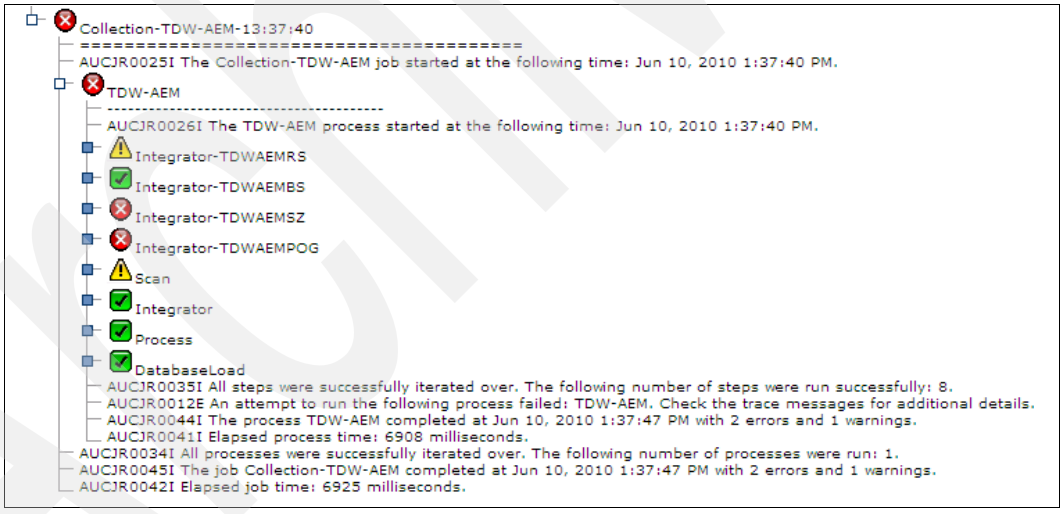


Figure 3-22 JobRunner log files

Note: You might receive error messages indicating that you are not monitoring certain resources, for example PDUs. In that case, the table does not exist Tivoli Data Warehouse, and a red indicator is shown. The remaining collections work.

Archived

3.7 Scenario walk-through

This scenario describes how IBM Tivoli Monitoring for Energy Management can be used to monitor and visualize thermal and power data combined with application and system data from a real-life data center.

3.7.1 Data Center environment

The data center used in this scenario consists of the following IT infrastructure:

- Rack Servers: IBM x3850 and x3950
- BladeCenter Chassis: IBM HS21, LS41
- Power Systems: IBM 550, 575
- System Z10

The data center also contains the following facilities equipment:

- Intelligent Power Distribution Unit (iPDU): APC, Eaton
- Uninterrupted Power Supply (UPS): APC, Eaton
- Computer Room Air Conditioner (CRAC): Liebert
- Chillers: JCI Metasys

Figure 3-23 shows the setup of the monitoring infrastructure for this data center. The various IT and facilities components of the data center are monitored using IBM Systems Director for Active Energy Manager. The AEM agent forwards the data to Tivoli Monitoring, where the monitoring data is stored in the Tivoli Data Warehouse.

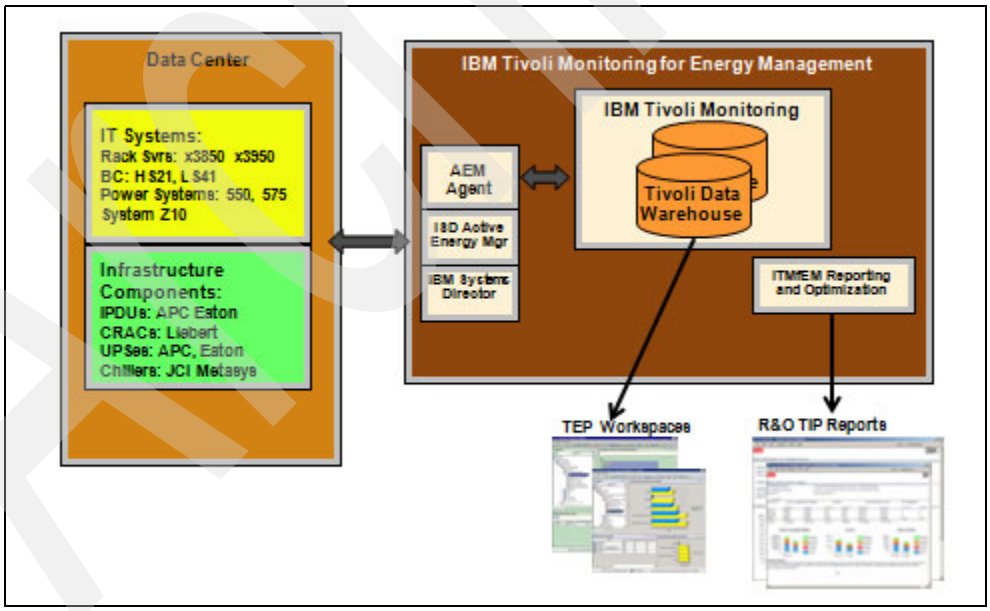


Figure 3-23 Data Center setup: Green IT scenario 1

The data can be viewed using various Tivoli Enterprise Portal workspaces and using the Reporting and Optimization (R&O) feature of Tivoli Monitoring for Energy Management we can analyze the data and display reports showing data center efficiency, energy cost, carbon footprint and many more. The Reporting and Optimization feature also provides features for

making recommendations for power savings, standby and power down recommendations for server pools, allowing you to reduce data center power consumption and associated cost.

3.7.2 Visualizing and reporting energy and thermal metrics

The data provided by the Tivoli Monitoring for Energy Management agents can be consulted using the ready-to-use Tivoli Monitoring for Energy Management Tivoli Enterprise Portal workspaces. In our example data center we are using the AEM agent. The workspaces for the AEM agent will be displayed on the navigator item of the system where the agent is installed (which is also the IBM Systems Director machine). Figure 3-24 shows an example of the Active Energy Manager workspaces. The figure shows the data for a specific BladeCenter (BC5MM). The views in this workspace show the average power consumption (AC and DC) and the thermal data (exhaust and ambient temperature) for this BladeCenter.

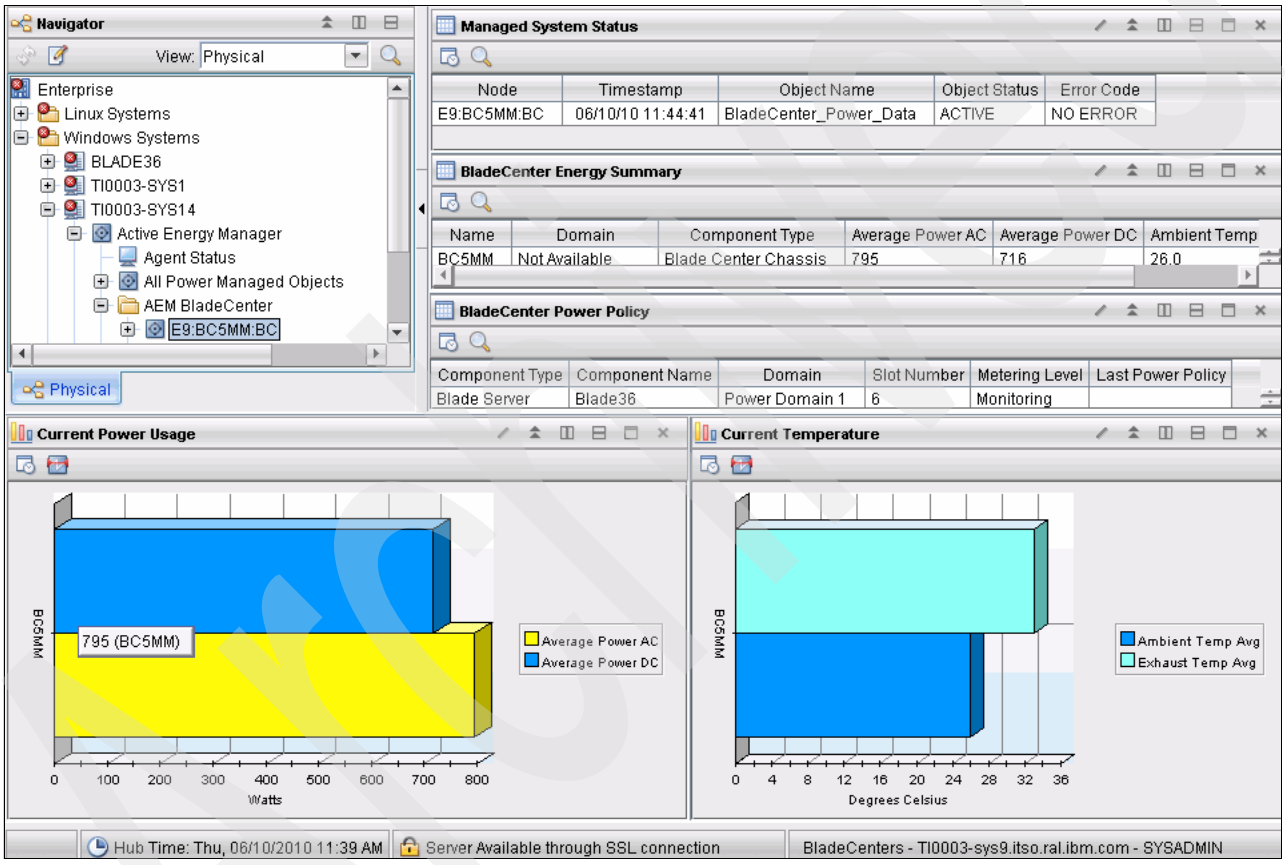


Figure 3-24 Default Active Energy Manager workspace

The AEM workspaces that are provided with Tivoli Monitoring for Energy Management are all described in *IBM Tivoli Monitoring for Energy Management: Active Energy Manager User's Guide*, SC23-9741.

The IBM Tivoli Monitoring administrator can create custom workspaces displaying the thermal and power data in various views. Before showing examples of custom workspaces, we briefly explain how to create custom workspaces in Tivoli Enterprise Portal.

Creating custom workspaces

This section uses a simple example to describe how to create a custom workspace. The basic basic steps are as follows:

- Create a new “navigator” named “IBM Green IT Data Center”
- In the new navigator create a new “navigator item” named “Real-Time Power/Thermal/CPU data”
- For the new navigator item create a new workspace. The new workspace contains 3 new views showing a plot diagram of CPU, power and thermal data

Perform the following steps to create a custom workspace:

1. Create a new navigator. By default, Tivoli Enterprise Portal includes two ready-to-use navigators: Physical navigator and Logical navigator. Enterprise is a node in the Physical navigator tree. Creating a new navigator is done by clicking the **Edit Navigator View** from the top menu bar, as shown in Figure 3-25.

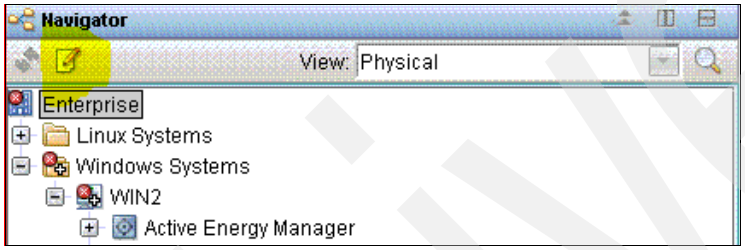


Figure 3-25 Create new navigator

2. In the Edit Navigator View window, click the **Create New Navigator** icon.
3. Specify the name and description of the new navigator (Figure 3-26 on page 45).

Archived

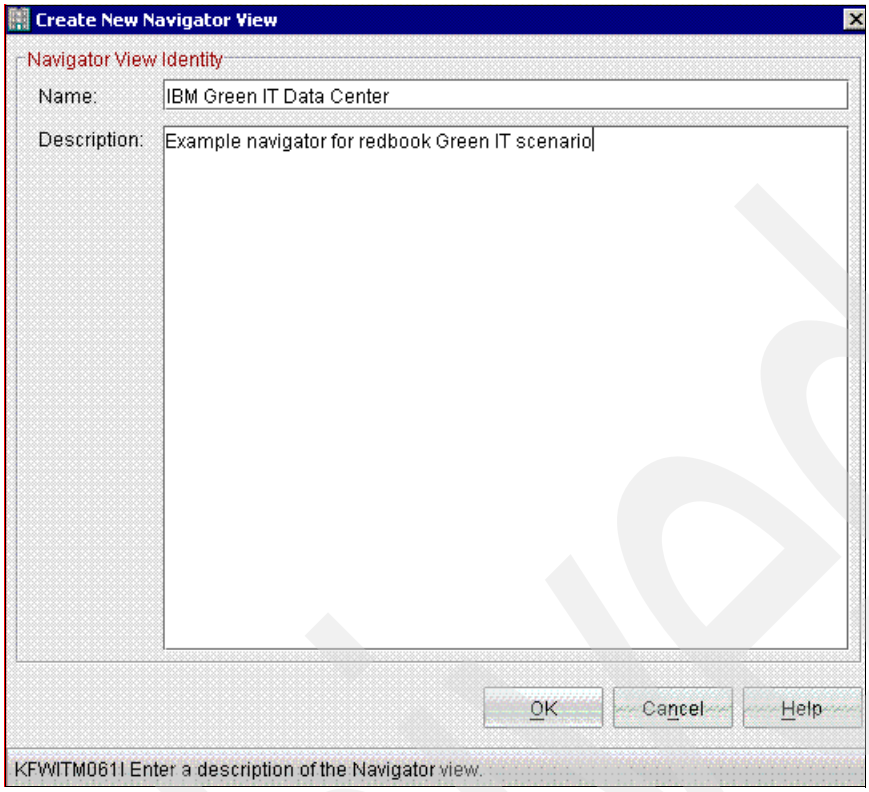


Figure 3-26 Create new navigator view

4. Click **OK**. The new navigator is now created and is initially empty.
5. Add navigator items to the navigator by clicking **New navigator item**, as shown in Figure 3-27.

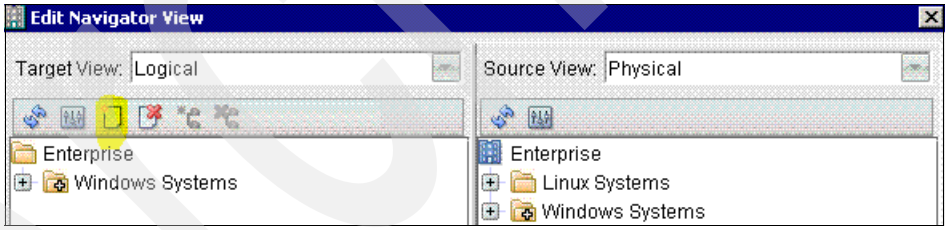


Figure 3-27 Creating a new navigator item

6. The definition of the new navigator item is shown in Figure 3-28 on page 46. Specify a name, a description and a list of “Assigned” systems. The assigned system list is important because it determines for which systems we can display data in the workspaces that we will define for this navigator item.

Archived

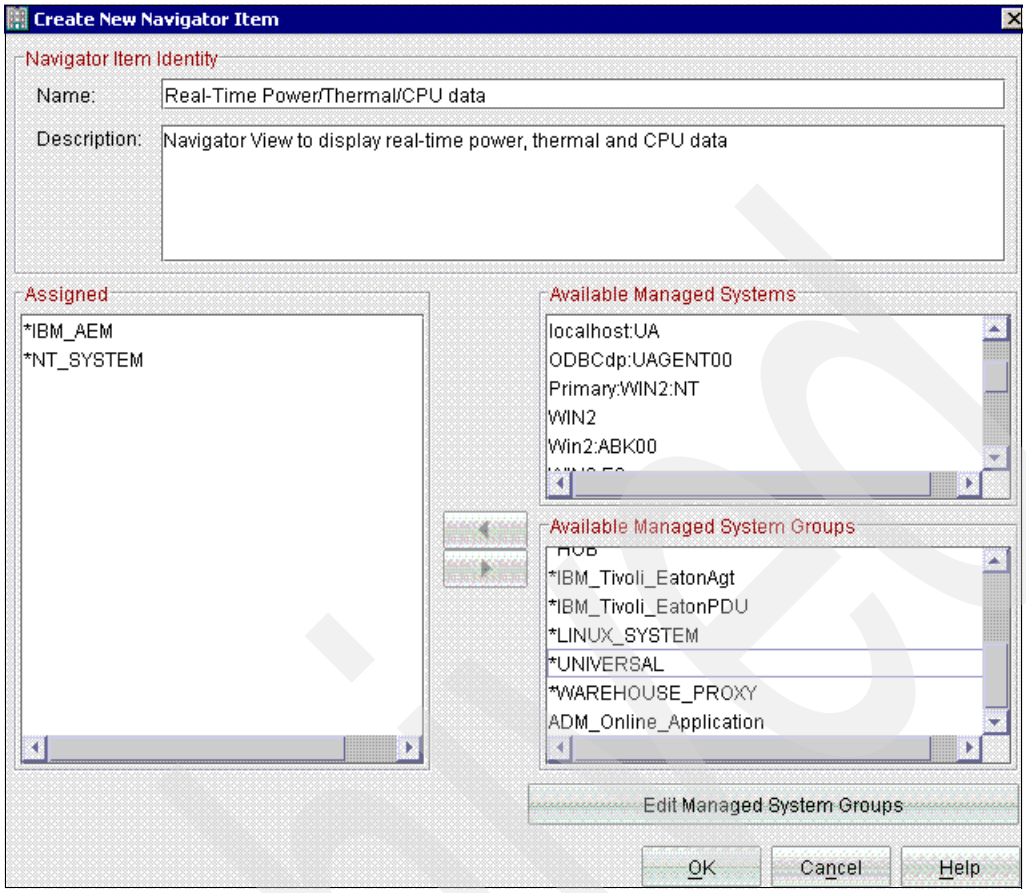


Figure 3-28 Creating a new navigator item

In the example, we have selected two Managed System Groups: IBM AEM and NT_SYSTEM. These are dynamic groups that contain all Active Energy Manager systems and Windows systems respectively.

7. Access the new navigator from the Tivoli Enterprise Portal by using the View drop-down list of the Tivoli Enterprise Portal client (Figure 3-29).

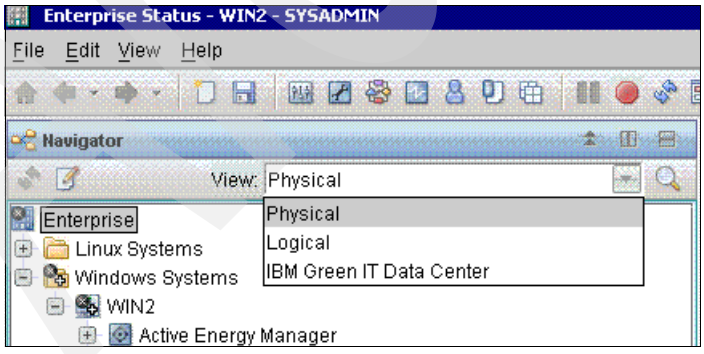


Figure 3-29 Opening the new navigator view

After clicking the navigator item you see a workspace as shown in Figure 3-30 on page 47.

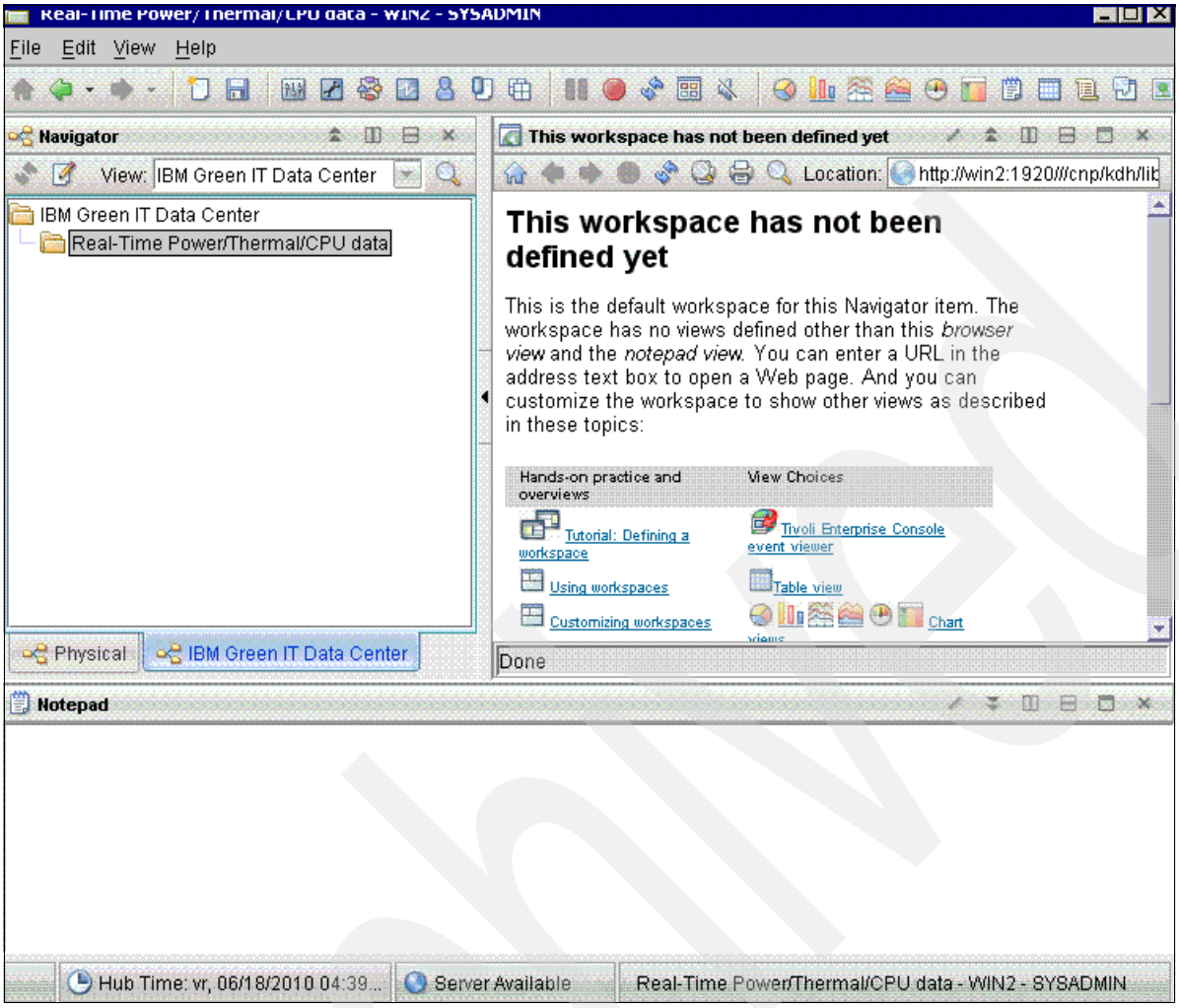


Figure 3-30 Opening the new navigator item

The workspace has not been defined yet. By default it contains two views: a web browser view and a Notepad view. In our example we want to create a workspace that contains three plot chart views, showing CPU, power and thermal data.

8. To create these three views, first split the default Notepad view (at the bottom of the workspace) in two separate view by selecting the **Split View** icon as highlighted in Figure 3-31.

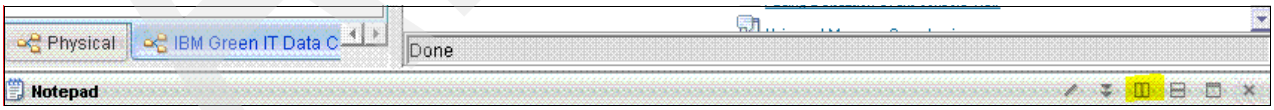


Figure 3-31 Splitting a view

9. Replace the web browser view at the top of the workspace with a plot chart view by selecting the **Plot Chart** view icon from the Tivoli Enterprise Portal menu bar (highlighted in Figure 3-32 on page 48) and drag it onto the web browser view. The web browser view is now replaced by a plot chart view.

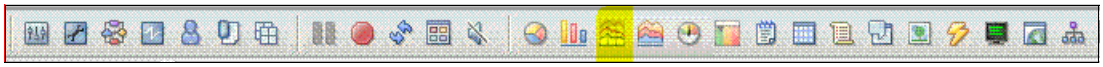


Figure 3-32 Adding a plot chart view

10. Define the properties for the plot chart view: When prompted to assign a query for the new view (Figure 3-33), click **Yes**. The query that is assigned to a view determines which data will be available for displaying in the particular view.

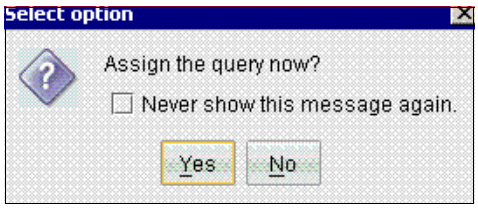


Figure 3-33 Assign query dialog

A window opens as shown in Figure 3-34.

11. Select **Click here to assign a query.**

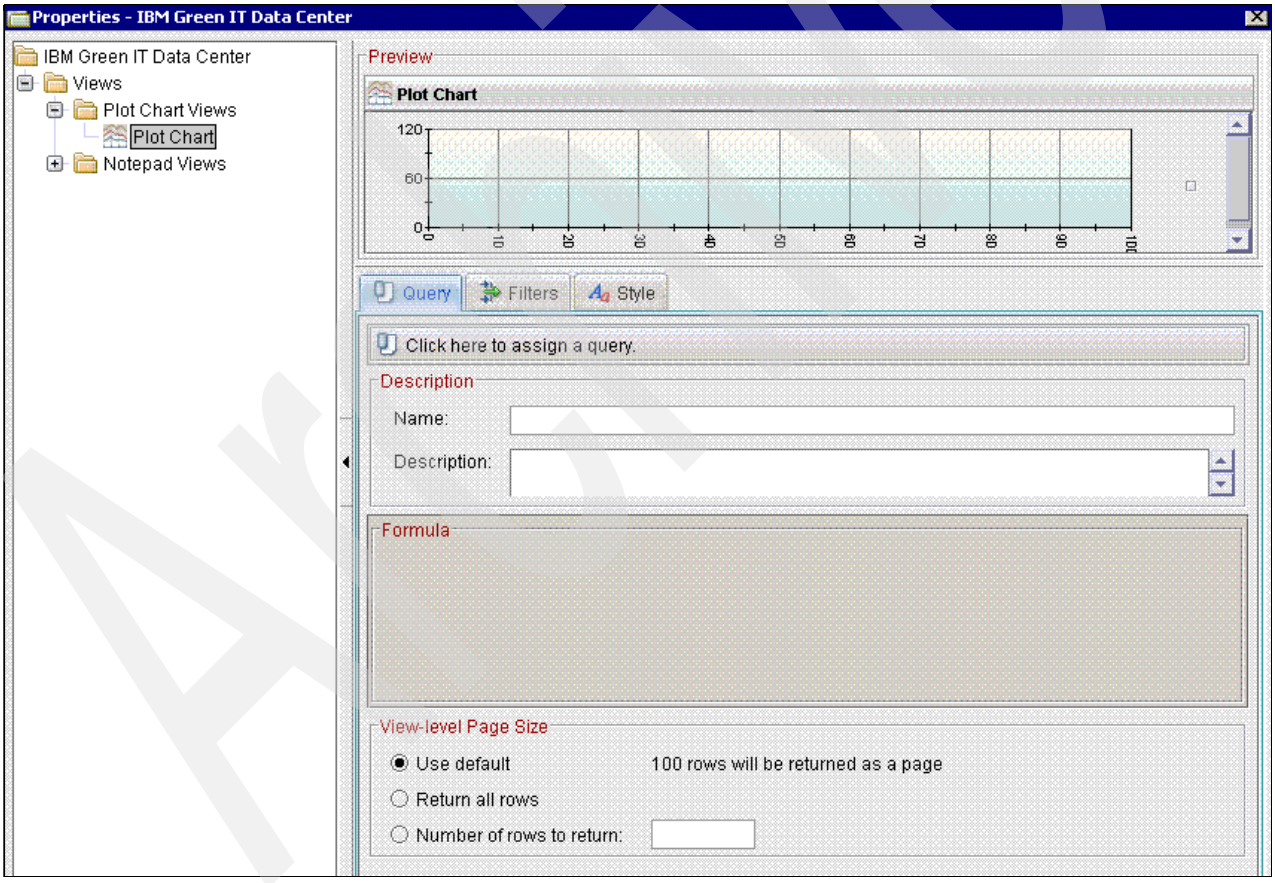


Figure 3-34 View properties

12. In our example, we want to display CPU data for our Windows servers in the plot chart view. We select the **Processor Overview** query in the Windows OS Processor object, as shown in Figure 3-35 on page 49.

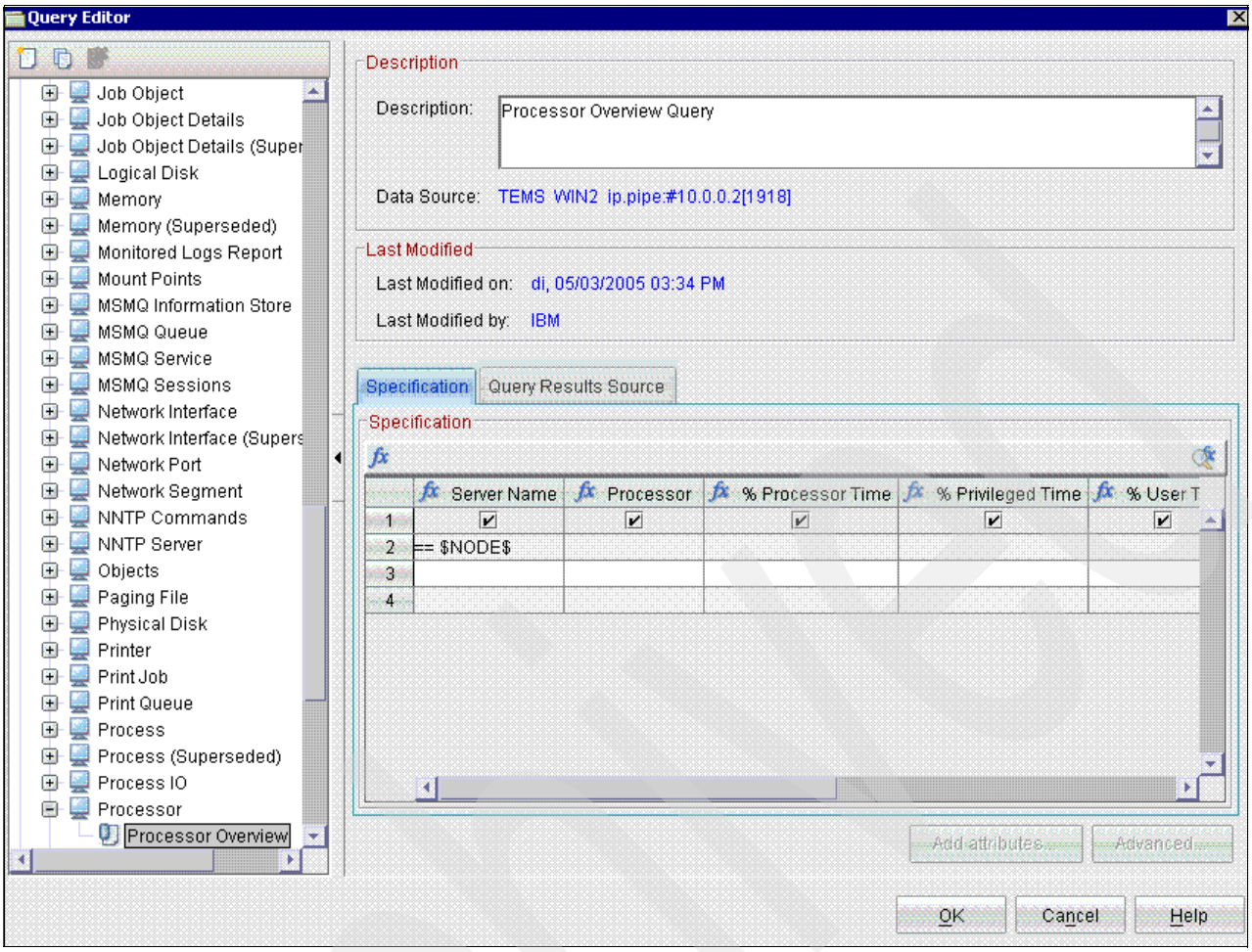


Figure 3-35 Query specification

The Processor Overview query returns a number of attributes such as `%Processor Time`, `%Privileged Time`, and more. In our plot chart view, we want to display only `%Processor Time`. To achieve this, we must define filters that are applied to the selected query.

13. Access the filters by clicking the **Filters** tab as shown in Figure 3-36 on page 50.

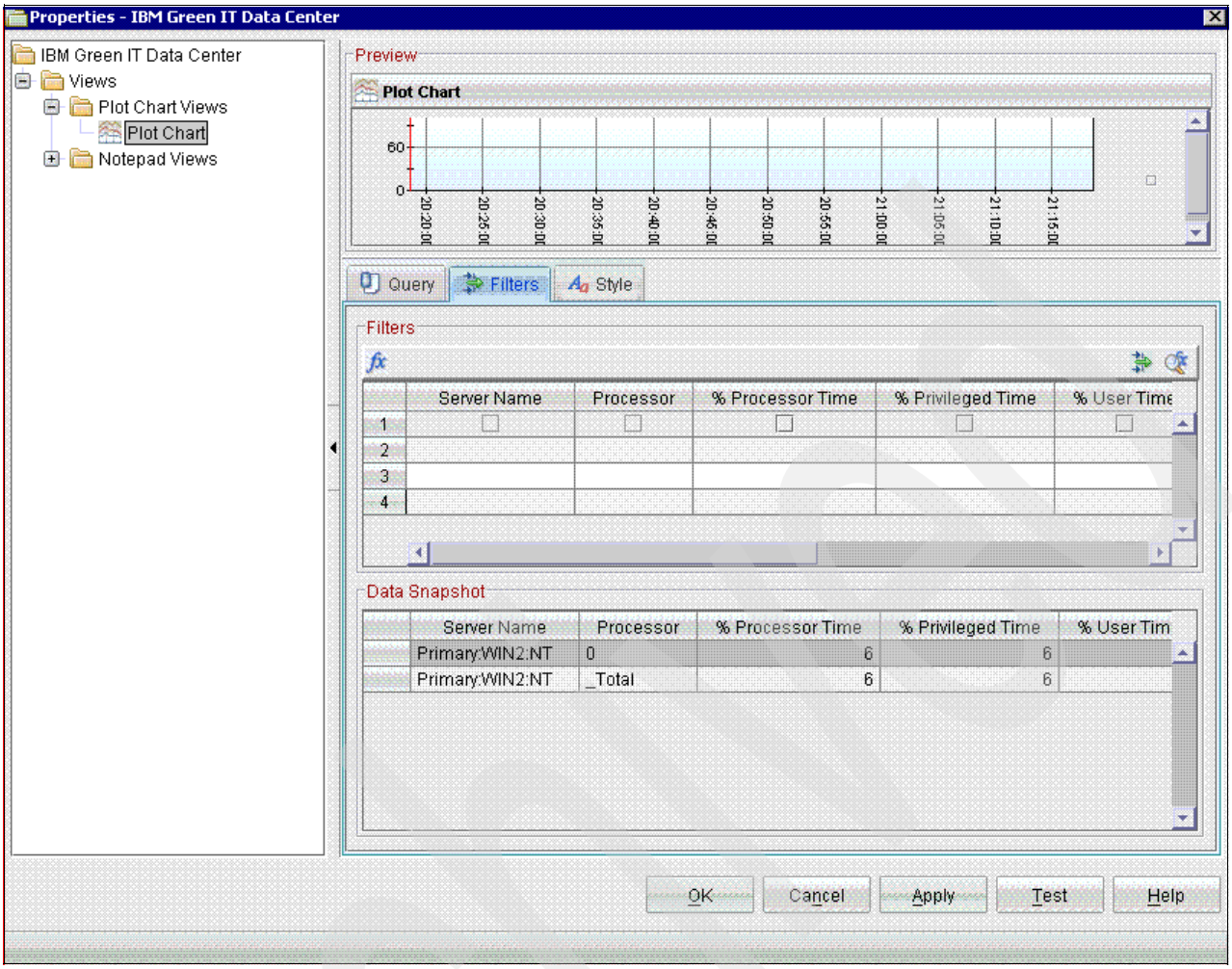


Figure 3-36 Defining query filters

14. For our example, we select the **%Processor Time** only for the `_Total` Processor (in a machine with multiple CPUs, the `_Total` is the average for all the CPUs). These filters are shown in Figure 3-37.

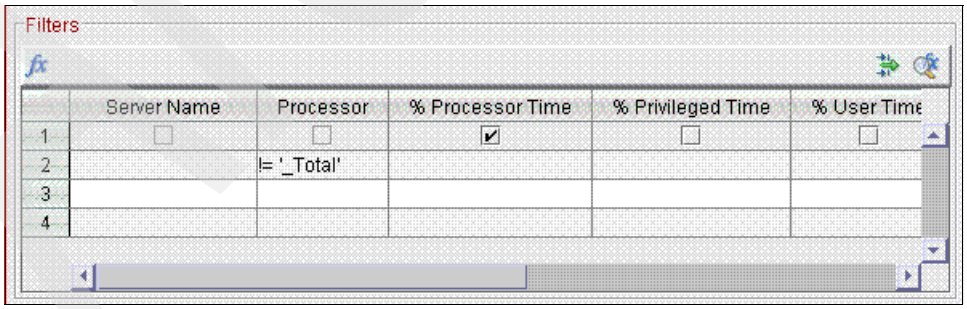


Figure 3-37 Filter specification

15. Besides specifying the query and the query filters, we must also define the Style of our view. Click the **Style** tab, shown in Figure 3-38 on page 51. This tab is where to define the plot-chart refresh interval (30 seconds in our example), the plot duration, and other settings (color, labels for x-y axis).

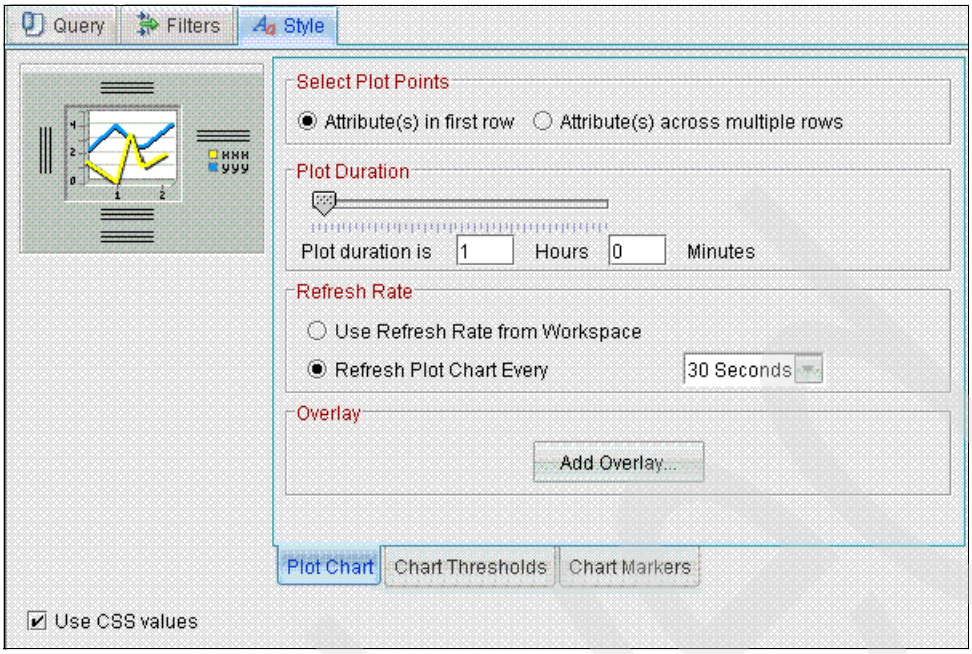


Figure 3-38 View style properties

After defining all the settings for our plot chart view, it will show data as shown in Figure 3-39 on page 52.

Archive

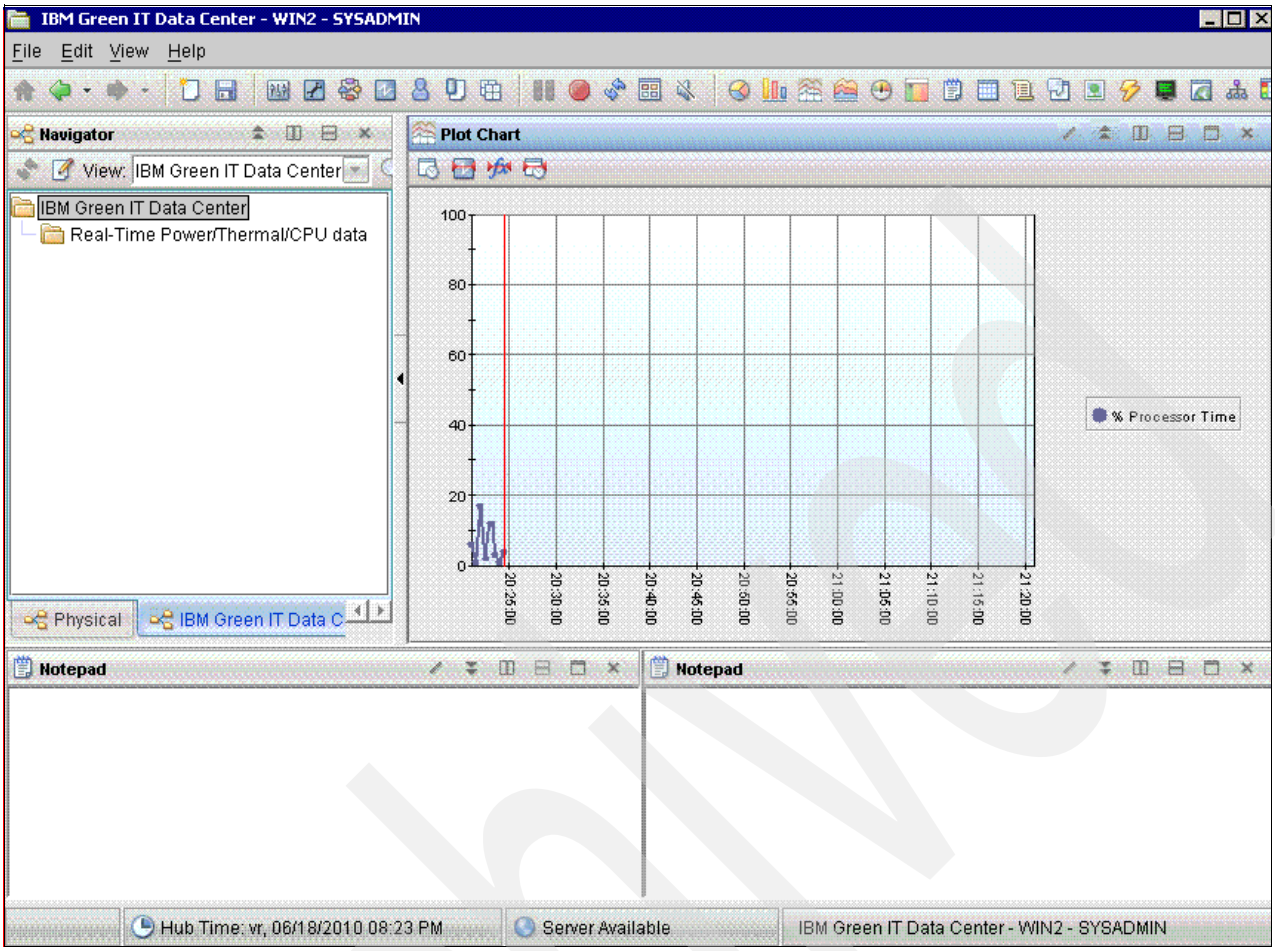


Figure 3-39 CPU Plot chart completed

16. Similarly you can define the other two views showing thermal and power data. The process of defining these views is similar to the example of the CPU plot chart view, we have just described. The main difference is the selection of the query for these views. The power and thermal data will be collected using a query of the Active Energy Manager agent. Figure 3-40 shows the query items that are available for the Active Energy Manager agent.

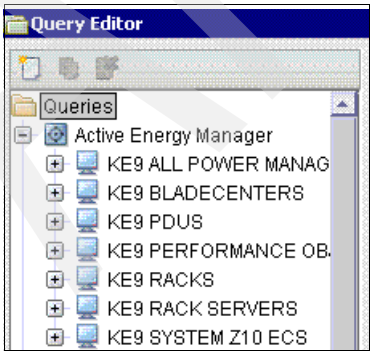


Figure 3-40 Selecting an Active Energy Manager query

Real-life examples of custom workspaces

The previous section, describes how to create a simple custom workspace. Figure 3-41 shows a real-life example of a custom workspace. This custom workspace contains three views. The upper view is a plot chart displaying the history of the total power consumption of the data center. The lower left view has a circular gauge showing the current power usage of the data center. The lower right view has a circular gauge showing the data center temperature.

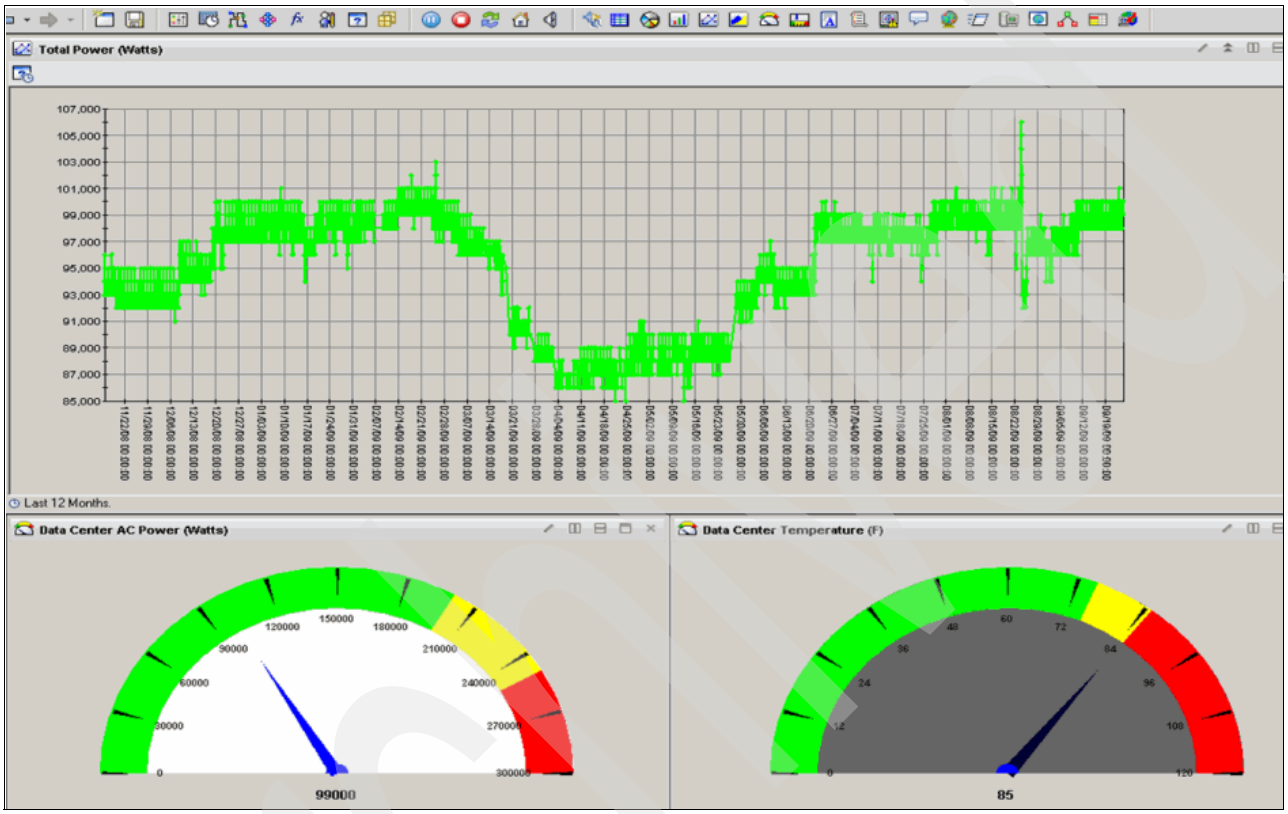


Figure 3-41 Total power and thermal data of the data center

Another example of a custom workspace is shown in Figure 3-42. In this example, two bar chart views are used to display CPU, memory usage, and number of guest machines (left view) and temperature and power usage (right view).

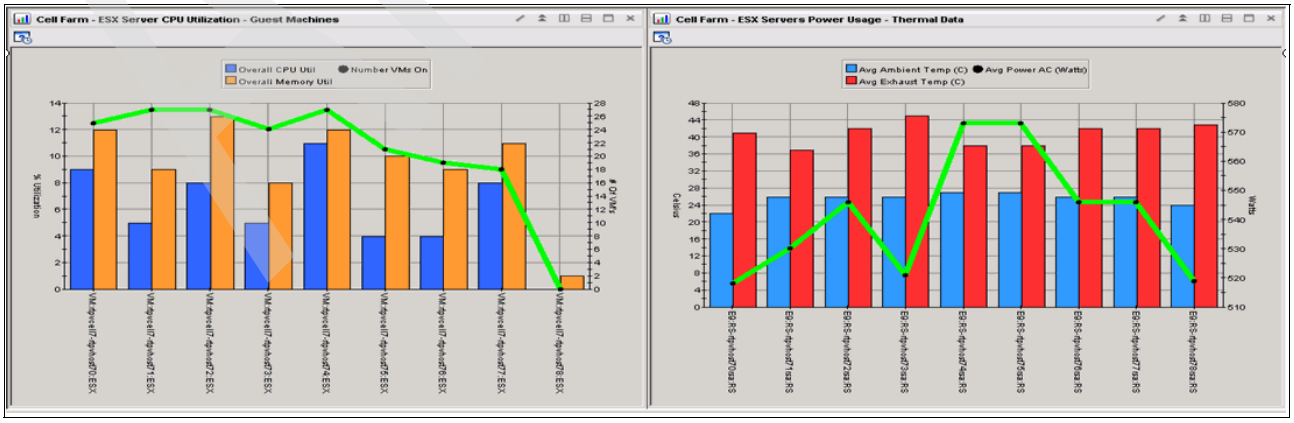


Figure 3-42 Blade Centers power and thermal data

Tip: Another typical scenario in Tivoli Monitoring for Energy Management environment is the ability to set a threshold and be alerted of it being exceeded.

3.7.3 Visualizing and reporting energy and thermal metrics using Tivoli Monitoring for Energy Management Reporting and Optimization

The reports provided with Reporting and Optimization are divided into a number of report sets. Figure 3-43 shows the report sets that are available with Tivoli Monitoring for Energy Management Reporting and Optimization.

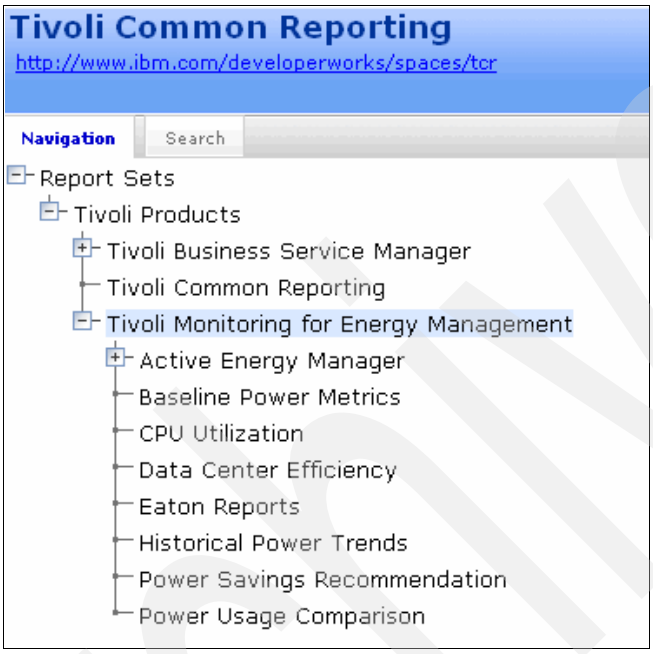


Figure 3-43 Reporting and Optimization Report sets

The reports and purpose of the reports are as follows:

- Active Energy Manager: Contains exception reports (showing devices that are being excluded from normal reports because of problem data), top energy-use reports and top heat-gain reports.
- Baseline Power Metrics: Contains the Data Center Power Usage report, displaying power usage details of the data center. See Figure 3-44 on page 55.
- CPU utilization: Contains the CPU Utilization of Servers report, which shows the CPU utilization of selected servers.
- Data Center Efficiency: Contains the DCIE report, which shows the Data Center Infrastructure Efficiency and PUE of the data center.
- Eaton Reports: Contains Eaton breaker reports.
- Historical Power Trends: Contains various reports displaying history data of power usage for various types of devices.

Archived

- Power Savings Recommendations: Contains reports that use calculated data to make recommendations for power-down, standby, and dynamic power-savings settings. These recommendations are based on the historical CPU data of servers within a server pool. These reports can also be used to forecast power consumption in a future date and time window.
- Power Usage Comparison: Contains the Data Center Power Usage Comparison report, which you use to compare two date periods for power usage, estimated cost, and carbon emission.

Figure 3-44 shows the Data Center Power Usage report. This report provides data concerning the total power usage, estimated cost and carbon emission of the data. It also shows how servers, facility devices, storage and network contribute to these totals.

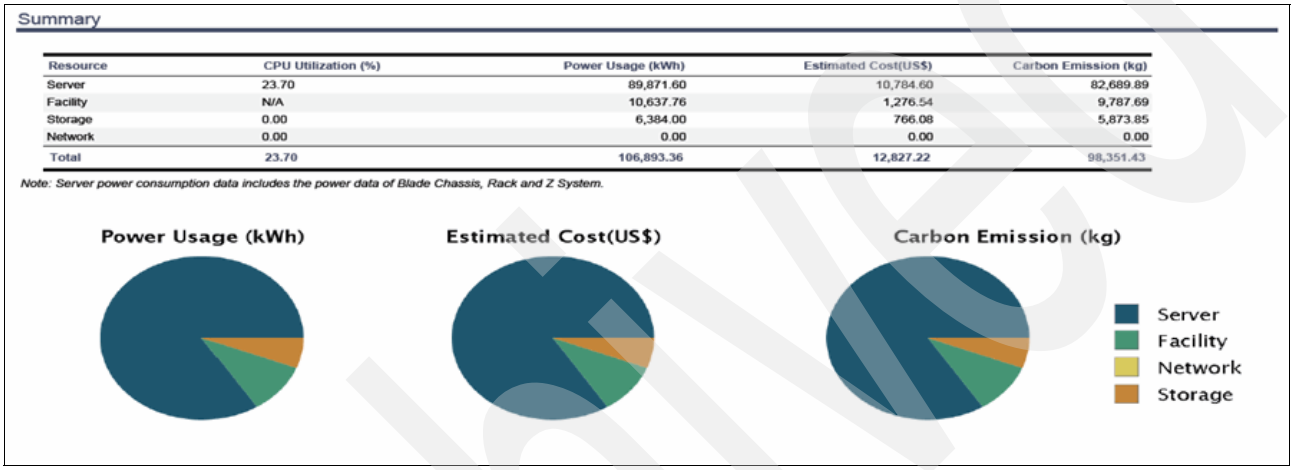


Figure 3-44 Reporting and Optimization Data Center Power Usage report

When executing the Reporting and Optimization reports, you must specify a number of parameters such as date and time ranges. In most reports, you may also allow specify a group of devices, as shown in Figure 3-45 on page 56. Creating server groups was described in “Configuring Tivoli Monitoring for Energy Management Reporting and Optimization Preferences settings” on page 33.

This feature is particularly useful because you may use it to filter the report data for a specific group of devices. For example, the report shown in Figure 3-44 can be filtered to show only data for the devices in the “North” group.

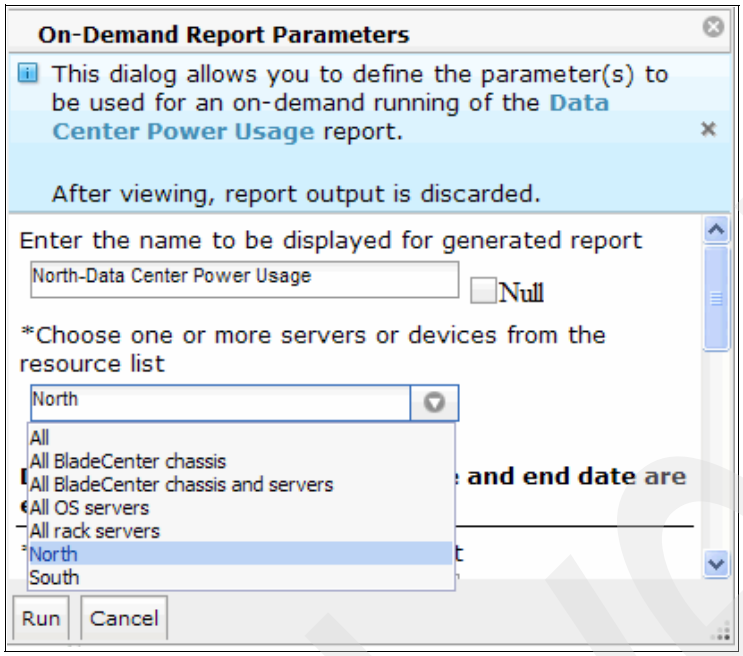


Figure 3-45 Reporting and Optimization Report Parameters

Figure 3-46 shows the Data Center Infrastructure Efficiency report.

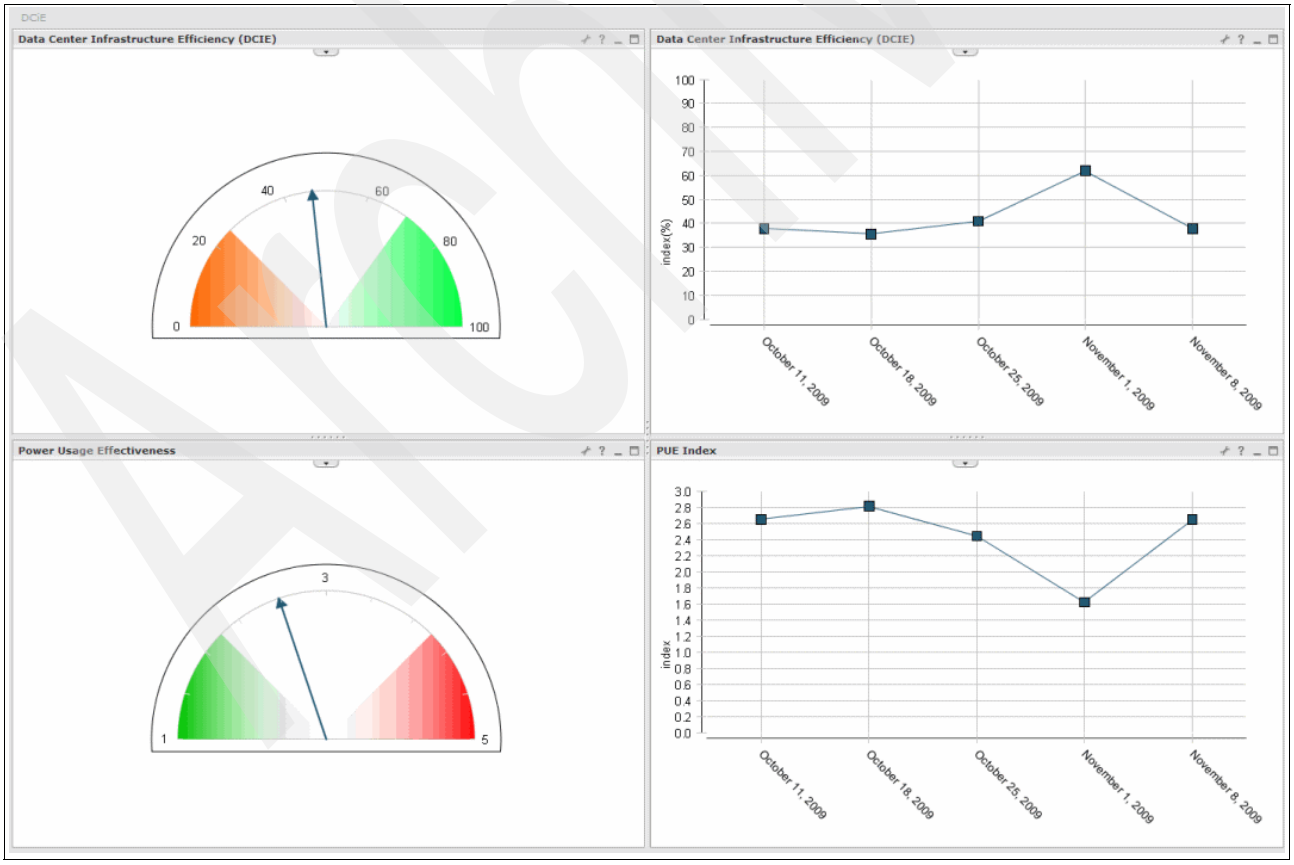


Figure 3-46 Reporting and Optimization Data Center Infrastructure Efficiency report

This report contains two important indicators of the “greenness” of a data center:

- DCIE: The data center infrastructure efficiency
- PUE: The power usage effectiveness

PUE is the original metric; DCiE was created to more easily understand the efficiency of the data center. The indicators are calculated using the following formulas:

```
DCiE = (IT equipment power / total facility power) x 100%
PUE = Total facility power / IT equipment power
```

IT equipment power includes the load that is associated with all of your IT equipment (such as servers, storage, and network equipment).

Total facility power includes IT equipment and everything that supports the IT equipment load such as PDU, CRAC, and UPS units.

A DCiE value of 33% (equivalent to a PUE of 3.0) suggests that the IT equipment consumes 33% of the power in the data center. Thus, for 100 dollars spent in energy, only 33 dollars are really used by IT equipment. Improvements in energy efficiency result in movement towards 100% DCiE, the ideal number.

A recent study initiated by IBM showed that the average DCiE is 44%. An excellent DCiE is more than 60%, thus many data centers can be optimized.

Figure 3-47 shows a scale to illustrate relative measurements of the DCiE and the PUE.

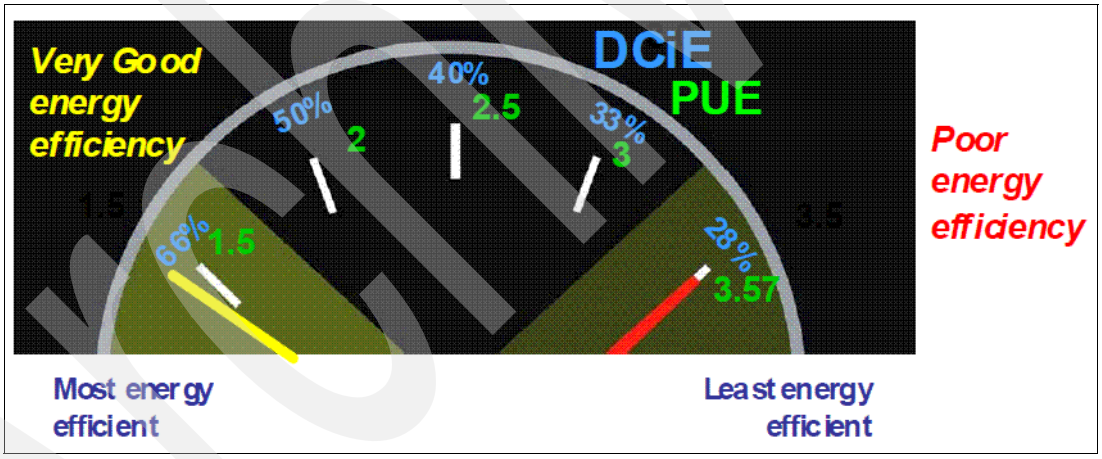


Figure 3-47 DCIE and PUE relative scale

An important feature of the Tivoli Monitoring for Energy Management Reporting and Optimization product is the recommendations reports. For example, the “Stand-by and Power-Down Recommendations for Server Pool” report displays recommendations for putting servers on standby or powering off.

When this report is generated, Reporting and Optimization evaluates the servers in a server pool during a time period and searches for servers with no or unvarying CPU load. Based on this date, it makes recommendations for placing certain servers in the pool in standby or power-off mode. Figure 3-48 on page 58 shows an example of this report. The report contains several recommendations for powering down certain servers during certain time windows.

Action	Target	Begin At	Duration	Repeat Interval	*Estimated Energy Savings(US$)	*Confidence	*Risk
PowerDown	A74E0000-F1C6-11DA-8D44-0004AC66008A	Fri Jun 12 01:00:00 IST 2009	4 Hour(s)	1 Day(s)	169.013 US$	75.00%	--
PowerDown	A74E0000-F1C6-11DA-8D44-0004AC66008A	Fri Jun 12 05:00:00 IST 2009	20 Hour(s)	1 Day(s)	845.064 US$	75.00%	--
PowerDown	A74E0000-F1C6-11DA-8D44-0004AC66008A	Tue Jun 16 05:00:00 IST 2009	1 Week(s)	1 Week(s)	997.315 US$	75.00%	--
PowerDown	A74E0000-F1C6-11DA-8D44-0004AC66008A	Fri Jun 26 05:00:00 IST 2009	2 Hour(s)	1 Month(s)	2.794 US$	75.00%	--
PowerDown	A74E0000-F1C6-11DA-8D44-0004AC66008A	Fri Jun 26 07:00:00 IST 2009	2 Hour(s)	1 Month(s)	2.794 US$	75.00%	--
PowerDown	A74E0000-F1C6-11DA-8D44-0004AC66008A	Fri Jun 26 09:00:00 IST 2009	2 Hour(s)	1 Month(s)	2.794 US$	75.00%	--
PowerDown	A74E0000-F1C6-11DA-8D44-0004AC66008A	Fri Jun 26 11:00:00 IST 2009	2 Hour(s)	1 Month(s)	2.794 US$	75.00%	--
PowerDown	A74E0000-F1C6-11DA-8D44-0004AC66008A	Fri Jun 26 13:00:00 IST 2009	2 Hour(s)	1 Month(s)	2.794 US$	75.00%	--
PowerDown	A74E0000-F1C6-11DA-8D44-0004AC66008A	Fri Jun 26 15:00:00 IST 2009	2 Hour(s)	1 Month(s)	2.794 US$	75.00%	--
PowerDown	A74E0000-F1C6-11DA-8D44-0004AC66008A	Fri Jun 26 17:00:00 IST 2009	2 Hour(s)	1 Month(s)	2.794 US$	75.00%	--
PowerDown	A74E0000-F1C6-	Fri Jun 26 19:00:00	2 Hour(s)	1 Month(s)	2.794 US$	75.00%	--

Figure 3-48 Reporting and Optimization Standby and Power-Down Recommendations for Server Pool report

A last example of a Reporting and Optimization report is shown in Figure 3-49 on page 59. It shows the “Data Center Power Usage Comparison” report, which you may use to compare two time periods. In the example, we compared the first half of October with the second half of October. As the report shows, during the second half of October the data center consumed far more energy compared to the first half of October.

Archived

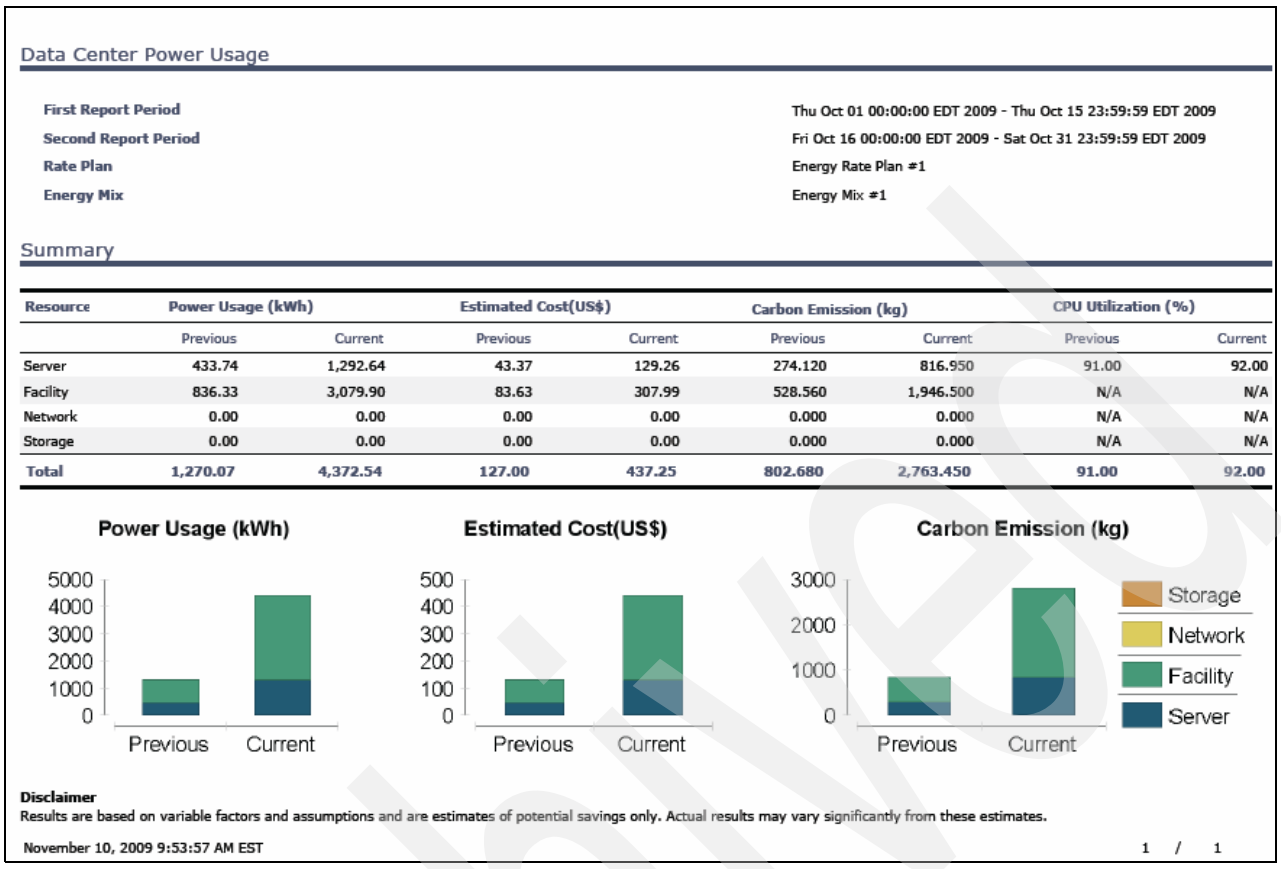


Figure 3-49 Reporting and Optimization Data Center Power Usage Comparison

3.7.4 Visualizing Tivoli Usage and Accounting Manager reports based on IBM Tivoli Monitoring for Energy Management

The reports provided by the Tivoli Data Warehouse collector for the Tivoli Monitoring for Energy Management use the following Tivoli Usage and Accounting Manager resource types:

- PWRAC
- PWRDC
- PWRAMTMP
- PWREXTMP
- PWRIN
- PWROUT

For accessing the ready-reports, access Tivoli Usage and Accounting Manager reporting server by using a URL similar to the following format, and click **Login**, as in Figure 3-50 on page 60:

`http://IP/tuam/`

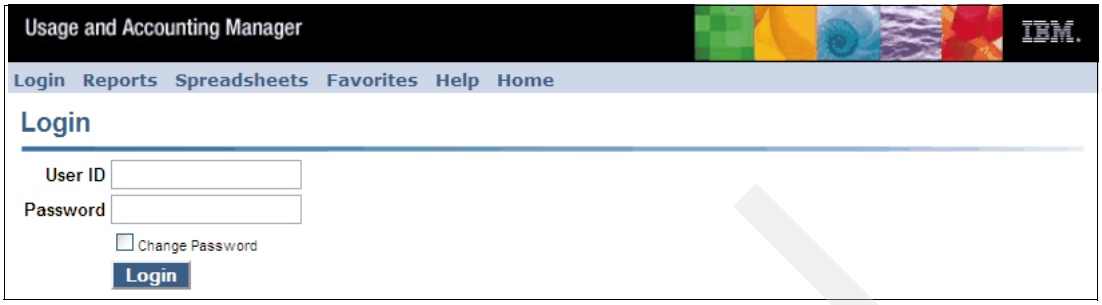


Figure 3-50 Login to Tivoli Usage and Accounting Manager Reporting Server

For access to the reports, click **Reports** → **Run Reports**. The available report types are listed (Figure 3-51). Use them to review invoices, trends and usage per account, and type of resource, for example.

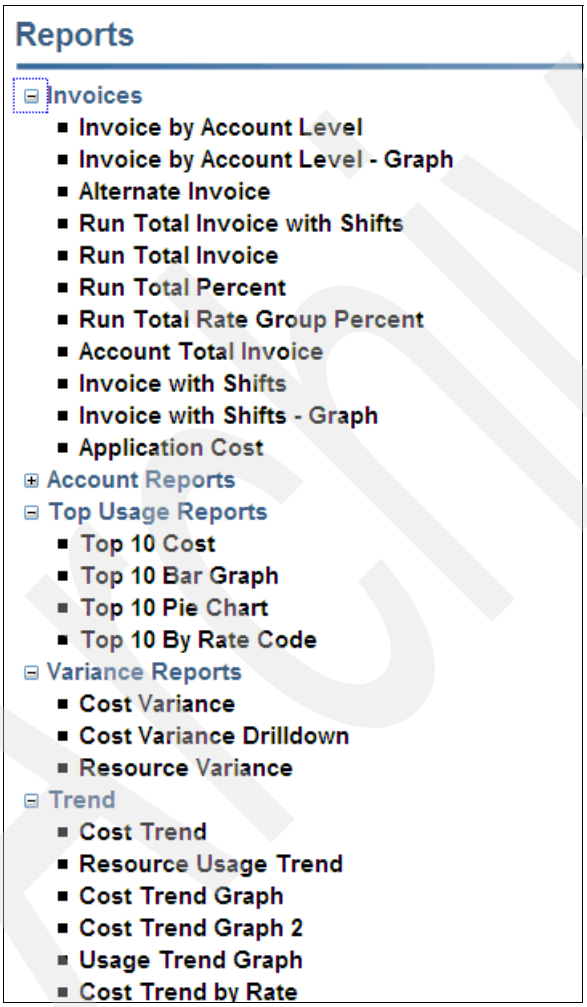
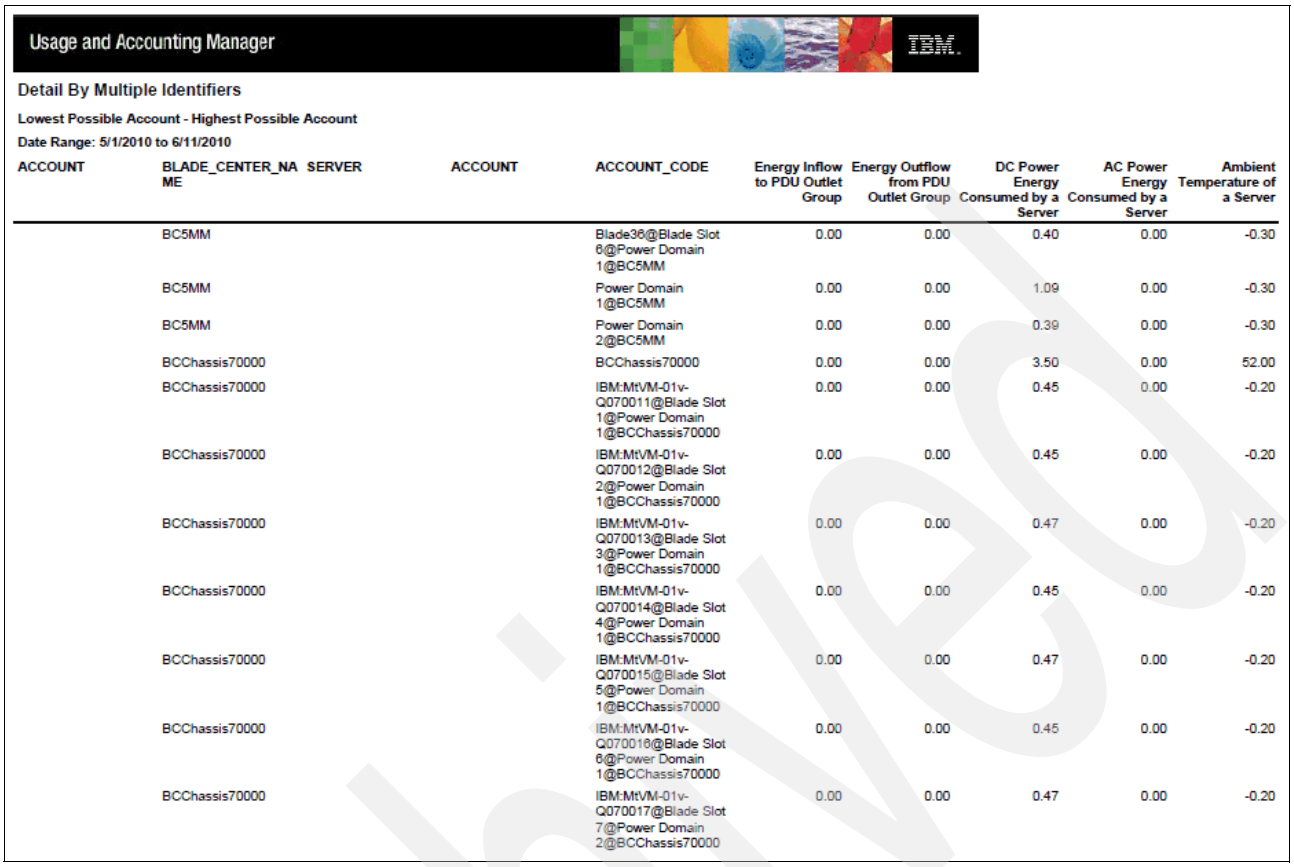


Figure 3-51 Tivoli Usage and Accounting Manager Report types

The sample report in Figure 3-52 on page 61 shows the energy consumed, and the temperature of each Account Code in a BladeCenter.

Archived



Usage and Accounting Manager

IBM.

Detail By Multiple Identifiers

Lowest Possible Account - Highest Possible Account

Date Range: 5/1/2010 to 6/11/2010

ACCOUNT	BLADE_CENTER_NAME	SERVER	ACCOUNT	ACCOUNT_CODE	Energy Inflow to PDU Outlet Group	Energy Outflow from PDU Outlet Group	DC Power Energy Consumed by a Server	AC Power Energy Consumed by a Server	Ambient Temperature of a Server
	BC5MM			Blade36@Blade Slot 6@Power Domain 1@BC5MM	0.00	0.00	0.40	0.00	-0.30
	BC5MM			Power Domain 1@BC5MM	0.00	0.00	1.09	0.00	-0.30
	BC5MM			Power Domain 2@BC5MM	0.00	0.00	0.39	0.00	-0.30
	BCChassis70000			BCChassis70000	0.00	0.00	3.50	0.00	52.00
	BCChassis70000			IBM:MtVM-01v-Q070011@Blade Slot 1@Power Domain 1@BCChassis70000	0.00	0.00	0.45	0.00	-0.20
	BCChassis70000			IBM:MtVM-01v-Q070012@Blade Slot 2@Power Domain 1@BCChassis70000	0.00	0.00	0.45	0.00	-0.20
	BCChassis70000			IBM:MtVM-01v-Q070013@Blade Slot 3@Power Domain 1@BCChassis70000	0.00	0.00	0.47	0.00	-0.20
	BCChassis70000			IBM:MtVM-01v-Q070014@Blade Slot 4@Power Domain 1@BCChassis70000	0.00	0.00	0.45	0.00	-0.20
	BCChassis70000			IBM:MtVM-01v-Q070015@Blade Slot 5@Power Domain 1@BCChassis70000	0.00	0.00	0.47	0.00	-0.20
	BCChassis70000			IBM:MtVM-01v-Q070016@Blade Slot 6@Power Domain 1@BCChassis70000	0.00	0.00	0.45	0.00	-0.20
	BCChassis70000			IBM:MtVM-01v-Q070017@Blade Slot 7@Power Domain 2@BCChassis70000	0.00	0.00	0.47	0.00	-0.20

Figure 3-52 Tivoli Usage and Accounting Manager report

Archived

3.8 Summary

Figure 3-53 shows the quick summary for this integration.

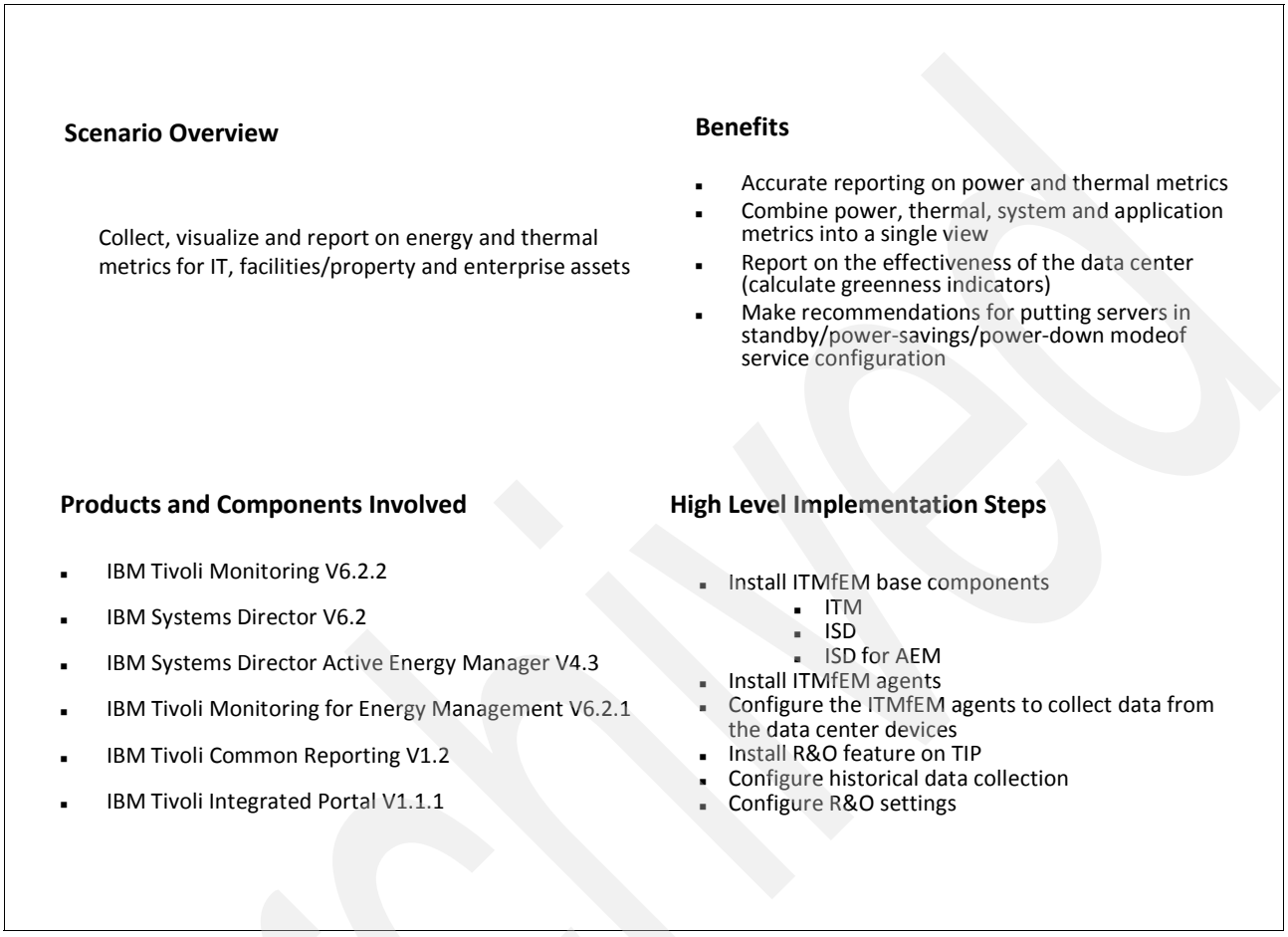


Figure 3-53 Quick summary

This scenario demonstrated how the various components of IBM Tivoli Monitoring for Energy Management can be used to accurately collect power and thermal data from a wide range of IT devices and facilities and property resources. We have shown how this data can be combined with system and application performance data to provide the administrators with an unprecedented integrated view of power, temperature and application performance data.

We have also demonstrated the capabilities of the Reporting and Optimization feature of Tivoli Monitoring for Energy Management. Reporting and Optimization reports are available to calculate indicators for greenness of a data center (PUE and DCIE), it can provide us with detailed information about the power consumption, estimated cost and carbon emission of our data center.

The Reporting and Optimization feature can also help us identify opportunities for saving energy and associated costs by determining which servers or server pools can be put on power savings mode, standby mode or completely powered off during certain time intervals.

Archived

4

Managing business service with energy and environment

This chapter describes a scenario where various Tivoli products are used to build an energy real-time dashboard and reports on environmental and energy uses.

This chapter contains the following topics:

- 4.1, “Scenario overview” on page 64
- 4.2, “Products involved” on page 65
- 4.3, “Benefits” on page 66
- 4.4, “Architectural diagram of integration” on page 68
- 4.5, “Implementation steps” on page 69
- 4.6, “Scenario walk-through” on page 90
- 4.7, “Summary” on page 101

Archived

© Copyright IBM Corp. 2011. All rights reserved.

4.1 Scenario overview

This scenario provides a unified dashboard that can help you make the right decisions in terms of saving costs to your company, put the focus on business value services, and manage them through a holistic view of all components, including energy and environmental information.

Data centers are not final objectives for companies but they are crucial for business services goals. Their management must be involved in the business management rules to reduce total costs and optimize their utilization, efficiency, and design.

Business service management (BSM) strategies had focused on establishing the relationship between business services status and performance and their IT components output. This vision was not complete, because energy and facilities status can cause a big impact on the business too. Now is the time to have a wider view that fits better with the reality:

- Other data center elements such as power distribution units (PDUs), computer room air conditioners (CRAC), uninterruptible power supplies (UPS), and heating, ventilating, and air conditioning (HVAC) units can affect business services availability and we are able to manage them.
- Energy costs for servers and cooling can seriously affect business services costs.[1] Measuring the energy consumed by each service, and analyzing its value for the business is important.
- Including new key performance indicators (KPIs) in the dashboards, such as temperature or data center infrastructure efficiency (DCiE), can help you make better decisions. For example, making the decision to start a new instance of an important business application on a separate server when the temperature is high on the current server.
- Determine and report on the costs of the services, based on the utilization and consumption of various resources (IT and non-IT components).

Figure 4-1 on page 65 shows various data that has been used for building the dashboard in this scenario:

[1] According to the Uptime Institute, the three-year cost of powering and cooling typical servers is currently one-and-a-half times the cost of purchasing server hardware. Kenneth G. Brill, “Data Center Energy Efficiency and Productivity”, The Uptime Institute, 2007

Archived

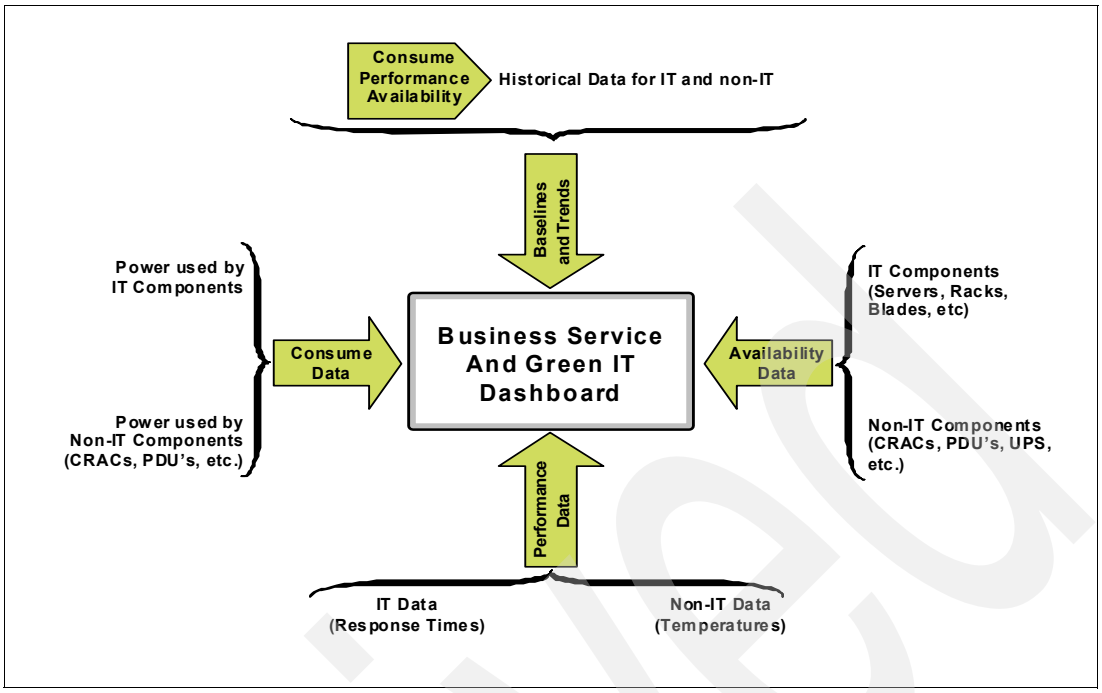


Figure 4-1 Information sources for BSM dashboard with energy and environment management

The use of business service management dashboards should be one of the first steps in any Green IT strategy, because they help to establish the relationship between data center capacity and utilization information, costs, and value for the business. Availability, performance, consumption data, and all the historical data are then key sources of information.

This scenario shows how to create your own BSM dashboard and how to integrate energy and environmental indicators on it to have a complete picture of your services.

4.2 Products involved

The products and product components involved in this scenario are as follows:

- Tivoli Business Service Manager V4.1.2 with FP0001
- IBM Tivoli Monitoring for Energy Management V6.2.1.01
- IBM Systems Director Active Energy Manager V4.3
- IBM Tivoli Application Dependency Discovery Manager V7.2
- IBM Tivoli Usage and Accounting Manager V7.1.2

In addition, the following product has been installed to support the products:

- IBM DB2 V9.7

Archive

See the following IBM websites for detailed lists of software and hardware requirements for the products:

- IBM Tivoli Business Service Manager:

  http://publib.boulder.ibm.com/infocenter/tivihelp/v3r1/index.jsp?topic=/com.ibm.tivoli.itbsm.doc/welcome.htm

- IBM Tivoli Monitoring for Energy Management:

  http://publib.boulder.ibm.com/infocenter/tivihelp/v15r1/index.jsp?topic=/com.ibm.itmem.doc_621if1/EM_welcome.htm

- IBM Systems Director

  http://www.ibm.com/systems/software/director/resources.html

- IBM Tivoli Application Dependency Discovery Manager:

  http://publib.boulder.ibm.com/infocenter/tivihelp/v10r1/index.jsp?topic=/com.ibm.taddm.doc_7.2/welcome_page/welcome.html

- IBM Tivoli Usage and Accounting Manager:

  http://publib.boulder.ibm.com/infocenter/tivihelp/v3r1/index.jsp?topic=/com.ibm.ituam.doc_7.1.2/welcome.htm

4.3 Benefits

Final objectives of green data centers are to save energy and cut costs so that the company can continue to grow.[2] Having these goals in mind, the scenario can provide the following benefits:

- For the IT operations director: Implementing a green data center helps to establish the relationship between business services, applications, IT devices, and power consume, that is critical to the following tasks:
  - Identify business services with high costs and little value for the business. These services could be suppressed or reduced to save costs to the company.
  - Obtain used versus available server capacity, and its trend, to allocate future demands.
  - Obtain the exact usage of the servers:
    - Are the servers being used for the services they were intended to?
    - Are any other applications or programs running on the server and that can impact the power consumption? According to U.S. Department of Energy, 1 watt of application computing requires more than 30 watts of power. This data is shown in Figure 4-2 on page 67.

[2] *The Green Data Center: Steps for the Journey*, REDP-4413

Archived

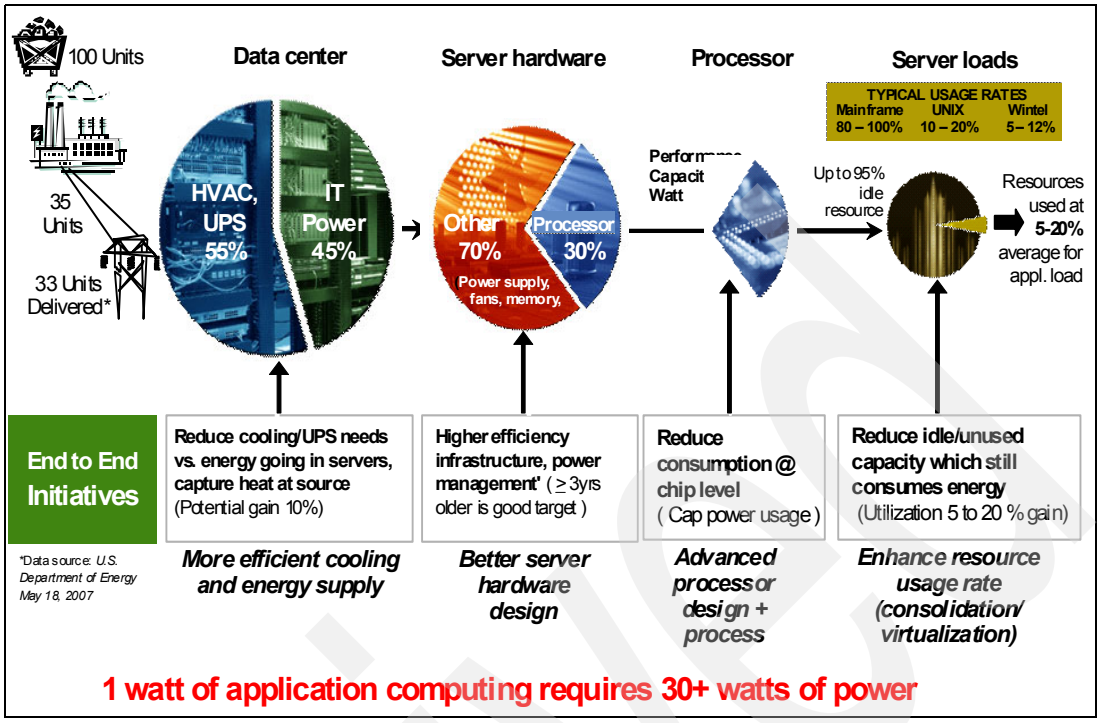


Figure 4-2 Where does all energy go?

- For the Chief information Officer (CIO): Using Tivoli for Green IT information, the CIO can make the following decisions:
  - Eliminate services that are not producing value to the business (they produce high costs and few benefits). This is an initial state of Green IT because it allows the following results:
    - Reduced number of servers, which means reduced initial and maintenance costs.
    - Reuse of existing servers for new services with higher value for the business, which means reduced acquisition, cooling, and space costs.

    Company acquisitions and mergers are also ideal scenarios for starting Green IT strategies. The redundancy of the services, servers, applications, locations makes the Green IT dashboard indispensable. This dashboard helps ease the decision of which redundancies are the most efficient for the business, and which ones can be removed.
  - Allocate energy costs to the consuming departments.
  - Reduce consumes while preserving service levels.
- For the systems operations manager: The Green IT dashboard offers the required information for making decisions about virtualization (that means another step in a Green IT strategy). Which services should be grouped in the same physical server?
  - Those that are critical for the business in the same server?
  - Those that have peak of use in different time periods?
- For IT operators: The Tivoli scenario shows the relationship between services, IT components and non-IT components. This information is crucial for saving costs in business services maintenance. A failure in a non-IT component, such as a UPS, can cause a failure in several servers. The IT operator must know that the root cause of the problem is not a failure on the servers, which translates into a reduced mean time to repair (MTTR). The IT operator can now open an incident to the facilities department.

- For the facilities manager: With the Tivoli portfolio for Green IT, the facilities department is now aligned with business. If a component fails manager knows which business services are being affected, and assigns the proper priority to the problem.
- For the data center manager: The information that is provided by Tivoli helps ease the decision of how to distribute servers in the data centers.

In summary, IT is uniquely positioned to offer the complete vision of the impact of infrastructure, energy, and environment in the business, as shown in Figure 4-3.

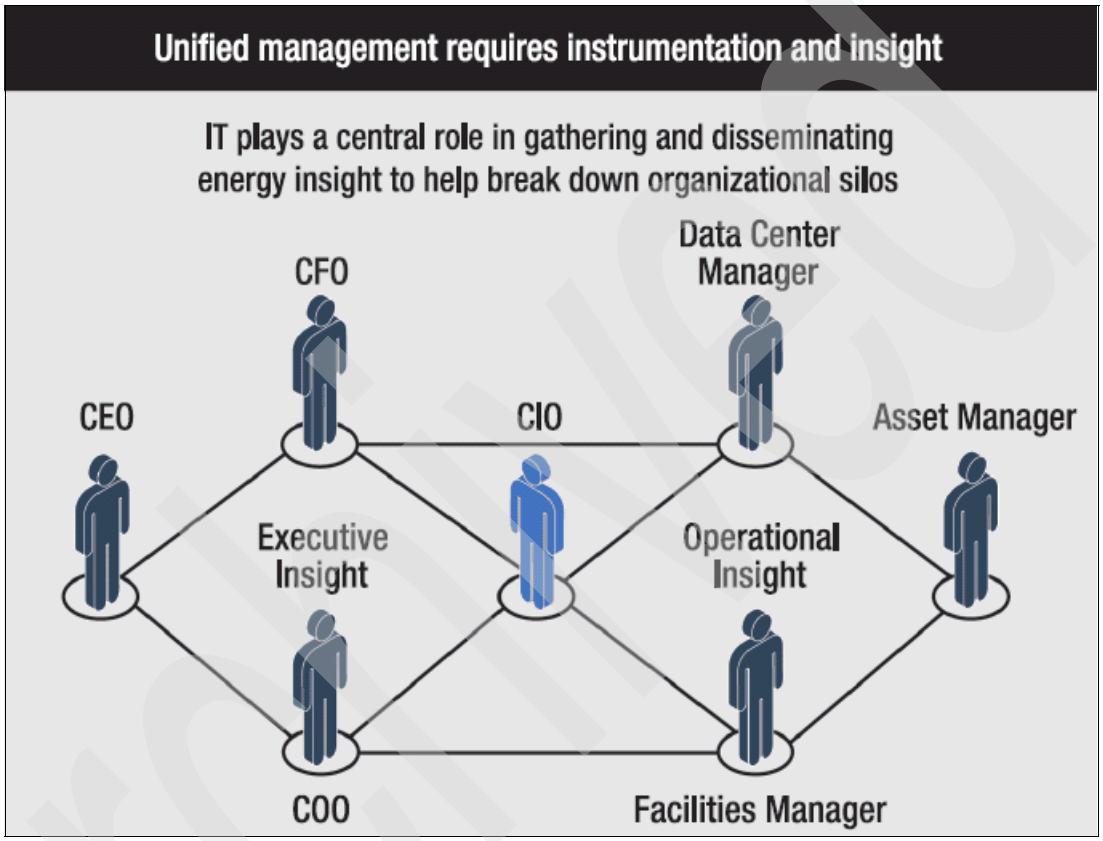


Figure 4-3 Central role of IT in the energy management for the business

4.4 Architectural diagram of integration

Figure 4-4 on page 69 shows the architectural diagram of this integration scenario. Tivoli Applications Dependency Discovery Manager (TADDM) automatically discovers the interdependencies between applications, middleware, servers, and network components; Tivoli Management System Discovery Library Adapter (TMS DLA) completes TADDM information with configuration data about the IBM Tivoli Monitoring environment including Active Energy Manager agent.

Tivoli Business Service Manager imports data discovered from TADDM and TMS DLA into the Service Component Repository and shows this data in a service tree in a ready-to-use dashboard named Energy Dashboard. Nevertheless, customers can create their own customized dashboards that include business services, IT and facilities components, and energy and environmental information.

Tivoli Business Service Manager calculates status and KPIs for the Energy Dashboard by using the information in Tivoli data warehouse of Tivoli Monitoring for Energy Management (ITMfEM).

Tivoli Usage and Accounting Manager (TUAM) has also been configured for using Tivoli Monitoring for Energy Management data related to energy, to allocate costs to applications, servers, and accounts and generate the appropriate reports. These reports are accessible from Tivoli Common Reporting (TCR) engine embedded on Tivoli Business Service Manager portal.

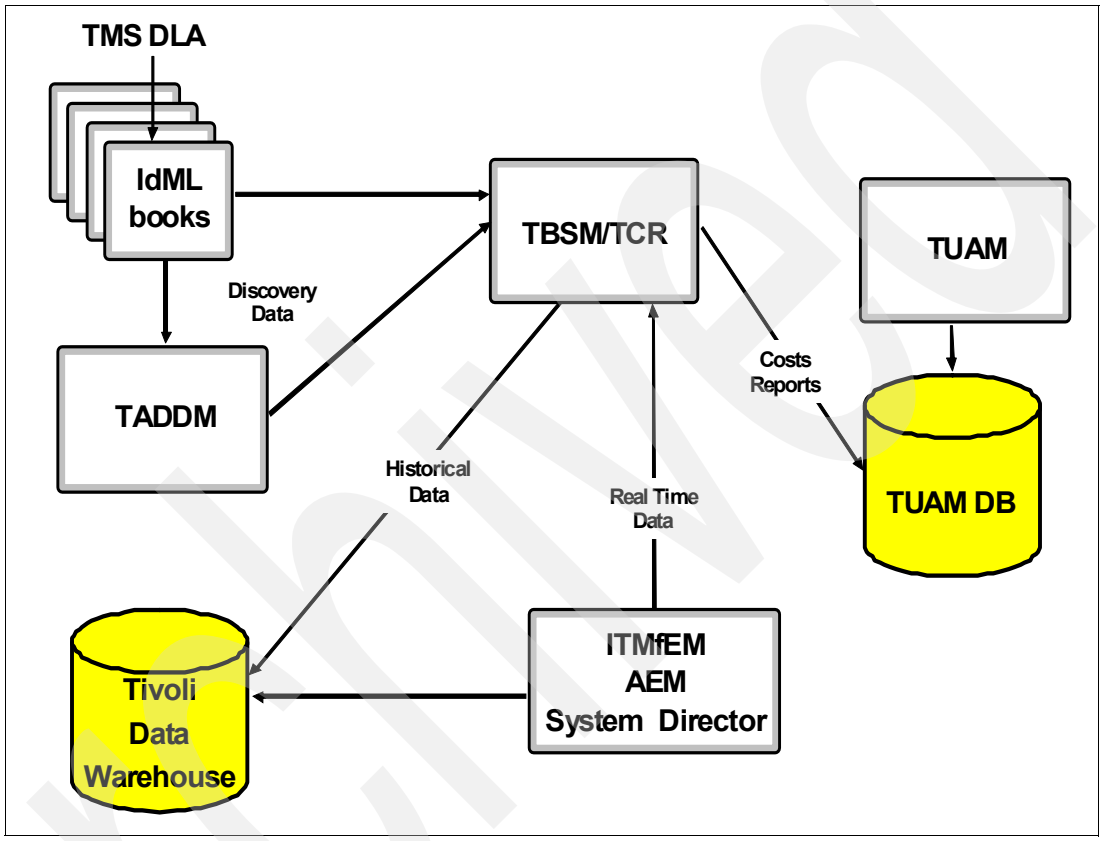


Figure 4-4 High level architecture for the scenario

4.5 Implementation steps

This section describes how to integrate the Tivoli products that are used in this scenario. It does not cover the complete installation of the products, only the installation steps that are relevant for the integration.

First we show you how to discover the resources and their relationships using TADDM and TMS DLA. Then we describe how to import all the discovered resources into Tivoli Business Service Manager Energy Dashboard and how Tivoli Business Service Manager uses Tivoli Monitoring for Energy Management information for showing power demand and temperatures. Finally, we show how to integrate Tivoli Usage and Accounting Manager reports into the reporting engine of Tivoli Business Service Manager.

Archived

4.5.1 Prerequisites

To have Tivoli Monitoring for Energy Management collecting energy and thermal data from the devices, be sure to first complete all the steps that are described in Chapter 3, “Collect, visualize, report on energy and thermal metrics for IT, facilities, property, enterprise assets” on page 19.

Note: If Tivoli Monitoring for Energy Management Reporting and Optimization is installed on the reporting engine of Tivoli Business Service Manager, the Tivoli Common Reporting installed path to provide during the installation is as follows (Figure 4-5):

```
/opt/IBM/tivoli/tip
```

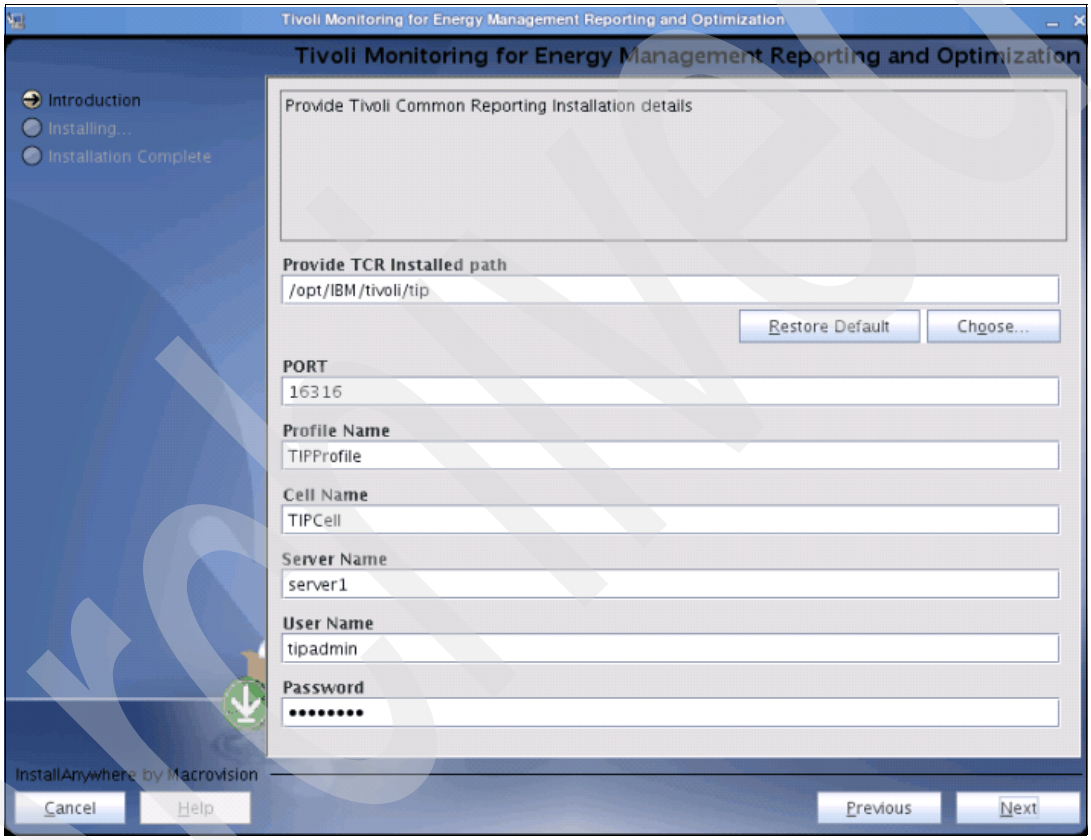


Figure 4-5 Path for Tivoli Common Reporting embedded into Tivoli Business Service Manager

Discovery Library Toolkit of Tivoli Business Service Manager is also required. The installation steps are defined in Chapter 5 of *Tivoli Integration Scenarios*, REDP-4401.

4.5.2 Discovering IBM Tivoli Energy Management devices in Tivoli Applications Dependency Discovery Manager and creating IdML book

Figure 4-6 on page 71 explains the integration process between IBM Tivoli Monitoring Energy Management data and Tivoli Applications Dependency Discovery Manager (TADDM).

Tivoli Monitoring for Energy Management currently uses the Discovery Library Adapter (DLA) technology to get its managed resources into TADDM and other consumers such as Tivoli Business Service Manager.

IBM Tivoli Monitoring delivers this DLA as part of the base installation; it is called TMS DLA. The DLA uses the DLA API to extract the discovered resources and produce formatted InDesign Markup Langauge (IdML) books. For more information about IdML data, see the Data integration chapter in *Integrating Tivoli Products*, SG24-7757.

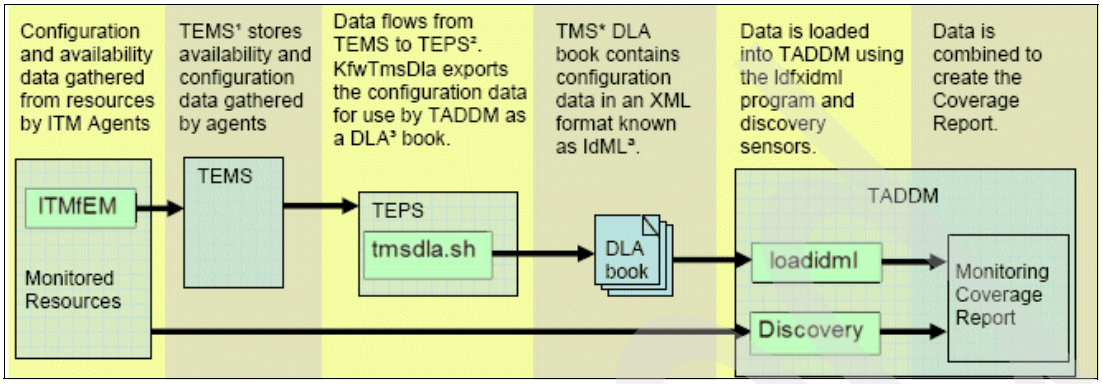


Figure 4-6 Data integration process between Tivoli Monitoring Services and TADDM

The required steps for this integration are as follows:

1. Discover all the IT resources in TADDM, as described in Chapter 4 of *IBM Tivoli Application Dependency Discovery Manager Capabilities and Best Practices*, SG24-7519.
2. Generate a TMS IdML book at the Tivoli Enterprise Portal Server installation:

   ```
   Install_dir/bin/itmcmd execute cq "tmsdla.sh"
   ```

   Note: To make a refresh-type IdML book use the following command:

   ```
   Install_dir/bin/itmcmd execute cq "tmsdla.sh -r"
   ```

   The book is generated in the following directory with `.xml` extension:

   ```
   Install_dir/li6263/cq/bin/tmsdla/
   ```

   Currently TMS DLA includes the discovery of the following types of devices for energy management:

   - Blades
   - BladeCenter
   - Rack servers
   - Certain power distribution units (PDUs)
   - Certain types of sensors

   Nevertheless, IBM expects to complete TMS DLA for including the discovery of several types of devices (among others):

   - Johnson Controls devices through Johnson Controls Metasys
   - Eaton Power Xpert devices, meters, UPS units and PDUs

   These plans can include the creation of new classes in the Common Data Model and the support of these new types of devices into TADDM UI.

   Note: The information about plans for new developments is intended to outline IBM general DLA direction; do not rely on it for making a purchasing decision. The information about new developments is not a commitment, promise, or legal obligation to deliver any material, code or functionality. The development, release, and timing of any features or functionality described remains at IBM discretion.

Archive

3. Copy IdML file in any directory of the TADDM server:
   - Grant access to the CMDB user to the new directory and `.xml` file.
   - Execute **`chmod 600`** on the new `.xml` file.
4. Load IdML book into TADDM by using the following command:

   `<TADDM_install_dir>/bin/loadidml.sh -g -f <path to the xml files> -h localhost`

   Figure 4-7 shows the relationship between a BladeCenter management module, the managed blade centers, and all its blades:

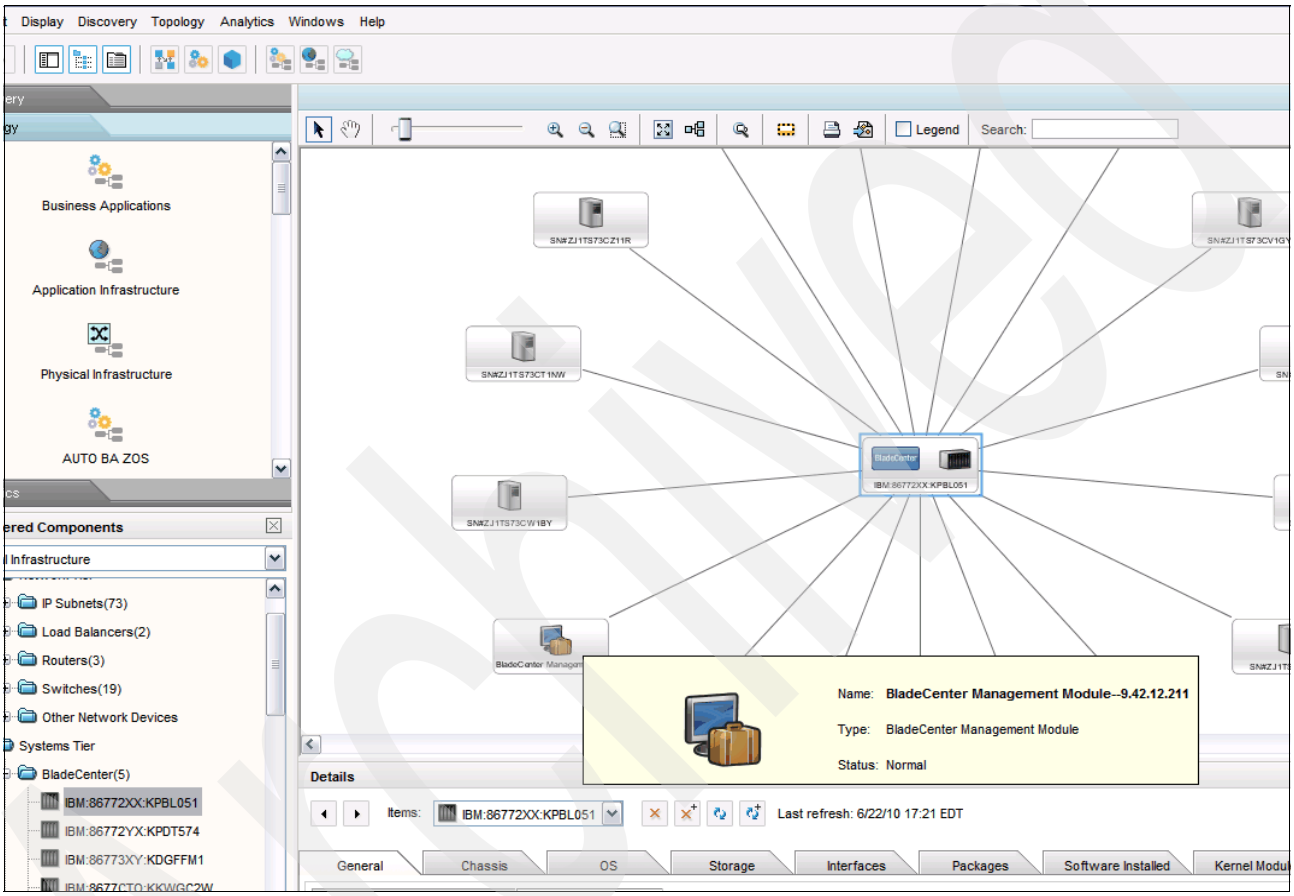


Figure 4-7 Topology and details for IT resources in TADDM

Note: The TMS DLA currently discovers Energy Management resources as TMS Agents. That is why only limited resource attributes are provided and no complex relationship information is available to TADDM. Without relationship information, the establishment of relationships between discovered Energy Management resources and other service instances cannot be performed programmatically during the importing of the TMS DLA book, into TADDM. This step needs to be administered by the customer. Currently, a more complete discovery process for Energy Management resources is being considered for future development.

4.5.3 Configuring Energy Dashboard at Tivoli Business Manager

The required steps to configure the Energy Dashboard are as follows:

1. Enable the discovery of the energy resources in the Discovery Library Toolkit:
   a. Open the following file:

      `$TBSM_HOME/XMLtoolkit/xml/CDM_TO_TBSM4x_MAP.xml`

   b. Remove the lines that contain the following string:

      "*Energy Management: REMOVE*"

   c. Restart Discovery Library Toolkit:

      ```
      $TBSM_HOME/XMLtoolkit/bin/tbsmrdr_start.sh
      $TBSM_HOME/XMLtoolkit/bin/tbsmrdr_stop.sh
      ```

2. Create IBM Tivoli Monitoring for Energy Management data source, that is, the Tivoli Data Warehouse where all the metrics are being stored, as follows:
   a. Log in to Tivoli Business Service Manager as an administrator user.
   b. In the task list, select **Administration** → **Service Configuration**.
   c. Select **DataSource** and create a new one that points to the Tivoli Data Warehouse database, The data source created should be similar to the one shown in Figure 4-8. Use **Test Connection** option for verifying the connection.

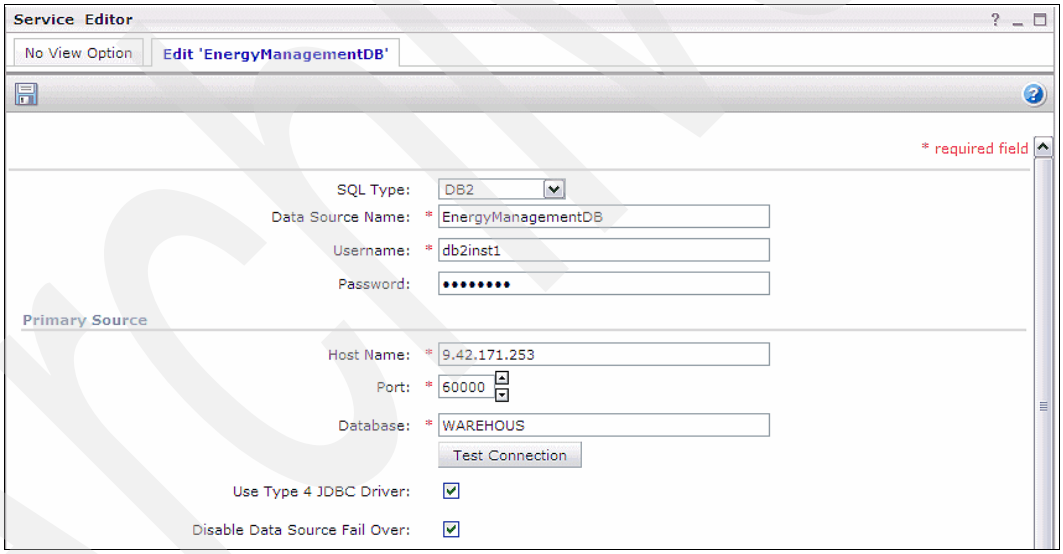


Figure 4-8 Data source connection

   d. The following collection of attribute groups in Tivoli Data Warehouse must be started before you perform the next step:
      - KE9 ALL BLADE CENTERS POWER® DATA
      - KE9 ALL CRAC DATA
      - KE9 ALL MANAGED OBJECTS POWER DATA
      - KE9 ALL PDUS POWER DATA
      - KE9 ALL PERFORMANCE OBJECT STATUS
      - KE9 ALL RACK SERVERS POWER DATA
      - KE9 ALL RACKS POWER DATA

Archived

- KE9 BC PERFORMANCE OBJECT STATUS
- KE9 BLADECENTER POWER DATA
- KE9 CRAC DATA
- KE9 PDU PERFORMANCE OBJECT STATUS
- KE9 PDU POWER DATA
- KE9 PERFORNACE OBJECT STATUS
- KE9 RACK POWER DATA
- KE9 RACK SERVER POWER DATA
- KE9 RK PERFORMANCE OBJECT STATUS
- KE9 RS PERFORMANCE OBJECT STATUS
- KE9 SYSTEM Z10 EC POWER DATA
- KE9 UPS POWER DATA

Note: If you are not monitoring, for example, uninterruptible power supplies (UPS), the table will not exist in the Tivoli Data Warehouse. Therefore, queries for the following data fetchers fail:

- EM_UPS_AEM_getPowerDemand
- EM_UPS_AEM_getTemperature

In this case, the unnecessary data fetchers shown in Figure 4-11 on page 76 must be disabled.

e. Import the energy management service data model into Tivoli Business Service Manager:

```
cat $TBSM_HOME/install/BSM_Templates.radsh | $TBSM_HOME/bin/rad_radshell

cat $TBSM_HOME/contrib/EnergyManagement/BSM_TemplatesEnergyManagement.radsh
| $TBSM_HOME/bin/rad_radshell
```

Archived

This step creates the following elements in Tivoli Business Service Manager:

- Templates (Figure 4-9)

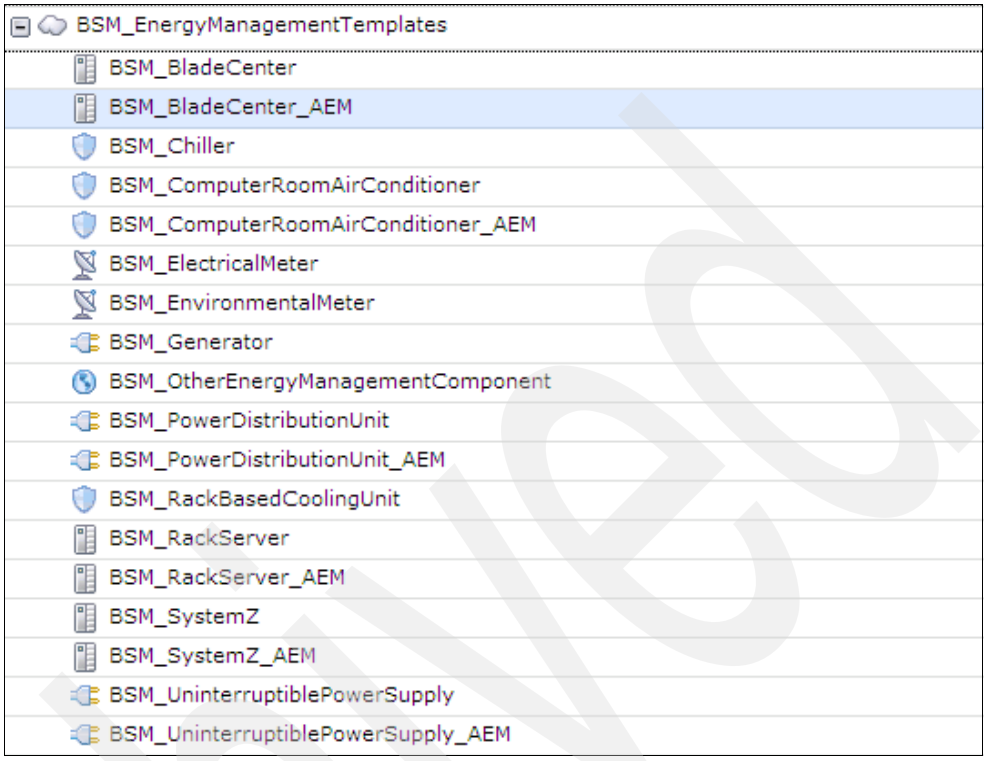


Figure 4-9 Energy templates in Tivoli Business Service Manager service model

- Parent services instances (Figure 4-10 on page 76) for Energy service tree

Archived

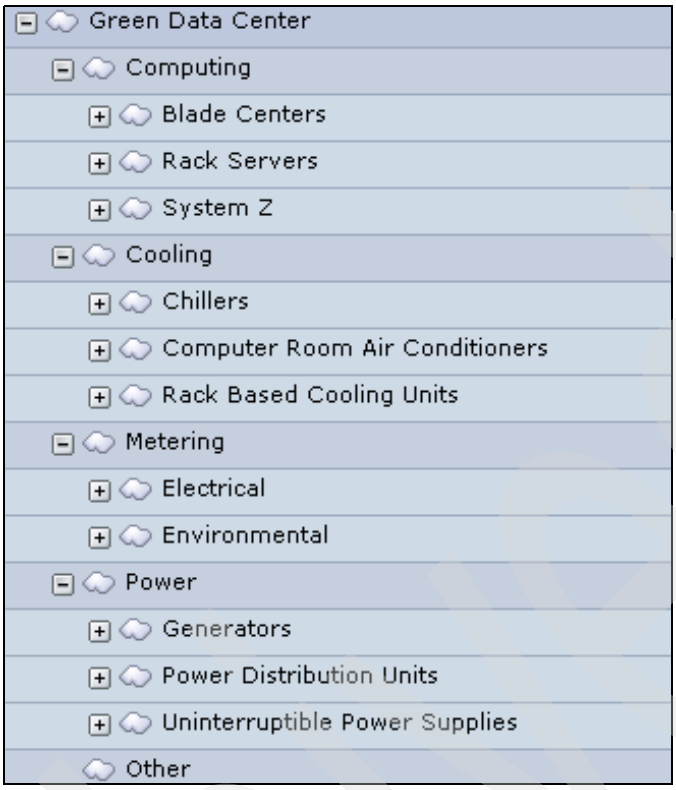


Figure 4-10 Basic service tree for Energy Dashboard

- Data Fetchers (Figure 4-11)

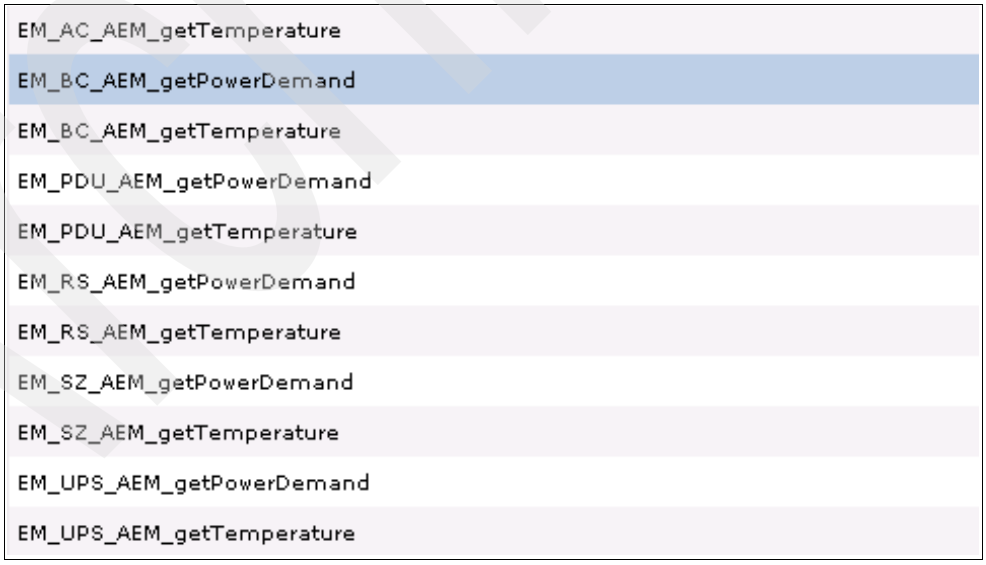


Figure 4-11 Data fetchers for Energy Dashboard

f. Configure the service tree template for Energy Management dashboard:

i. In TBMS Data Server, edit the `treeTemplates.xml` file, which is in the following location:

`<PRODUCT_INSTALLATION_PATH>/profiles/TBSMProfile/installedApps/TBSMCell/tbsm.ear/sla.war/av/xmlconfig`

ii. Add the contents of the `treeTemplates.xml.update` file in the `$TBSM_HOME/contrib/EnergyManagement` directory to the `treeTemplates.xml` file.

Insert the contents of the `treeTemplates.xml.update` file between the last `</treeTemplate>` tag and the final `</canvasConfig>` tag.

More information about this step is at the following address:

http://publib.boulder.ibm.com/infocenter/tivihelp/v3r1/index.jsp?topic=/com.ibm.tivoli.itbsm.doc/customization/bsmc_engc_install.html

iii. Restart Tivoli Business Service Manager data and dashboard servers.

The service tree template shown in Figure 4-12, called *ServiceEnergyManagement* must appear in Tivoli Business Service Manager.

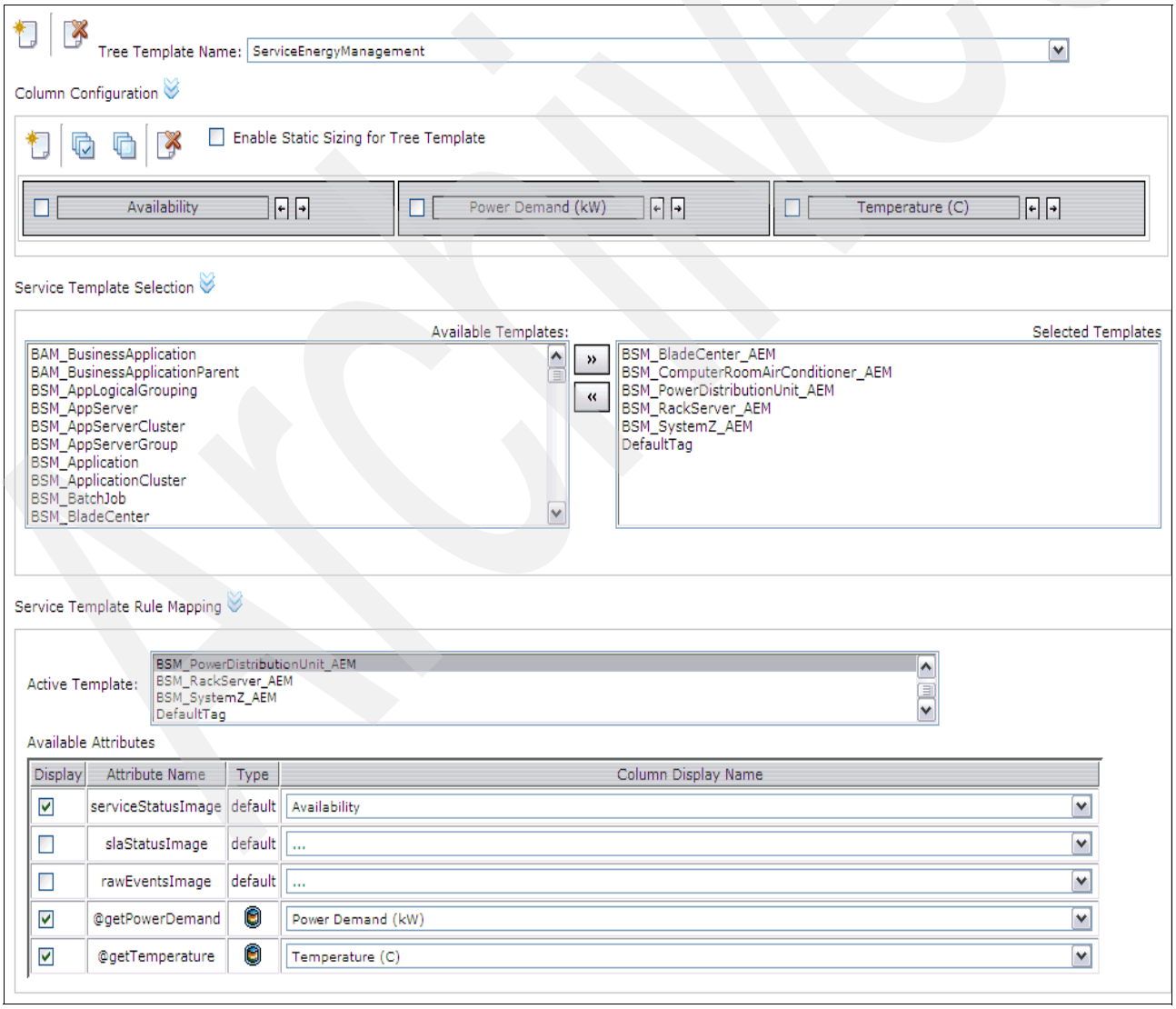


Figure 4-12 Service tree template for Energy Dashboard

g. Add the scorecard column policy by using the steps at the following web address:

   http://publib.boulder.ibm.com/infocenter/tivihelp/v3r1/index.jsp?topic=/com.ibm.tivoli.itbsm.doc/customization/bsmc_engc_install.html

h. Create the Energy Dashboard page:

   i. Log in to Tivoli Integrated Portal as a user with administrator roles.

   ii. In the task list, select **Settings** → **Page Management**, and click **New Page**.

   iii. Split the page in five sections, three vertical and two horizontal sections as it is described at the following web address:

   http://publib.boulder.ibm.com/infocenter/tivihelp/v3r1/index.jsp?topic=/com.ibm.tivoli.itbsm.doc/customization/bsmc_engc_install.html

   The final page must have a similar structure to the one shown in Figure 4-13.

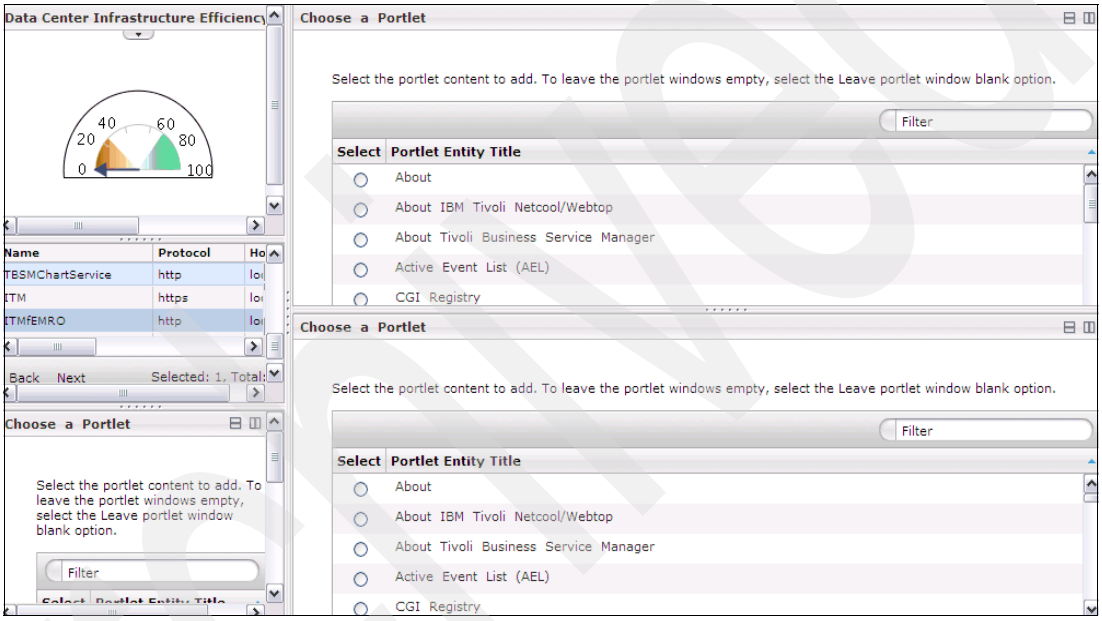


Figure 4-13 Energy dashboard page structure

   iv. For all the vertical sections, select a Charting portlet, as shown in Figure 4-14, and click **OK**.

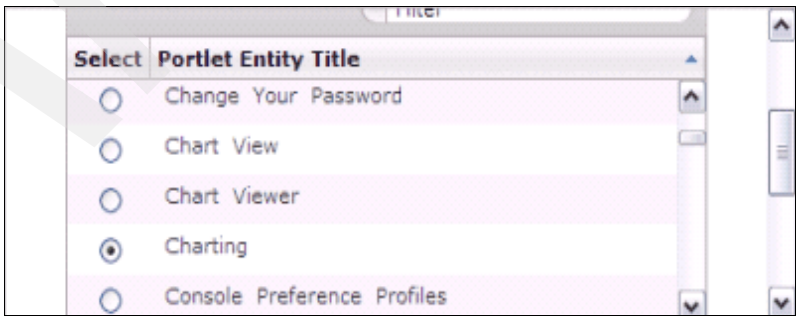


Figure 4-14 Charting portlet selection

v. Select **Tivoli Charts**, as shown in Figure 4-15.

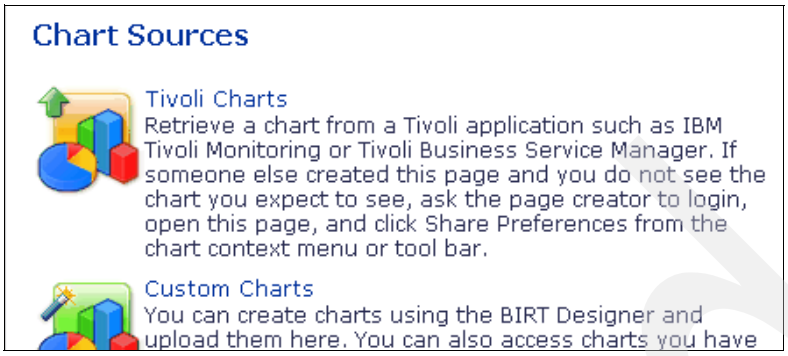


Figure 4-15 Tivoli Charts selection

vi. Select the **Reporting and Optimization** web connection (ITMfEMRO) and click **Next**, as in Figure 4-16.

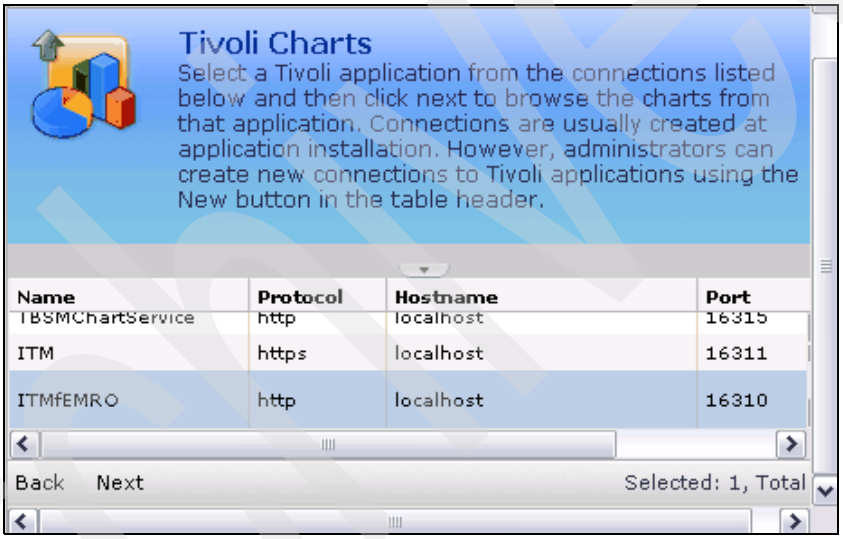


Figure 4-16 Tivoli Monitoring for energy management web service

Archived

vii. Select a group from the drop-down list for choosing the time range for the chart, as in Figure 4-17.

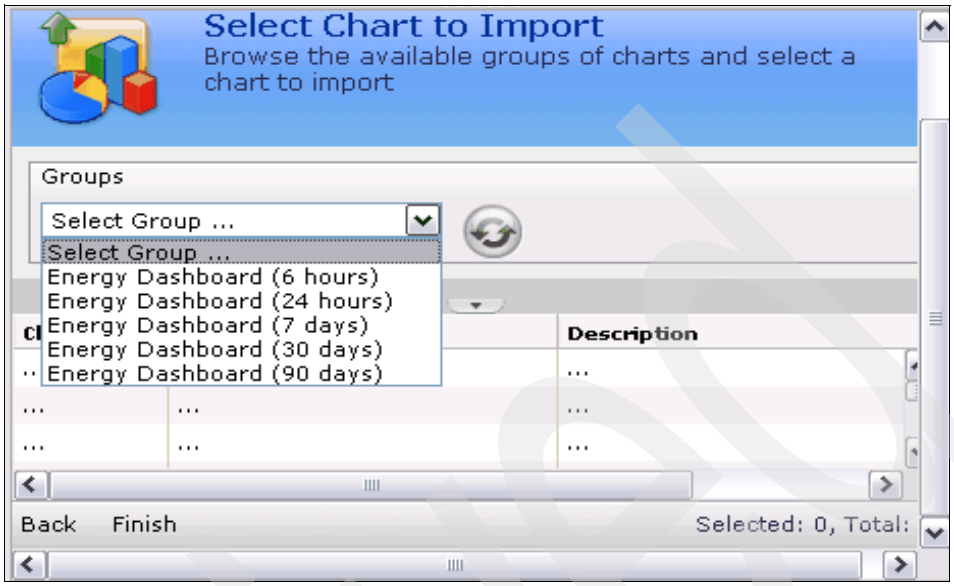


Figure 4-17 Group selection of Tivoli Monitoring for Energy Management attributes

viii. Select the chart type to include in each of the vertical sections, as in Figure 4-18:

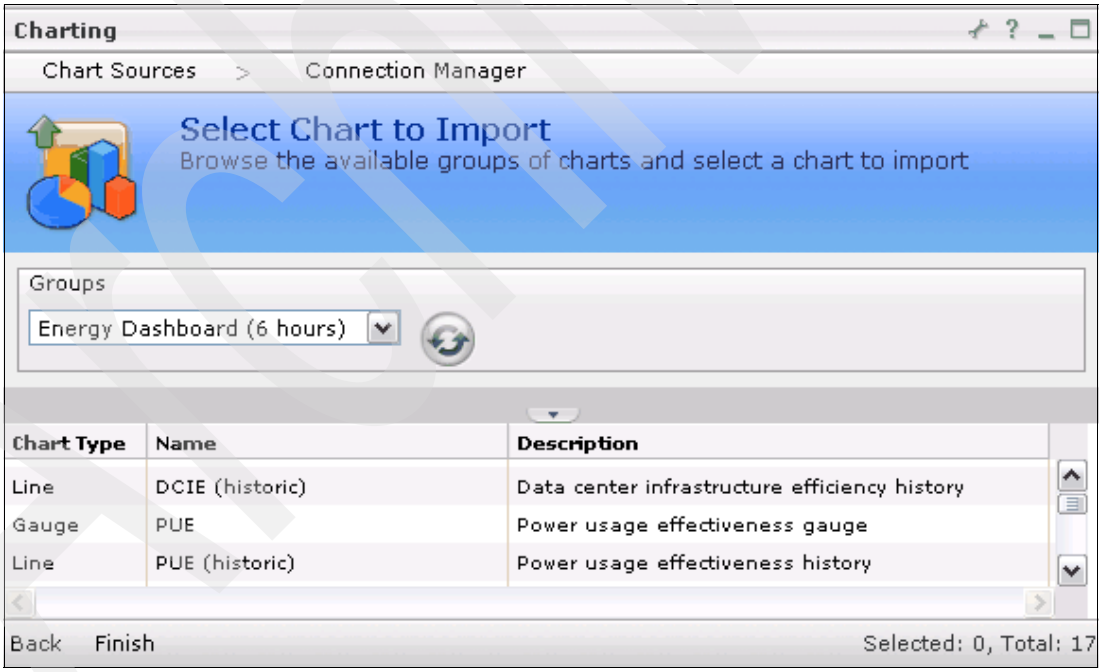


Figure 4-18 Chart type selection

ix. For the upper horizontal section, select **Service Tree** portlet. For the last one, select **Service Details** portlet.

The result is similar to Figure 4-19 on page 81.

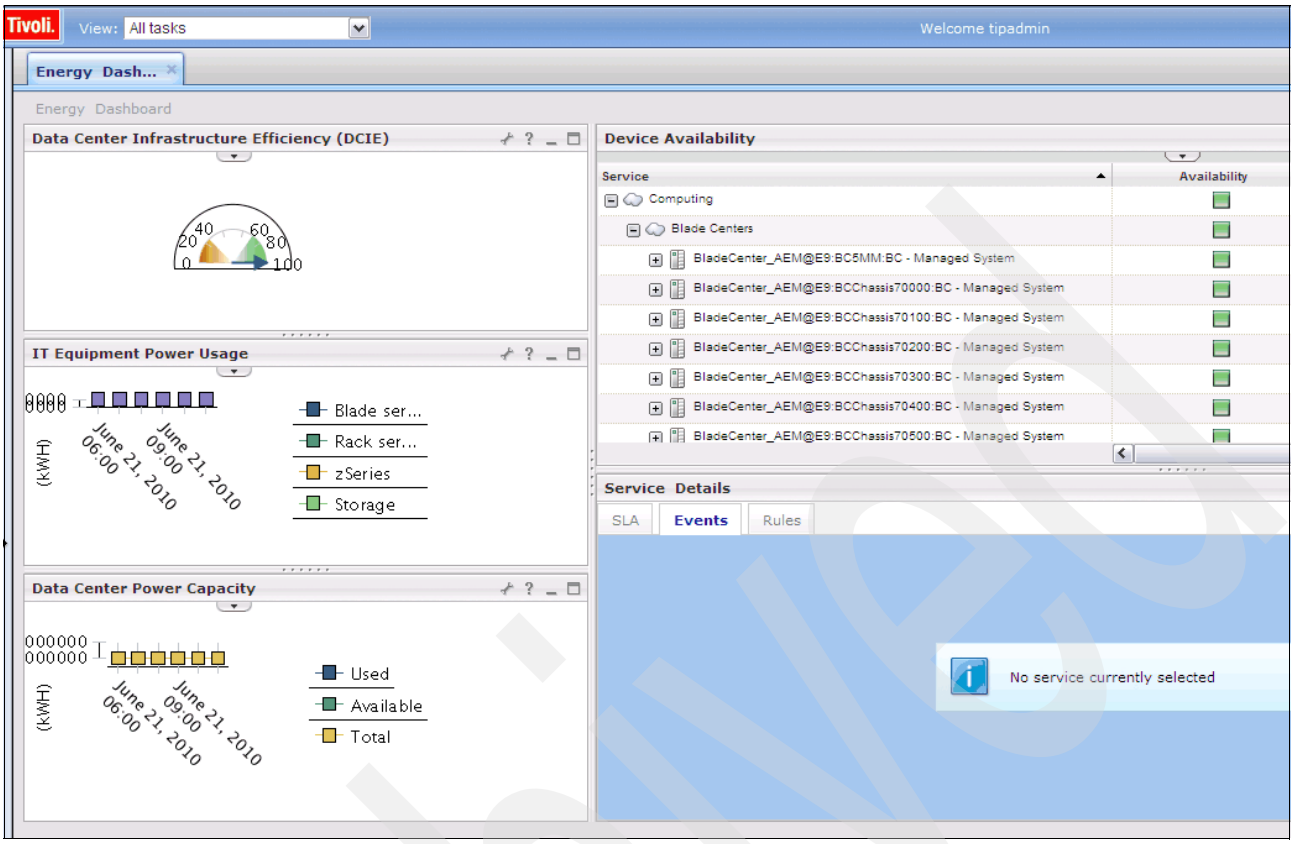


Figure 4-19 Energy management dashboard

4.5.4 Importing discovered resources into Tivoli Business Service Manager

The resources discovered in 4.5.2, “Discovering IBM Tivoli Energy Management devices in Tivoli Applications Dependency Discovery Manager and creating IdML book” on page 70 must be loaded into Tivoli Business Service Manager.

Note: The Discovery Library Toolkit must be installed in Tivoli Business Service Manager and configured to work with TADDM and IdML books: The DL_ImportSource property in the following file must be set to `all`:

`/opt/IBM/tivoli/tbsm/XMLtoolkit/bin/xmltoolkitsvc.properties`

The steps required for installing Discovery Library Toolkit for Tivoli Business Service Manager are described in *Integrating Tivoli Products*, SG24-7757.

Perform the following steps for the success of the integration:

1. For TADDM integration, perform the following steps:
   a. Copy the following files from TADDM server:

      TADDM_install_dir`/sdk/lib/taddm-api-client.jar`
      TADDM_install_dir`/sdk/lib/platform-model.jar`

   b. Paste the copied files into the specified folder of Tivoli Business Service Manager server:

      `$TBSM_HOME/XMLtoolkit/sdk/clientlib`

c. Open the following properties file:

   TADDM_install_dir/etc/collation.properties

   In this file replace the default TADDM Server host name with the actual one in the following line:

   com.collation.clientproxy.rmi.server.hostname=<*IP address of TADDM Server*>

2. For IdML integration, copy the IdML book (obtained in4.5.2, “Discovering IBM Tivoli Energy Management devices in Tivoli Applications Dependency Discovery Manager and creating IdML book” on page 70) into the following directory:

   /opt/IBM/tivoli/tbsm/discovery/dlbooks

3. Start the discovery library toolkit:

   /opt/IBM/tivoli/tbsm/XMLtoolkit/bin/tbsmrdr_start.sh

4. View the log file (/opt/IBM/tivoli/tbsm/XMLtoolkit/log/msgGTM_XT.log) to be sure it contains information similar to the following lines:

   ```
   Book TMSDISC620-A.TI0003-sys9.itso.ral.ibm.com.2010-06-18T21.37.52Z.refresh.xml
   processed successfully

   L5306I: CMDB import completed successfully
   ```

   These lines mean that IdML book was successfully imported, and that TADDM information was successfully imported.

The TMS DLA book, obtained from IBM Tivoli Monitoring for Energy Management contains the following fields:

- managedSystemName: The Tivoli Monitoring Managed System name for the agent
- product code: The two-letter Tivoli Monitoring product code for the agent

  For Tivoli Monitoring for Energy Management, the code is E9.
- affinity: The affinity code corresponding to the productCode

  Several of the following affinities are included in IBM Tivoli Monitoring for Energy Management:
  - %IBM.AEM_BLCEN
  - %IBM.AEM_RKSRV
  - %IBM.AEM_ALL
  - %IBM.AEM_PDU
  - %IBM.AEM_SNS
- softwareVersion: The version of the agent bundle
- Label: Label of the agent

Archived

Tivoli Business Service Manager uses the criteria shown in Table 4-1 for assigning service templates to the IdML discovered resources.

Table 4-1 Common data model classes to Tivoli Business Service Manager templates

ManagedSystemName	Label	Class	Template
like “E9%:BC”	BladeCenter	tbsm:BladeCenter	BSM_BladeCenter_AEM
like “E9%:RS”	RackServer	tbsm:RackServer	BSM_RackServer_AEM
like “E9%:SZ”	SystemZ	tbsm:SystemZ	BSM_SystemZ_AEM
like “%:CHR	Chiller	tbsm:Chiller	BSM_Chiller
like “E9%:AC	CRAC	tbsm:CRAC	BSM_ComputerRoomAir Conditioner_AEM
like “%:AIR	CRAC	tbsm:CRAC	BSM_ComputerRoomAir Conditioner
like “%:ARU	AirRemovalUnit	tbsm:RackBasedC oolingUnit	BSM_RackBasedCooling Unit
like “%:SNS	Sensor	tbsm:Sensor	BSM_EnvironmentalMete r
like “%:IEM	Integrated Event Monitor	tbsm:Sensor	BSM_EnvironmentalMete r
like “%:EEM	External Event Monitor	tbsm:Sensor	BSM_EnvironmentalMete r
like “%:GEN	Generator	tbsm:Generator	BSM_Generator
like “%:PDU	PDU	tbsm:PDU	BSM_PowerDistributionU nit_AEM
like “%:rPD	RackPDU	tbsm:PDU	BSM_PowerDistributionU nit
like “%:xPD	ISxPDU	tbsm:PDU	BSM_PowerDistributionU nit
like “%:UPS	UPS	tbsm:UPS	BSM_UninterruptiblePow erSupply_AEM
like “%:OTH	EMComponent	tbsm:OtherEnergy MgmtComponent	BSM_OtherEnergyManag ementComponent

The following address has more information about how to assign service templates to TADDM resources:

http://www.ibm.com/developerworks/wikis/display/tivolibsm/Tivoli+Application+Dependency+Discovery+Manager+and+Tivoli+Business+Service+Manager+Integration+Solution+Guide

Archive

4.5.5 Integrating Tivoli Usage and Accounting Manager BIRT Reports into Tivoli Common Reporting engine of Tivoli Business Manager

Use the following steps to import IBM Tivoli Usage and Accounting Reports, based on BIRT, into Tivoli Common Reporting engine:

1. Install BIRT Designer v2.2.1. Download it from either of the following locations:

   From OPAL:

   http://www.ibm.com/software/brandcatalog/ismlibrary/details?catalog.label=1TW10OT02

   Directly from BIRT site:

   http://archive.eclipse.org/birt/downloads/build.php?build=R-R1-2_2_1-200710010630

2. Create a new Report project, as shown in Figure 4-20.

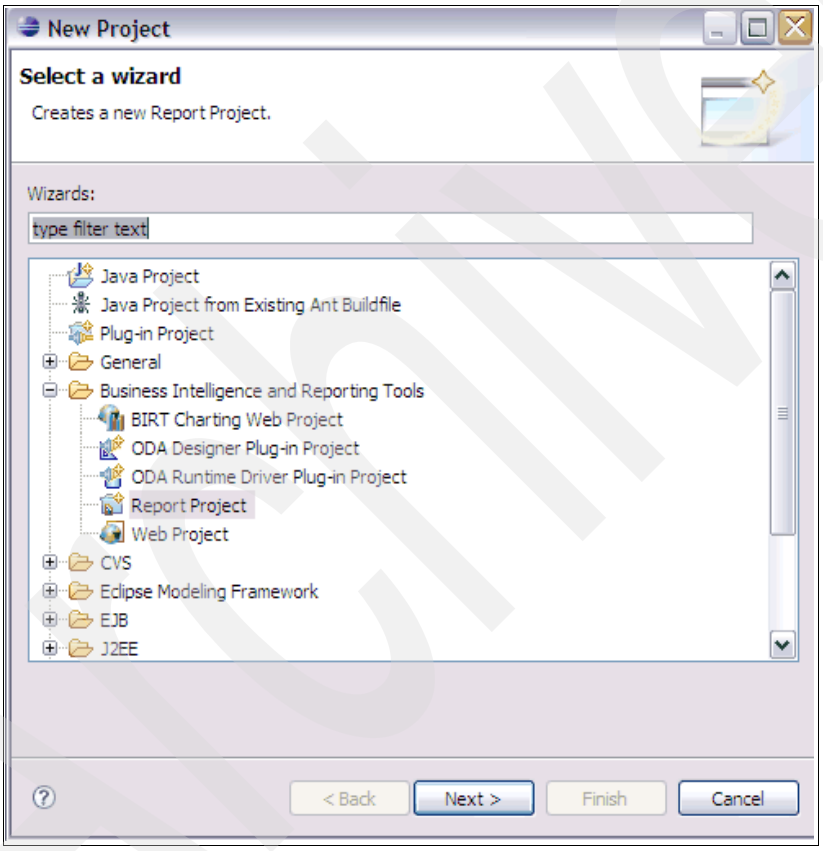


Figure 4-20 BIRT Report project

3. Copy the resources and reports from Tivoli Usage and Accounting Manager Server into a folder in the BIRT workstation, and create a `.zip` file from them:

   Resources are in the following location:

   `/opt/ibm/tuam/server/reportsbirt/resources`

   Reports are in the following location:

   `/opt/ibm/tuam/server/reportsbirt/db2/standard`

4. Import resources and reports into the report project of BIRT, as described in Figure 4-21 and Figure 4-22 on page 86, by selecting the `.zip` file.

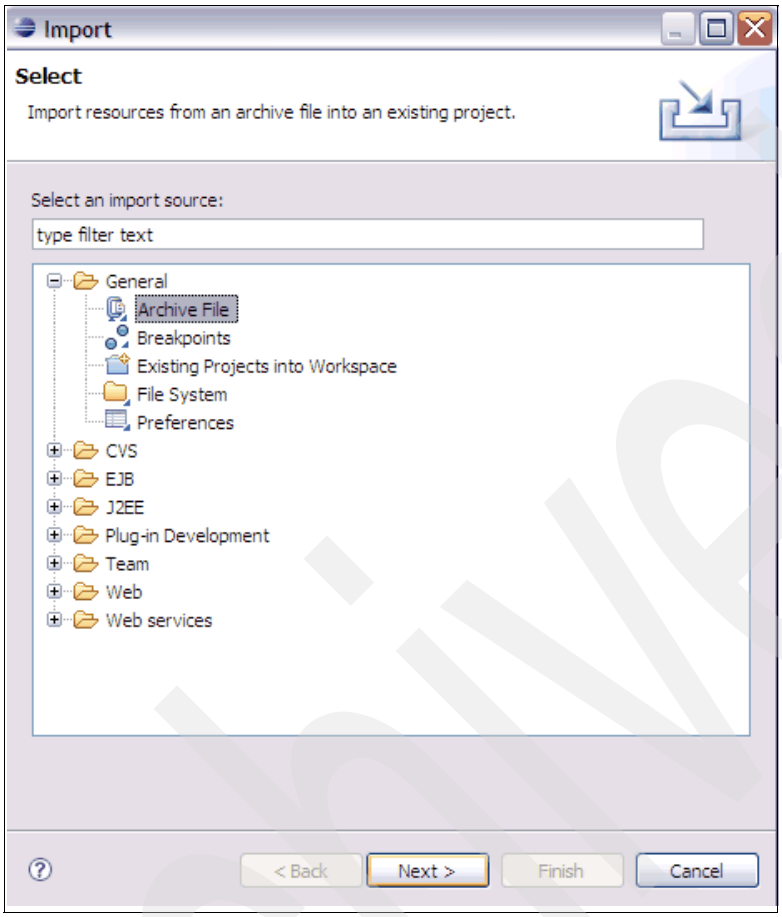


Figure 4-21 Importing from file

Archive

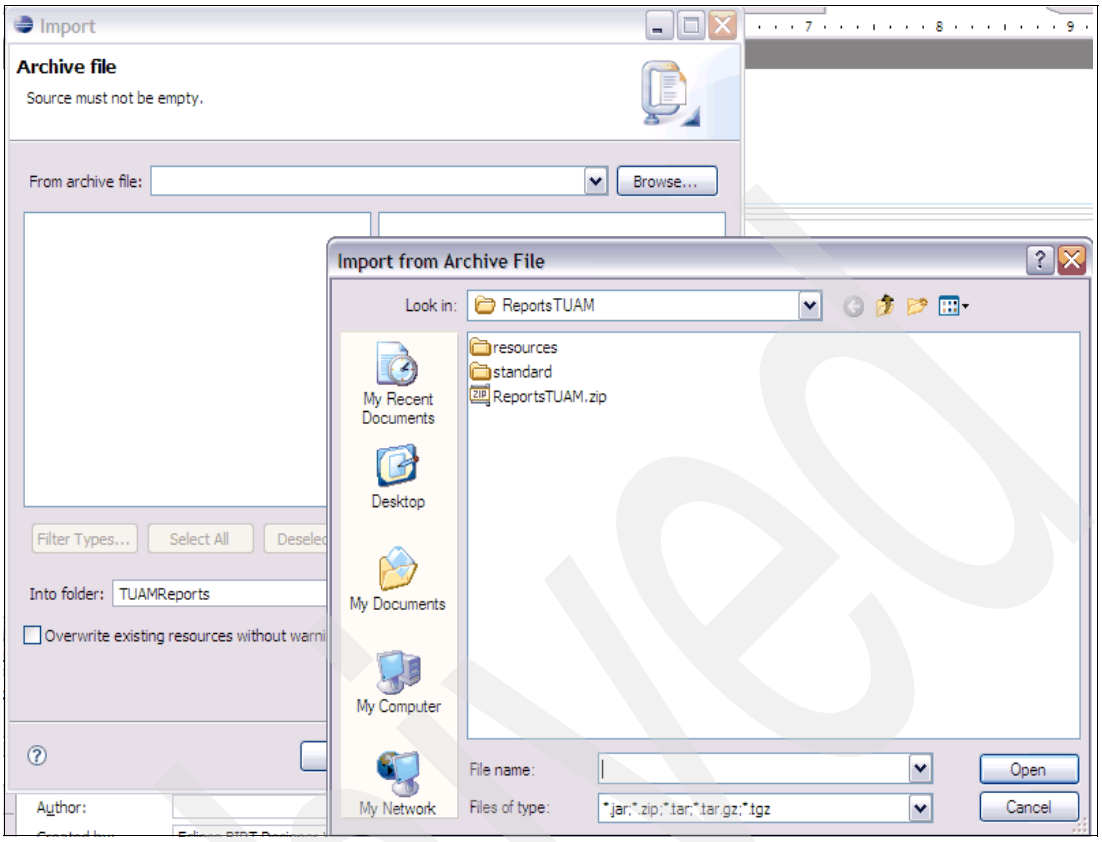


Figure 4-22 Selecting the .zip file

BIRT Report Navigator tab must have the components shown in Figure 4-23 on page 87 after the import.

Archive

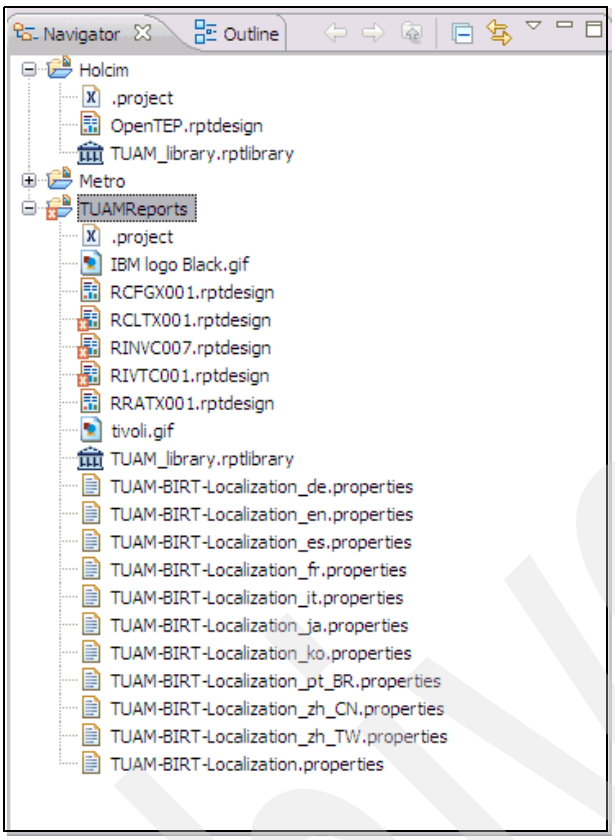


Figure 4-23 BIRT designer navigator structure

5. Edit the data source information of the report project and link to Tivoli Usage and Accounting Manager database, as in Figure 4-24 on page 88.

Archived

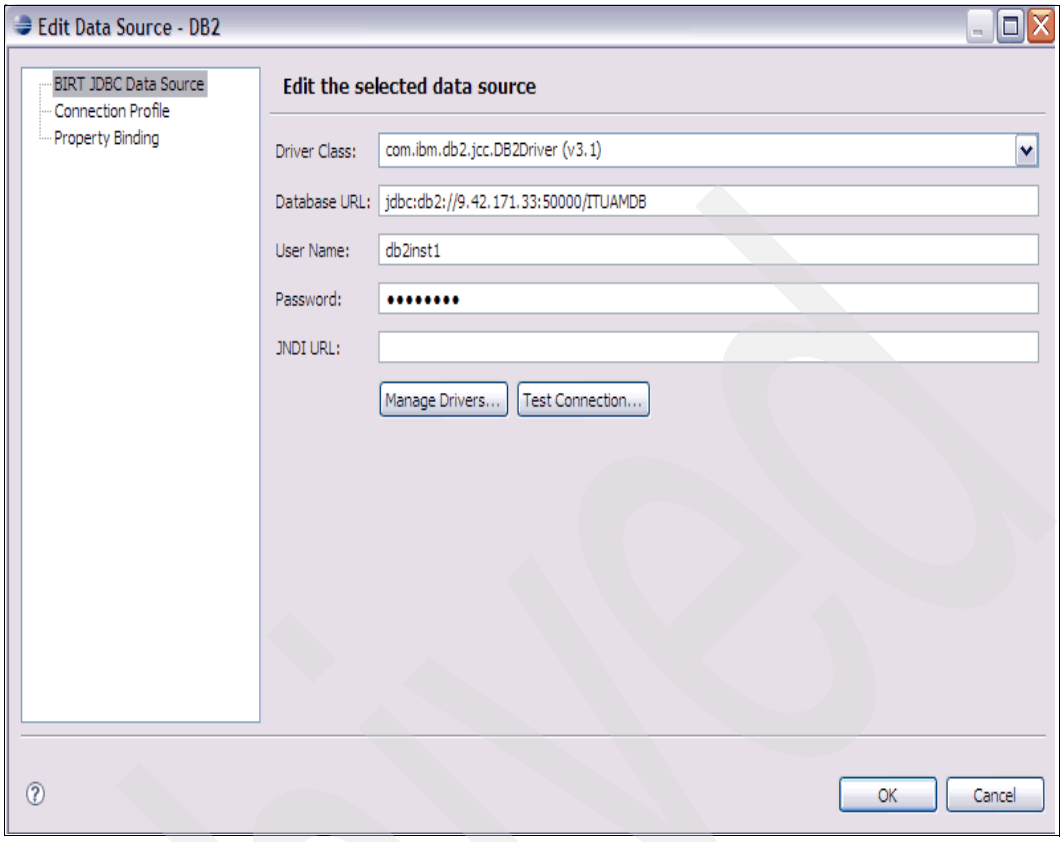


Figure 4-24 Data source information

6. Edit data sets to be sure that they are using the correct qualifier (the correct owner of the tables). Modify it, if it is wrong and test the reports with BIRT Preview.
7. Export reports (files with the `.rptdesing` extension) as Archive File, as shown in Figure 4-25 on page 89 and specify a `.zip` file name and a folder.

Archive

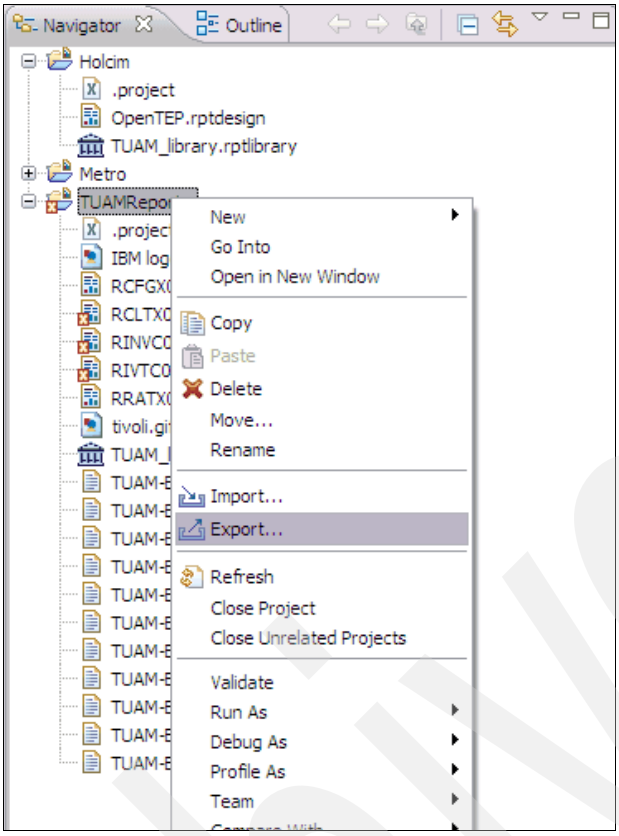


Figure 4-25 Export reports

8. Log in into Tivoli Common Reporting and, in the Task list, select **Reporting** → **Common Reporting** option.
9. Right-click **Report Sets** and select **Import Report Package**, as shown in Figure 4-26.

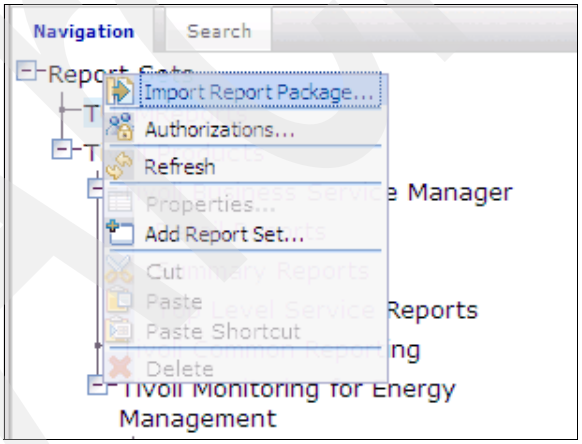


Figure 4-26 Import report package in Tivoli Common Reporting

Tivoli Usage and Accounting Manager reports appear in Tivoli Common Reporting as in Figure 4-27 on page 90.

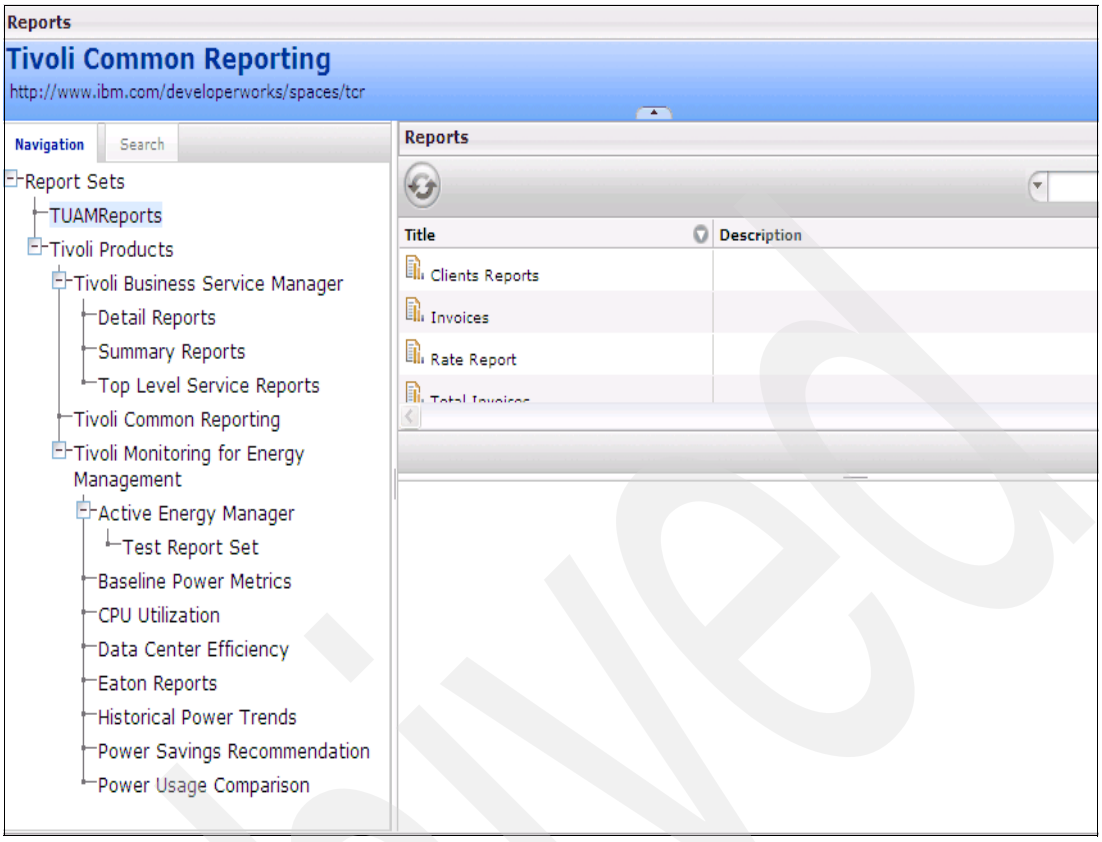


Figure 4-27 Tivoli Usage and Accounting Manager reports into Tivoli Common Reporting

4.6 Scenario walk-through

This scenario describes how Energy Dashboard built in Tivoli Business Service Manager with information from Tivoli for Energy Management and the integration with Tivoli Applications Dependency Discovery Manager and Tivoli Usage and Accounting Manager can be used for the following tasks:

- View charts that visualize aggregated power and thermal metrics of your data center.
- From the charts, launch to reports that support the data center metrics on the dashboard and help to control the costs of the data center resources.
- View power and thermal metrics for individual resources and make use of the information for data center optimization
- View the health status of individual resources based on color-coded indicators that are calculated by the use of threshold for thermal and power data.
- View related events for individual resources, and link events of IT and non-IT infrastructure with business services.
- From individual resources, launch to additional managed resource information:
  - Tivoli Enterprise Portal for technical dashboard
  - Tivoli Applications Dependency Discovery Manager for configuration details
- Launch to Tivoli Usage and Accounting Manager reports to control total business services costs.

The Energy Dashboard serves as a best practice implementation but can be customized to fit your own specific needs. For example, energy resources can be incorporated into user-defined service models to tie business services to infrastructure energy metrics, as in Figure 4-28, with a billing service.

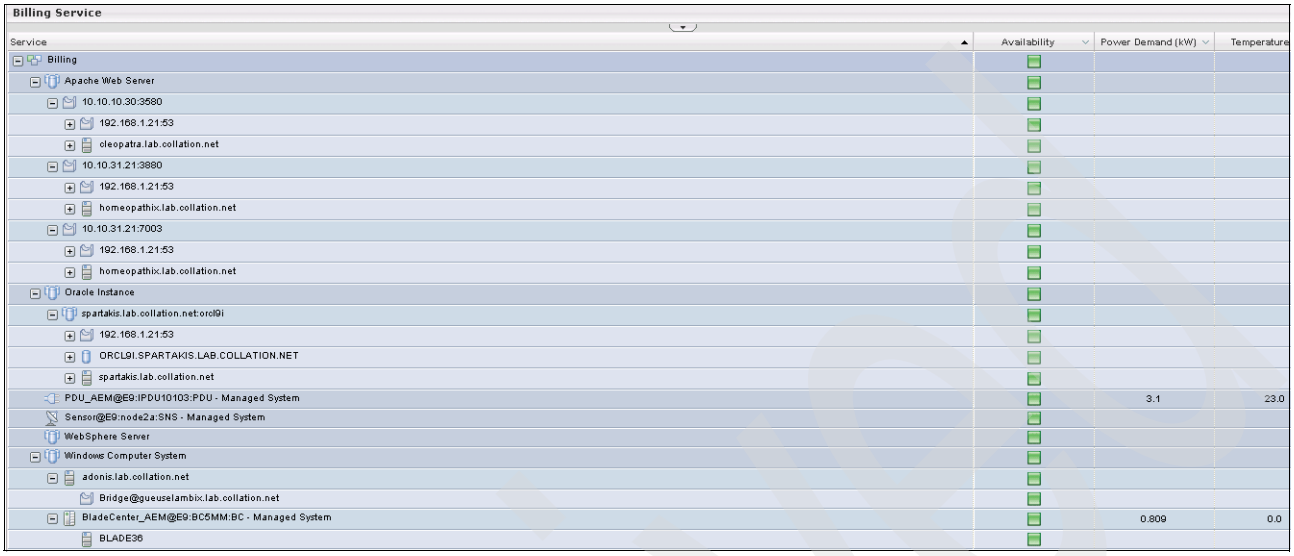


Figure 4-28 Using energy metrics in business service model

The Energy Dashboard described in this chapter consists of five windows or portlets, that show charts with power and thermal information from Tivoli Monitoring for Energy Management Reporting and Optimization, device metrics in a Tivoli Business Service Manager service tree, and service details for event management. An example is shown in Figure 4-29.

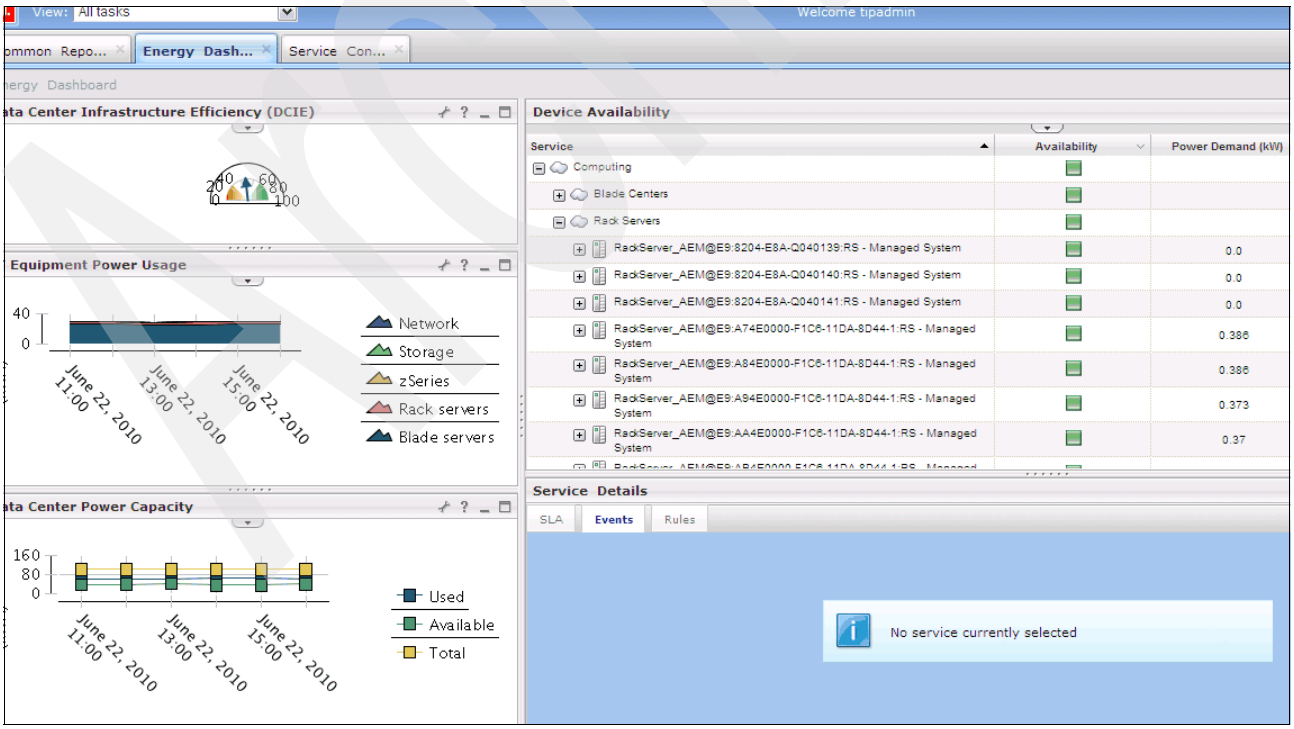


Figure 4-29 Energy Dashboard

4.6.1 Visualizing aggregated power and thermal metrics, and launching to reports

Energy Dashboard charts are used to visualize power data and thermal metrics for the data center. For this scenario, we selected three charts, and we also explain how the charts can be customized to include the type of information you might need.

The three charts for our Energy Dashboard are shown in Figure 4-30.

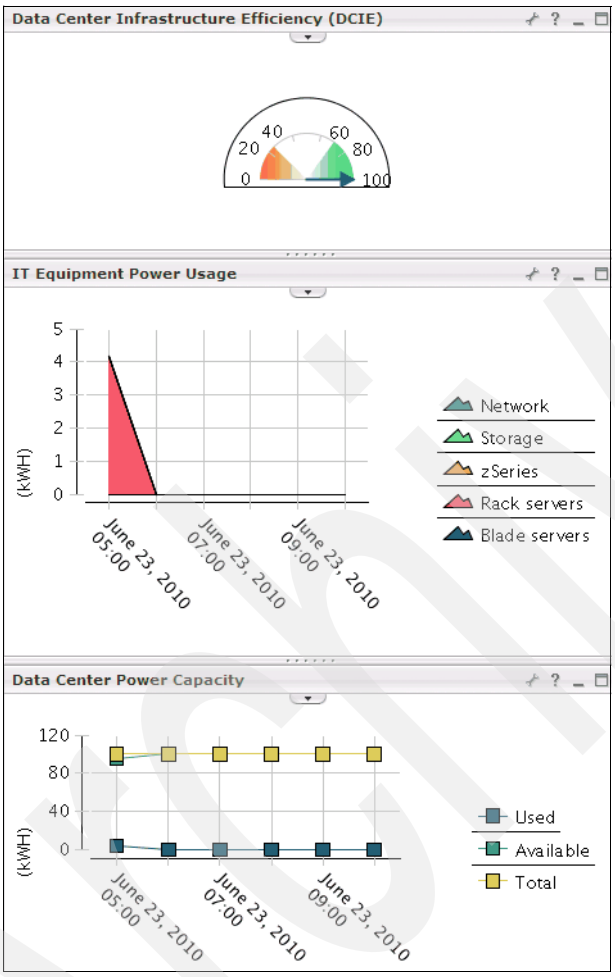


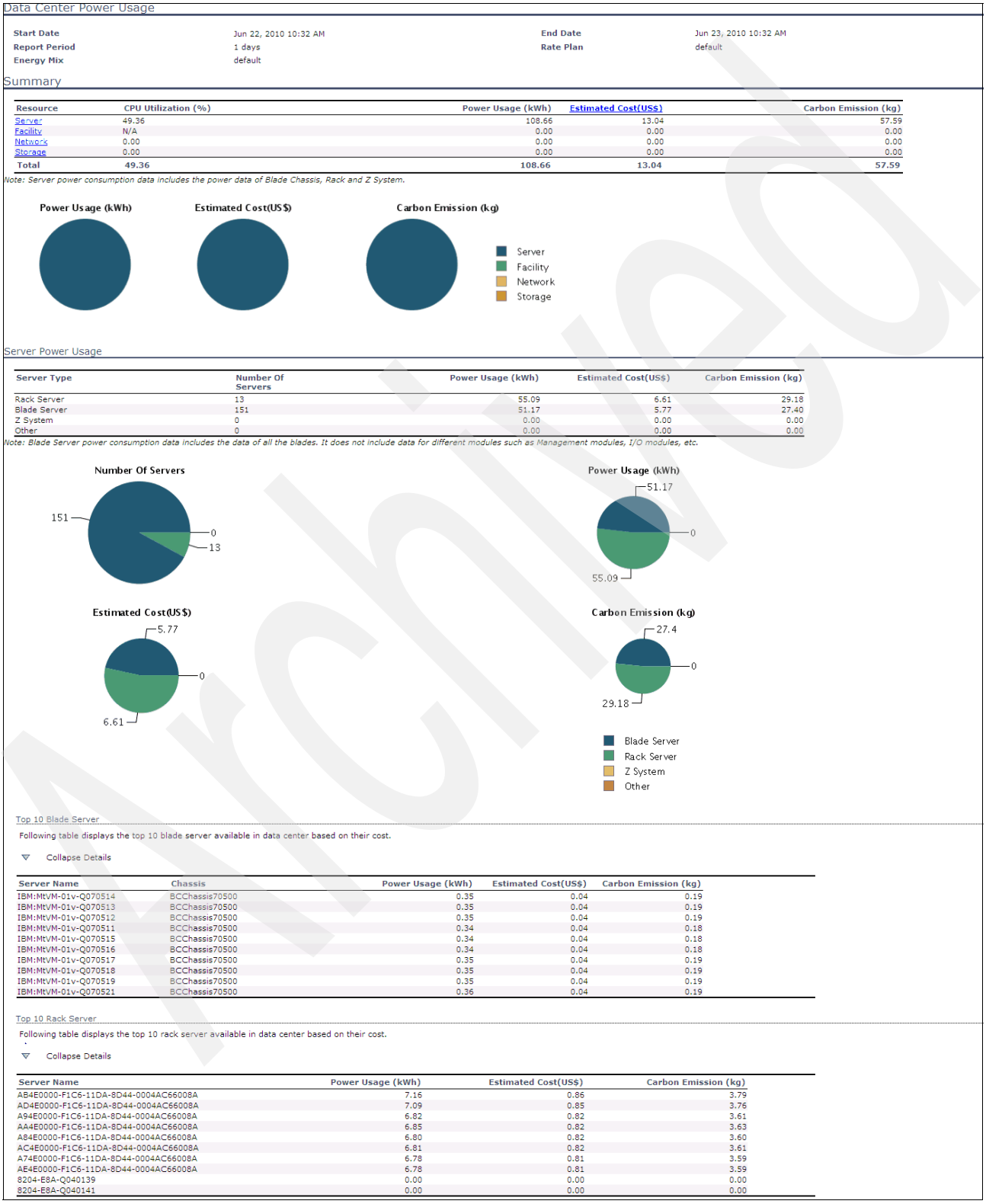
Figure 4-30 Charting portlets

The charts are as follows:

- Data Center Infrastructure Efficiency (DCIE): This chart shows the average value of the DCIE for the last six hours. The DCIE index represents the power that is consumed by the IT equipment divided by the total power consumed by the entire data center, including the IT equipment, and then multiplied by 100. For example, a DCiE value of 50% suggests that the IT equipment consumes 50% of the power in the data center. This information can help you to control your data center consumption.
- IT Equipment Power Usage: This chart shows, over time, the average of KW/h used per type of IT component, network, rack servers, blade servers, and so on. This information can help you identify the types of devices with higher consumptions in your data center.
- Data Center Power Capacity: This chart shows, over time, the relationship between your power capacity and your power usage.

Archived

Clicking in any chart causes launching to specific Tivoli Monitoring for Energy Management reports. For example, clicking in IT Equipment Power Usage chart opens the report in Figure 4-31 to help you understand the costs that are related to the power consumption and the top resources in terms of costs.



Data Center Power Usage

Start Date	Jun 22, 2010 10:32 AM	**End Date**	Jun 23, 2010 10:32 AM
Report Period	1 days	**Rate Plan**	default
Energy Mix	default		

Summary

Resource	CPU Utilization (%)	Power Usage (kWh)	Estimated Cost(US$)	Carbon Emission (kg)
Server	49.36	108.66	13.04	57.59
Facility	N/A	0.00	0.00	0.00
Network	0.00	0.00	0.00	0.00
Storage	0.00	0.00	0.00	0.00
Total	**49.36**	**108.66**	**13.04**	**57.59**

Note: Server power consumption data includes the power data of Blade Chassis, Rack and Z System.

Server Power Usage

Server Type	Number Of Servers	Power Usage (kWh)	Estimated Cost(US$)	Carbon Emission (kg)
Rack Server	13	55.09	6.61	29.18
Blade Server	151	51.17	5.77	27.40
Z System	0	0.00	0.00	0.00
Other	0	0.00	0.00	0.00

Note: Blade Server power consumption data includes the data of all the blades. It does not include data for different modules such as Management modules, I/O modules, etc.

Top 10 Blade Server

Following table displays the top 10 blade server available in data center based on their cost.

Collapse Details

Server Name	Chassis	Power Usage (kWh)	Estimated Cost(US$)	Carbon Emission (kg)
IBM:MtVM-01v-Q070514	BCChassis70500	0.35	0.04	0.19
IBM:MtVM-01v-Q070513	BCChassis70500	0.35	0.04	0.19
IBM:MtVM-01v-Q070512	BCChassis70500	0.35	0.04	0.19
IBM:MtVM-01v-Q070511	BCChassis70500	0.34	0.04	0.18
IBM:MtVM-01v-Q070515	BCChassis70500	0.34	0.04	0.18
IBM:MtVM-01v-Q070516	BCChassis70500	0.34	0.04	0.18
IBM:MtVM-01v-Q070517	BCChassis70500	0.35	0.04	0.19
IBM:MtVM-01v-Q070518	BCChassis70500	0.35	0.04	0.19
IBM:MtVM-01v-Q070519	BCChassis70500	0.35	0.04	0.19
IBM:MtVM-01v-Q070521	BCChassis70500	0.36	0.04	0.19

Top 10 Rack Server

Following table displays the top 10 rack server available in data center based on their cost.

Collapse Details

Server Name	Power Usage (kWh)	Estimated Cost(US$)	Carbon Emission (kg)
AB4E0000-F1C6-11DA-8D44-0004AC66008A	7.16	0.86	3.79
AD4E0000-F1C6-11DA-8D44-0004AC66008A	7.09	0.85	3.76
A94E0000-F1C6-11DA-8D44-0004AC66008A	6.82	0.82	3.61
AA4E0000-F1C6-11DA-8D44-0004AC66008A	6.85	0.82	3.63
A84E0000-F1C6-11DA-8D44-0004AC66008A	6.80	0.82	3.60
AC4E0000-F1C6-11DA-8D44-0004AC66008A	6.81	0.82	3.61
A74E0000-F1C6-11DA-8D44-0004AC66008A	6.78	0.81	3.59
AE4E0000-F1C6-11DA-8D44-0004AC66008A	6.78	0.81	3.59
8204-E8A-Q040139	0.00	0.00	0.00
8204-E8A-Q040141	0.00	0.00	0.00

Figure 4-31 Tivoli Monitoring for Energy Management report launching from Tivoli Business Service Manager charts

Archived

Clicking in the Data Center Infrastructure Efficiency (DCIE) chart takes you to a report that shows you the DCIE evolution for the last days, as in Figure 4-32.

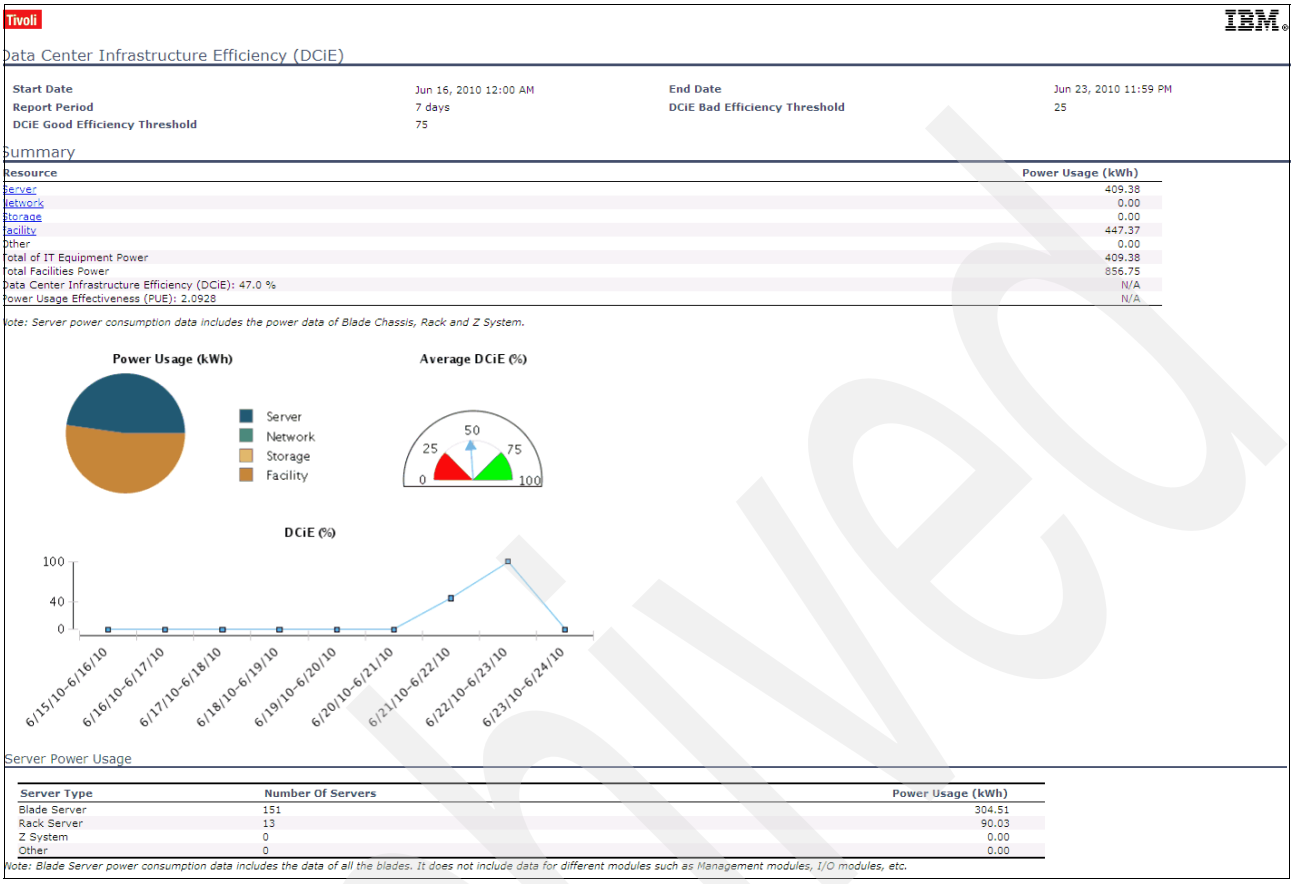


Figure 4-32 DCIE trend report

4.6.2 Viewing energy metrics and health status of resources

Energy Dashboard service tree shows a complete view of all the individual devices of the data center, their relationships and health status, and also displays KPIs of power and temperature for each of them.

Figure 4-33 on page 95 shows several elements of the service tree for Energy Dashboard.

Archived

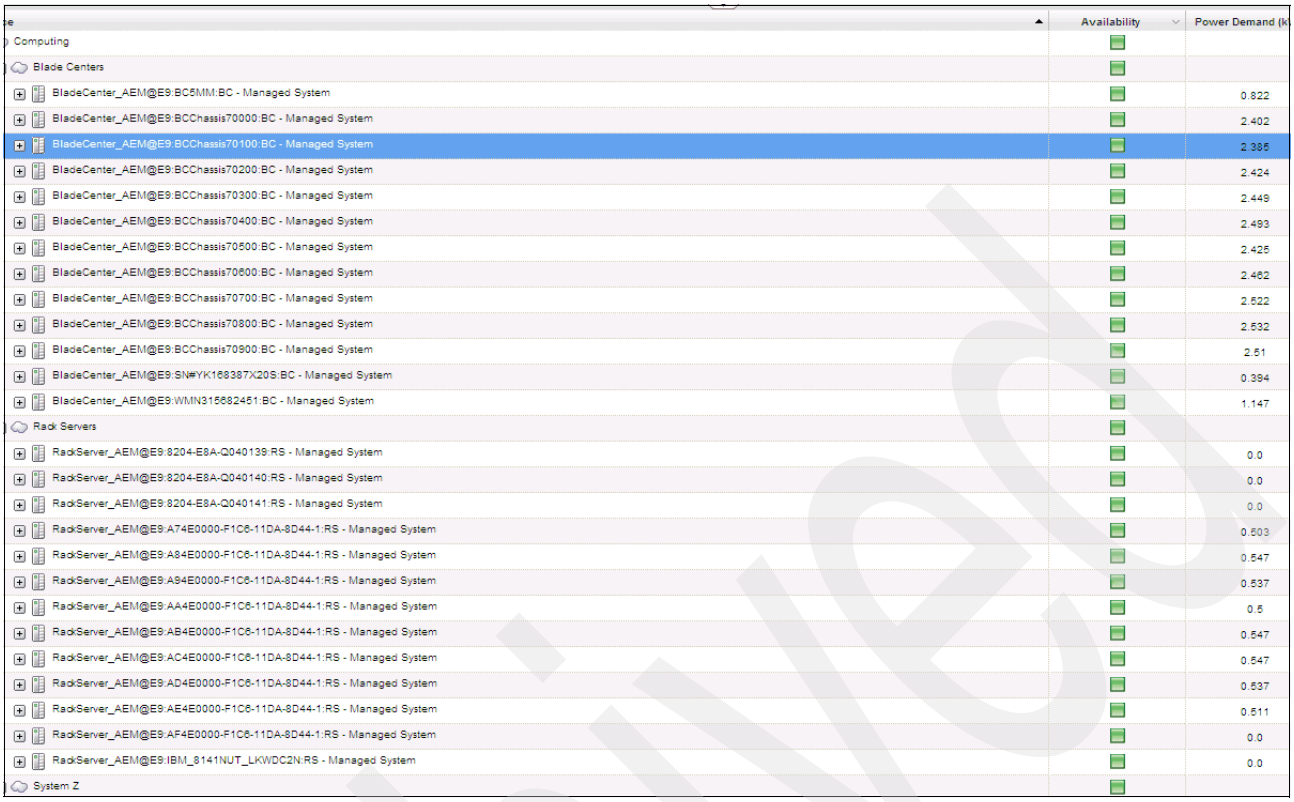


Figure 4-33 Service tree for Energy Dashboard

The Service Tree scorecard contains the following columns:

- Availability: A color (green, yellow or red) indicates the current state of the device. This state can be obtained from several sources: incoming events, environmental or energy metrics, and others.
- Power Demand (KW): The current power demand is expressed in KW of (or on, according to the type of device) the device.
- Temperature (ºC): Indicates the current ambient or exhaust temperature of a device (depending on the type of device)

In the scorecard, power and thermal metrics are highlighted with a background of red or yellow if defined status thresholds for each device type are exceeded. See Figure 4-34.

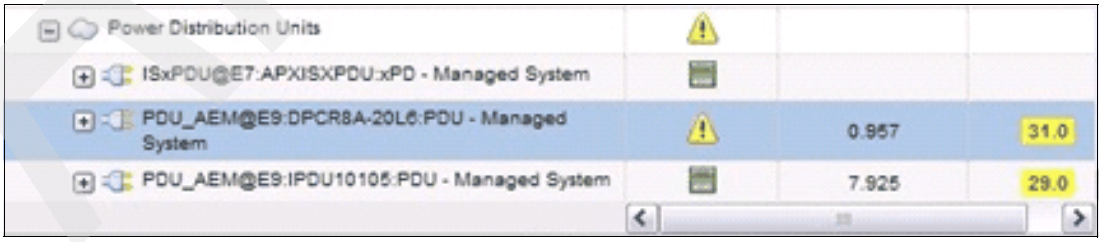


Figure 4-34 Temperature highlighted because warning threshold is reached

The warning and critical thresholds for each type of device are defined in each energy template. For example, all the PDUs have default thresholds for warning and critical status, defined in the BSM_PowerDistributionUnit_AEM template. These default threshold and

warning settings are in the Additional tab of the template edition, as in Figure 4-35 on page 96.

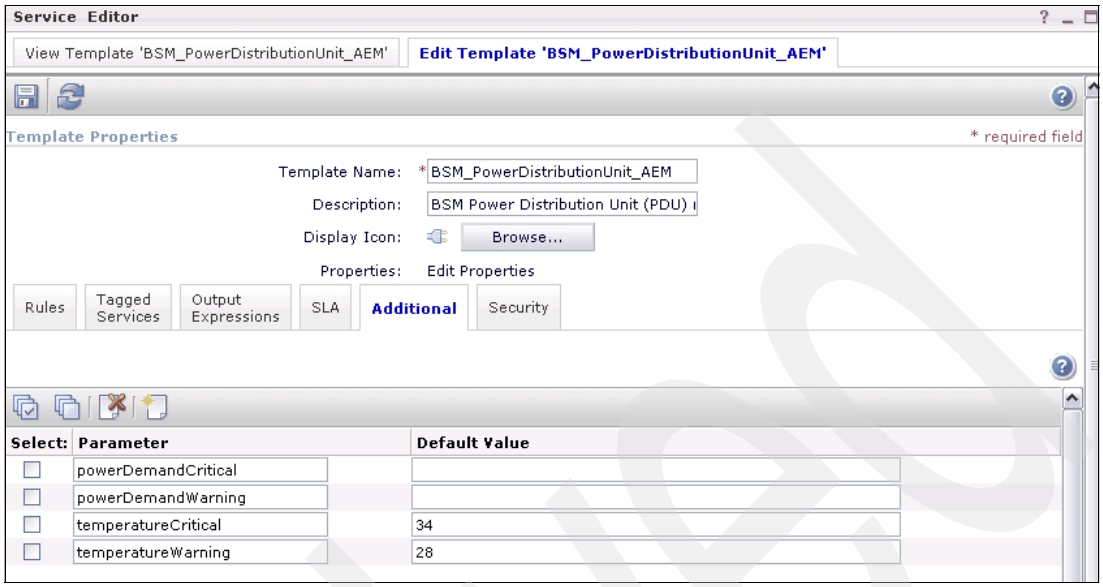


Figure 4-35 Temperature and power demand thresholds defined per template

The default value of the thresholds can be modified for each service instance, in the Additional tab for the service editor, as shown in Figure 4-36.

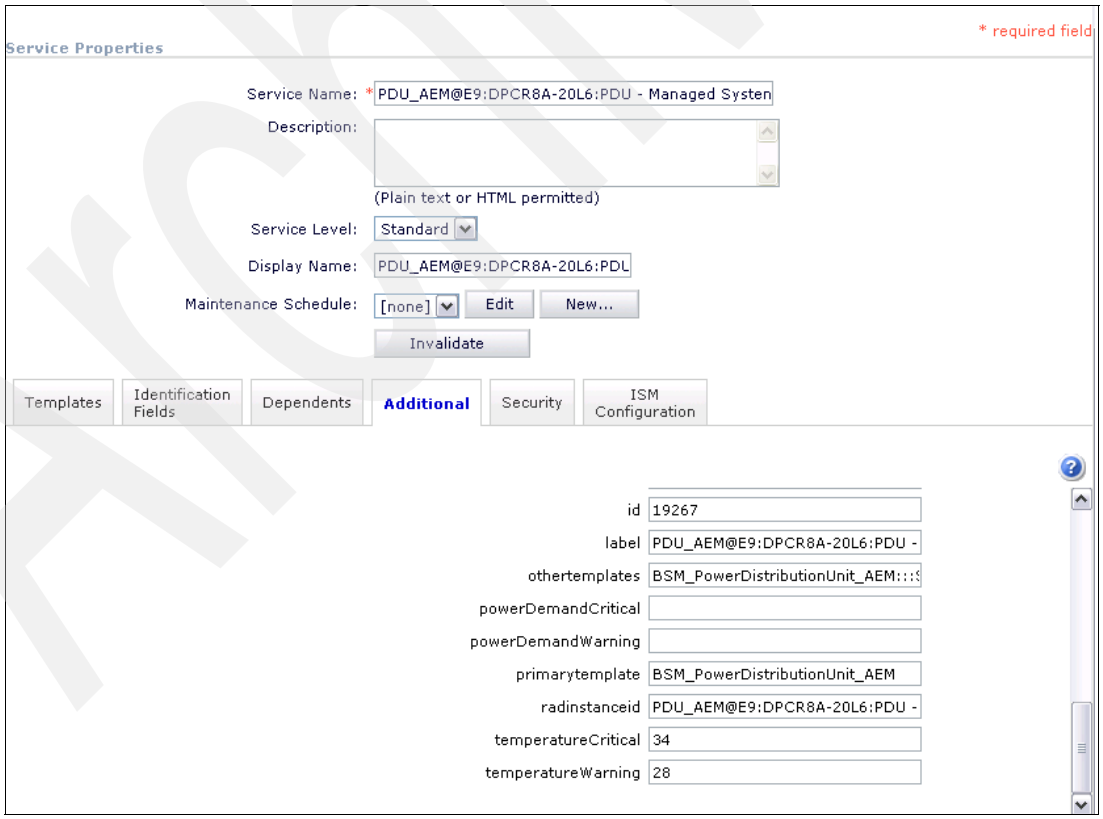


Figure 4-36 Threshold definition for service instances

4.6.3 Launching to additional managed resource information

Energy Dashboard allows the launching to Tivoli Enterprise Portal to see more technical details about IBM Tivoli Monitoring Energy Management, as shown in Figure 4-37.

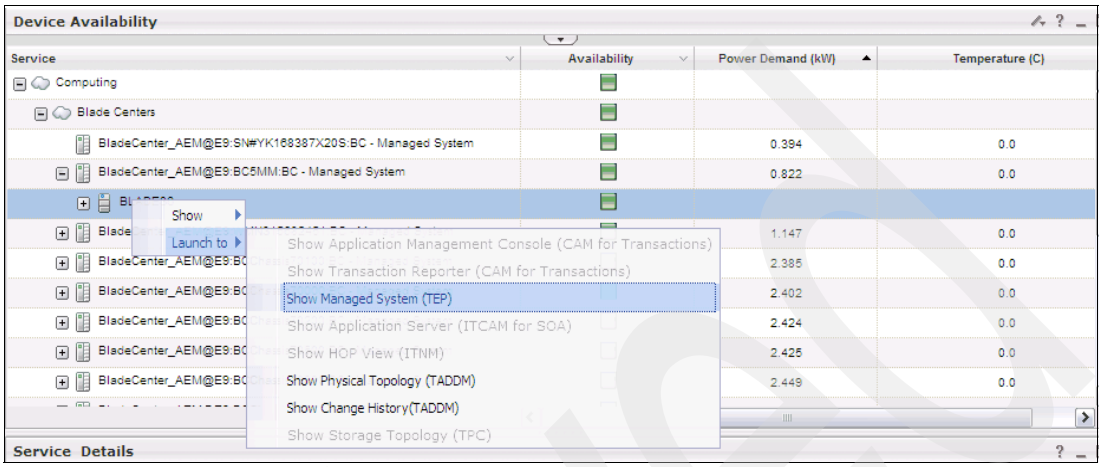


Figure 4-37 Launching to Tivoli Enterprise Portal

Tivoli Enterprise Portal provides technical views or workspaces per device, as shown in Figure 4-38. More details about Tivoli Monitoring for Energy Management workspaces is in 3.7.2, “Visualizing and reporting energy and thermal metrics” on page 43.

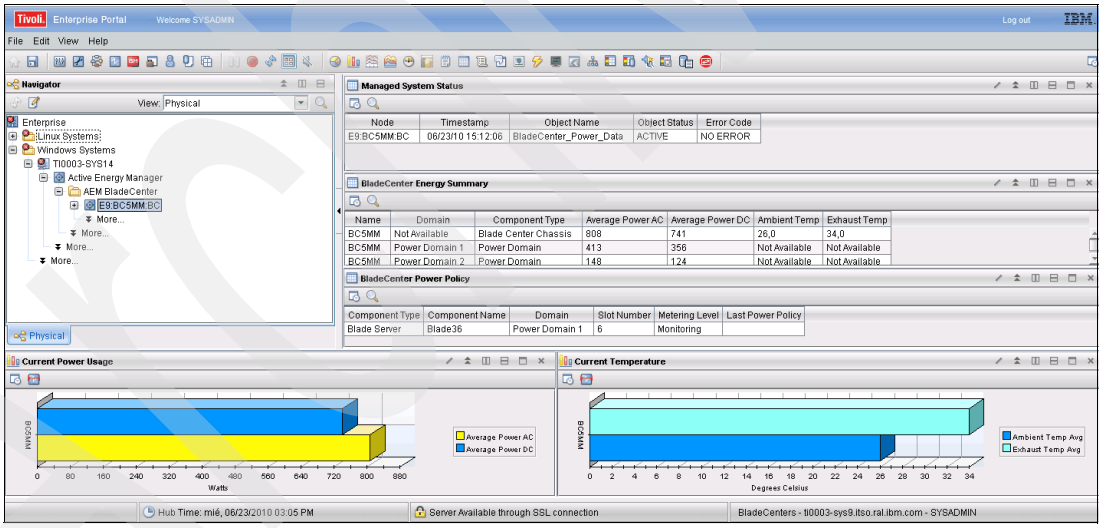


Figure 4-38 Tivoli Enterprise Portal

Energy Dashboard can be used to launch to Tivoli Applications Dependency Discovery Manager(TADDM), as is shown in Figure 4-39 on page 98.

Archive

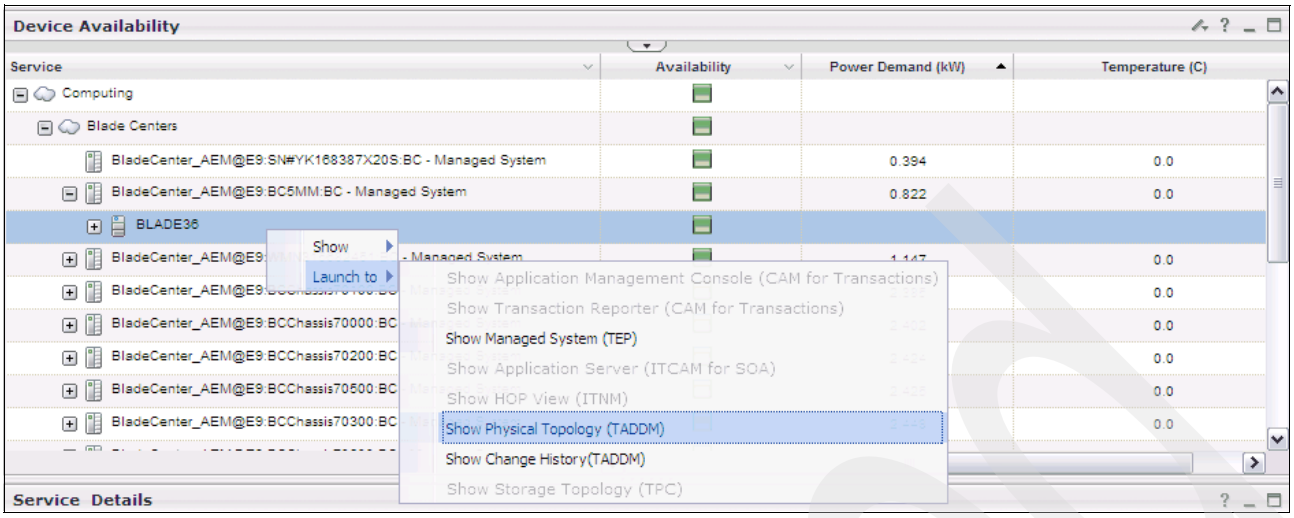


Figure 4-39 Launching to Tivoli Applications Dependency Discovery Manager

From a device, you can see TADDM configuration details (Figure 4-40).

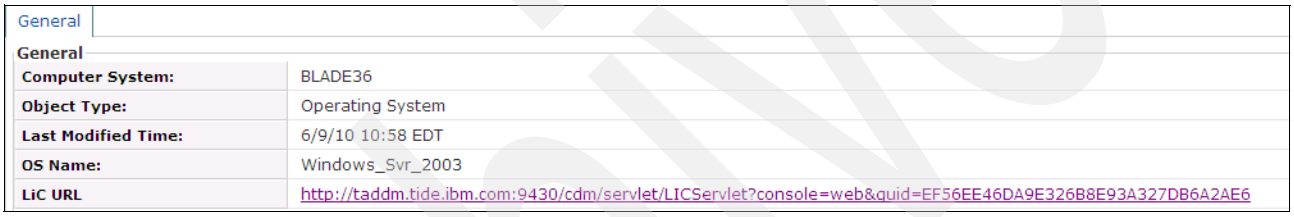


Figure 4-40 TADDM configuration item details page

You can also see changes in the configuration of the device (Figure 4-41).

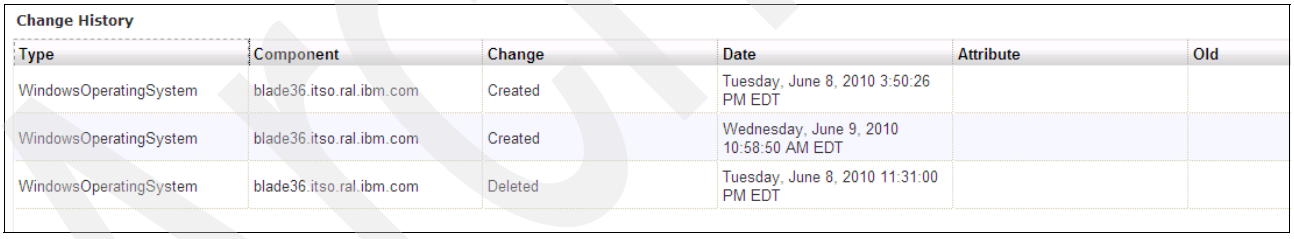


Figure 4-41 TADDM changes to a configuration item

4.6.4 Launching to Tivoli Usage and Accounting Manager reports from Tivoli Common Reporting

IBM Tivoli Usage and Accounting Manager can use Tivoli Monitoring for Energy Management information to report on the amount of power consumed, when it was consumed, and which services consumed it.

Although Tivoli Usage and Accounting Manager uses its own report engine, several reports have also been designed with BIRT and can be loaded into the reporting engine of Tivoli Business Service Manager, as is explained in 4.5.5, “Integrating Tivoli Usage and Accounting Manager BIRT Reports into Tivoli Common Reporting engine of Tivoli Business Manager” on page 84.

Tivoli Usage and Accounting Manager reports that are loaded into Tivoli Business Service Manager can be displayed by clicking **Reporting** → **Common Reporting** in the Task list, as in Figure 4-42.

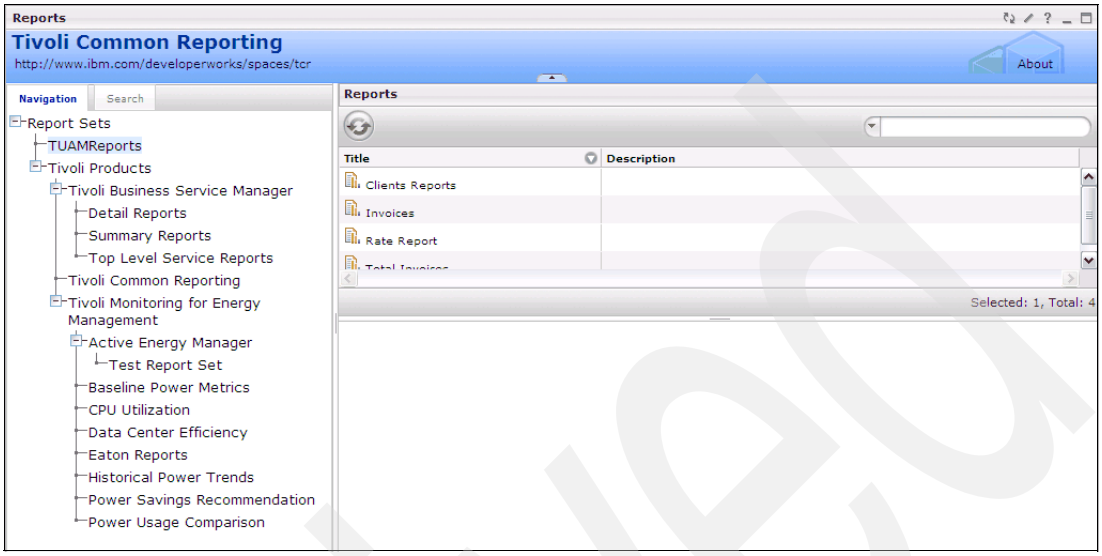


Figure 4-42 Tivoli Usage and Accounting Manager reports

Several reports that can be launched from Tivoli Common Reporting are as follows:

- Top accounts in terms of costs (Figure 4-43)

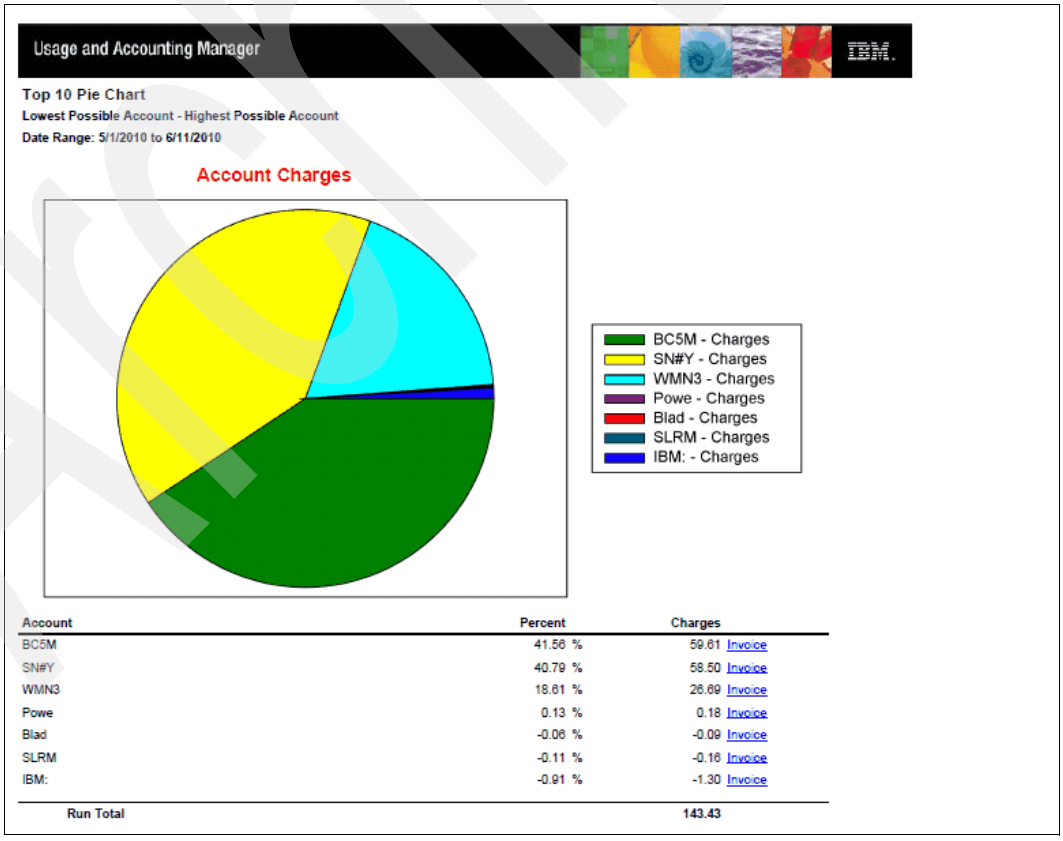
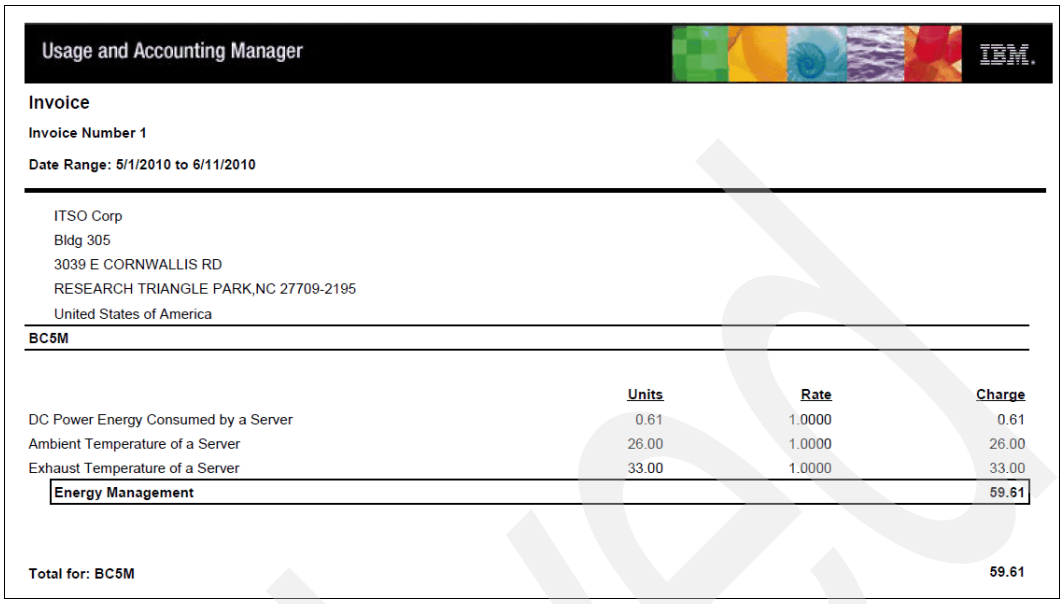


Figure 4-43 Tivoli Usage and Accounting Manager Top accounts charges report

Archived

- Invoice linked to a specific account (Figure 4-44)


Usage and Accounting Manager

IBM.

Invoice

Invoice Number 1

Date Range: 5/1/2010 to 6/11/2010

ITSO Corp
Bldg 305
3039 E CORNWALLIS RD
RESEARCH TRIANGLE PARK,NC 27709-2195
United States of America

BC5M

	Units	Rate	Charge
DC Power Energy Consumed by a Server	0.61	1.0000	0.61
Ambient Temperature of a Server	26.00	1.0000	26.00
Exhaust Temperature of a Server	33.00	1.0000	33.00
Energy Management			**59.61**

Total for: BC5M **59.61**

Figure 4-44 Tivoli Usage and Accounting Manager Invoice report

Archived

4.7 Summary

Figure 4-45 shows the quick summary for this integration.

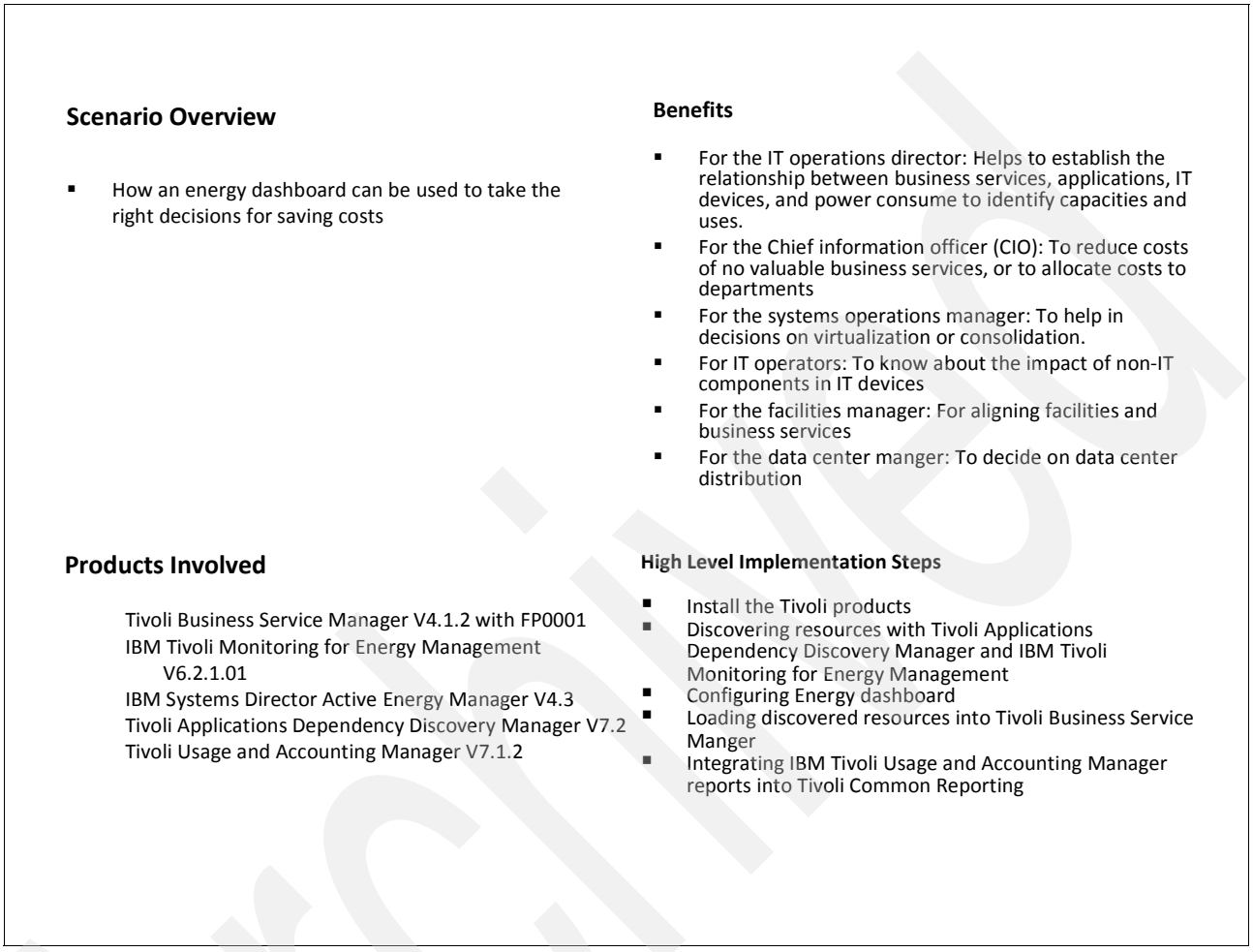


Figure 4-45 Quick summary

In this scenario, we demonstrated how you can use a unified dashboard to make the right decisions in terms of saving costs to your company, put the focus on business value services, and manage them through a holistic view of all their components, including energy and environmental information.

We first showed you how to discover the resources and their relationships using TADDM and TMS DLA. Then, we described how to import all the discovered resources into Tivoli Business Service Manager Energy Dashboard and how Tivoli Business Service Manager uses Tivoli Monitoring for Energy Management information for showing power demand and temperatures.

Finally, we explained how to integrate Tivoli Usage and Accounting Manager reports into the reporting engine of Tivoli Business Service Manager.

Archived

Archived

5

Green IT scenario for Asset Management

This chapter provides a detailed overview about the configuration of Green IT based Asset Management provided by IBM products.

IBM Maximo Asset Management for Energy Optimization helps you monitor the data center environment to identify ways to optimize energy consumption. You can view the layout and a heat map of your data center. Then, you can use the provided service management functions to address related issues, reducing energy costs, and eliminating associated risks.

The chapter guides you through the steps to configure Maximo Asset Management for Energy Optimization and IBM Tivoli Monitoring for Energy Management to obtain asset and sensor information. This information is used to build graphical views of the data center layout and heat map. You use these views to monitor the conditions in your data center and to identify potential problems. Click an asset on the layout view to retrieve a list of related assets (such as servers mounted on a rack), view meter readings, or create a work order for the asset.

This chapter contains the following topics:

- 5.1, “Scenario overview” on page 104
- 5.2, “Products involved” on page 104
- 5.3, “Benefits” on page 105
- 5.4, “High-level implementation steps” on page 106
- 5.5, “Implementing Tivoli Monitoring for Energy Management into Maximo Asset Management for Energy Optimization” on page 106
- 5.6, “Architectural diagram of integration” on page 118
- 5.7, “Scenario walk-through” on page 120
- 5.8, “Summary” on page 124

© Copyright IBM Corp. 2011. All rights reserved.

5.1 Scenario overview

The scenario has been designed to enable proper maintenance of your building facilities and the components included. This way helps to ensure optimal performance of your assets by optimizing the lifetime, reduction of energy waste, and in general a reduced carbon footprint by using less energy. Figure 5-1 shows you several modern Asset Management practices implemented in enterprises.

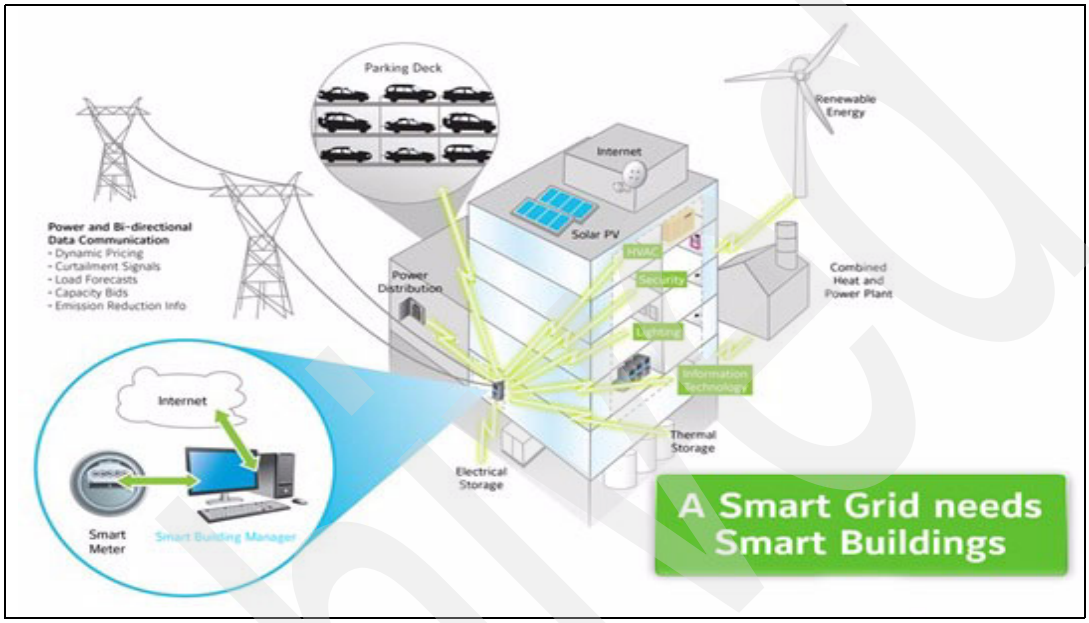


Figure 5-1 Asset Management today

We show how to track your assets and how to automate the maintenance for your components.

5.2 Products involved

The products and product components involved in this scenario are as follows:

- IBM Maximo Asset Management for Energy Optimization V 7.1.0.0
- IBM Tivoli Integration Composer V7.2.0.0
- IBM Tivoli Asset Management for IT V 7.2.0.0
- IBM Tivoli Naming Reconciliation Services 1.0.0
- Common PMP 7.1.1.5

In addition, the following products have been installed to support the products:

- IBM WebSphere Application Server 6.1
- Windows Server 2003 5.2 Service Pack 2

A detailed list of software and hardware requirements for IBM Maximo Asset Management for Energy Optimization is published on the IBM website at:

`http://www.ibm.com/software/tivoli/products/maximo-asset-mgmt-energy-optimize/requirements.html?S_CMP=rnav`

5.3 Benefits

IBM Maximo Asset Management for Energy Optimization performs a regular scheduled maintenance on building facilities and components. This scheduling ensures maintenance is performed as required to optimize the energy usage. The end-to-end service process tracks and schedules maintenance requests, hosted all in one system. This way optimizes your site travel requirements and provides instant access to your status and history data.

The benefits of this scenario are as follows:

- Reduce energy costs and eliminate service outages by identifying and eliminating energy inefficiencies.
- Visualize energy and environmental metrics to identify temperature hot-spots in a simple map overview.
- Remediate energy issues by generating service requests to take action on energy issues
- Link energy metrics with asset information to manage energy as a part of asset management processes, such as condition monitoring and maintenance.
- Ability to create “what if” scenarios. Real-time energy, thermal, and IT dashboards can be used by the IT and operations staff to conduct what-if scenarios on environmental and energy impact and analyze how altering the physical layout and cooling set-points affect the thermal map of the data center. This technique helps organizations determine the most energy-efficient data center layout and facilities equipment settings.

Figure 5-2 shows the Asset Management process improvements over the years.

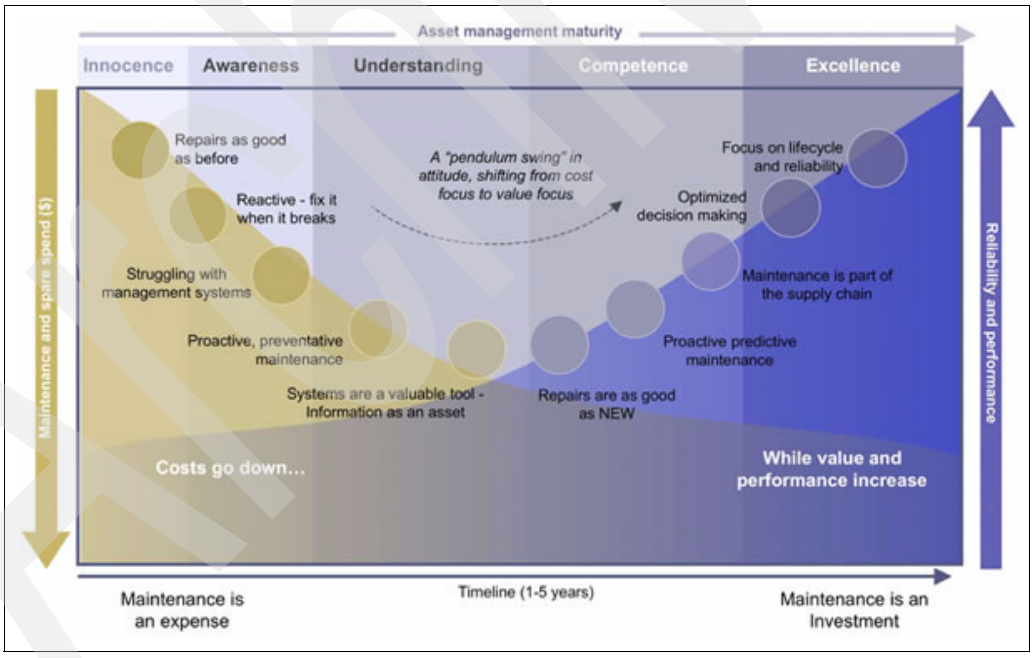


Figure 5-2 Asset Management improvements

A detailed description is at the following address:

http://www.ibm.com/software/tivoli/products/maximo-asset-mgmt-energy-optimize/

5.4 High-level implementation steps

This chapter guides you through the high-level steps that are required to accomplish the scenario in your environment.

This chapter does not give you a detailed installation instructions but does outline the main steps and highlight the steps that differ from a standard installation without integration. Figure 5-3 shows a simplified integration diagram for this scenario.

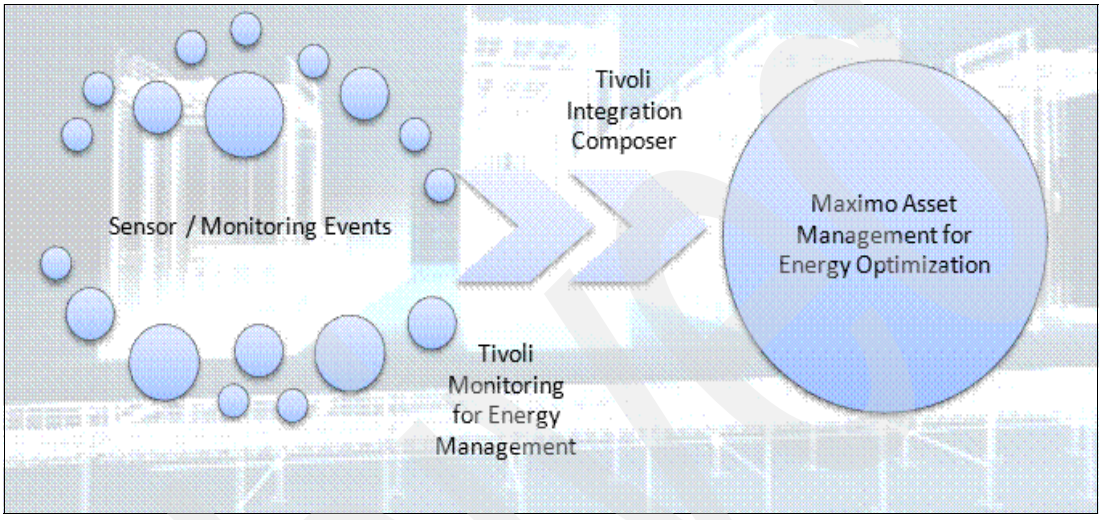


Figure 5-3 Simplified integration diagram

Based on the system requirements, we assume that you have the products in place and configured according to the product guidelines. See 5.2, “Products involved” on page 104.

5.5 Implementing Tivoli Monitoring for Energy Management into Maximo Asset Management for Energy Optimization

The Tivoli Monitoring for Energy Management provides the data from a variety of devices or assets hosted in your facility. The IBM Tivoli Integration Composer integration adapter is used to map the monitoring data to the asset information in Maximo Asset Management for Energy Optimization.

This section describes how to configure the integration.

5.5.1 Configuring IBM Tivoli Integration Composer for data connection

Define the data source connections that connects Integration Composer to Tivoli Monitoring for Energy Management and Maximo Asset Management for Energy Optimization.

The Tivoli Integration Composer uses a JDBC driver to connect to data sources. Maximo Asset Management for Energy Optimization provides two data sources:

- The ITM61 data source is used for connection to the source, which is in our scenario the Tivoli Monitoring for Energy Management database.
- The MEO71 data source is used for connection to the target system, which is the Maximo Asset Management for Energy Optimization database.

To define the mapping, you must create the connections for both data sources as shown in Figure 5-4.

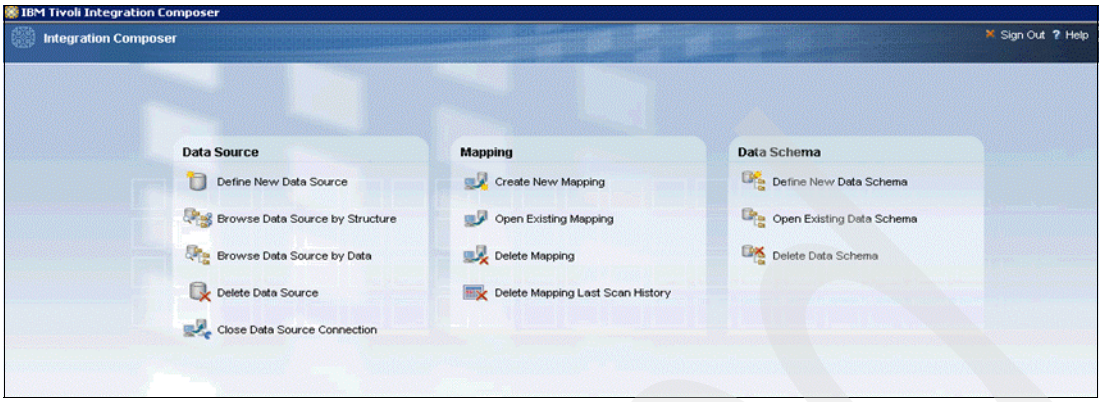


Figure 5-4 Tivoli Integration Composer

You must have database administrative rights to access the user interface. Integration Composer stores each database user ID that you enter when defining connectivity to the source and target data sources, but it does not store the passwords. After you define your data source connections, the only parameter that the Integration Composer user interface requests from you in real time is the password for the data source (that is, the database password for database users).

Source data connection procedure

The source data connection procedure is as follows (see and Figure 5-6 on page 108:

1. Sign in to Integration Composer using the Maximo database user ID and password.

   Tip: The default database name is MAXDB71 and the default user is MAXIMO.

2. Under Data Source, select **Define New Data Source**.
3. In the Define a New Data Source window, select **Tivoli Monitoring 6.1**, and then click **Next**. Type `ITM61` as the data source name, click **Next**, and then confirm that you want to overwrite the data source.
4. In Connection Method, select **IBM Tivoli Monitoring**. Specify the parameters for connecting to Tivoli Monitoring for Energy Management. Click **Test Connection to verify the connection**.
5. Click **Finish**.

Figure 5-5 shows how to define a new data source.

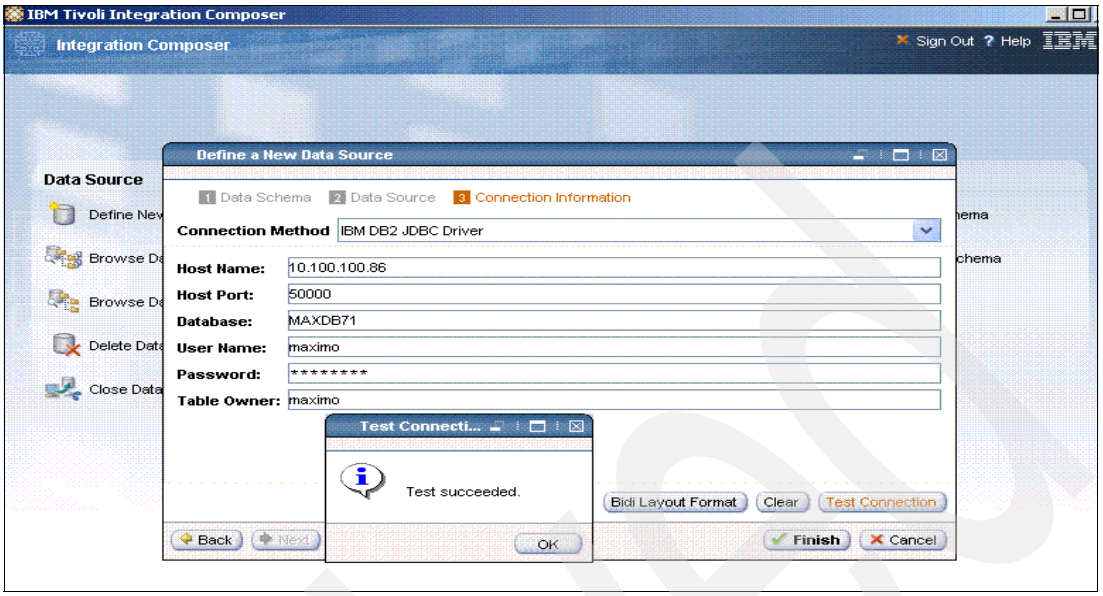


Figure 5-5 Tivoli Integration Composer: Define a new data source

Figure 5-6 shows how to select the data source.

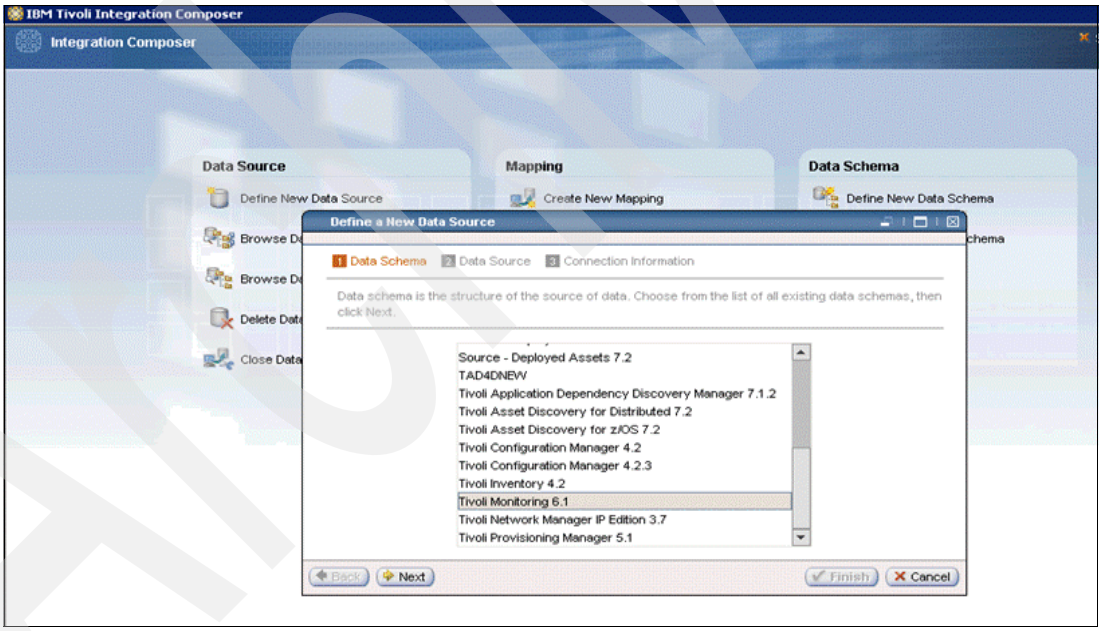


Figure 5-6 Tivoli Integration Composer: Define source data

Target data connection procedure

The source data connection procedure is as follows (see Figure 5-7 on page 109):

1. In the Integration Composer window, select **Define New Data Source**.
2. In the Define a New Data Source window, select **Deployed Asset 7.2**, and then click **Next**. Type `ME071` as the data source name, click **Next**, and then, confirm that you want to overwrite the data source.

3. In Connection Method, select the option for the database in which Maximo Asset Management for Energy Optimization is running. Specify the parameters for connecting to the Maximo Asset Management for Energy Optimization database. Click **Test Connection** to verify the connection.
4. Click **Finish**.

Figure 5-7 shows the options to select the target.

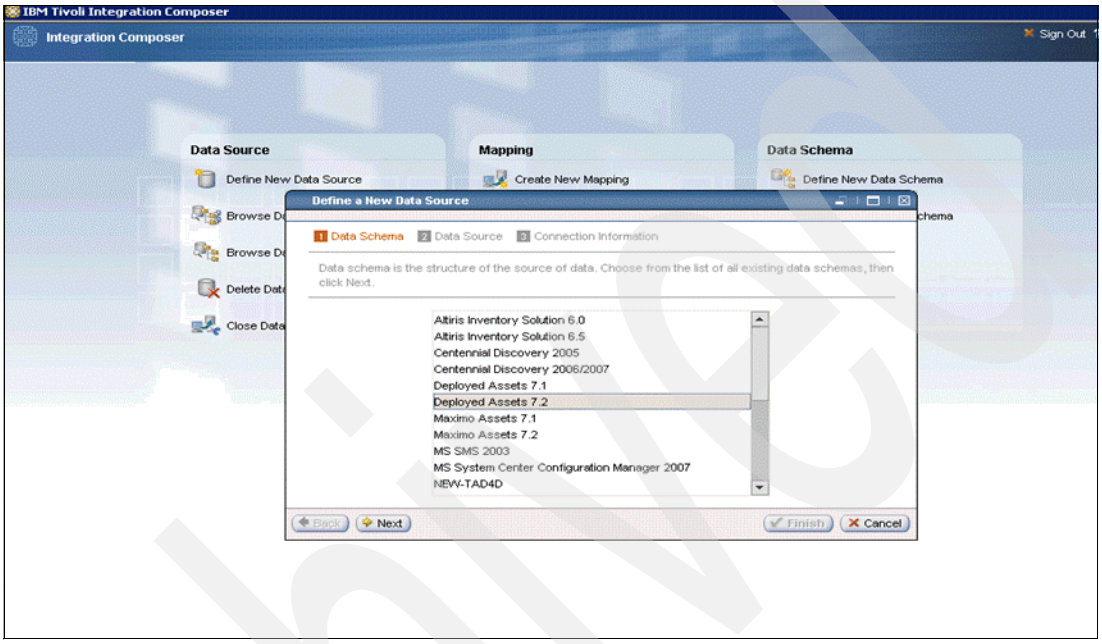


Figure 5-7 Tivoli Integration Composer: Define target data

5.5.2 Configuring the mapping in Tivoli Monitoring for Energy

The integration adapter requires a mapping to load data from Tivoli Monitoring for Energy Management to Maximo Asset Management for Energy Optimization.

Connection between source and target database

The mapping connects the source and the target database. The mapping information about the mapping expressions is stored in a mapping file and must be imported.

The mapping procedure is as follows:

1. Sign in to Integration Composer using a valid user ID and password.
2. Select **Create New Mapping**.
3. In the Source list, select the data source that you defined for Tivoli Monitoring for Energy Management, for example, ITM61.
4. In the Target list, select the data source that you defined for Maximo Asset Management for Energy Optimization.
5. Type a name for the new mapping, and then click **OK**. If necessary, enter the password for the connection to the source and target database, if requested.
6. In the Mapping window, select the **Import action**.
7. Select the `ITM61toDPA72.fsn` file, and then click **Open**.
8. After the file is imported, click **Save**, and then click **Close**.

Figure 5-8 shows the .fsn file import process.

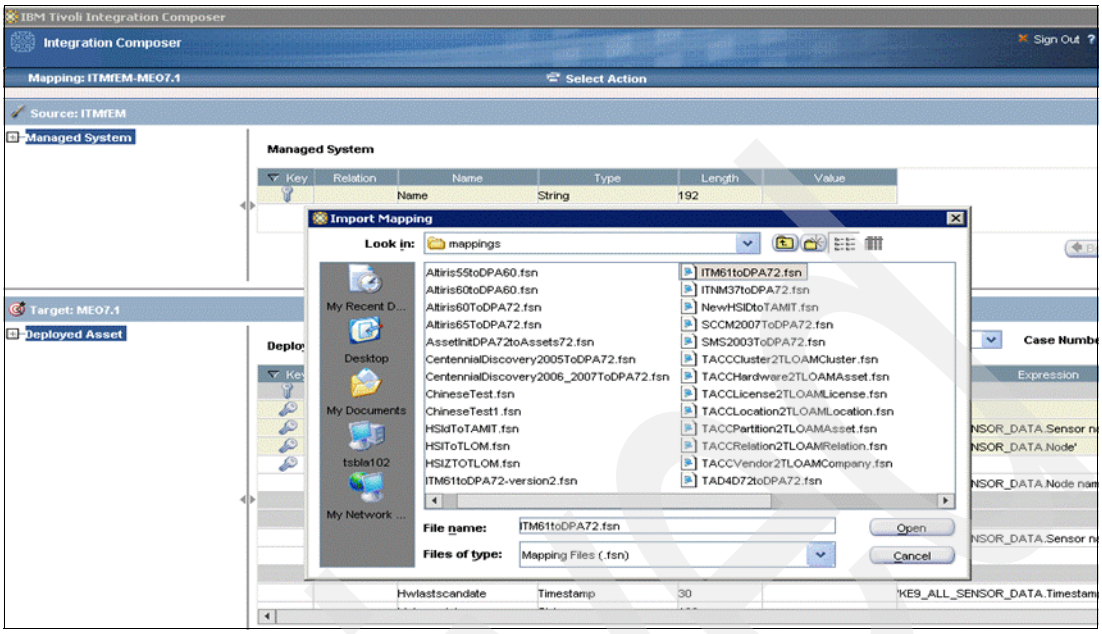


Figure 5-8 Tivoli Integration Composer: Mapping process

5.5.3 Configuring Tivoli Maximo Asset Management for Energy Optimization

With the given setup you can run the integration adapter mapping to load asset data from Tivoli Monitoring for Energy Management into Maximo Asset Management for Energy Optimization.

Mapping configuration

To execute the data mapping, run the integration adapter mapping to load data from Tivoli Monitoring for Energy Management into the Deployed Assets tables in Maximo Asset Management for Energy Optimization. You can use several options to set up the mapping execution to fit to your daily needs.

One-time mapping execution

You must create a mapping in Tivoli Integration Composer between Tivoli Monitoring for Energy Management and Maximo Asset Management for Energy Optimization.

Run the mapping command once to load the asset data to the Maximo Asset Management for Energy Optimization Deployed Assets tables.

Use any scheduling method to run this command periodically to keep your database updated. Perform the following steps:

1. On the computer where Tivoli Integration Composer is installed, open a command prompt and go to the `bin` subdirectory of the Tivoli Integration Composer installation directory.
2. Enter the following command, depending on the OS:
   - For Windows:

     ```
     commandLine.bat <mapping name> <Integration Composer database userid>
     <Integration Composer database password> <Tivoli Monitoring userid> <Tivoli
     Monitoring password> <Maximo for Energy Optimization database userid>
     <Maximo for Energy Optimization database password>
     ```

The execution is shown Example 5-1.

Example 5-1 Tivoli Integration Composer: Windows command line execution

```
C:\Integration Composer\comandLine.bat ITMfEM-ME07.1 maximo db2admin test
test maximo db2admin
```

– For UNIX:

```
commandLine.sh <mapping name> <Integration Composer database userid>
<Integration Composer database password> <Tivoli Monitoring userid> <Tivoli
Monitoring password> <Maximo for Energy Optimization database userid>
<Maximo for Energy Optimization database password>
```

3. Follow the standard reconciliation and promotion process shown in Figure 5-9 to promote the deployed assets that were loaded to authorized assets.

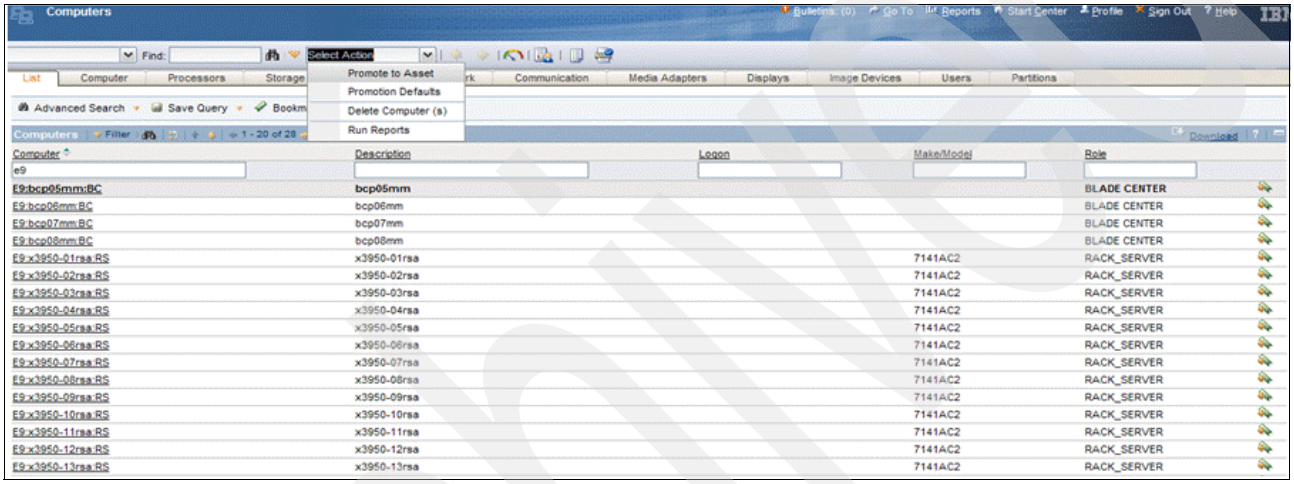


Figure 5-9 Maximo: Reconciliation and promotion process

Further information is in the Maximo online help.

Scheduled data load

Configure and run the cron task to load meter information from Tivoli Monitoring for Energy Management into Maximo Asset Management for Energy Optimization.

The PLUSEITMMETERS cron task obtains meter data from Tivoli Monitoring for Energy Management and sets the corresponding meters for the Maximo Asset Management for Energy Optimization assets.

Configure the integration endpoint and external system for communication with Tivoli Monitoring for Energy Management:

1. In the End Points application, select the **PLUSEITMEP** endpoint.
2. On the End Point tab, specify the values for the URL, user name, and password for your Tivoli Monitoring for Energy Management system.
3. In the External Systems application, select the **PLUSEITMES** system.
4. On the System tab, specify the values for the URL, user name, and password for your Tivoli Monitoring for Energy Management system.
5. In the Cron Task Setup application, select the **PLUSEITMMETERS** cron task. An instance of the cron task is provided, as shown in Figure 5-10 on page 112.

6. Click Set Schedule and select the interval for running the cron task to load the meters, and then click **Save Cron Task Definition**.

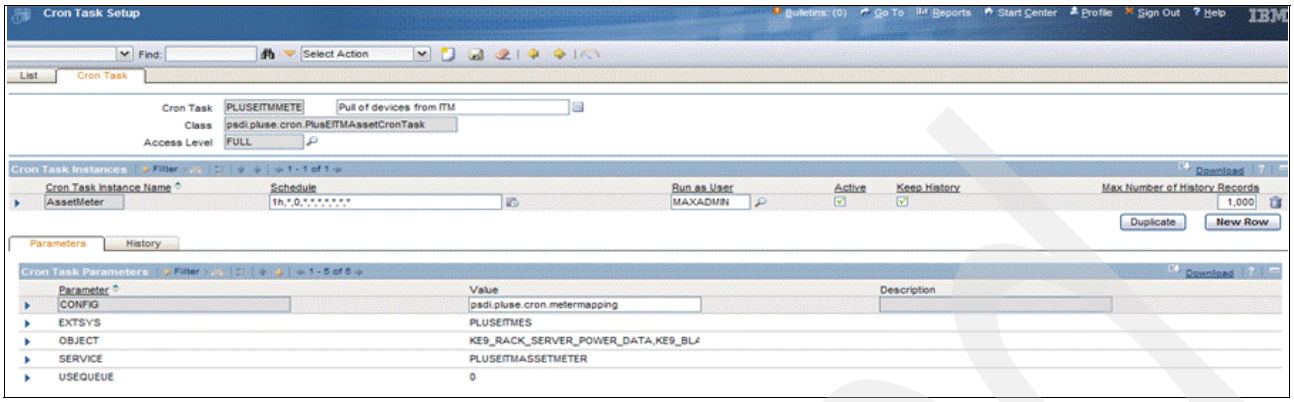


Figure 5-10 Maximo: Cron Task configuration

5.5.4 Loading meter information from Tivoli Monitoring for Energy Management into Maximo Asset Management for Energy Optimization

Maximo Asset Management for Energy Optimization obtains meter data from Tivoli Monitoring for Energy Management metrics.

Tivoli Monitoring for Energy Management does not have the “concept” of a meter name that relates to the individual metrics that are captured for a device. Meter information is obtained from the following types of Tivoli Monitoring for Energy Management objects:

- KE9_RACK_SERVER_POWER_DATA
- KE9_BLADECENTER_POWER_DATA
- KE9_PDU_POWER_DATA
- KE9_ALL_SENSOR_DATA

Metrics information from Tivoli Monitoring for Energy Management is mapped to the corresponding Maximo Asset Management for Energy Optimization meter name. For example, suppose the following information is pulled from Tivoli Monitoring for Energy Management for a KE9 Rack Server:

```
<Ambient_Temp_Avg dt="number">21.0</Ambient_Temp_Avg>
<Ambient_Temp_Max dt="number">21.0</Ambient_Temp_Max>
<Ambient_Temp_Min dt="number">21.0</Ambient_Temp_Min>
```

The metric to be recorded is Ambient_Temp_Avg. Therefore, the meter name that is created in Maximo Asset Management for Energy Optimization for that server is AMB_AVG_C.

The following types of meters are defined for Tivoli Monitoring for Energy Management:

AMB_AVG_C	Ambient Average Temperature (Celsius)
AMB_AVG_F	Ambient Average Temperature (Fahrenheit)
EXH_AVG_C	Exhaust Average Temperature (Celsius)
EXH_AVG_F	Exhaust Average Temperature (Fahrenheit)
AVG_PWR_AC	Average Power AC
AVG_PWR_DC	Average Power DC
HUMIDITY_R	Relative Humidity

Archive

5.5.5 Loading temperature metrics data from Tivoli Monitoring for Energy Management into Maximo Asset Management for Energy Optimization

You must load the metrics data into Maximo Asset Management for Energy Optimization. Configure and run the cron task to load temperature metrics data from Tivoli Monitoring for Energy Management into Maximo Asset Management for Energy Optimization.

Make sure that your assets exists and linked to the meters to which the metrics refer.

The PLUSEITMMETRICS cron task obtains metrics data from Tivoli Monitoring for Energy Management. Metrics information is obtained from the following types of Tivoli Monitoring for Energy Management objects:

- KE9_RACK_SERVER_POWER_DATA
- KE9_BLADECENTER_POWER_DATA
- KE9_PDU_POWER_DATA
- KE9_ALL_SENSOR_DATA

Configuring metrics data integration

Verify the configuration of the integration endpoint and external system for communication with Tivoli Monitoring for Energy Management:

1. In the End Points application, select the **PLUSEITMEP** endpoint.
2. On the End Point tab, ensure that the values for the URL, user name, and password correspond to your Tivoli Monitoring for Energy Management system.
3. In the External Systems application, select the **PLUSEITMES** system.
4. On the System tab, ensure that the values for the URL, user name, and password correspond to your Tivoli Monitoring for Energy Management system.
5. In the Cron Task Setup application, select the **PLUSEITMMETRICS** cron task, as shown in Figure 5-11
6. Click **Set Schedule** and select the interval for running the cron task to load the metrics, and then, click **Save Cron Task Definition**.

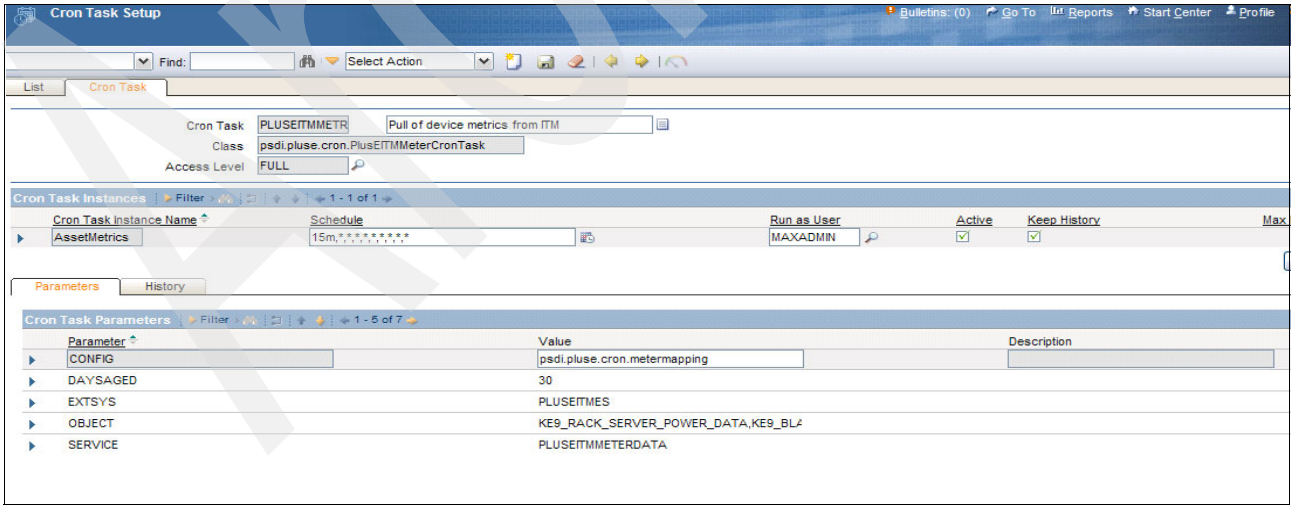


Figure 5-11 Maximo: Cron task configuration for metric data

5.5.6 Configuring data centers

Before you can use Maximo Asset Management for Energy Optimization, you must configure a data center location and its assets.

Configuring data center locations

To display the data center views, configure the data center location by setting its dimensions in the Locations application.

When you configure the dimension attributes for a location, Maximo Asset Management for Energy Optimization identifies that location as a data center. Perform the following steps:

1. In the Locations application, create a location or select the location that you want to set as a data center location.
2. Select the **Modify Data Center Properties** action.
3. In the Modify Data Center Properties window, select the unit of measure for the data center dimensions in the Layout Unit of Measure field. Ensure that the unit of measure that you selected has a conversion to the FEET unit (ft) of measure configured.
4. In the Meters application, select **Select Action** → **Unit of Measure and Conversion** → **Add/Modify Conversions**.

   If a conversion to FEET is not configured, create one by clicking **New Row**.
5. Specify the data center dimensions in the Length, Width, and Height fields. Regardless of the unit of measure that you select, the resolution for the length and width on the data center views is 1 foot (30.48 cm). The resolution for height is 0.5 foot (15.24 cm).
6. Select the unit of measure for the temperature sensors in the Sensor Unit of Measure field. This unit of measure is used only for manual temperature sensors
7. Click **Save Location**.

Optional: The Absolute Color Scale check box enables you to set the upper and lower temperature boundaries of the heat map. By default, the colors are relative to the maximum and minimum temperature readings in the data center. If this check box is selected, the `mxe.meo.maxtemp` and `mxe.meo.mintemp` system properties specify the maximum and minimum temperatures.

Configuring data center assets

Before you can display assets in the data center view, you must assign the assets to a data center location. You must also set the attributes that describe the dimensions and the position of the asset in the data center.

Note: The location of the asset that you want to configure must be a data center location.

The configuration of a data center involves two main steps:

- Surveying the data center.
- Configuring asset hierarchies.

Surveying the data center

Surveying the data center involves the following tasks:

1. A template worksheet is provided to help with the physical collection of data center asset information that needs to be entered into Asset Management for Energy Optimization (MEO). You can find this worksheet in the following location:

   `meo_install_dir\tools\maximo\en\xcenter\Data center Layout Worksheet.xls.`

2. Pulling device information from IBM Tivoli Monitoring for Energy Management using the provided Tivoli Integration Composer adapter.
3. Promotion and reconciliation of the devices from Tivoli Monitoring for Energy Management to Maximo Assets.

Besides IT assets, you can use the provided asset types to configure other elements in your data center. You can configure walls, furniture, and perforated floor tiles, and assets that are not monitored by software, such as racks and air conditioners. The unit of measure for the coordinates and dimensions is the same that is defined for the data center location.

Note: If you try to change the coordinates or dimensions of an asset while the meter readings for that asset are being updated, an "update by another user" message is indicated. To minimize this problem, stop the PLUSEMETRICS **`cron`** task instances during data center setup.

To proceed with this step, select the asset that you want to configure in the Assets application, or create an asset:

1. In the Data Center Details section of the Asset tab, select a data center type for the asset. Data center types define the colors and shapes of the assets in the layout view.
2. Specify the coordinates of the position of the asset in the X, Y, and Z fields. The origin for the coordinates is the lower left corner of the layout view.
3. Specify the dimensions of the asset in the Width, Length, and Height fields.
4. Select the direction of the front of the asset in the view.
5. For rack-mounted assets, enter the slot position of the asset. Optionally, select the group to which the asset belongs.
6. Click **Save Asset**.

Configuring asset hierarchies

Configure assets in the data center that are related, such as a rack and the servers installed in that rack, in a single operation.

Maximo Asset Management for Energy Optimization identifies asset hierarchies by the X and Y coordinates of the assets. The data center layout view does not recognize any parent-child relationships that are configured for the assets. Therefore, you can configure assets that are physically located one above the other but that do not share a parent asset. For example, you can configure servers that are installed on shelves.

When you configure racks and the servers that are mounted on those racks, enter the same X and Y coordinates for the rack and all the servers.

You can set the coordinates and dimensions of multiple data center assets by using the Modify Data Center Assets action:

1. In the Locations application, select the data center location for which you want to set the asset attributes, and then select the Modify Data Center Assets action.
2. In the Modify Data Center Assets window, perform the following steps:
   a. Select the assets that you want to modify.
   b. In the fields at the bottom of the window, specify the values for the coordinates and any other attributes that you want to set.
   c. Click **Apply New Values**. Attributes that are blank are not changed.
   d. Repeat the steps for other groups of assets for which you want to set common attributes.
3. Click **Save**.

5.5.7 Using data center graphical views

You can monitor your data center and resolve energy issues from the Data Center tab in the Assets and Locations applications. When you select a data center location in the Locations application, or when you select an asset that is assigned to a data center location in the Assets application, the Data Center tab is added to the application window. It has the following information:

- The top part of the Data Center tab shows the data center location name, description, and site. You can change the description by clicking the icon next to it.
- The middle part shows the data center views. The layout view is shown by default. To display the heat view, select **Heat** on the drop-down list above the views.

When both views, Layout and Heat, are selected, they are shown superimposed. The actions that are available when you click anywhere inside the view depend on which view is in the foreground. You can switch the views by clicking the view.

Note: To remove any of the views, select it in the list, and then click it.

The graphical views always show the measures in feet. If you select a different unit of measure for your data center, this unit of measure is converted to feet for display.

Layout view

The Layout view shows the data center layout with all the assets that are assigned to the data center location and that have their coordinate and dimension attributes set.

All X and Y coordinate values that are entered for assets and locations are converted to feet and displayed on a grid of tiles that are fixed at two feet by two feet. All asset size values (width, length, and height) are also converted to feet.

The colors and shapes of the assets depend on their data center types. If you select Legend in the left list on the bottom area, a list of all data center types opens. By selecting or clearing the check boxes in the visible column, you can choose which types of assets are shown in the layout view.

If you select an asset type in the center list and then select Elements in the left list on the bottom area, a list opens of all assets of that type that are configured in this data center.

If you click an asset in the layout view, a list of all assets with the same type of the selected asset opens, with the asset that you selected highlighted.

If the point where you click has a rack and its servers configured, the rack is highlighted. To view all assets in the selected point, select **Retrieve Assets for Selected Map Location** in the right-most list at the bottom part of the tab, then click **Go**. A window opens showing all assets at the point selected. Click the arrow next to an asset in the list to switch to the Assets application with that asset selected.

With an asset selected on the layout view, you can also choose the following actions on the right-most list:

- Show Meter Readings, to view the meter reading for the asset
- Create work order for selected asset, to create a work order pre-filled with the asset name and location, and with the following description: Generated for Meter Readings (Click **Long Description**).
- **Go to** → **Deployed Assets**, to switch to the appropriate Deployed Assets application (Computer or Network Device, depending on the type of asset), with that asset selected.

Heat view

This view shows a heat map of the data center which is an interpolation of the most recent temperature sensor readings.

The heat map update period, in minutes, is specified by the `mex.meo.heatrefresh` system property. You can also update the map manually by clicking. The sensor readings are obtained fromTivoli Monitoring for Energy Management by a **`cron`** task. The heat map changes only if the **`cron`** task was executed since the last time the map was updated. Set the `mex.meo.heatrefresh` property to a value synchronized to that of the frequency of the metrics load **`cron`** task.

The colors in the map can be relative to a fixed range of temperatures, or to the last minimum and maximum temperature readings. This behavior is determined by the setting of the Absolute Color Scale check box in the data center location properties.

You can view the heat map for various heights by selecting the height in the list above the map.

When the heat view is in focus, the bottom part of the Data Center tab shows a vertical heat slice view. This view consists of the following elements:

- A vertical section of the heat map of the data center, taken along the X (width) axis

  Select the Y (length) coordinate for the cut by dragging the horizontal cross-hair line on the top heat map view.
- A vertical line showing the temperature readings at the selected X coordinate

  Select the coordinate by dragging the vertical cross-hair line on the top heat map view.
- A solid line showing the temperature graphic for the height selected
- A dotted line showing the height selected

Note: To prevent displaying temperature readings that have not been updated for a long time, set the `mxe.com.heattime` system property. This property defines the maximum time, in minutes, that meter readings are used for building the heat map. Any readings taken more than `mxe.com.heattime` minutes ago are not considered for building the heat map.

Archived

Configuring manual temperature sensors

To improve the accuracy of the heat map view, you can configure external sensors and enter their readings to be displayed in the heat view.

Consider providing manual sensor readings in locations where there are not enough sensor readings (such as data center perimeters). Providing manual sensors improves the overall appearance of the heat map.

Perform the following steps:

1. In the Locations application, select the location for which you want to include sensor information, and then select the **Modify Data Center Properties** action.
2. Select the sensor for which you want to add or change information, or add a new sensor, and then enter the coordinates of the location of the sensor. The origin of the coordinates is the lower left corner of the data center view.
3. Enter the temperature reading for the sensor. The unit of measure for all sensors is that set for the data center location.
4. Click **OK**.

For more product-related help information, go to the following web address:

https://www.ibm.com/developerworks/wikis/display/tivolidoccentral/IBM+Maximo+Asset+Management+for+Energy+Optimization

5.6 Architectural diagram of integration

Figure 5-12 shows the products used and gives the architectural overview of the components.

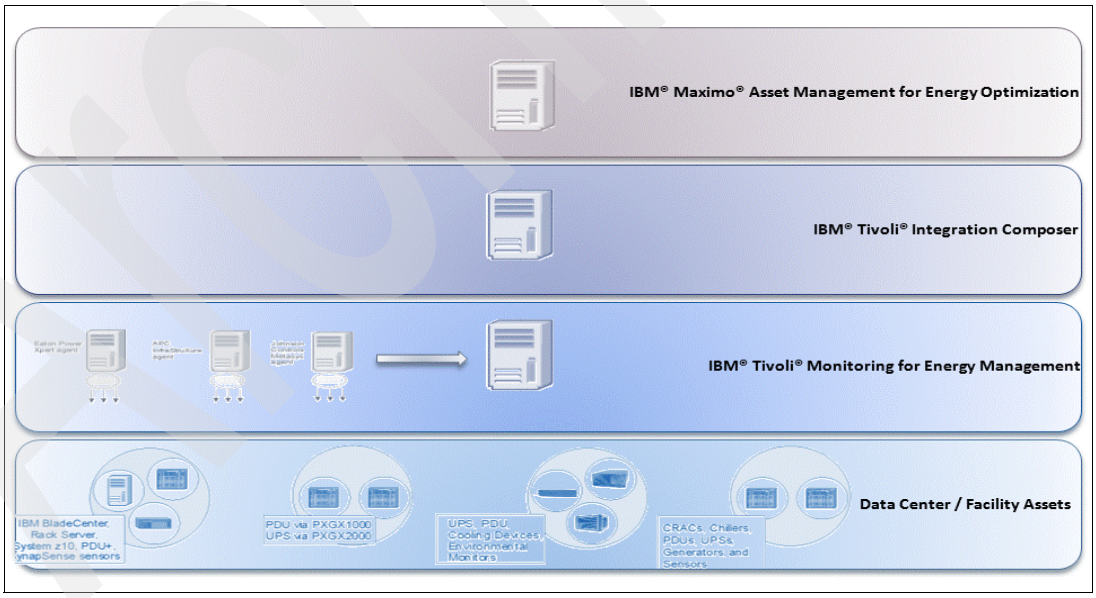


Figure 5-12 Data Center product overview

Figure 5-13 explains the components that are used in the specified products to configure the described scenario.

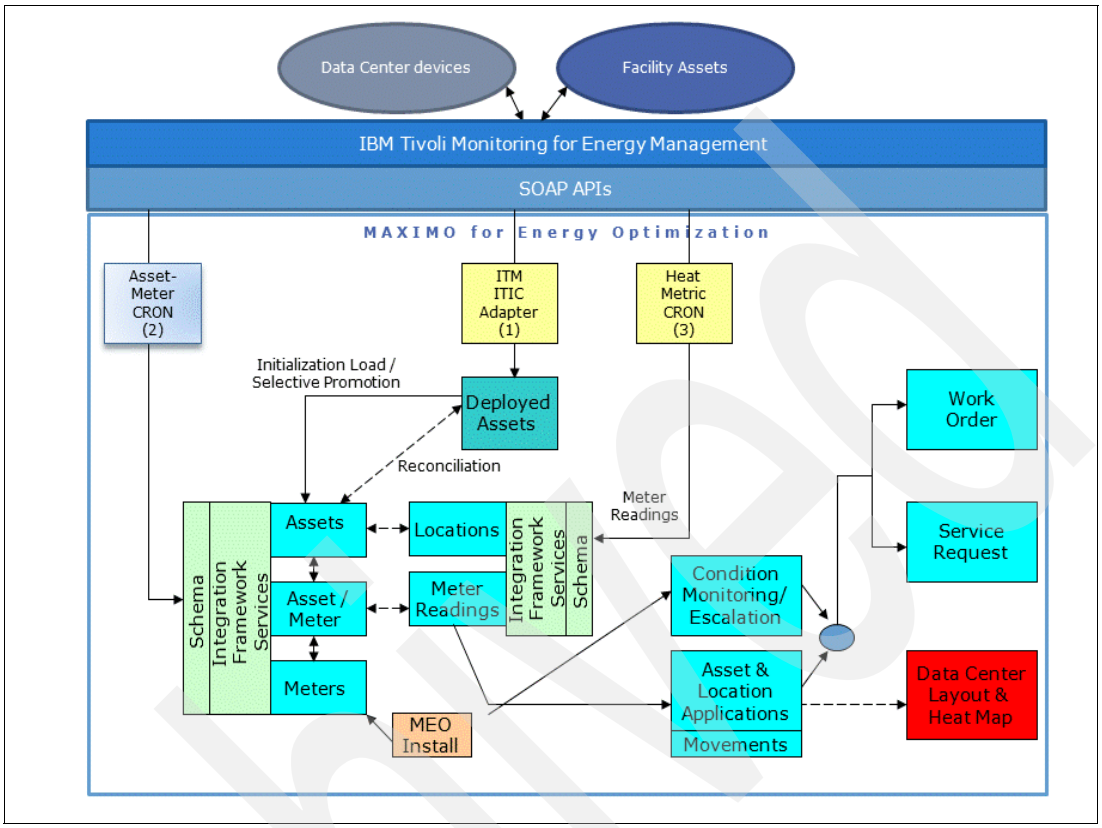


Figure 5-13 Component overview: Maximo for Energy Optimization

Archived

5.7 Scenario walk-through

Increasingly efficient, high-density computers provide an opportunity to get more value from every square foot of space in your data center.

The scenario walk-through has several examples of how you can achieve your goal with a focus on energy efficiency in an overall Green IT context. Figure 5-14 shows the visualization of the temperature as a main instrument for visualization.

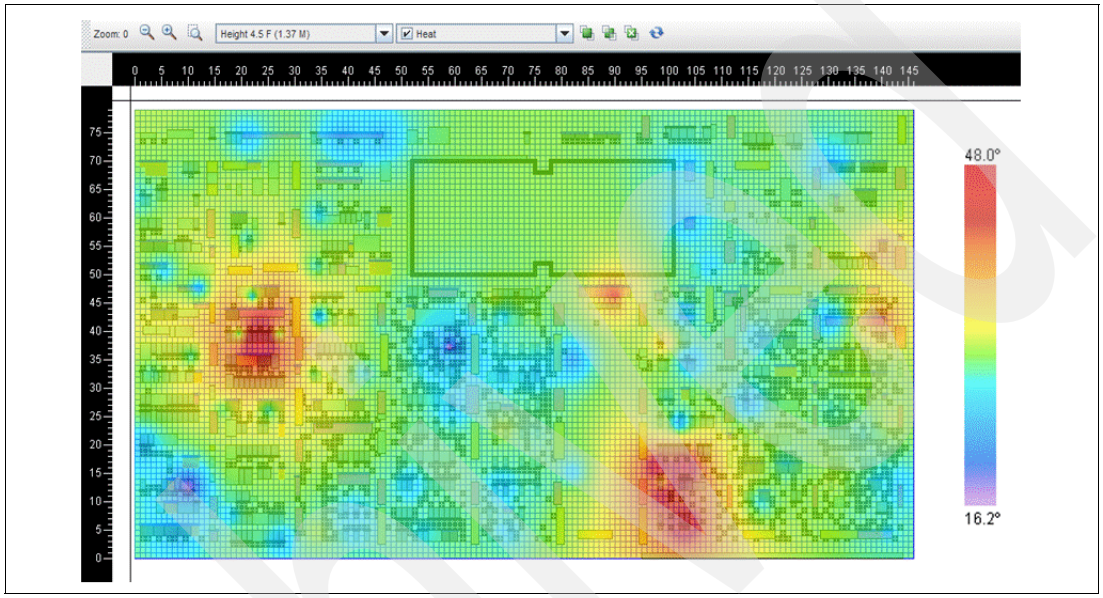


Figure 5-14 Maximo heat map: data center overview

For the walk-through, we assume that products (listed in 5.2, “Products involved” on page 104) are in place and are configured (as described in 5.4, “High-level implementation steps” on page 106).

5.7.1 New assets from strategy to ready

Daily, you receive new requests for new service, new servers, and new assets to be placed in your data center. The Maximo heat map can help you in several ways:

- Where to place your assets

  The heat map visualizes the heat in your data center. With this information, you can balance the temperature by selecting the right place for your new assets.

- Where to move your assets

  Because you have several servers and further assets in your heat map, you can identify warmer and colder areas of your data center. The heat map visualizes this information, helping you to move your assets accordingly.

These two typical scenarios are used to explain the required implementation steps from an end-to-end perspective.

Because you have the products installed and configured, you get all your requests (install, move, change, delete) for new assets through the Maximo service order. According to your service, several service work orders are created to cover your job plans for installing new assets and updating asset information in your data center. To find the correct place for the

new servers, the job plan includes a task to select the appropriate location from the heat map of Maximo Asset Management for Energy Optimization. This technique ensures that the server is placed into the correct area, from an energy perspective. See Figure 5-15.

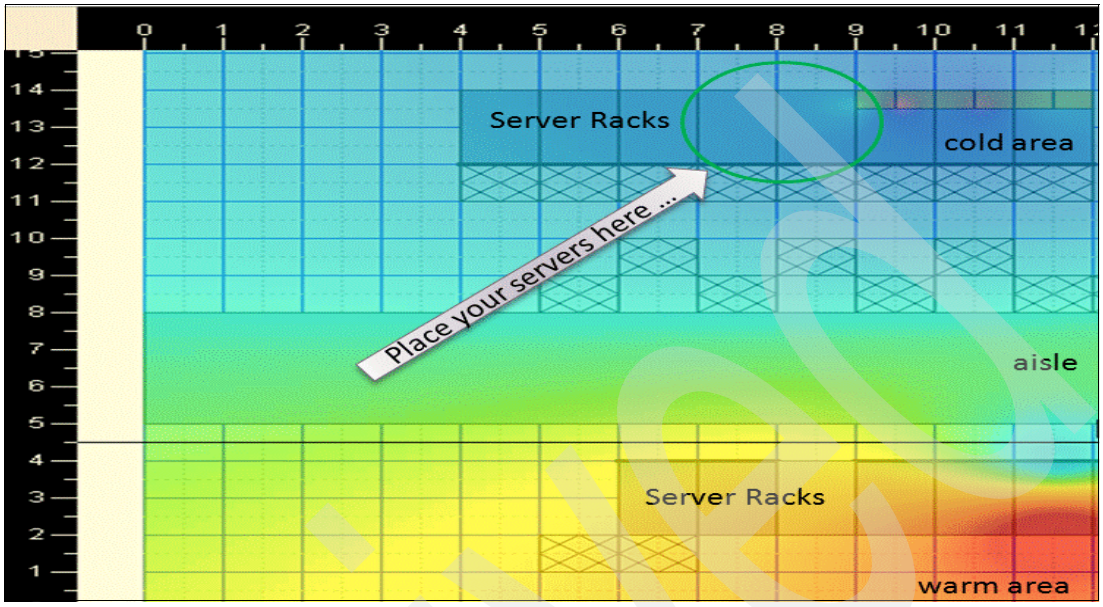


Figure 5-15 Maximo Asset Management for Energy Optimization: heat map

The same heat map can be used to relocate your assets within the data center. The process starts, based on your heat monitoring, and the heat map can help you to identify heat bottlenecks. Figure 5-16 shows a bottleneck identified. Warm area (red color) is the bottleneck.

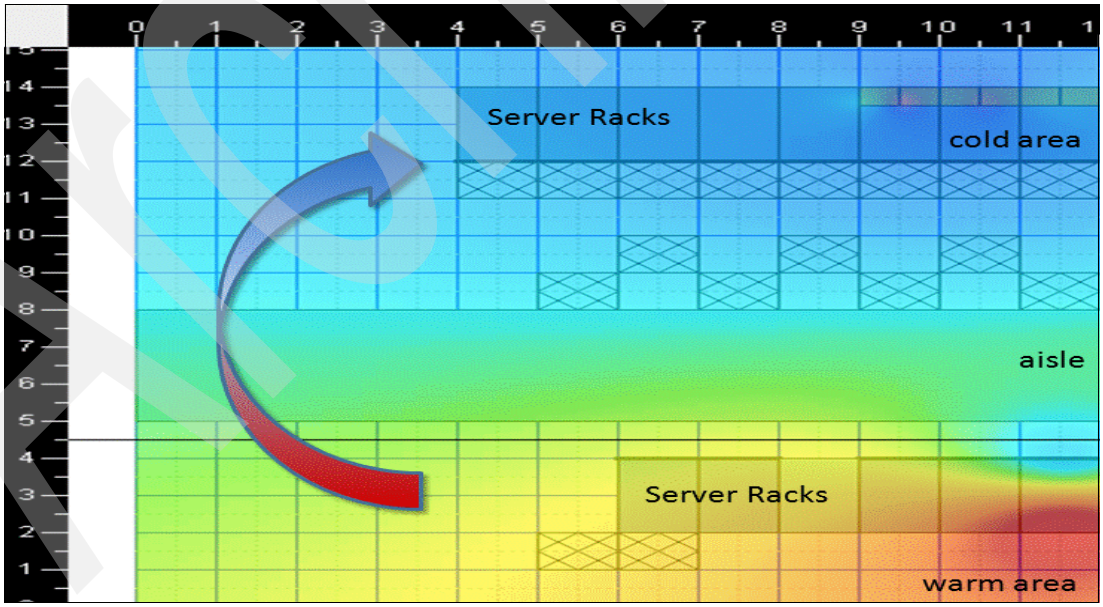


Figure 5-16 Asset relocation

The process for *moving* the asset is applied through a service order to create the required service work orders, and runs similar to the described process for *placing*.

To apply the scenario to a virtualized environment, the Maximo Asset Management for Energy Optimization heat visualization can help to place your virtual instances according to the needed power and cooling data. The heat map supports you to start the placing process and to give you the daily data to balance your heat production in comparison with the cooling capacity.

5.7.2 Event Management based on Energy Management

Fast reaction and event management is crucial to your environment's health index to deliver your service, fulfilling the needed service level. Energy and especially heat data measured with sensors in your data center can help you to have short reaction time between asset related interruptions and starting your solution process.

Combine heat and asset information

If you discover an unexpected increase of temperature in the heat map, you may select the appropriate area, and the assets location are shown. Condition monitoring of assets notifies you when conditions go outside the defined metric levels. Figure 5-17 shows the drill-down options from heat to asset.

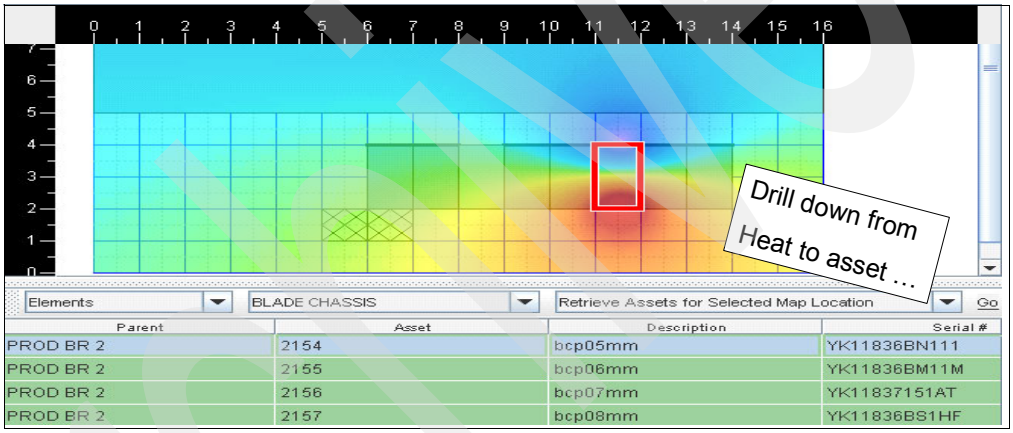


Figure 5-17 Maximo heat map drill down

Because the information has been processed in your event and incident processes, the process chain starts, for example to schedule a technician visit or a maintenace plan.

Use of historical data

Because the thermal data is linked to the asset data, reports can be created to present several areas. Figure 5-18 on page 123 shows a BIRT report about asset power consumption.

Archived

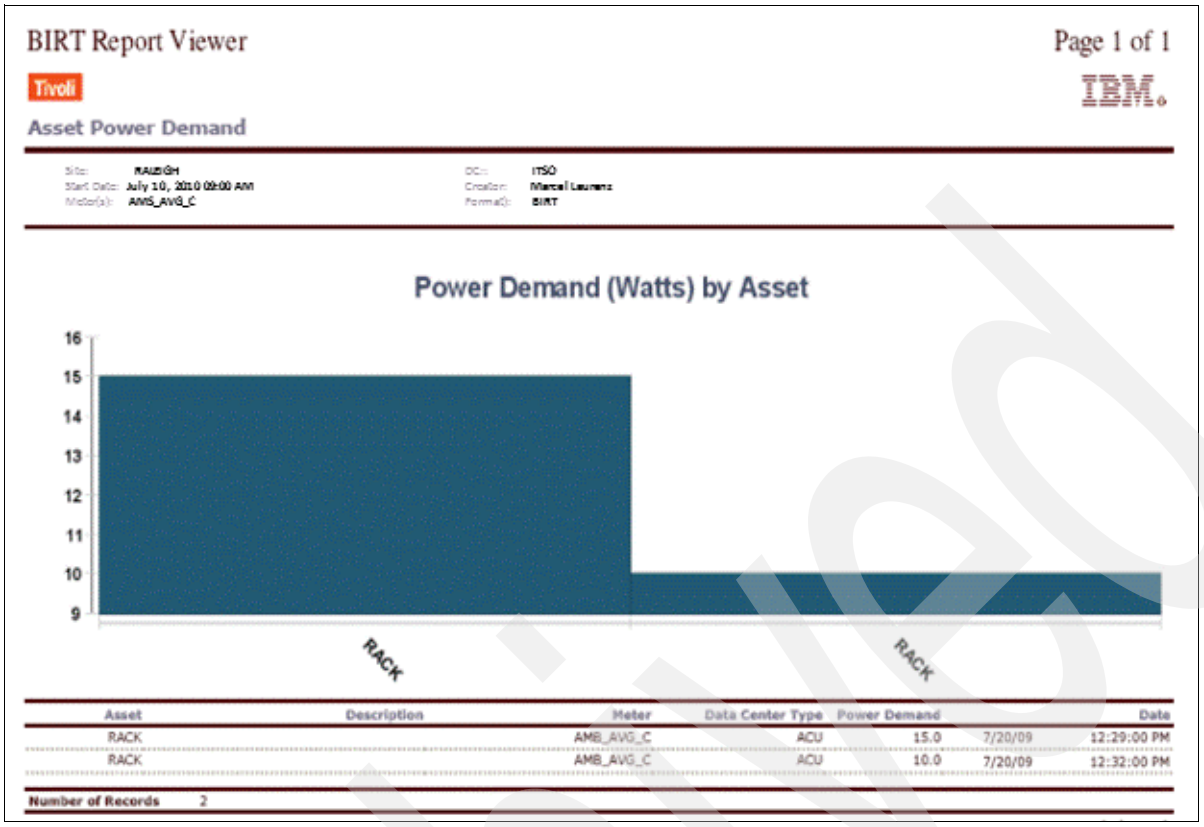


Figure 5-18 BIRT report: power demand by asset

Archived

5.8 Summary

Figure 5-19 shows the quick summary of the solution.

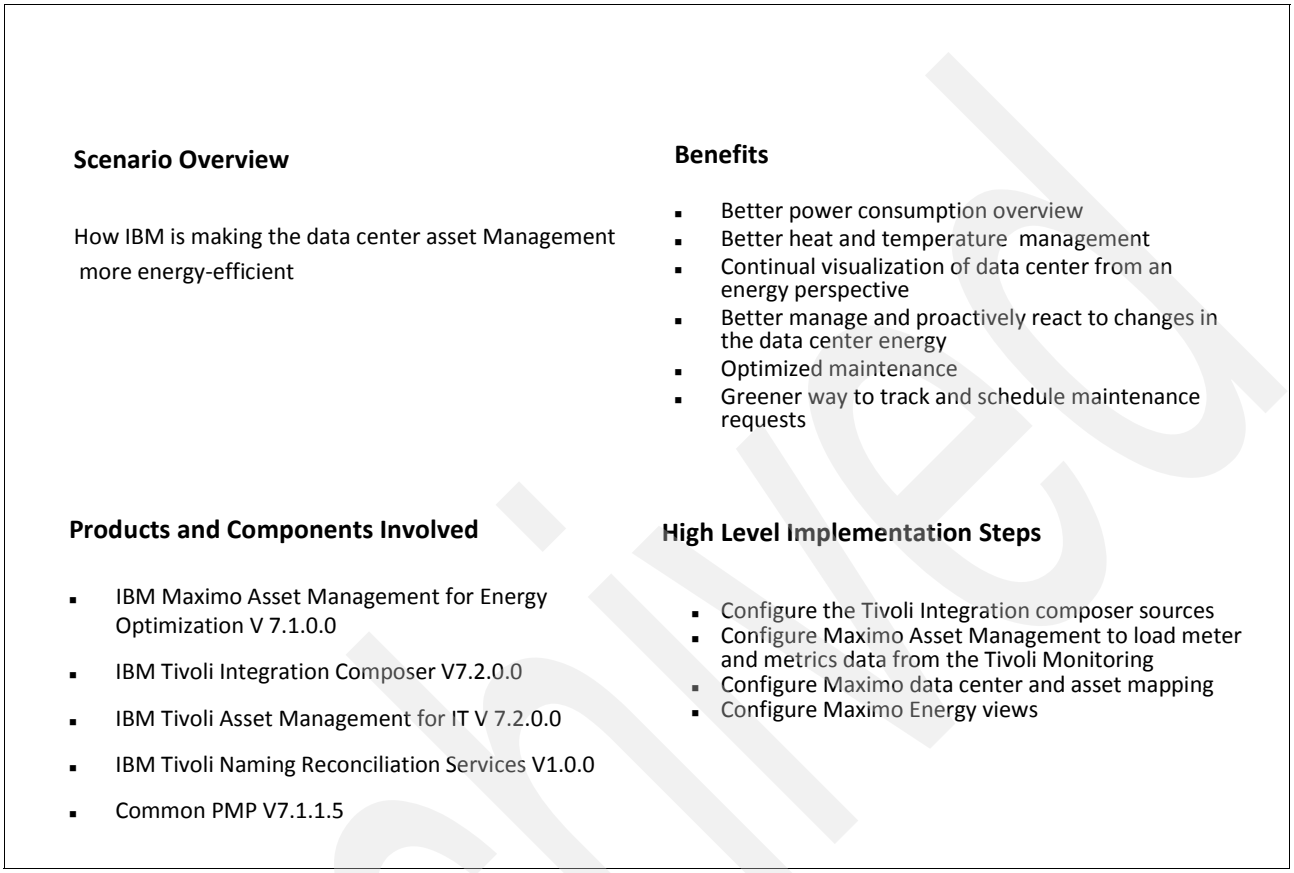


Figure 5-19 Quick summary

In this chapter, we described how you can configure Maximo Asset Management for Energy Optimization and IBM Tivoli Monitoring for Energy Management to obtain asset and sensor information from your data center. This information is used to build graphical views of the data center layout and heat map. You can use these views to monitor the conditions in your data center and to identify potential problems.

In the scenario walk-through section, we showed you two common use cases of this graphical view:

- Where to place your assets: Choosing the right place for your new assets to balance your temperature.
- Where to move your assets: Relocating your assets inside the data center taking into consideration warmer and colder areas of your data center.

Archived

6

Tivoli integration scenario for Virtual Infrastructure

This chapter describes the use of Tivoli products to create a fully automated business service. It provides the means for the IT department to have a faster deployment process for new virtual servers and their monitoring. This chapter does acknowledge the approval part of the change-management process and requires several manual actions, although you may further automate these, also.

This chapter contains the following topics:

- 6.1, “Scenario overview” on page 126
- 6.2, “Products involved” on page 126
- 6.3, “Benefits” on page 126
- 6.4, “Integration architecture” on page 127
- 6.5, “Implementation steps” on page 128
- 6.6, “Scenario walk-through” on page 136
- 6.7, “Summary” on page 165

© Copyright IBM Corp. 2011. All rights reserved.

6.1 Scenario overview

This scenario demonstrates the use of Tivoli products that are listed (in 6.2, “Products involved” on page 126).

We use the following actions to demonstrate the scenario:

- “Creating a service in TADDM that contains a Virtual Center application” on page 136
- “Checking for existence of service in Tivoli Business Service Manager” on page 148
- “Showing launch to IBM Tivoli Monitoring and Tivoli Application Dependency Discovery Manager (preconfigured functionality)” on page 150
- “Creating new virtual machine on the Virtual Center” on page 152
- “Auto-discovering the virtual machine by using TADDM rediscovery” on page 153
- “Accepting (adding) virtual machine to existing Virtual Center Application” on page 157
- “Verifying that virtual machine connects to IBM Tivoli Monitoring server” on page 159
- “Verifying this virtual machine appears in Tivoli Business Service Manager” on page 162

6.2 Products involved

Products and product components involved in this scenario are as follows:

- IBM Tivoli Netcool/OMNIbus 7.3
- IBM Tivoli Application and Dependency Manager 7.2
- IBM Tivoli Monitoring for Virtual Servers 6.2.2
- IBM Tivoli Business Service Manager 4.2.1
- LDAP server on Microsoft Active Directory

In addition, you must have a VMware Virtual Infrastructure for this scenario.

Tivoli Common Reporting integration: Using Tivoli Common Reporting in this integration scenario is also possible. For detailed information, see “*Tivoli Monitoring Virtual Servers - VMware VI Agent reports*” at the following web address:

`http://www.ibm.com/software/brandcatalog/ismlibrary/details?catalog.label=1TW10TM7I`

6.3 Benefits

This scenario has the following benefits:

- Faster deployment process for new virtual servers and their monitoring
- Increase in availability of business processes and prevention of errors because of highly integrated automation
- Faster resolution of errors because of configuration changes
- Rapid ticketing either by the Operator tool from the Tivoli Netcool/OMNIbus desktop or through automated action

Archived

Notes:

- Generally, 80% of faults are caused by change. In our scenario, TADDM forwards events about configuration to Tivoli Netcool/OMNIbus where they may be correlated with other faults to assist in identifying root cause and accelerating the time to resolve. Launch-in-context tools from the IBM Tivoli Netcool/OMNIbus desktop allow the operator to view item details and change history to further assist in problem diagnosis and resolution.
- Rapid ticketing either by Operator tool from the Tivoli Netcool/OMNIbus desktop or also through automated action is a significant benefit in service assurance. Tivoli Netcool/OMNIbus has a simple to deploy bidirectional integration with Tivoli Service Request Manager for raising trouble tickets and subsequent status change. See Chapter 16, “Taking automatic actions based on predefined policies” on page 415 for a full discussion of this capability.

6.4 Integration architecture

Figure 6-1 shows the overall integration architecture for this scenario.

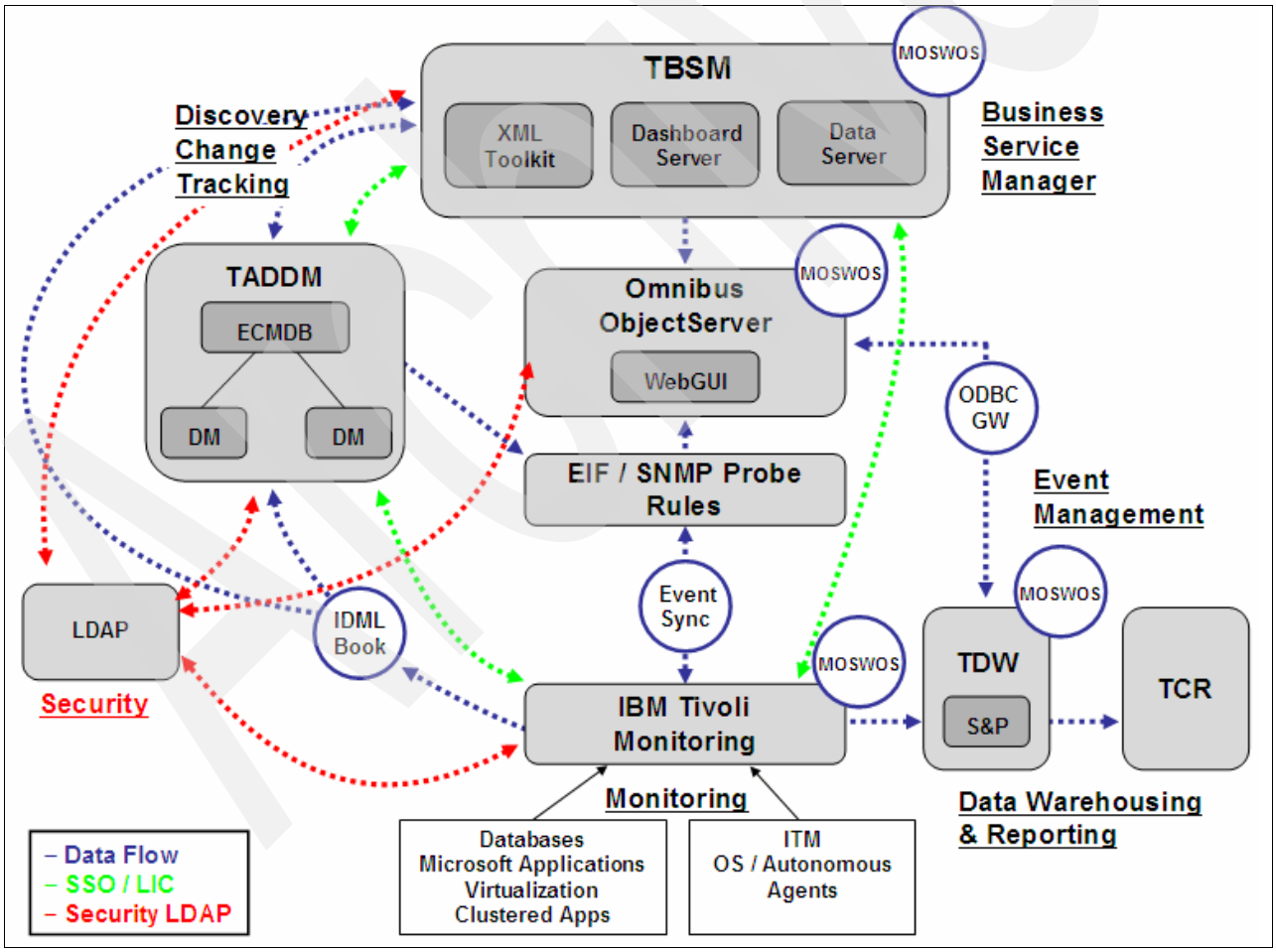


Figure 6-1 Architecture overview

6.5 Implementation steps

In this section, we describe the implementation steps of this scenario.

6.5.1 IBM Tivoli Monitoring for Virtual Servers

The IBM Tivoli Monitoring product monitors and manages system applications on a variety of operating systems, tracks the availability and performance of your enterprise system, providing reports to track trends and troubleshoot. IBM Tivoli Monitoring for Virtual Servers is an extension to the IBM Tivoli Monitoring family; it centrally monitors server virtualization, consolidation, resource performance and availability at the enterprise level.

See the following resources:

- Instructions for how to install IBM Tivoli Monitoring environment are in the *IBM Tivoli Monitoring Installation and Setup Guide*:

  .http://publib.boulder.ibm.com/infocenter/tivihelp/v15r1/index.jsp?topic=/com.ibm.itm.doc_6.2.2/welcome.htm

- See section 3.3 in *Integrating Tivoli Products*, SG24-7757 for important configuration information.
- The installation of IBM Tivoli Monitoring for Virtual Servers, follows the same process as *IBM Tivoli Monitoring Installation and Setup Guide.*

Archived

After completing all installation steps, open the Tivoli Desktop Console. You see a window that is similar to the one in Figure 6-2.

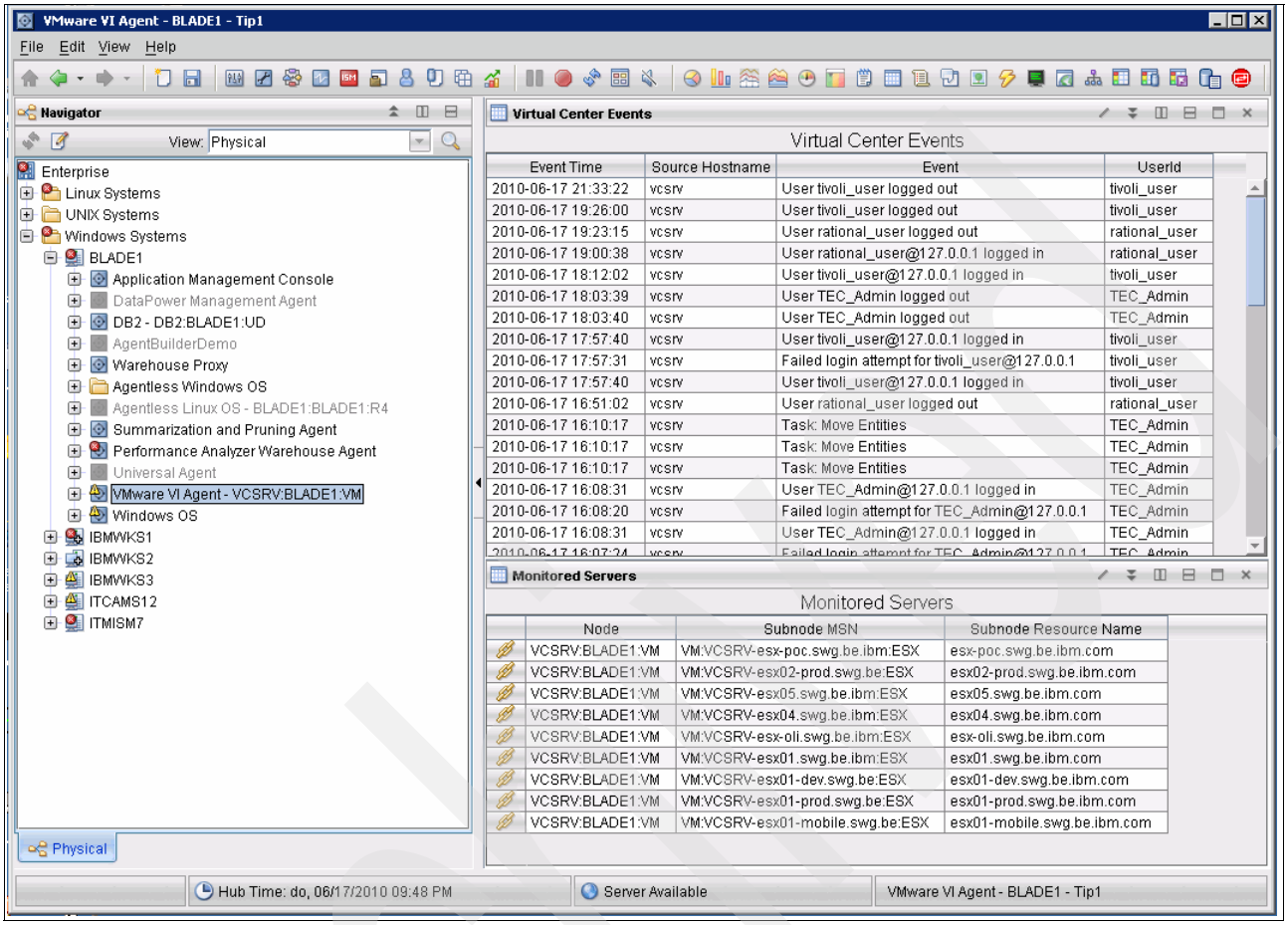


Figure 6-2 Example of IBM Tivoli Monitoring for Virtual Server agent

6.5.2 IBM Tivoli Monitoring security integration

Figure 6-3 shows the IBM Tivoli Monitoring security integration for this scenario.

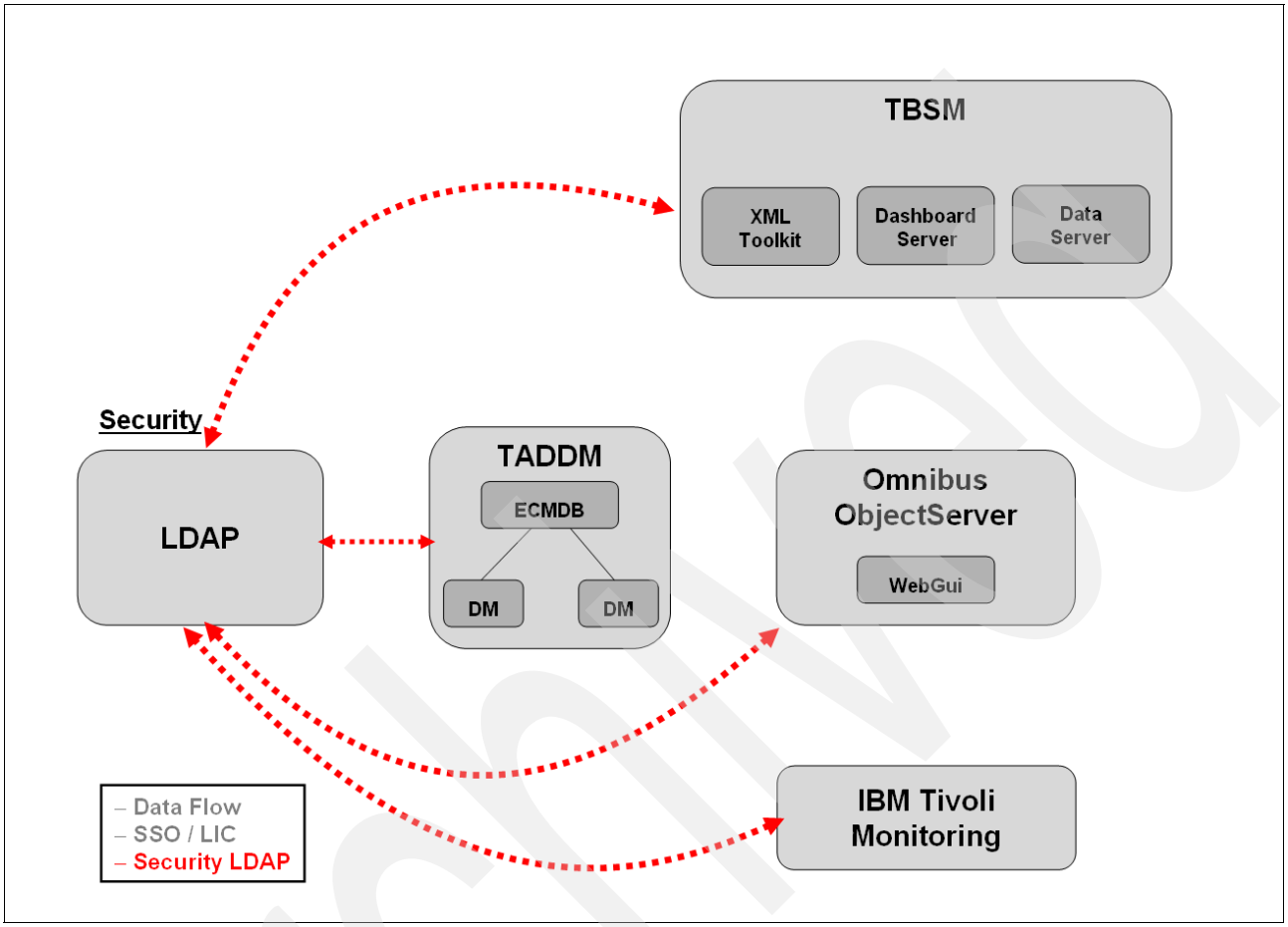


Figure 6-3 Security architecture

Note: Another possibility is to bring alerts into Tivoli Netcool/OMNIbus and TADDM from an external source by using any of the available probes and monitors to poll in external data. There is no difference in terms of security settings configurations described here.

IBM Tivoli Monitoring supports external authentication by the Hub Tivoli Enterprise Monitoring Server and Tivoli Enterprise Portal Server. The Tivoli Enterprise Portal Server can be configured to authenticate users in a Lightweight Directory Access Protocol (LDAP) repository in the Federated Repositories of the WebSphere Application Server. This support permits you to share user authentication information among IBM Tivoli Monitoring and other products.

Also supported is single sign-on (SSO) capability between IBM Tivoli Monitoring and other Tivoli applications. This support allows you to launch other web-based Tivoli applications from the Tivoli Enterprise Portal and to launch Tivoli Enterprise Portal from other applications without reentering the login credentials.

The SSO solution requires Lightweight Third Party Authentication (LTPA) tokens or keys to be exported and imported between the applications. Participating SSO applications pass LTPA

tokens using browser cookies. These tokens are encrypted and signed so that they cannot be decoded without the correct keys.

Information about how configure LDAP, SSO, and import and export LTPA tokens or keys, is in section 4.7 in *Integrating Tivoli Products*, SG24-7757.

Important: During the security configuration of IBM Tivoli Monitoring using Active Directory for LDAP authentication, we found that the following user filter in the Hub Tivoli Enterprise Monitoring Server configuration is required. In the **Enter required LDAP user filter** field, enter the following text (see Figure 6-4):

```
(&(cn=%v)(objectclass=user))
```

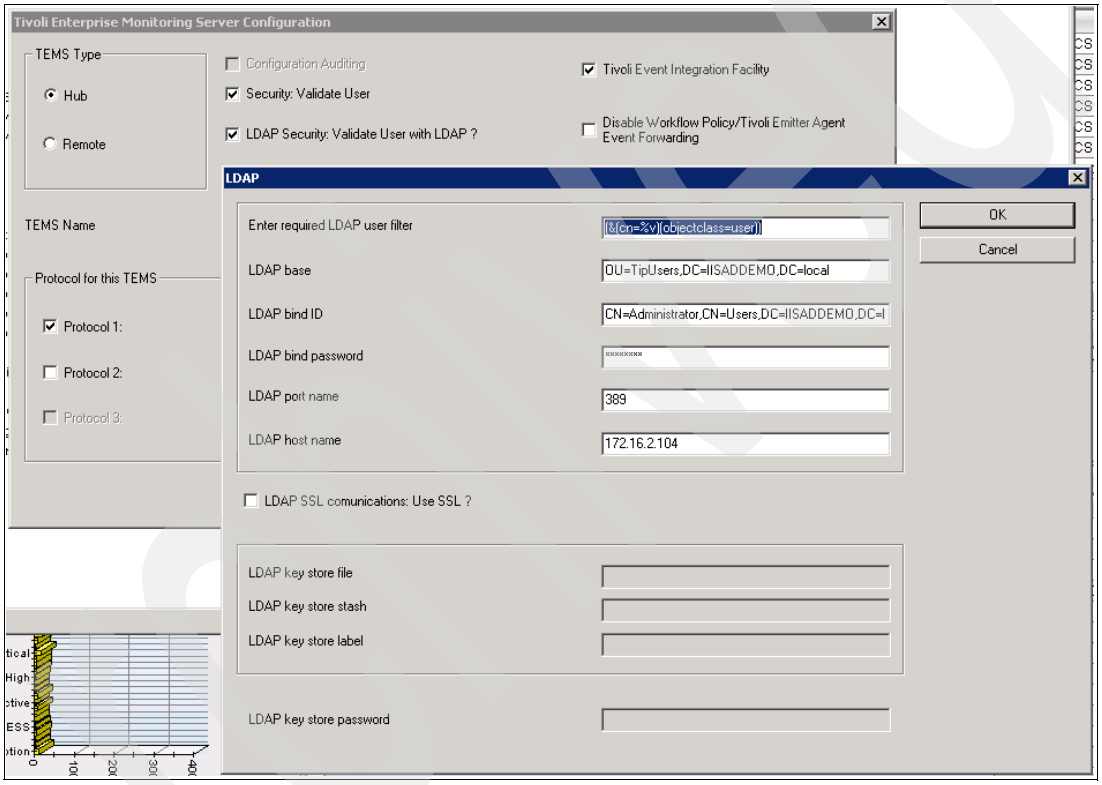


Figure 6-4 Filter configuration

Note: After you have completed the LDAP configuration and if only the wasadmin or local users are shown in Manager User, you must check the following file and include the information such as the one shown in Example 6-1 and in Example 6-2:

```
C:\IBM\ITM\CNPSJ\profiles\ITMProfile\config\cells\ITMCell\wim\config\wimconfig.xml
```

This information is documented at the following web address:

http://www.ibm.com/support/docview.wss?uid=swg21418276

Figure 6-5 on page 132 shows the Integration Solutions Console console with only the wasadmin user configured in LDAP.

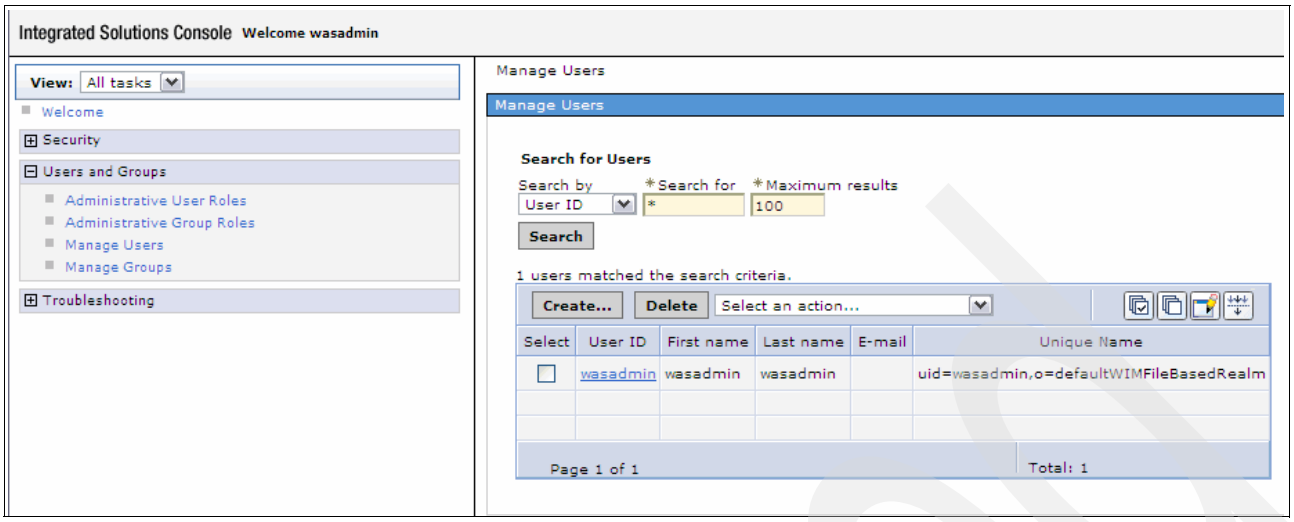


Figure 6-5 LDAP with only wasadmin user

Example 6-1 is a transcription of original `wimconfig.xml` configuration file.

Example 6-1 Original wimconfig.xml configuration file

```
<config:memberAttributes name="member" objectClass="group" scope="direct"/>
      </config:groupConfiguration>
      <config:attributeConfiguration>
        <config:attributes defaultAttribute="cn" name="cn">
          <config:entityTypes>Group</config:entityTypes>
        </config:attributes>
        <config:attributes defaultValue="8" name="groupType">
          <config:entityTypes>Group</config:entityTypes>
        </config:attributes>
        <config:attributes name="unicodePwd" propertyName="password" syntax="unicodePwd"/>
```

Example 6-2 is the new `wimconfig.xml` configuration file.

Example 6-2 New wimconfig.xml configuration file

```
<config:memberAttributes name="member" objectClass="group" scope="direct"/>
      </config:groupConfiguration>
      <config:attributeConfiguration>
        <config:attributes defaultValue="544" name="userAccountControl">
          <config:entityTypes>PersonAccount</config:entityTypes>
        </config:attributes>
        <config:attributes name="samAccountName" propertyName="uid">
          <config:entityTypes>PersonAccount</config:entityTypes>
        </config:attributes>
        <config:attributes defaultAttribute="cn" name="samAccountName">
          <config:entityTypes>Group</config:entityTypes>
        </config:attributes>
        <config:attributes defaultValue="8" name="groupType">
          <config:entityTypes>Group</config:entityTypes>
        </config:attributes>
        <config:attributes name="unicodePwd" propertyName="password"
syntax="unicodePwd"/>
```

Archive

Turning on the launch-in-context function of IBM Tivoli Monitoring is documented in section 6.3.3 of *Integrating Tivoli Products*, SG24-7757.

Figure 6-6 shows the data integration architecture for our scenario.

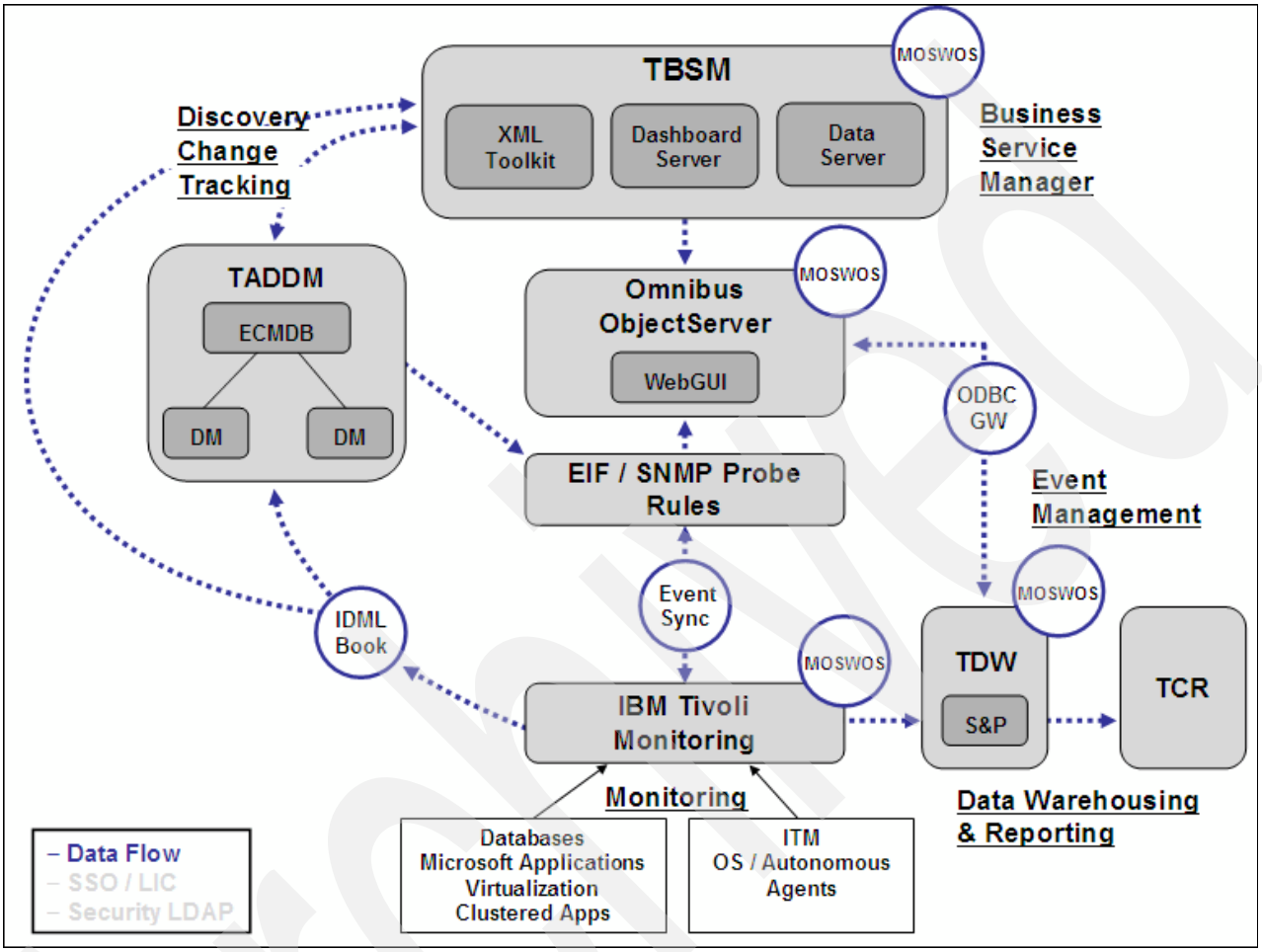


Figure 6-6 Data integration architecture

Why fault management is important: The latest IBM Tivoli Monitoring release and fix packs enhance the bidirectional integration of IBM Tivoli Monitoring and IIBM Tivoli Netcool/OMNIbus for situation-forwarding in the context of fault management.

The last release of IBM Tivoli Netcool/OMNIbus V7.3 contains, in the extensions directory, configurations for correlation of VMWare hypervisor monitor events from IBM Tivoli Monitoring with virtual server events that are collected by IBM Tivoli Netcool/OMNIbus. The provided configurations and correlations include patented techniques supporting that correlation and the launch-in-context tools to assist operator investigation and resolution.

IBM Tivoli Netcool/OMNIbus provides essential infrastructure within IBM Tivoli Business Service Manager for Service Management. Separately entitled for fault management, it provides the essential foundation of service assurance. Without solid fault management, prioritization, and resolution, the upper levels of Service Management bring little benefit to an organization.

6.5.3 Tivoli Netcool/OMNIbus

Perform the following steps for Tivoli Netcool/OMNIbus:

1. For the Tivoli Netcool/OMNIbus installation, see the standard installation guide:

   http://publib.boulder.ibm.com/infocenter/tivihelp/v8r1/index.jsp?topic=/com.ibm.netcool_OMNIbus.doc_7.3.0/omnibus/wip/install/concept/omn_ins_installingupgradinguninstallingunix.html

2. Optional: You can configure the Tivoli Netcool/OMNIbus server to also use LDAP. See section 4.6 in *Integrating Tivoli Products*, SG24-7757.

3. Install the Tivoli Event Integration Facility (EIF) probe as described in section 5.3.1 of the *IBM Tivoli Netcool/OMNIbus Installation and Deployment Guide*, SC23-6370.

4. Configure the EIF probe for Tivoli Applications Dependency Discovery Manager (TADDM) events as documented in the Tivoli Netcool/OMNIbus guide:

   http://publib.boulder.ibm.com/infocenter/tivihelp/v8r1/index.jsp?topic=/com.ibm.netcool_OMNIbus.doc_7.3.0/omnibus/wip/install/concept/omn_con_ext_enablingtaddmevents.html

5. Configure the EIF probe to perform event management for a virtual environment (itmvirtualization), as documented in the Tivoli Netcool/OMNIbus guide:

   http://publib.boulder.ibm.com/infocenter/tivihelp/v8r1/index.jsp?topic=/com.ibm.netcool_OMNIbus.doc_7.3.0/omnibus/wip/install/concept/omn_con_ext_managingvirtualization.html

Archived

Note: The IBM Tivoli Monitoring virtualization rules are created as a stand-alone rules file. We modified these rules to comply to the “include” file structure. We performed the following changes in an existing `tivoli_eif.rules` file:

- Added the following line, under the `default: rules` line:

  ```
  "tivoli_eif_virtualization.rules"
  ```

- Added the following text to the beginning of the `tivoli_eif.rules`:

  ```
  alerts_target = registertarget( "NCOMS", "", "alerts.status", "alerts.details" )
  vmstatus_target = registertarget( "NCOMS", "", "custom.vmstatus", "" )
  ```

- Removed the following rules from the `tivoli_eif_virtualization.rules`:

  ```
  alerts_target = registertarget( "NCOMS", "", "alerts.status", "alerts.details" )
  vmstatus_target = registertarget( "NCOMS", "", "custom.vmstatus", "" )
  if( match( @Manager, "ProbeWatch" ) )
  {
          switch(@Summary)
          {
          case "Running ...":
                  @Severity = 1
                  @AlertGroup = "probestat"
                  @Type = 2
          case "Going Down ...":
                  @Severity = 5
                  @AlertGroup = "probestat"
                  @Type = 1
          default:
                  @Severity = 1
          }
          @AlertKey = @Agent
          @Summary = @Agent + " probe on " + @Node + ": " + @Summary
  }
  else
  {
  ```

- Removed the last right closing brace (**}**) from the same file.

6. Configure the event forwarding on the IBM Tivoli Monitoring server towards the EIF probe on the Tivoli Netcool/OMNIbus server.

Tivoli Application Dependency Discovery Manager

Perform the following steps for TADDM:

1. For the TADDM V7.2 installation, see the standard installation guide that is available at:

   http://publib.boulder.ibm.com/infocenter/tivihelp/v10r1/topic/com.ibm.taddm.doc_7.2/InstallGuide/c_cmdb_install_introducing.html

2. Discover a VMware Virtual Center using TADDM. This discovered information is used for creating a Business Application and a Business Service during the scenario walk-through.

For more information about the VMware Virtual Center sensor, see the TADDM V.2 sensor guide at:

http://publib.boulder.ibm.com/infocenter/tivihelp/v10r1/index.jsp?topic=/com.ibm.taddm.doc_7.2/SensorGuideRef/r_cmdb_sensor_vmwarevirtualcenter.html

Tivoli Business Service Manager

Perform the following steps for Tivoli Business Service Manager:

1. Install Tivoli Business Service Manager, which is described at the following website:

   http://publib.boulder.ibm.com/infocenter/tivihelp/v3r1/index.jsp?topic=/com.ibm.tivoli.itbsm.doc/installguide/bsmi_t_installing.html.

2. Point the new Tivoli Business Service Manager instance to an existing Tivoli Netcool/OMNIbus installation.
3. Install The XML Toolkit as documented in Chapter 5 of *Integrating Tivoli Products*, SG24-7757. In this scenario, we are using only the TADDM to Tivoli Business Service Manager part of the XMLToolkit. We are not using the Tivoli Monitoring to Tivoli Business Service Manager DLA part of the XMLToolkit.

6.6 Scenario walk-through

This scenario walks through the automatic updating of Tivoli Application Dependency Discovery Manager services and relevant IBM Tivoli Monitoring updates.

6.6.1 Creating a service in TADDM that contains a Virtual Center application

Create a service in TADDM that contains a Virtual Center application as follows:

1. Log into Tivoli Application Dependency Discovery Manager. To login to the TADDM Product Console, point your Internet browser to the following URL, replacing `taddm2.swg.be.ibm.com` with the DNS name of the server that is running the TADDM application.

   `http://taddm2.swg.be.ibm.com:9430/`

   See Figure 6-7 on page 137.

Archived

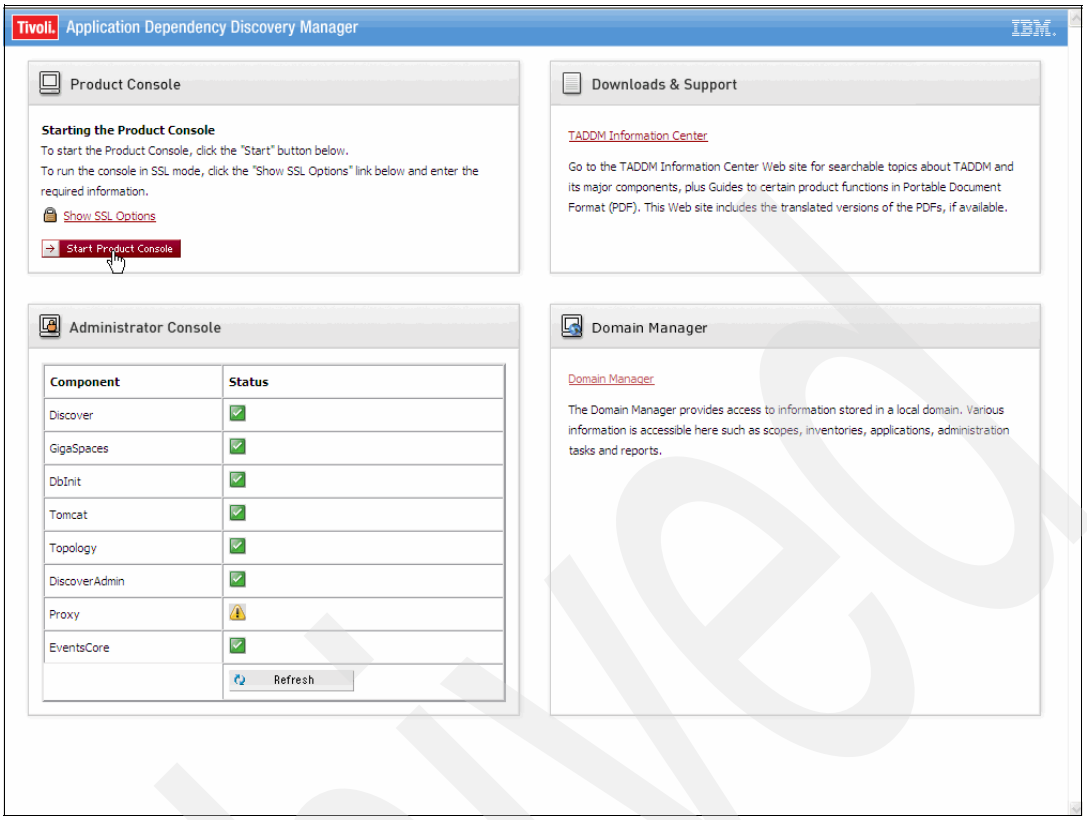


Figure 6-7 Start Product Console

Note: To understand the prerequisites for using the TADDM V7.2 'Product Console, see the following information center page:

`http://publib.boulder.ibm.com/infocenter/tivihelp/v10r1/index jsp?topic=/com.ibm.taddm.doc_7.2/UserGuide/t_cmdb_prereqs.html`

Figure 6-8 on page 138 shows the TADDM Product Console login widow.

Archive



Figure 6-8 TADDM Product Console login window

2. Enter information. The default user name for logging in to the TADDM Product Console is `administrator` and the default password is `collation`. Then, click **Login**.

   After successful login, the Overview window for the Product Console opens, as shown in Figure 6-9 on page 139.

Archive

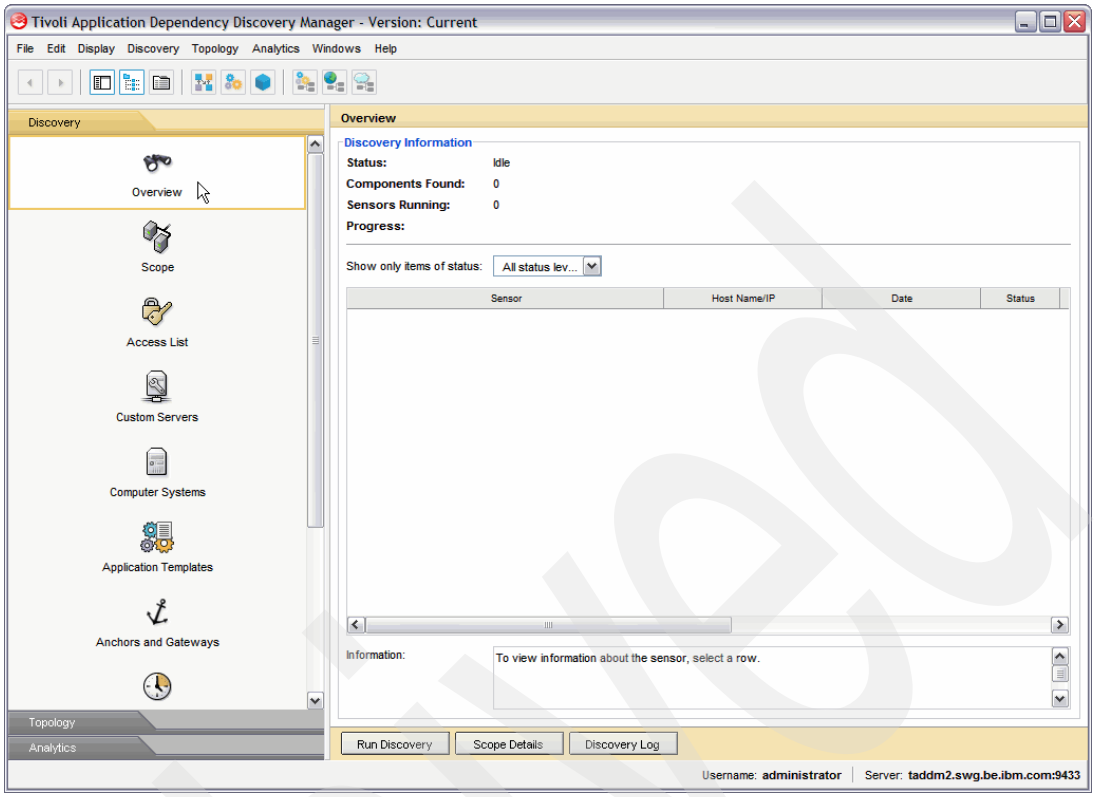


Figure 6-9 Overview window of TADDM Product Console

3. Optional: Navigate to the Business Applications window by going to the Topology, as shown in Figure 6-10 on page 140, to verify whether any Business Applications exist already.

Archive

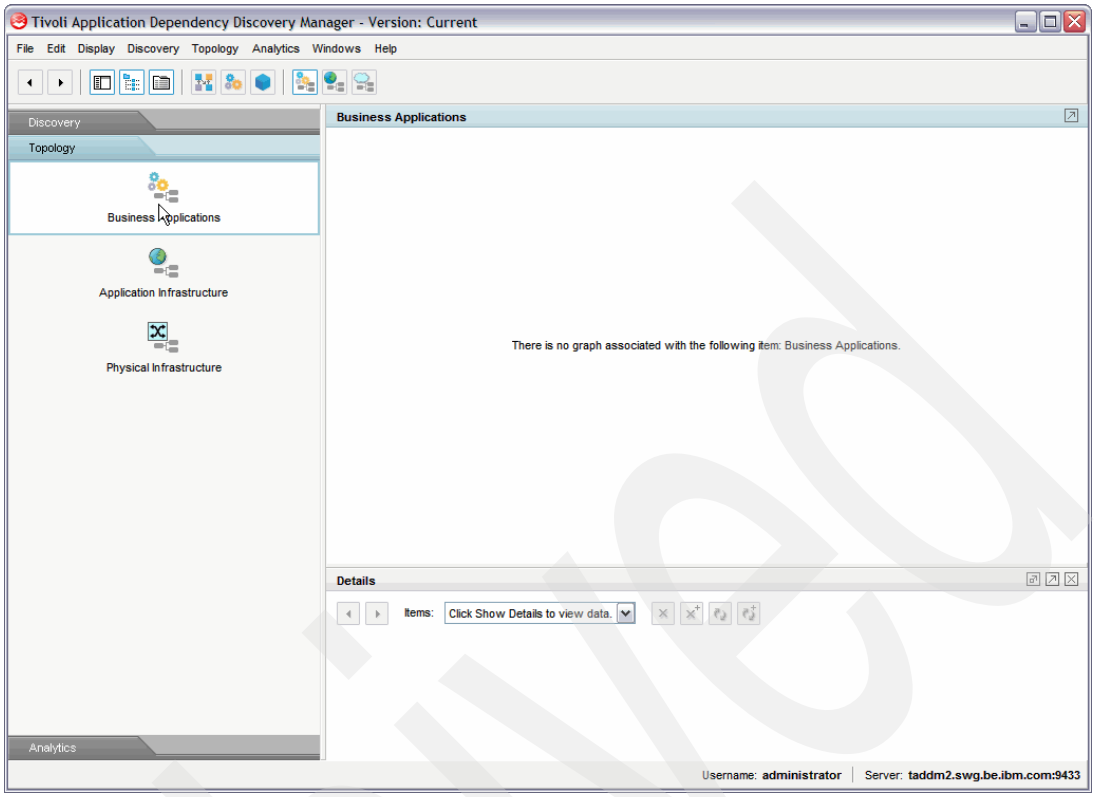


Figure 6-10 Business Applications window (under Topology)

4. Create a Business Application. Click **Edit** → **Create Business Application** in the TADDM Product Console. The Create Business Application Wizard starts and opens a window (Figure 6-11 on page 141).

   Provide a name for the Business Application and optionally any description and a URL for the same. In this scenario, we use the name `VMware Infrastructure` for the Business Application. Click **Next** to proceed further.

Archived

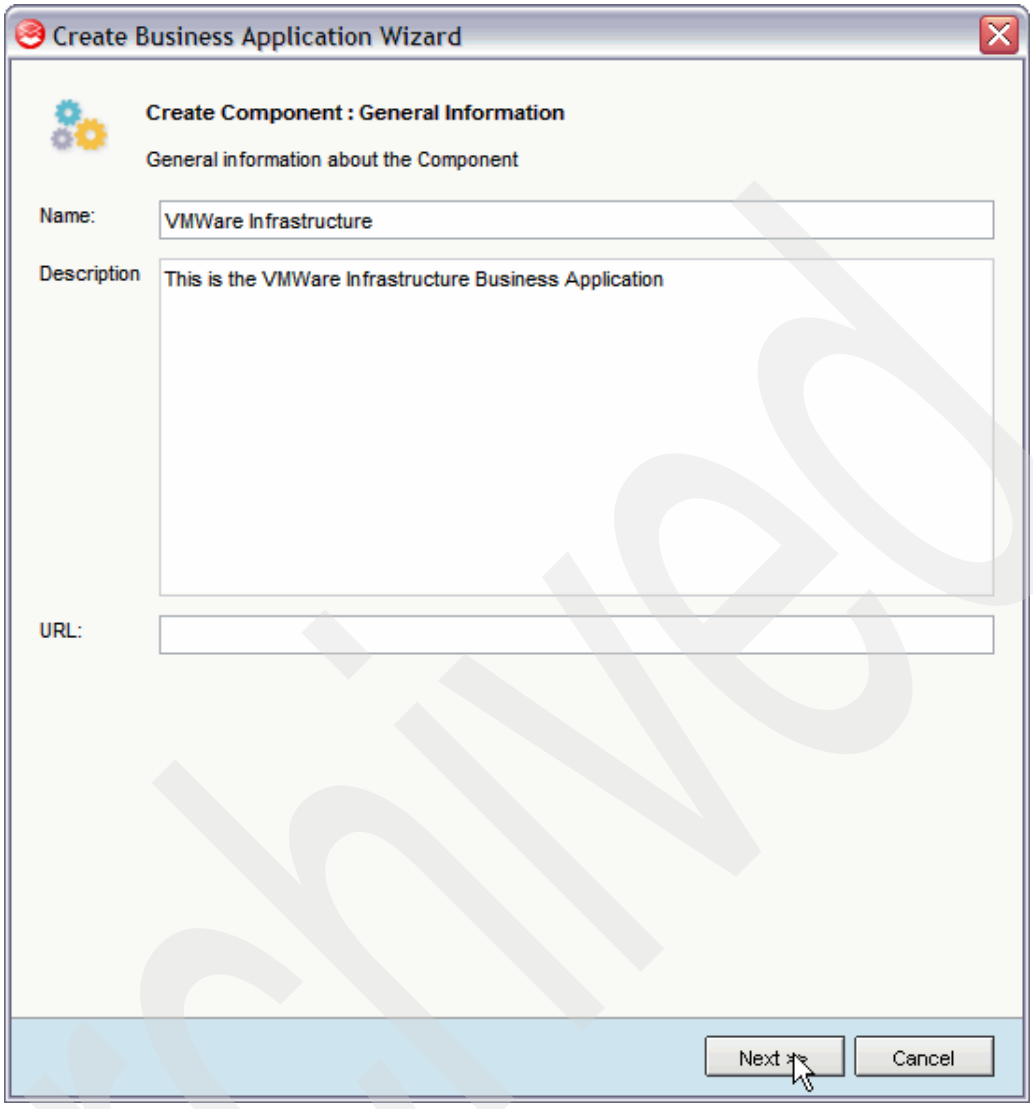


Figure 6-11 Create Business Application Wizard - Business Application name

5. The wizard prompts for selecting the components that realize this application. Figure 6-12 on page 142, shows that we selected the components that are part of the VMware Infrastructure. These are the same components that were discovered earlier (“Tivoli Application Dependency Discovery Manager” on page 135). This information includes all the components that are part of the VMware Infrastructure, including the VMware Virtual Center server, VMware ESX servers, and the respective virtual machines (VMs). When you are done, click **Next**.

Archived

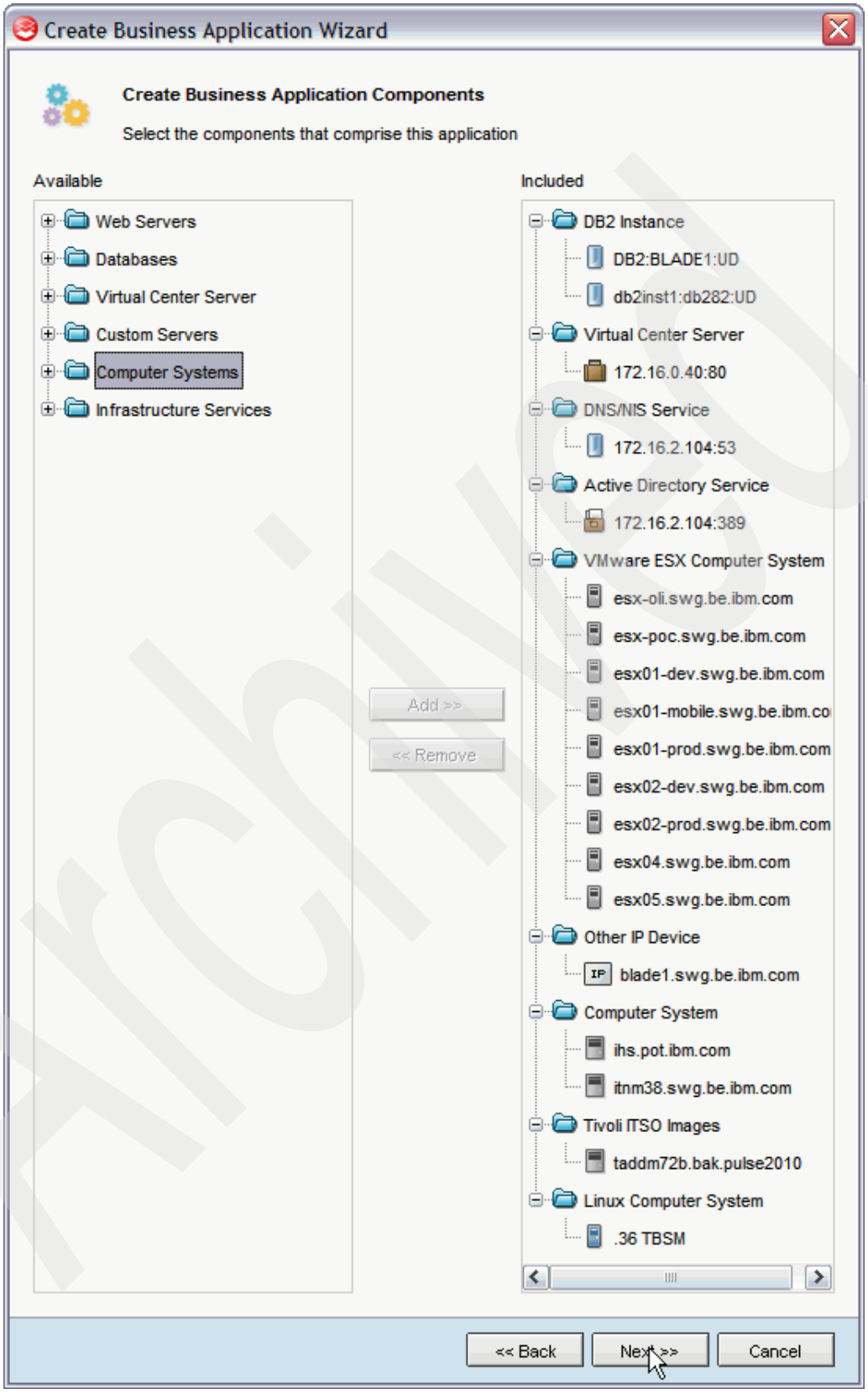


Figure 6-12 Select the Business Application components

Archive

6. Specify additional details related to the Business Application on the Business Application Administration information window (Figure 6-13).

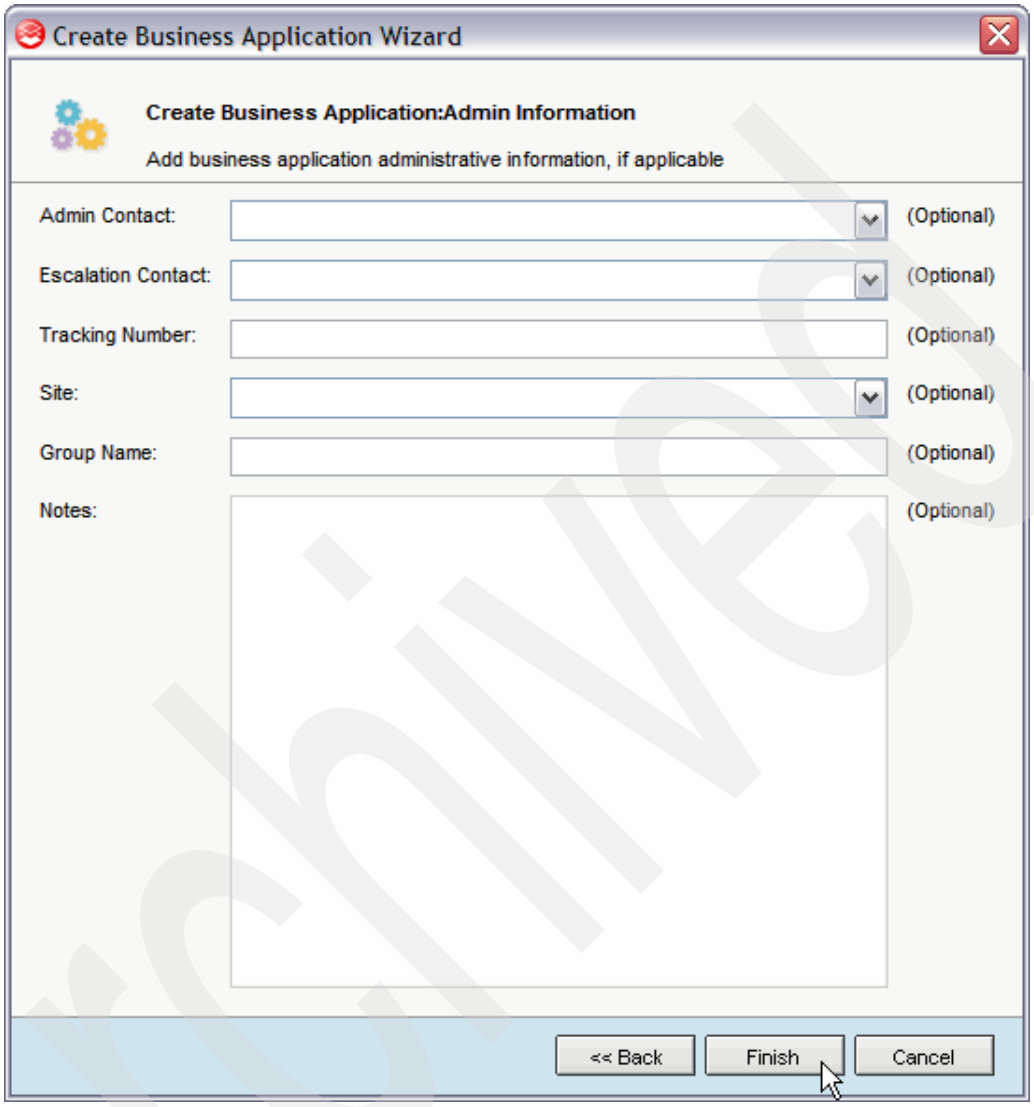


Figure 6-13 Optional details for the Business Application

7. View the newly created VMware Infrastructure Business Application by going to **Topology** → **Business Applications** (Figure 6-14 on page 144).

Archive

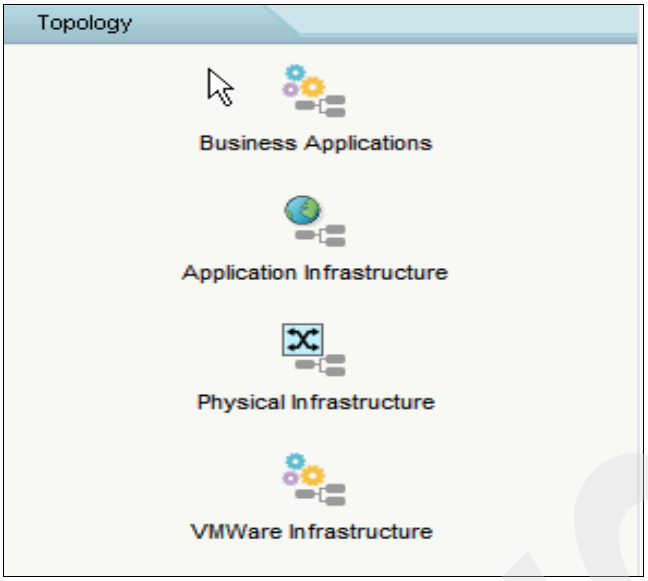


Figure 6-14 Vmware Infrastructure Business Application visible under Topology

8. Create a Business Service. Click **Edit** → **Create Business Service** in the TADDM Product Console. The Create Business Service Wizard starts and opens the window shown in Figure 6-15. Provide a name for the Business Service and optionally any description and a URL for the same. In this scenario, we use the name VMware Infrastructure Service for the Business Service. Click **Next** to proceed.

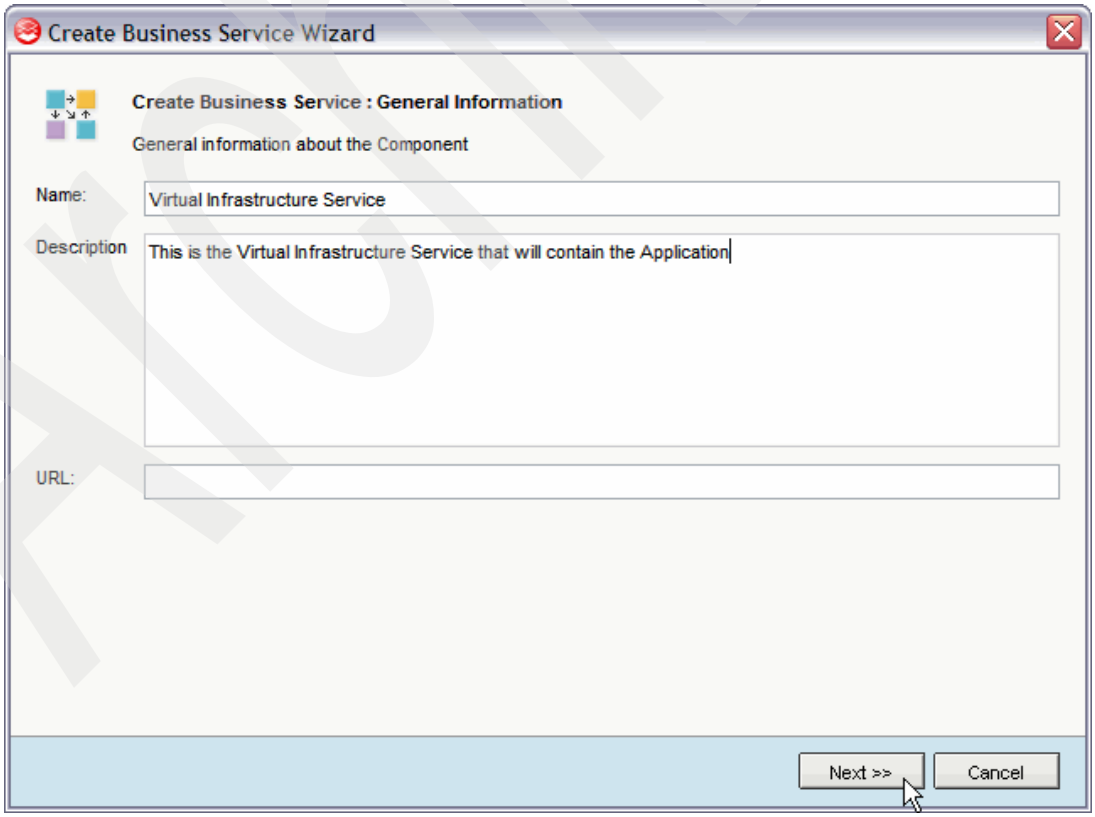


Figure 6-15 Create Business Service Wizard starting window

9. The wizard prompts for selecting the components of this Business Service (Figure 6-16). We select the VMware Infrastructure Business Application, which was created in the previous step.

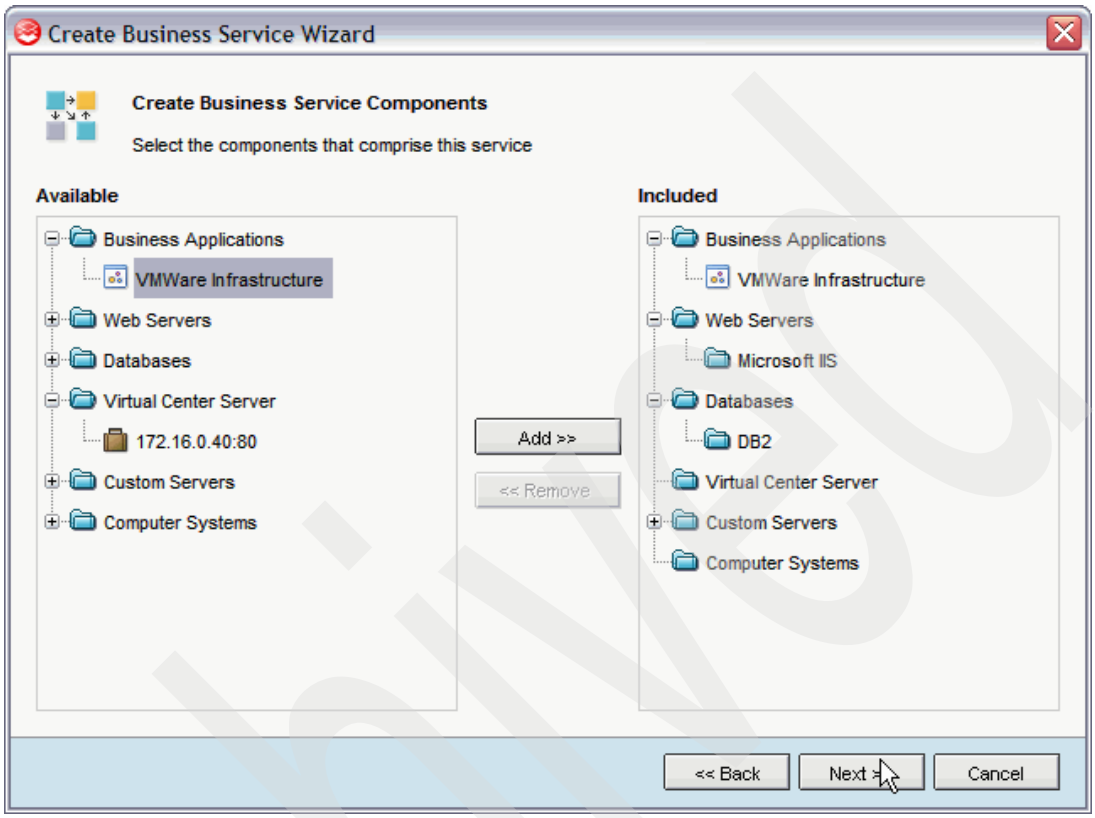


Figure 6-16 Select the components of the Business Service

10. Specify additional detail related to the Business Service in the Business Application Administration information window, as shown in Figure 6-17 on page 146.

Archive

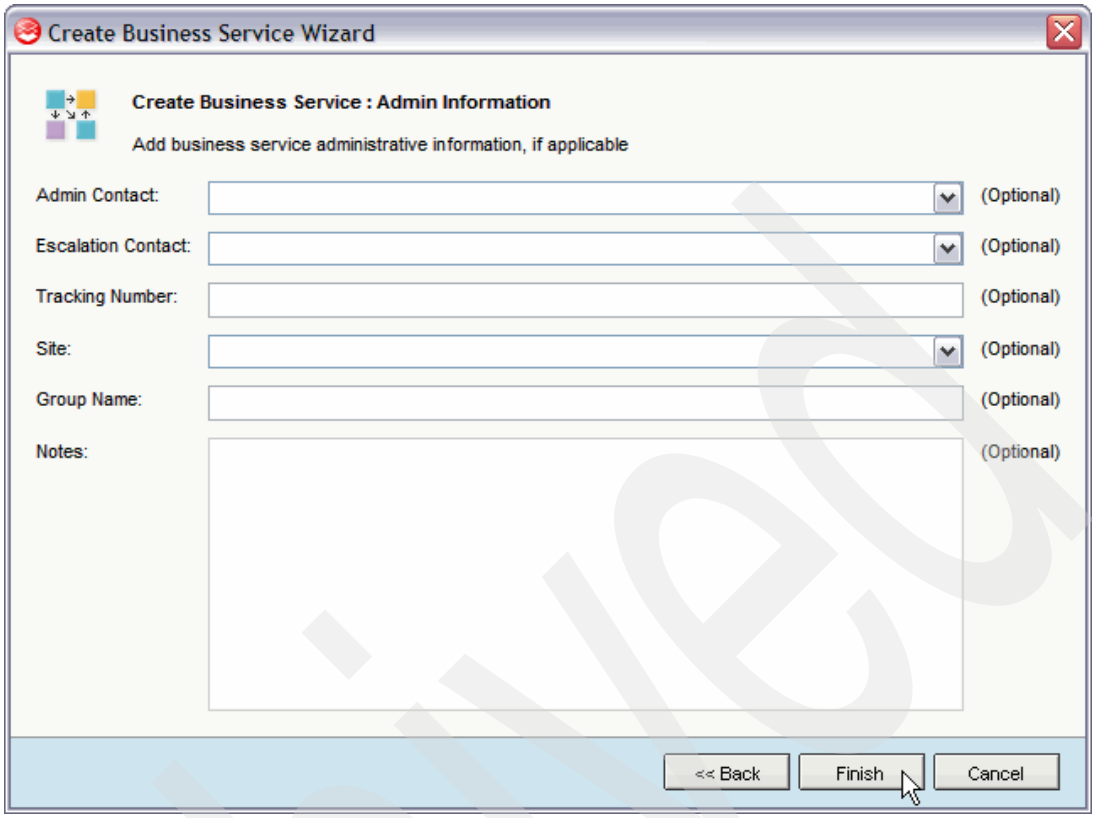


Figure 6-17 Optional detail for the Business Service

11. View the newly created VMware Infrastructure Service Business Service by navigating to **Discovered Components** → **Business Services** or by navigating to **Topology** → **Virtual Infrastructure Service**, as shown in Figure 6-18 on page 147.

Archive

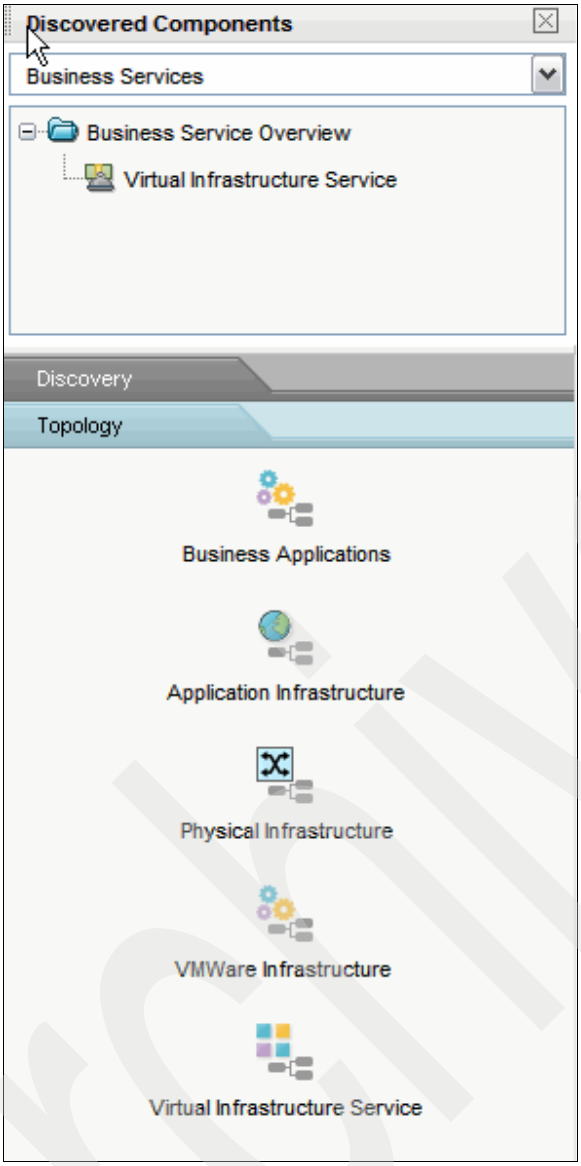


Figure 6-18 VMware Infrastructure Service visible under Topology

12. Visualize the entire Business Application topology as follows.
    a. Navigate to **Topology** → **Business Applications**.

       The VMWare Infrastructure Business Application and any other defined Business Applications are listed on the main display pane (on the right side).

    b. Right-click the 'VMWare Infrastructure' Business Application and select the **Show Physical Topology** option. The topology information is displayed, as shown in Figure 6-19 on page 148.

Archived

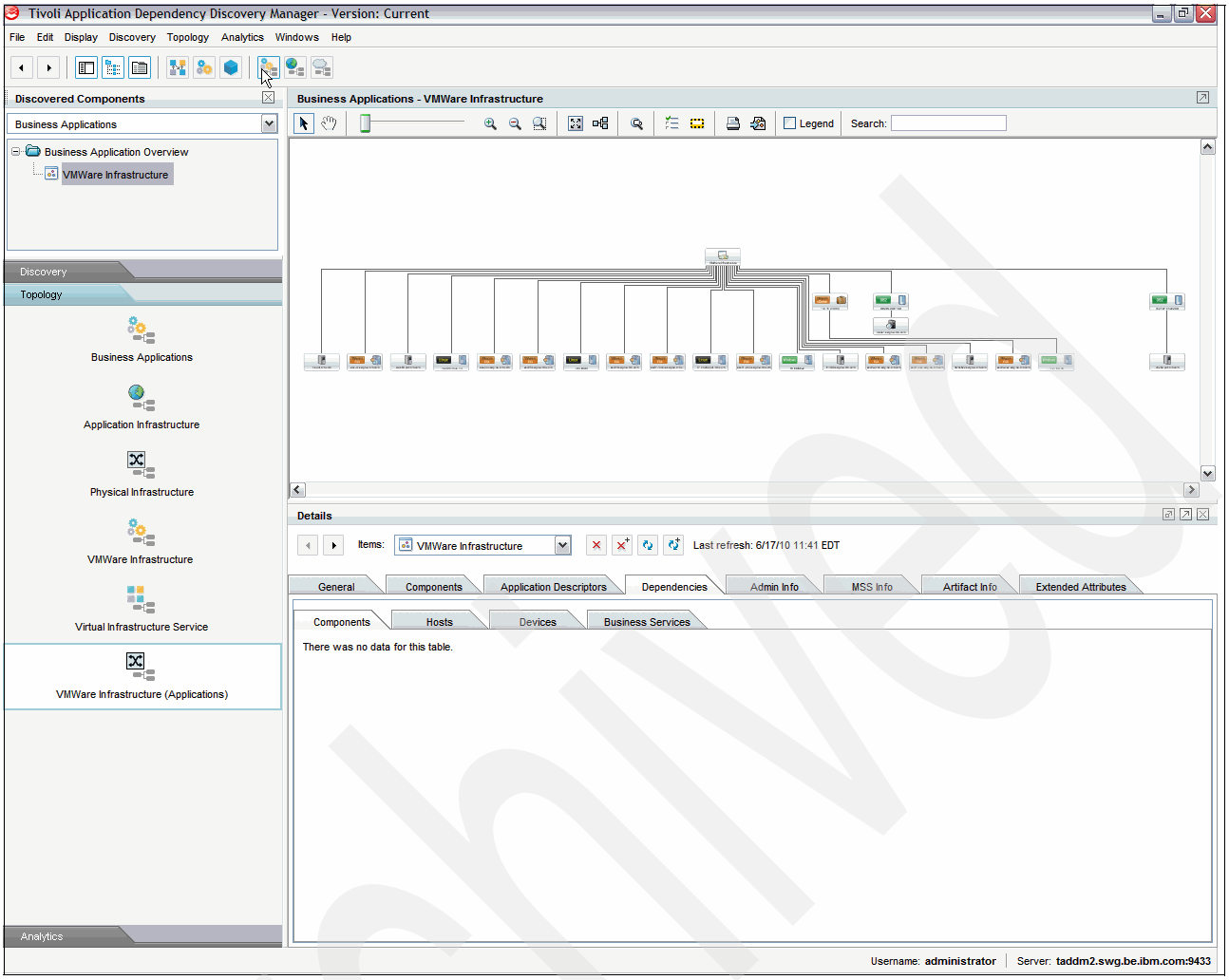


Figure 6-19 VMware infrastructure application topology

6.6.2 Checking for existence of service in Tivoli Business Service Manager

Perform the following steps to check for the existence of this service in Tivoli Business Service Manager:

1. Log in to the Tivoli Business Service Manager web GUI by using as the Tipadmin user by using the following address:

   `http://swg.be.ibm.com:16315/ibm/console/`

   See Figure 6-20 on page 149.

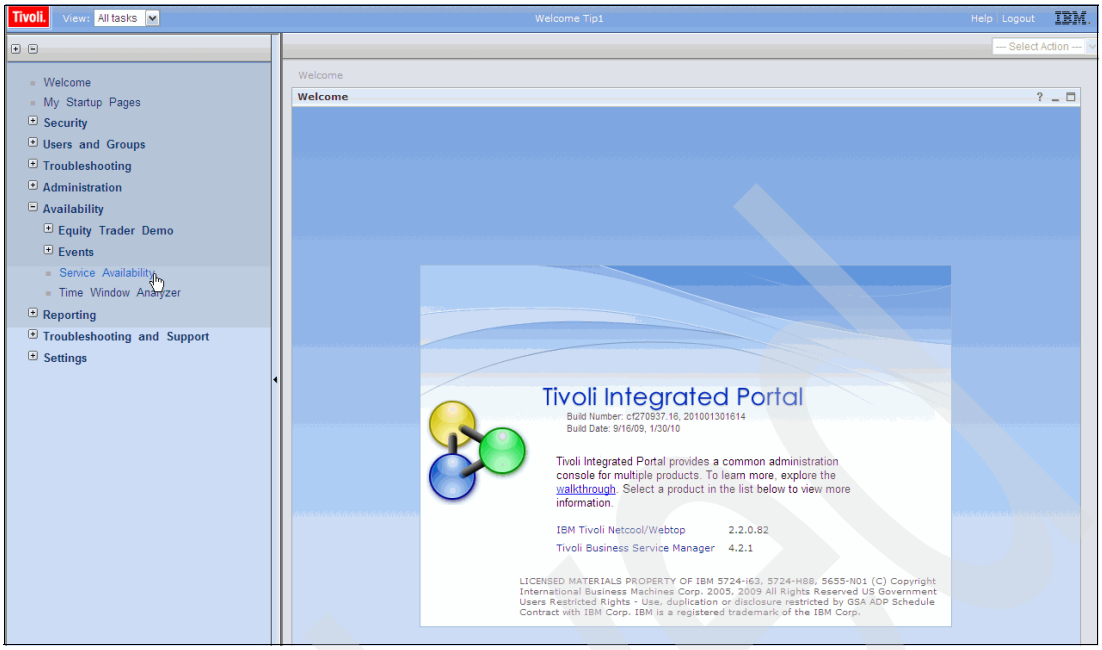


Figure 6-20 Default window

2. Click **Availability** → **Service Availability** → **Imported Business Service**. Figure 6-21 shows the Service view that is automatically being imported by using the XMLToolkit connected to the Tivoli Application Dependency Discovery Manager server.

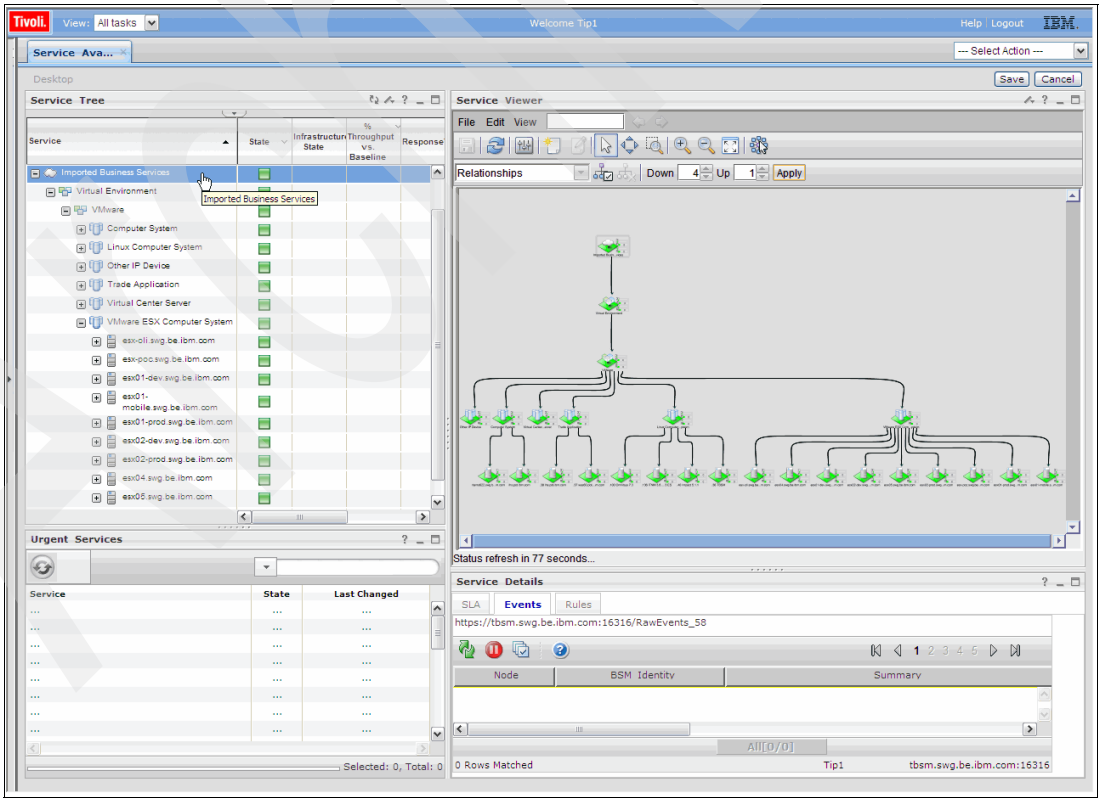


Figure 6-21 Imported Business Service View

3. Obtain more details about the service view content by clicking the maximize button on the Service Viewer portlet. See Figure 6-22.

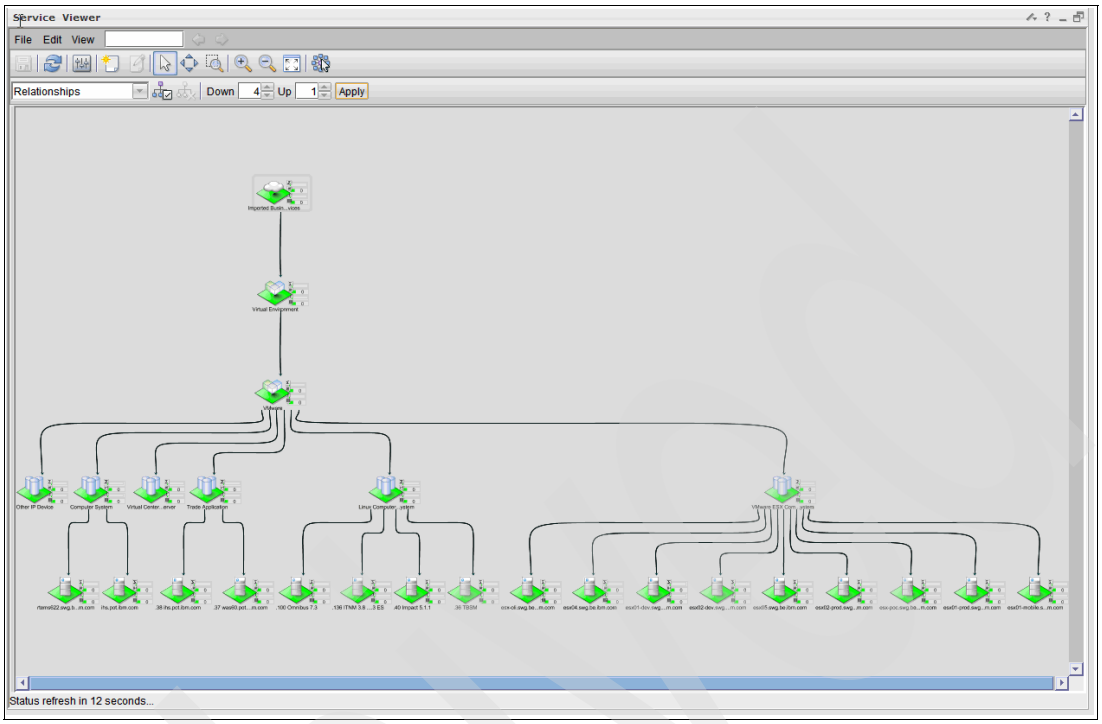


Figure 6-22 Service Viewer

6.6.3 Showing launch to IBM Tivoli Monitoring and Tivoli Application Dependency Discovery Manager (preconfigured functionality)

The Service Viewer provides the launch-in-context functionality of Tivoli Business Service Manager, as seen in Figure 6-23.

Enabling launch-in-context requires Tivoli Application Dependency Discovery Manager having imported monitoring data through the DLA books into Tivoli Business Service Manager. Figure 6-23 shows a server that is being monitored by IBM Tivoli Monitoring and has been discovered by Tivoli Application Dependency Discovery Manager.

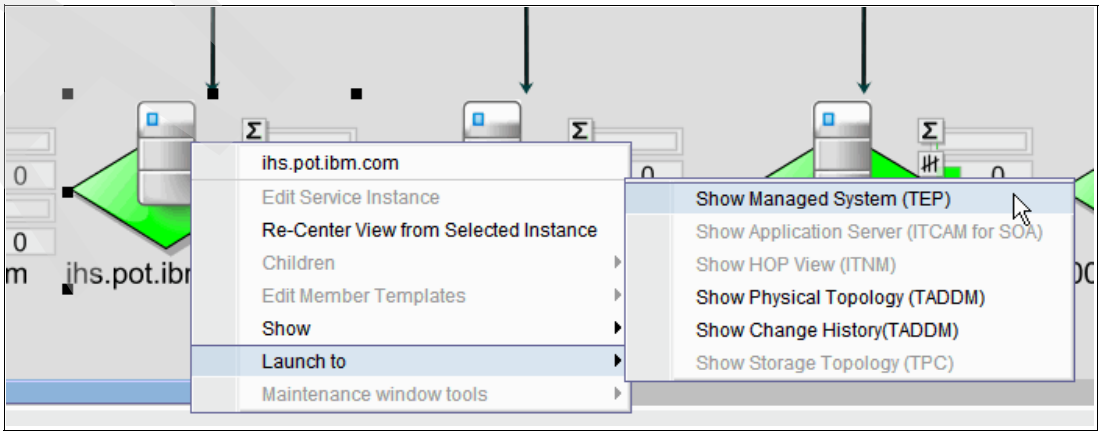


Figure 6-23 Launch-in-context

The **Show Managed System (TEP)** launch-in-context opens the IBM Tivoli Monitoring portal and shows the agents that are related to that specific server (Figure 6-24).

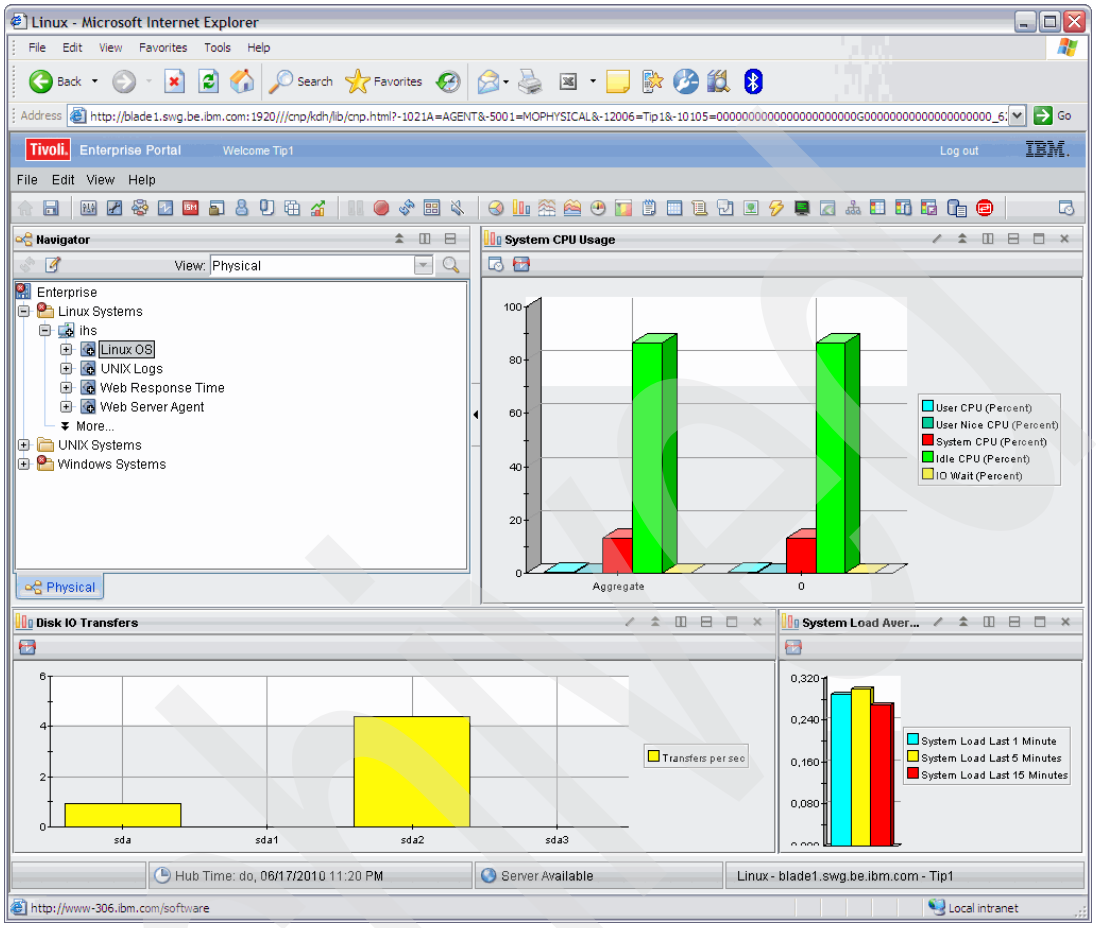


Figure 6-24 IBM Tivoli Monitoring workspace

The **Show Physical Topology (TADDM)** launch opens the Tivoli Application Dependency Discovery Manager's Domain manager page showing the Details View of the selected server (Figure 6-25 on page 152).

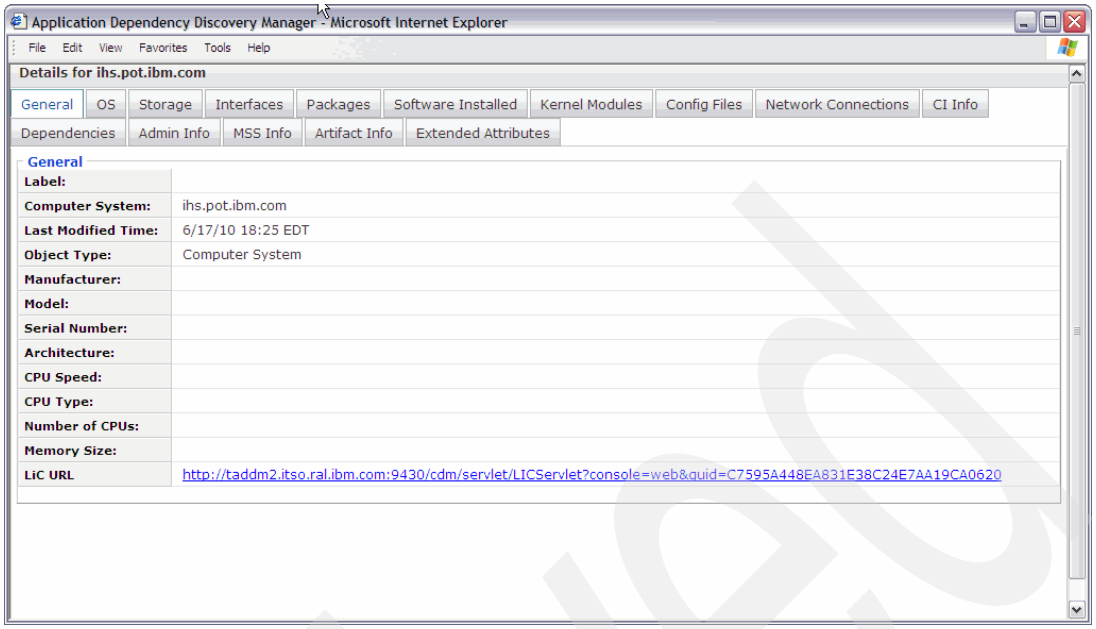


Figure 6-25 Details View of a server

The **Show Change History (TADDM)** launch opens the Tivoli Application Dependency Discovery Manager domain manager pages that show the change history view of this selected server (Figure 6-26).

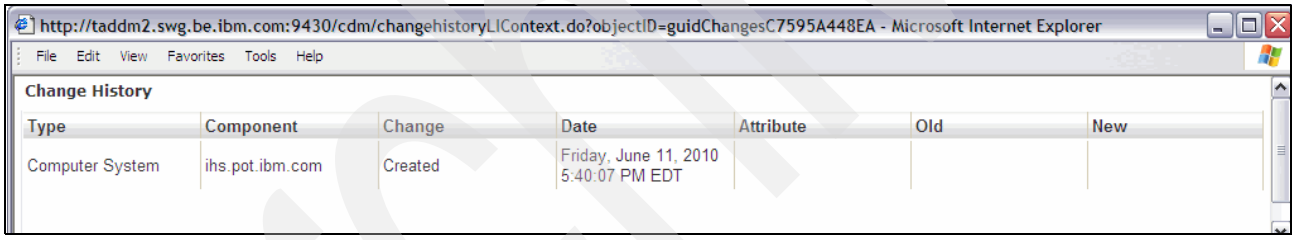


Figure 6-26 Change History View

Note: Another possibility is to perform a launch-in-context from TADDM to IBM Tivoli Monitoring by loading a DLA from IBM Tivoli Monitoring into TADDM. For more information, see section 5.2.4 of *Integrating Tivoli Products*, SG24-7757.

6.6.4 Creating new virtual machine on the Virtual Center

Create a new Virtual Machine on the Virtual Infrastructure center, by cloning a template virtual machine that contains the OS agent that is pre-installed. See Figure 6-27 on page 153.

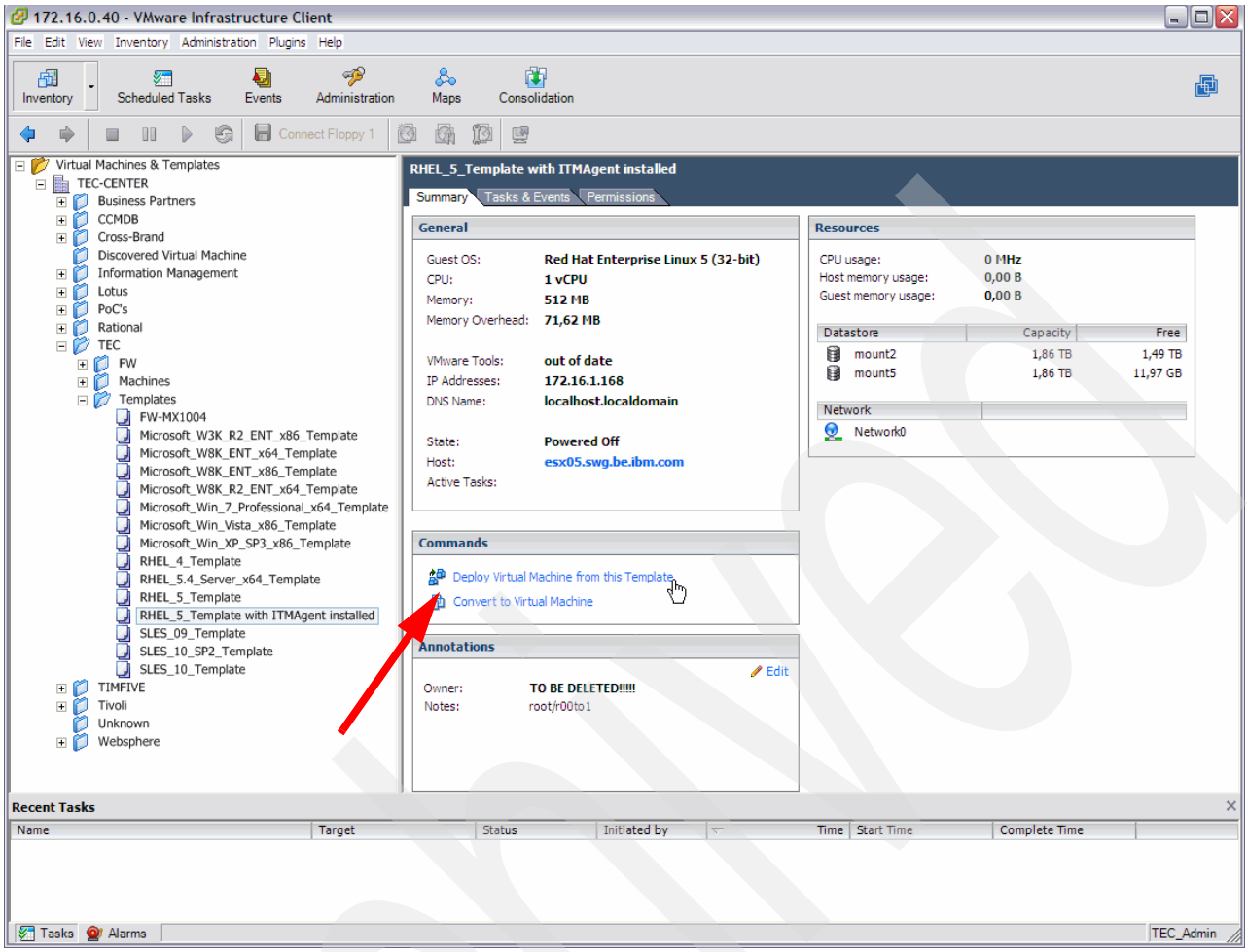


Figure 6-27 Create virtual machines

> **Note:** Another integration step in this scenario is to enable the IBM Tivoli Monitoring for virtual servers situation to be generated when a virtual machine is created. This situation can be forwarded to Tivoli Netcool/OMNIbus to notify about its creation in real-time and can be used to have the user perform the rediscovery (described in the next section) and update the repository (TADDM). In many cases, various people might create the virtual with Virtual Center and be responsible for TADDM. This way can help with synchronization.

6.6.5 Auto-discovering the virtual machine by using TADDM rediscovery

Perform the following steps:

1. Initiate a TADDM discovery of the VMware Virtual Center, which was discovered earlier (see “Tivoli Application Dependency Discovery Manager” on page 135), and scan for any changes to detect the change of the new virtual machine creation. To initiate a discovery, select **Discovery** → **Run Discovery**, as shown in Figure 6-28 on page 154.

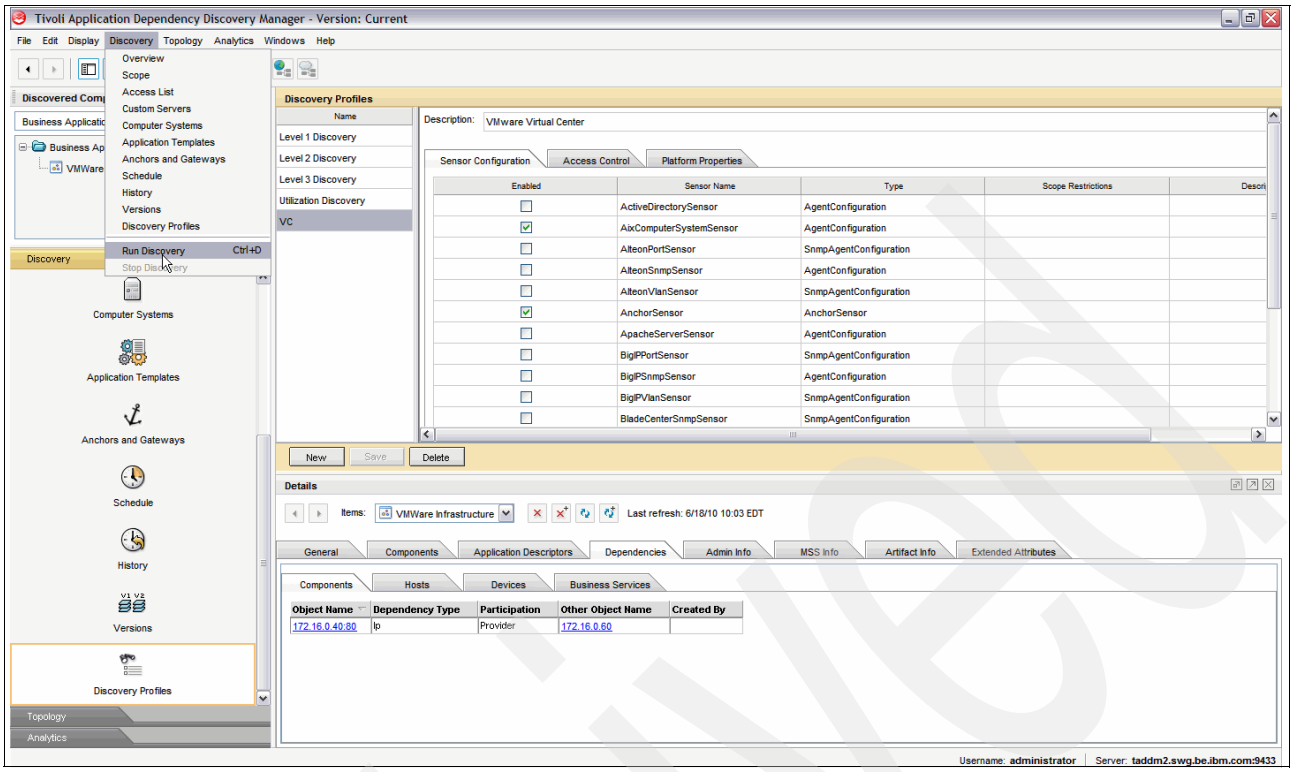


Figure 6-28 Starting the TADDM discovery process

2. Select the scope and profile to be used for the discovery, as shown in Figure 6-29 on page 155, and then click **OK** to start the discovery.

> **Note:** VMware VC is the name of a scope defined by us while testing this scenario. We added the IP address (or host name) of the VMware Virtual Center, which needs to be discovered, into this scope. You might need to define a scope in your environment before you initiate the discovery process.

Archive

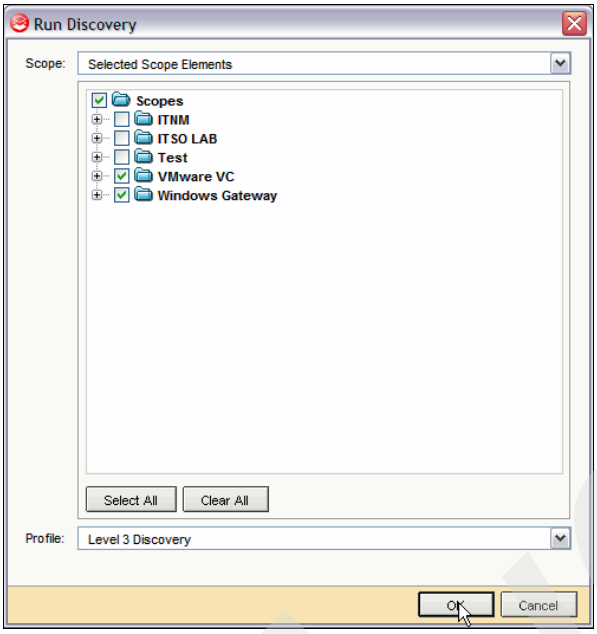


Figure 6-29 Selecting the scope and profile for the discovery

3. The discovery progress is displayed in the window (Figure 6-31 on page 156). At the bottom of the window the discovery status (Running) and the summary of errors (2) is also displayed.

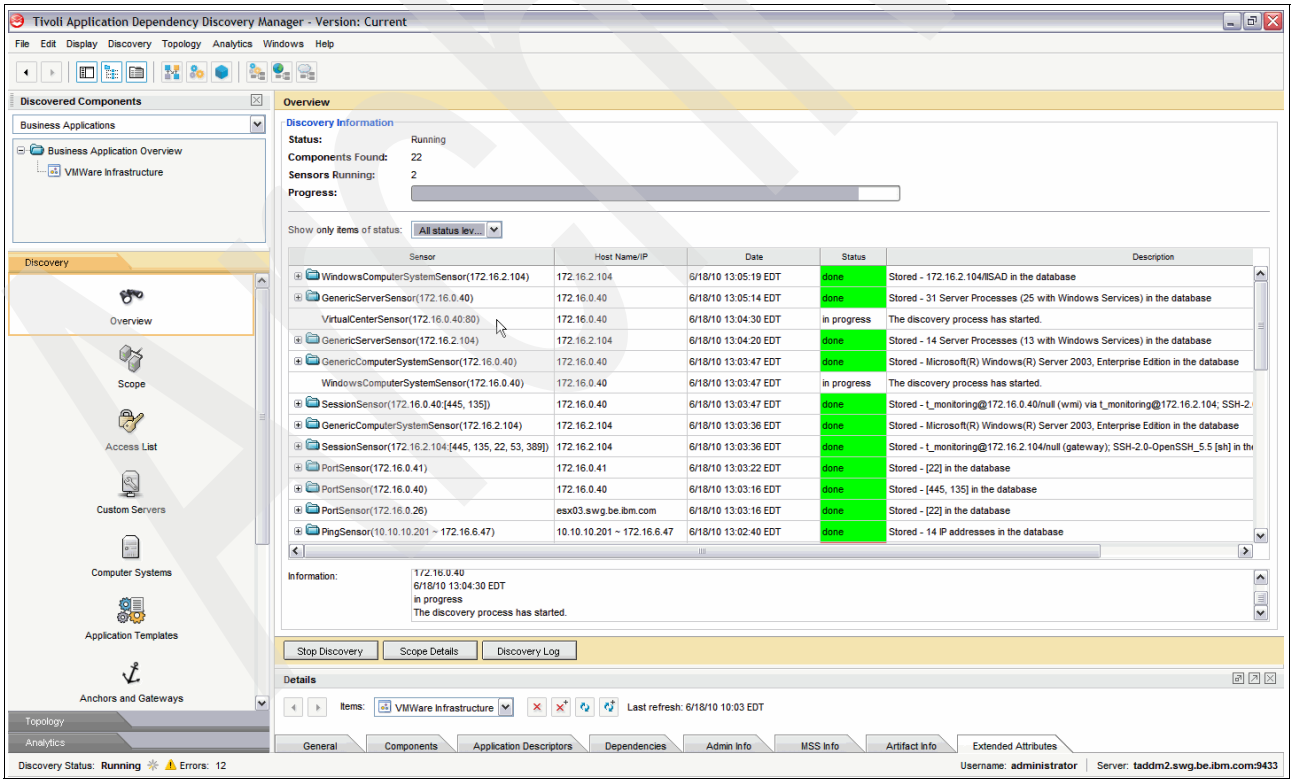


Figure 6-30 Discovery in progress

4. After the discovery is completed, various ways are available to determine whether the new Virtual Machine that defined on the VMware VC is detected by TADDM. One method is to use the **List/Search** facility under the **Discovered Components**, as shown in Figure 6-31.

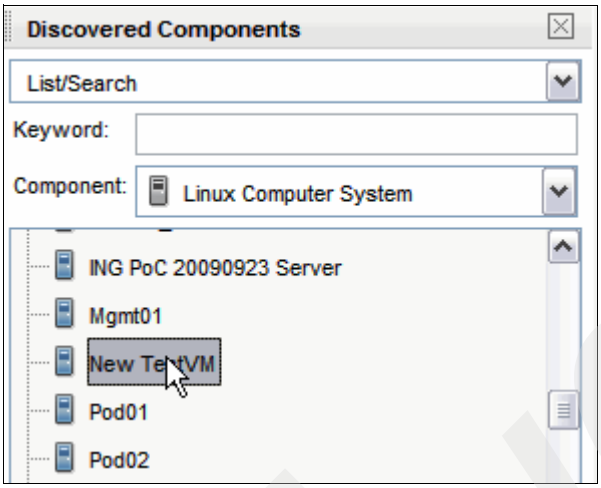


Figure 6-31 New Virtual Machine is discovered by TADDM

When Tivoli Application Dependency Discovery Manager discovers new machines, it sends change events to the Tivoli Netcool/OMNIbus server. These events can be used to trigger actions to be performed on these new machines, or to alert change-management that something has changed or been added, as shown in Figure 6-32.

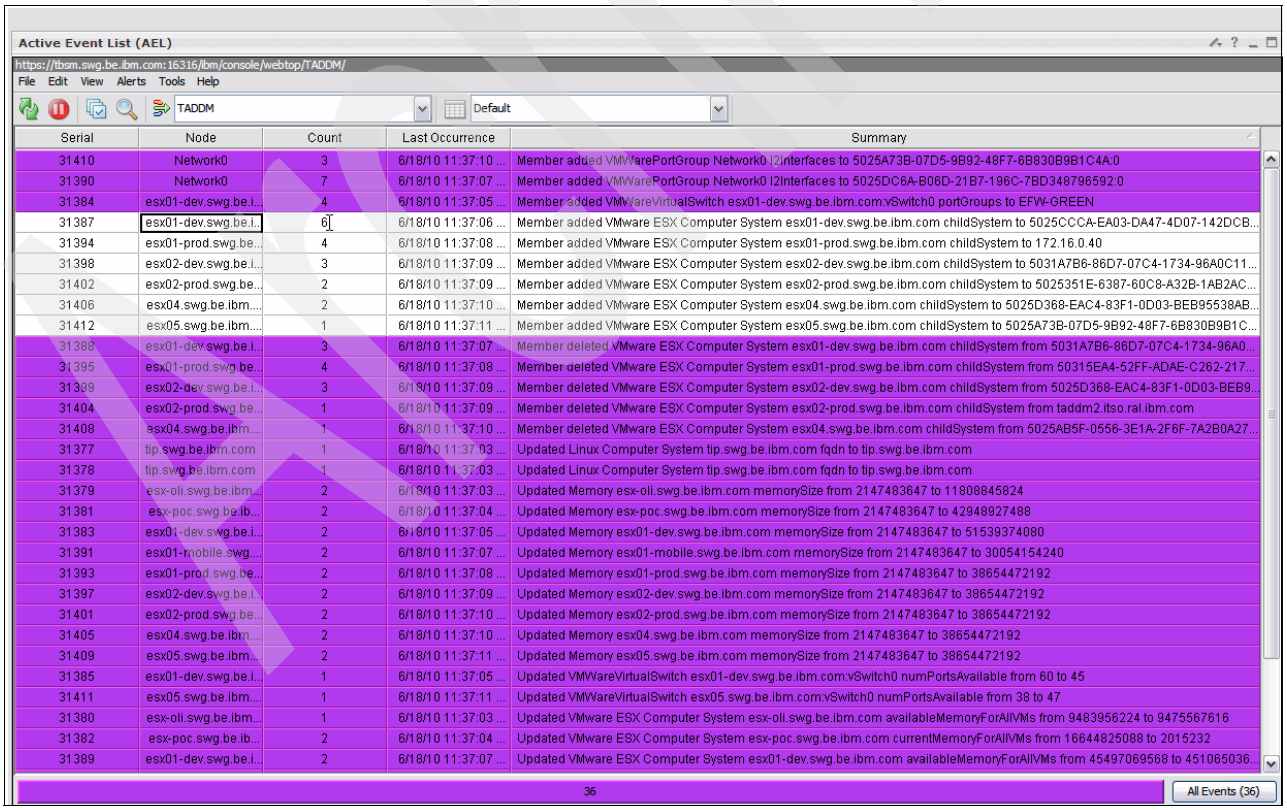


Figure 6-32 Change events shown in Active Event List

6.6.6 Accepting (adding) virtual machine to existing Virtual Center Application

The change detected by TADDM (new VM) can be accepted by adding the newly discovered virtual machine into the “VMWare Infrastructure” Business Application created earlier. The process is to simply edit the Business Application and add the newly discovered virtual machine into the “Included” list of entries for the Business Application, as shown in Figure 6-33 and Figure 6-34 on page 158 respectively.

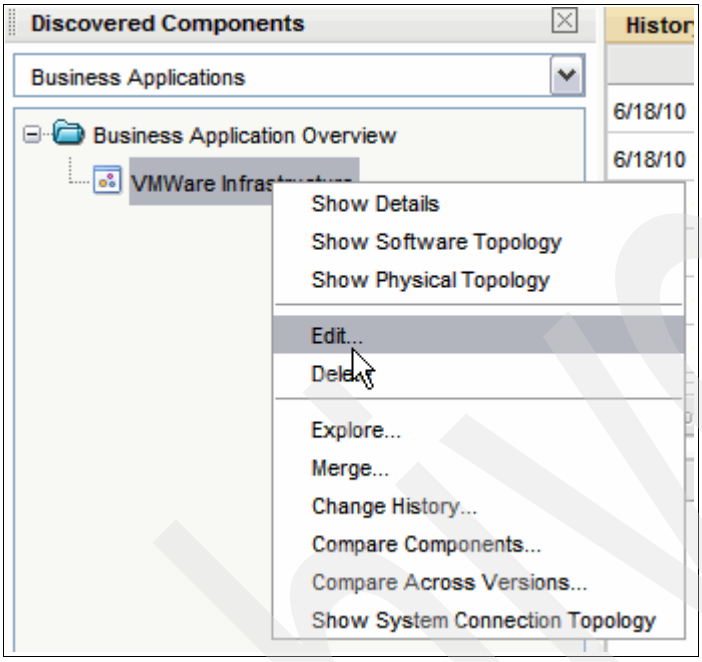


Figure 6-33 Add the newly discovered virtual machine into the VMware Infrastructure Business Application

Archived

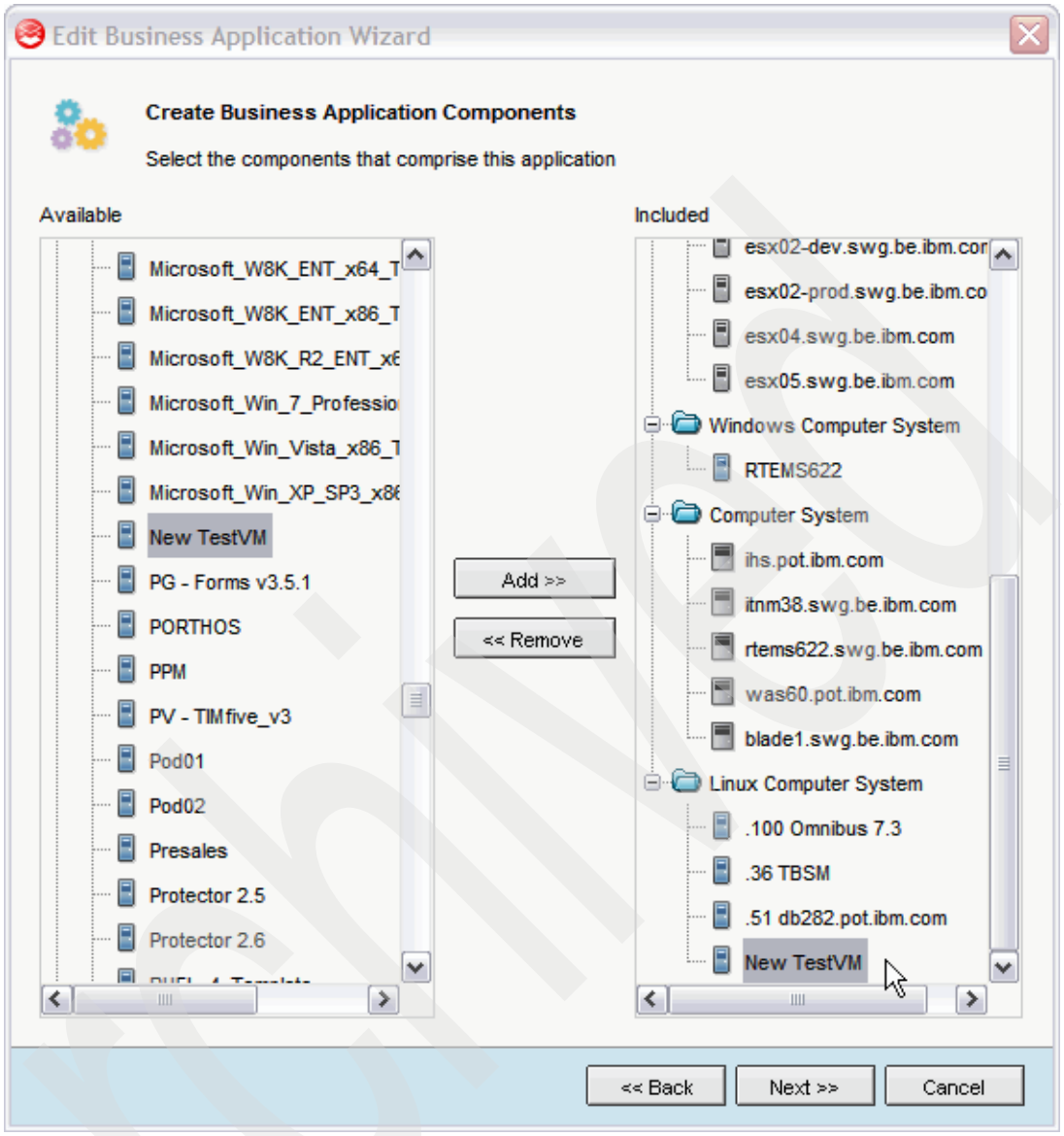


Figure 6-34 Newly discovered Virtual Machine is being added to the Business Application

Verify the addition of the new virtual machine into the “VMWare Infrastructure” Business Application by navigating to **Topology** → **VMWare Infrastructure** (on the left side), as shown in Figure 6-35 on page 159 (look for the virtual machine pointed to by the mouse pointer).

Archive

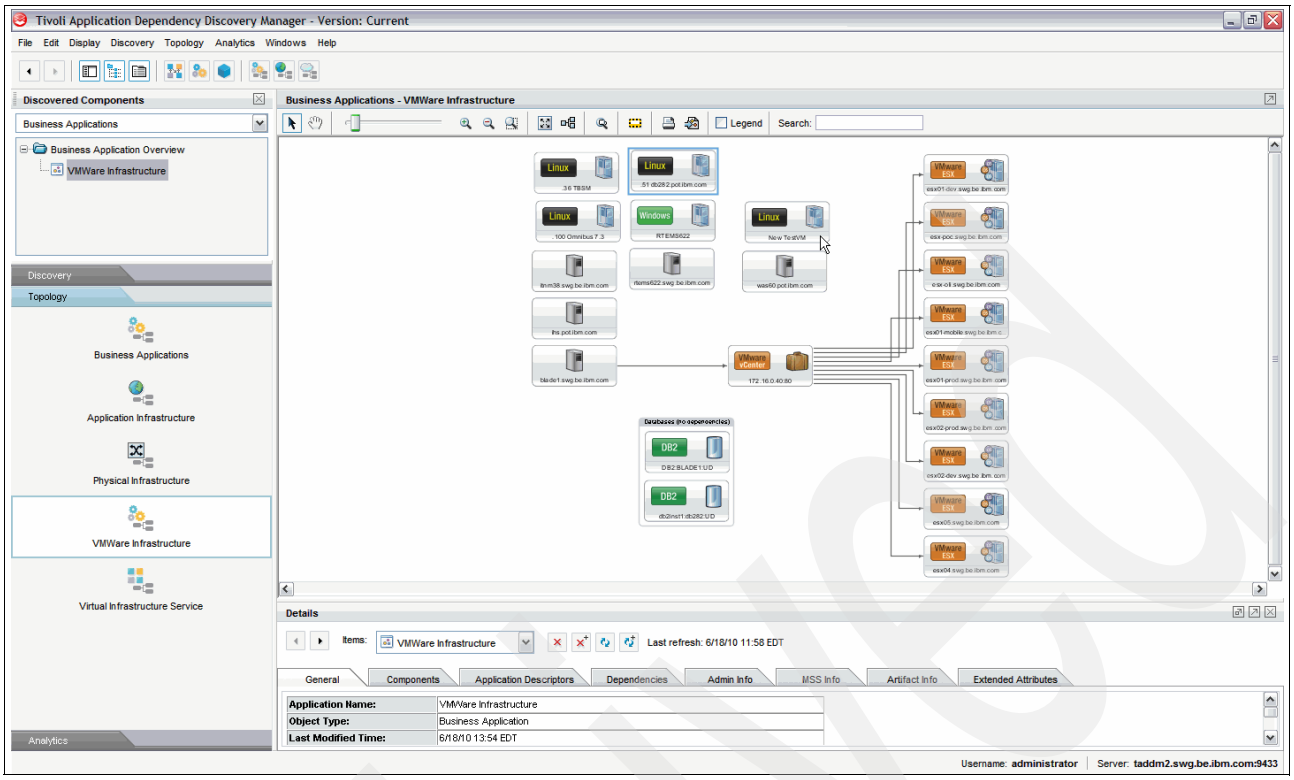


Figure 6-35 Business Application Topology reflects the newly discovered VM

> **Tip:** In TADDM V7.2, the addition of the newly created virtual machine into the "VMWare Infrastructure" Business Application can be automated by defining Application Templates. For more information, go to the following blog:
>
> https://www-950.ibm.com/blogs/taddm/entry/using_application_descriptor_files_to_build_business_applications_in_taddm?lang=en_us
>
> TADDM V7.2.1 offers an enhanced facility for the procedure described in the blog, by enabling the user to create rules in Model Query Language (MQL), and the output of those rules would be automatically added to the Business Application.

6.6.7 Verifying that virtual machine connects to IBM Tivoli Monitoring server

The test VM was built using a cloned Linux virtual machine that contains the OS agent, pre-installed. The added benefit is of not having to deploy an OS agent on this machine.

> **Note:** You may also deploy an OS agent using the IBM Tivoli Monitoring **`tacmd createnode`** command, and the Agent Depot functionality.

Figure 6-36 on page 160 shows the default agents and the resulting workspaces of the new virtual machine.

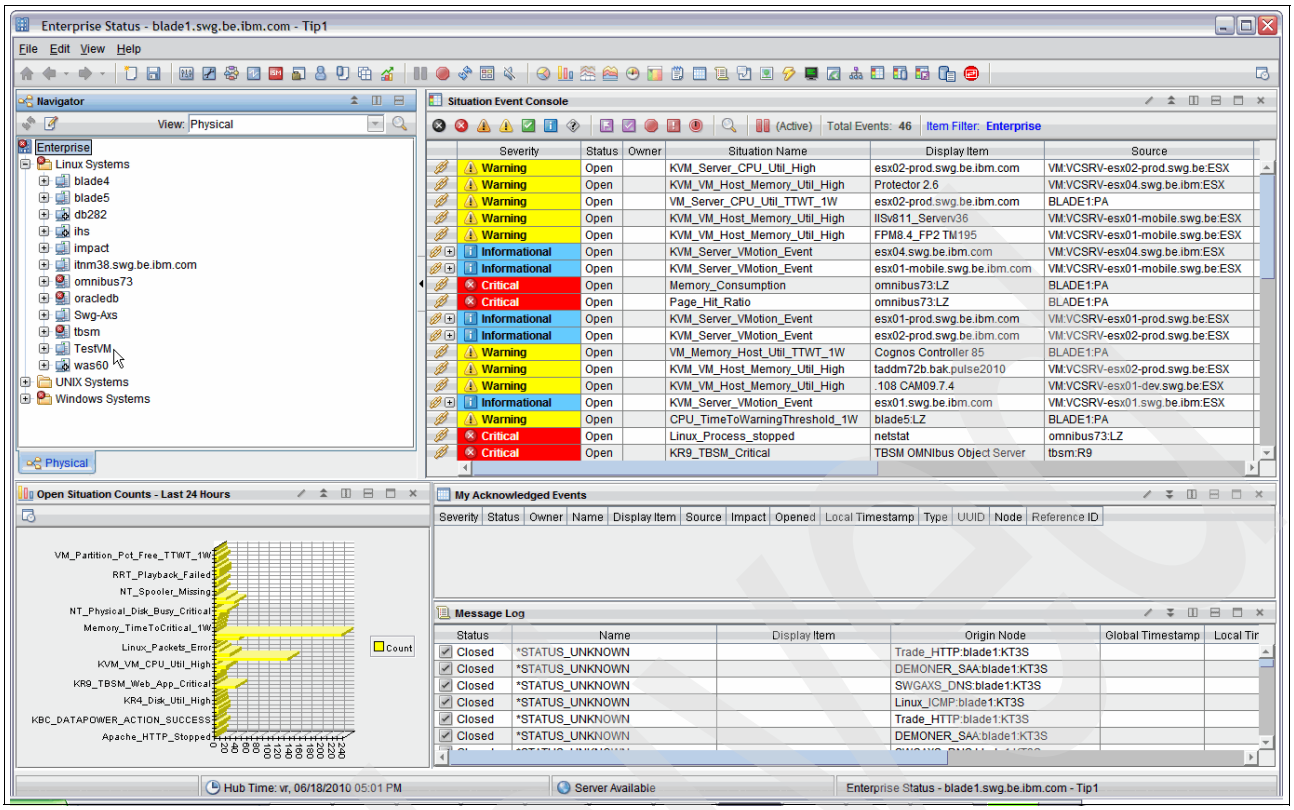


Figure 6-36 Workspace showing new machine

After the new virtual machine has connected to the IBM Tivoli Monitoring server, run the following command on the IBM Tivoli Monitoring server:

```
KfwTmsDla.exe /x y:\itm_book_18_06_18_35.xml
```

To re-create the monitored resources that will be imported into Tivoli Application Dependency Discovery Manager again, run the following command by using the same file that was created on the IBM Tivoli Monitoring server:

```
loadidml.sh -f /mnt/share1/itm_book_18_06_18_35.xml
```

This command updates the launch-in-context information that is stored in the Tivoli Application Dependency Discovery Manager server.

Figure 6-37 on page 161 shows the workspace displaying default agents.

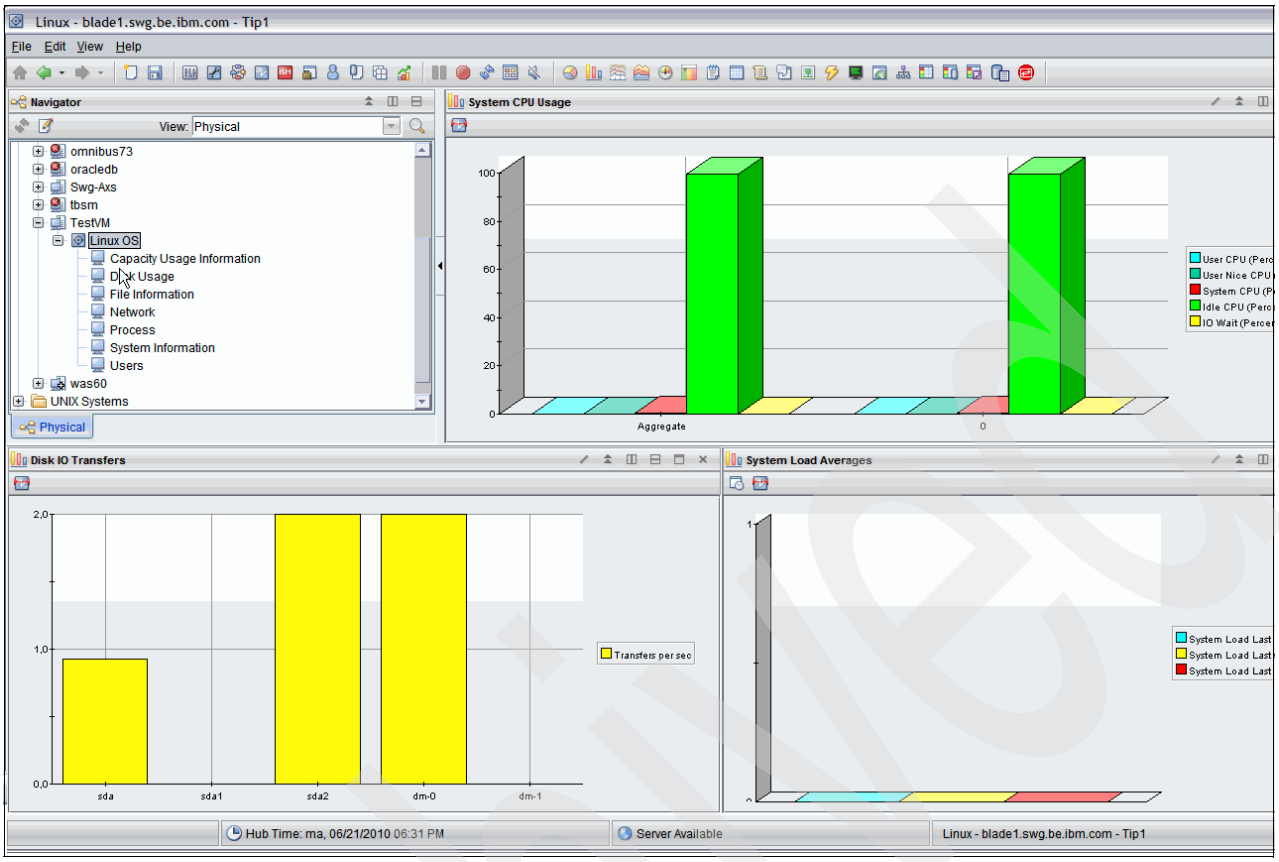


Figure 6-37 Workspace showing default agents

Note: The IBM Tivoli Management System Discovery Library Adapter (TMS DLA), when loaded into TADDM, gives the Monitoring Coverage report indicating which systems are monitored by IBM Tivoli Monitoring and which are not. The list of systems that are not monitored can be passed to the IBM Tivoli Monitoring **tacmd** command to automatically deploy IBM Tivoli Monitoring OS agents on them for monitoring coverage. This function works only for IBM Tivoli Monitoring OS agents.

Also, when a new VM is created, the same will automatically show up in the VMware VI Agent. But in some cases, depending on the business requirement for monitoring, you might still want to install the VM Agent inside the VM. For example, if you want to monitor the OS inside the VM, and are interested in attributes such as list of running processes, the respective OS agent must be installed inside the VM because the VMware VI Agent does not monitor such attributes. Certain situations still exist that require execution of the **tacmd** command or a similar agent provisioning mechanism.

For more information, go to the following web address:

http://www.ibm.com/developerworks/wikis/display/tivoliaddm/Identifying+Unmanaged+Systems+and+Deploying+Agents

6.6.8 Verifying this virtual machine appears in Tivoli Business Service Manager

If the XMLToolkit is still running, the TESTVM virtual machine is listed in the Service Component Repository (**Administration** → **Service Configuration**). Perform the following steps:

1. Navigate to **Service Component Repository** → **Component Registry** → **Servers** → **Unix** as shown in Figure 6-38.

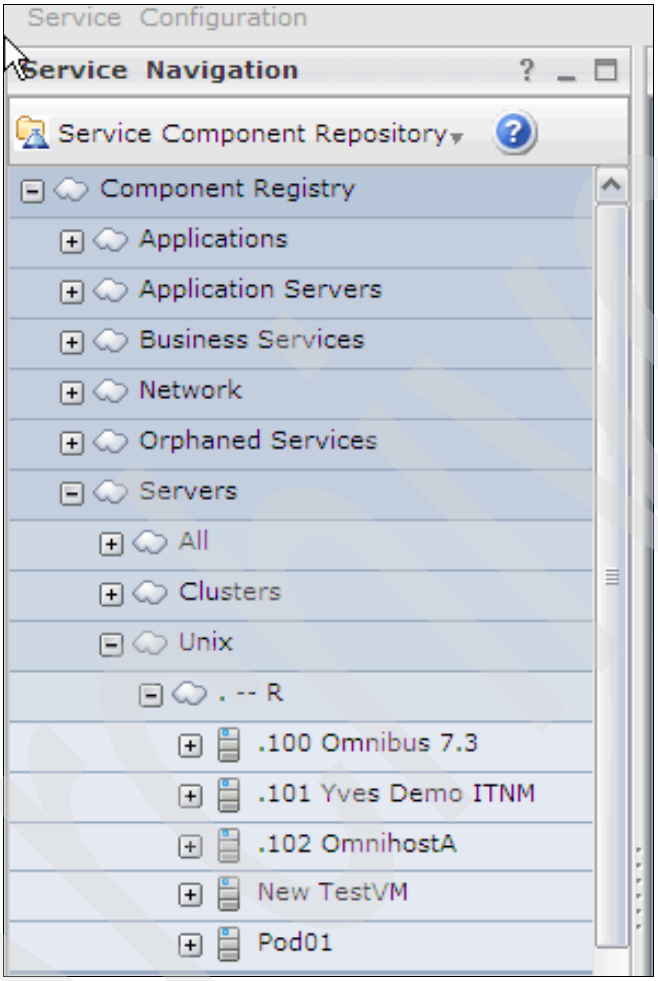


Figure 6-38 Service Component Registry

2. You may now use this Service Component Repository entry to create a new service or add to an existing service without having to configure the event relationships from IBM Tivoli Monitoring. The launch-in-context data is also be auto-populated by using this entry, as shown in Figure 6-39 on page 163.

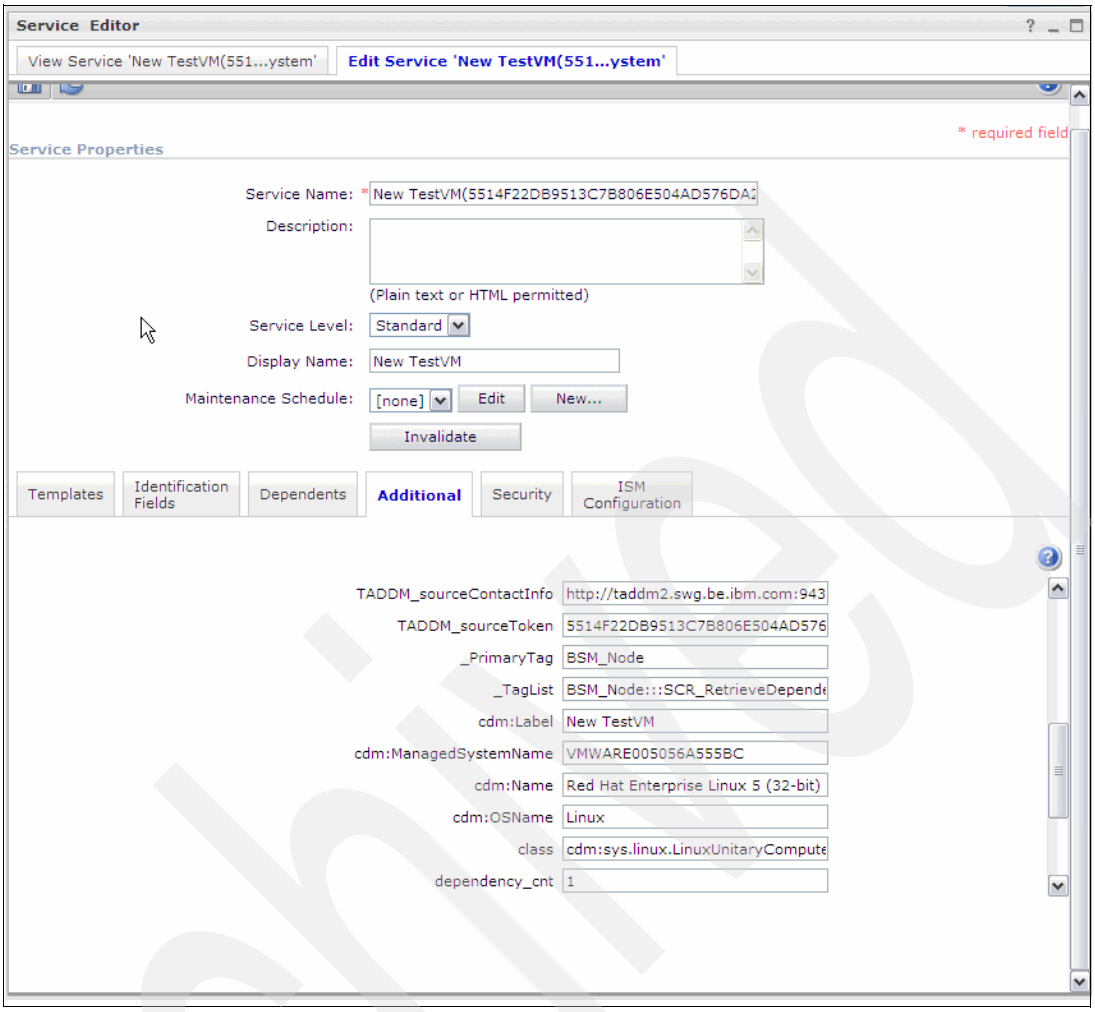


Figure 6-39 Launch-in-context data

The server is also automatically added to an existing Tivoli Application Dependency Discovery Manager Application or Service after running the **`cmdbdiscovery.sh -b -r`** from the `XMLToolkit/bin` directory. This way, any changes to an existing TADDM service are updated in Tivoli Business Service Manager.

Archive

Command description: Schedule, cancel, or query the status of an import request. To get the command syntax, run the following command:

```
./cmdbdiscovery.sh -h
```

Usage: **`cmdbdiscovery -b [-r] [-e] | -c | -s | -r [-e] | -v`**

- **-b** Start a bulk discovery -c cancel a bulk discovery
- **-s** Query discovery status
- **-r** Run the bulk or delta immediately rather than waiting for the next polling interval. If not paired with -b, then applies to a delta request.
- **-e** Before initiating the import, rebuild all of the explicit relationships in the TADDM database. This process can be lengthy, so it should only be used when you believe that the relationship database is corrupted. TADDM 7.1.1 or later required.
- **-v** Apply the latest XML configuration changes against the next time that the toolkit is started.

3. Verify the existence of this new component of the TADDM service instance by navigating to the Imported Business Services in the Service Tree, and then expanding the Service Tree to where the server has been placed in your business context. Figure 6-40 shows the server that was added to the TADDM Service.

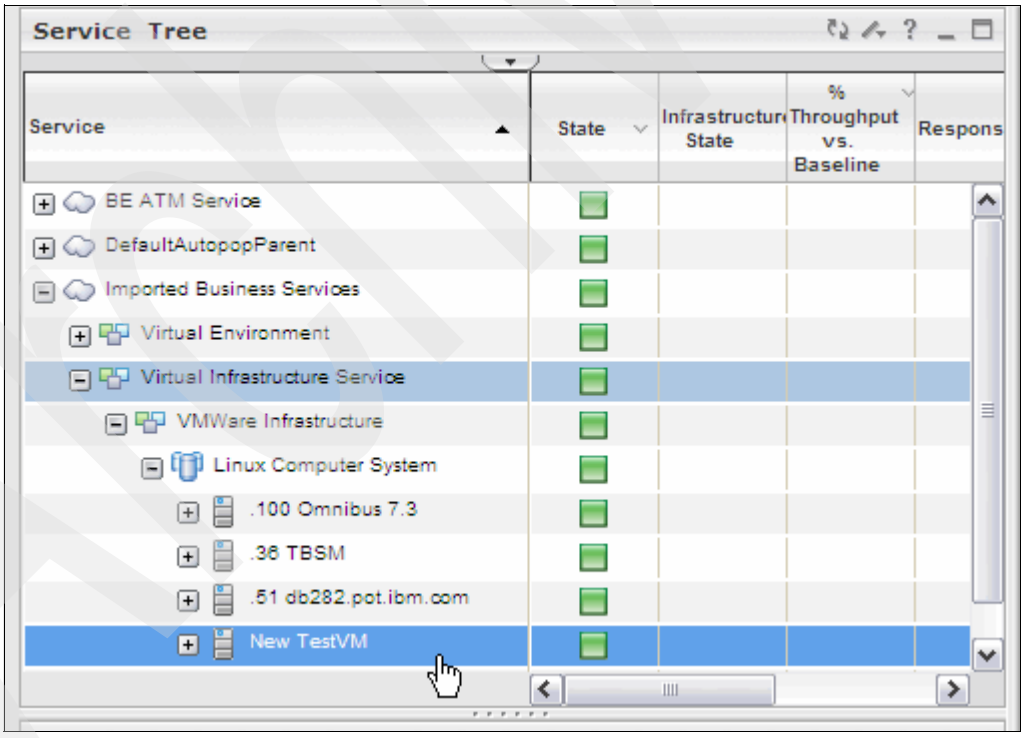


Figure 6-40 Updated Service Tree

6.7 Summary

Figure 6-41 shows the quick summary of the solution.

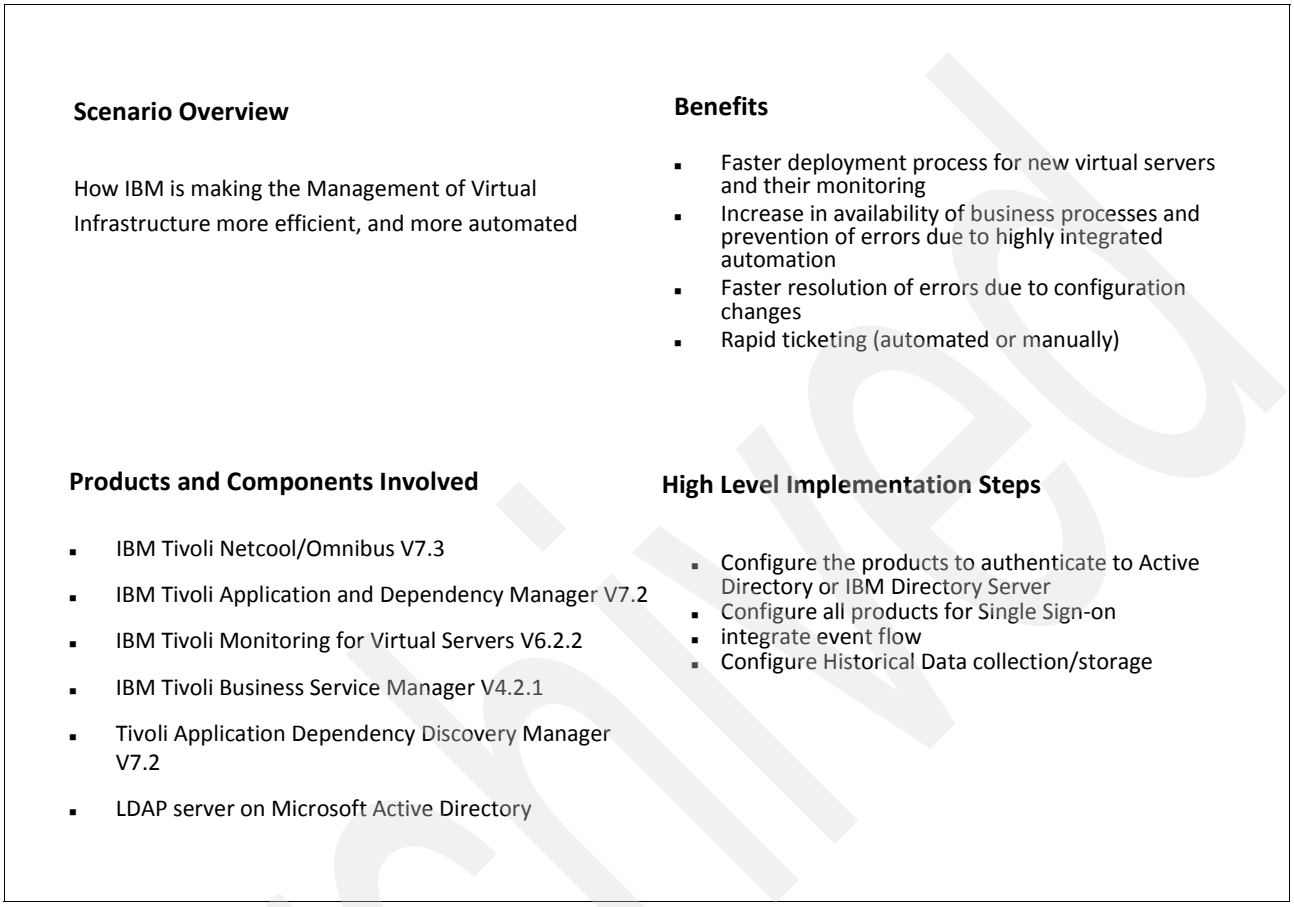


Figure 6-41 Quick summary

Archived

Archived

Part 3

Tivoli Integrated Portal integration scenarios

In this part, we cover the Tivoli Integrated Portal integration scenarios.

This part contains the following chapters:

- Chapter 7, “Operational drill down” on page 169
- Chapter 8, “Tivoli reporting integration: IBM Tivoli Monitoring and TADDM” on page 199
- Chapter 9, “Tivoli reporting integration: IBM Tivoli Monitoring and IBM Tivoli Service Request Manager” on page 245
- Chapter 10, “Protecting Tivoli Integrated Portal with Tivoli Access Manager for e-business” on page 291
- Chapter 11, “WebGUI launch to IBM Tivoli Monitoring” on page 303

© Copyright IBM Corp. 2011. All rights reserved.

Archived

7

Operational drill down

This chapter describes how various Tivoli products can be integrated seamlessly through Tivoli Integrated Portal. We provide a couple of scenarios where we show how an operator can use Tivoli Integrated Portal console to monitor business systems and be able to drill down to solve potential problems.

This chapter contains the following topics:

- 7.1, “Scenario overview” on page 170
- 7.2, “Products involved” on page 170
- 7.3, “Benefits” on page 171
- 7.4, “Scenario setup” on page 171
- 7.5, “Sample walk-through” on page 187
- 7.6, “Summary” on page 197

Archived

© Copyright IBM Corp. 2011. All rights reserved.

7.1 Scenario overview

This scenario, shown in Figure 7-1, describes how Tivoli Integrated Portal can be used as a single console for various Tivoli products (such as Tivoli Business Service Manager, Tivoli Netcool/OMNIbus, and Tivoli Network Manager) to have a dashboard view of the systems so that the operator can be alerted quickly when systems are not operating as they should be.

This scenario also describes how the integration between Tivoli Business Service Manager, IBM Tivoli Composite Application Manager for Transactions and Tivoli Monitoring helps the operators to seamlessly drill down to solve potential problems.

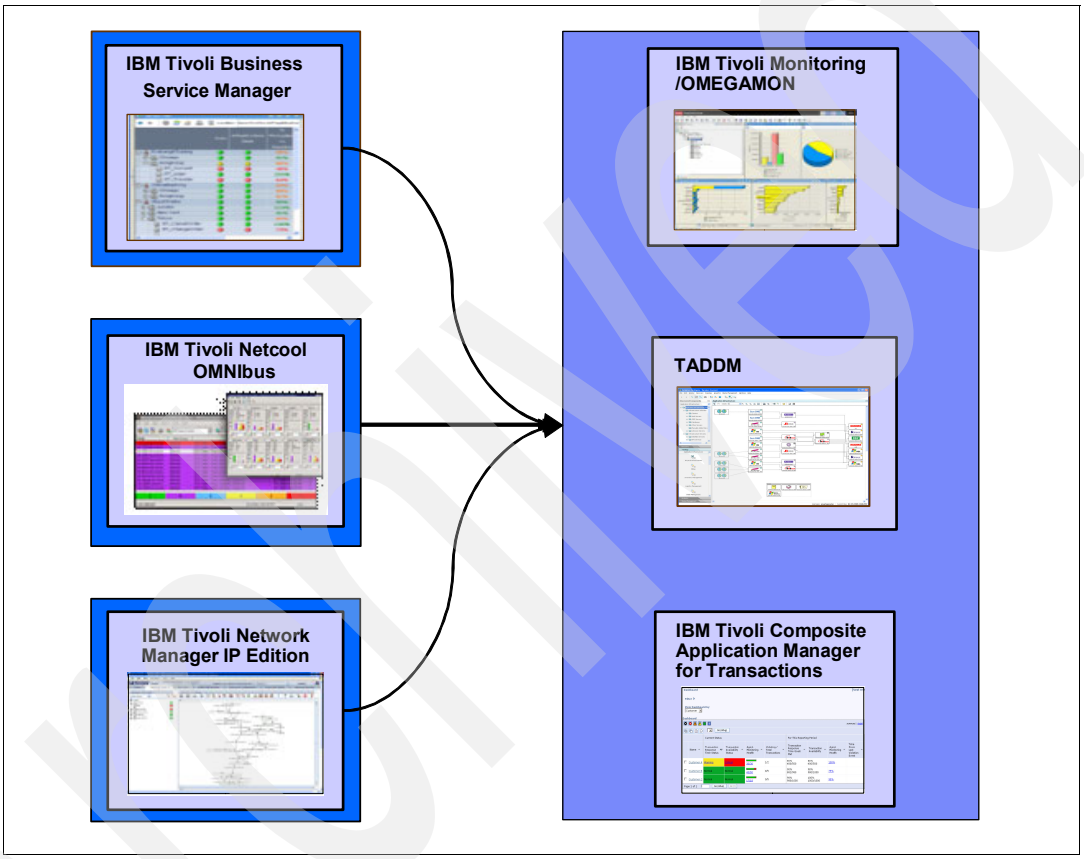


Figure 7-1 Scenario overview

7.2 Products involved

The products involved in this scenario are as follows:

- IBM Tivoli Netcool/OMNIbus 7.3 delivers a real-time, central point of management for all IT and network operations, maximizing service availability and reliability through automated event correlation, problem isolation, and resolution capabilities, enabling rapid identification and resolution of the most critical problems. It collects, consolidates, and correlates events and alarms from a wide variety of IT and networking environments together with events arising from performance and network management applications, providing a consolidated view of the entire managed environment

- IBM Tivoli Business Service Manager 4.2.1 delivers the real-time information that is needed to respond to alerts effectively in line with business requirements and optionally to meet service level agreements.
- IBM Tivoli Network Manager IP Edition 3.8 provides features necessary to manage network environments, including next generation network technology deployed by communication service provider, enterprise networks, and data center networking. These features include network discovery, device polling, including storage of polled SNMP data for reporting and analysis, and network topology visualization. In addition, Tivoli Network Manager is seamlessly integrated with Tivoli Netcool/OMNIbus to enrich network events with topology and other network data, and perform additional event correlation and root-cause analysis in real time to focus network operators on network problems affecting service.
- IBM Tivoli Monitoring 6.2.2 monitors resources (managed systems) and sends alerts when given conditions are met.
- IBM Tivoli Composite Application Manager for Transactions 7.2.0.1 delivers a comprehensive, unified transaction tracking management system that runs on a single, consolidated infrastructure with a tightly integrated user interface.

7.3 Benefits

The benefit of having various Tivoli products on a single console is that it reduces the number of consoles that the operators need to be familiar with. The customer also can get dashboard views of the systems, monitor the business systems, and be able to seamlessly drill down to solve potential problems.

Single sign-on and having all the information in one place and the ability of launch in context to other products makes the job of the operator easier and speeds time to resolution. The launch in context passes the name of the resource automatically which makes it less error-prone. This is important for complex environments with lots of server.

This integration lowers the cost of ownership through central user and role administration, and lowers training costs because administrators and operators familiar with the Tivoli Integrated Portal console can adopt new products more easily. It also increases operator efficiency by providing business service dashboards with drill-down features to subject matter expert views, to help solve problems quickly.

7.4 Scenario setup

In this section, we discuss the scenario setup, which involves the following steps:

1. Install IBM Tivoli Netcool/OMNIbus, IBM Tivoli Business Service Manager, and IBM Tivoli Network Manager IP Edition into a single Tivoli Integrated Portal environment.

   This step is an installation overview, with high-level steps.
2. Create custom dashboard view for network management
3. Integrate IBM Tivoli Netcool/OMNIbus, IBM Tivoli Business Service Manager and IBM Tivoli Composite Application Manager for Transactions
4. Create custom dashboard view for transaction management

Archived

7.4.1 Installing products into a single Tivoli Integrated Portal environment

In this section, although we do not describe the details of the individual product installations, we do explain the high-level steps, the common default variables, and the post-installation steps.

Installation of IBM Tivoli Network Manager IP Edition FP2 patch does not work on a system that has an existing Tivoli Business Service Manager 4.2.1. To avoid these issues, we install the products in the following order:

1. Install IBM Tivoli Netcool/OMNIbus 7.3.
2. Install IBM Tivoli Network Manager IP Edition 3.8 with FP2.
3. Install IBM Tivoli Business Service Manager 4.2.1 with FP1.

The products are installed as non-root user. Although IBM Tivoli Netcool/OMNIbus and IBM Tivoli Network Manager can be installed as root, we set up as non-root user to be able to integrate with IBM Tivoli Business Service Manager, which is installed only as non-root user.

Note: For this scenario, we install both GUI and core components on a single server. However, we can have a distributed environment with IBM Tivoli Netcool/OMNIbus core, IBM Tivoli Network Manager core, and IBM Tivoli Business Data Server installed on separate servers. For single Tivoli Integrated Portal integrated instance, we install only the IBM Tivoli Netcool/OMNIbus WebGUI, IBM Tivoli Network Manager GUI, and Tivoli Business Service Manager Dashboard server on a single server.

Install IBM Tivoli Netcool/OMNIbus 7.3

The IBM Tivoli Netcool/OMNIbus core and IBM Tivoli Netcool/OMNIbus WebGUI components are distributed as two separate installation packages. If OMNIbus core and WebGUI must be installed on the same machine, create a `WebGUI` directory in the directory that has `launchpad.exe` file (Windows) or `launchpad.sh` file (UNIX or Linux) of the IBM Tivoli Netcool/OMNIbus core. This step is needed for the launchpad program to be able to detect the installers for both core and WebGUI components.

The steps are as follows:

1. Perform the default installation of IBM Tivoli Netcool/OMNIbus core. Be sure the host name can be properly resolved through DNS or the hosts file.

   IBM Tivoli Netcool/OMNIbus core installs the components listed in Figure 7-2 on page 173.

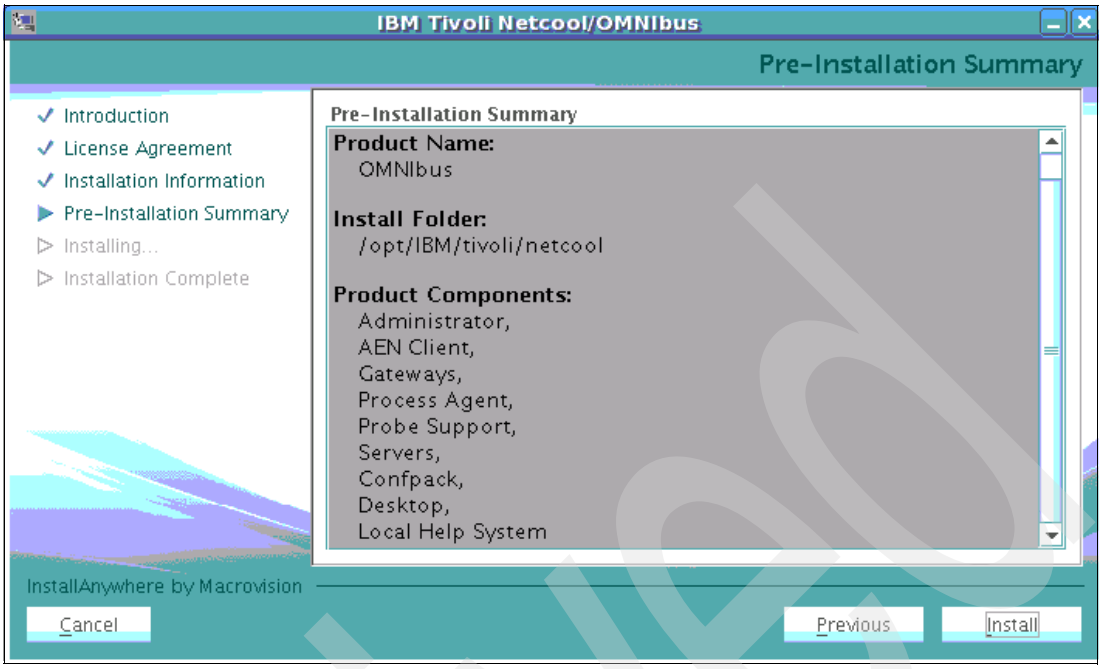


Figure 7-2 IBM Tivoli Netcool/OMNIbus core

2. Set the environment variables NCHOME, OMNIHOME and PATH after the IBM Tivoli Netcool/OMNIbus core installation, based on your installation path. See Example 7-1.

   Example 7-1 Set the environment variables

```
setenv NCHOME /opt/IBM/tivoli/netcool
setenv OMNIHOME $NCOME/omnibus
setenv PATH $OMNIHOME/bin:$PATH
```

3. Run the following command to list all the dynamic dependencies on a non-Windows setup and obtain the missing libraries:

   ```
   ldd $NCHOME/omnibus/platform/arch/bin/nco *
   ```

4. Install the missing libraries.
5. Ensure that the IBM Tivoli Netcool/OMNIbus database is created and is started before starting the installation.

   The default port for Tivoli Integrated Portal is 16310. Tivoli Integrated Portal uses 10 port numbers. If default port is 16310, it uses 16310, 16311, 16312, 16313, 16315, 16316, 16318, 16320, 16322, and 16323.

   Ensure that the ports are available before starting the installation. If they are not available, change the default port.
6. Perform the default installation of IBM Tivoli Netcool/OMNIbus WebGUI.
7. WebGUI default installation installs the components listed in Figure 7-3 on page 174.

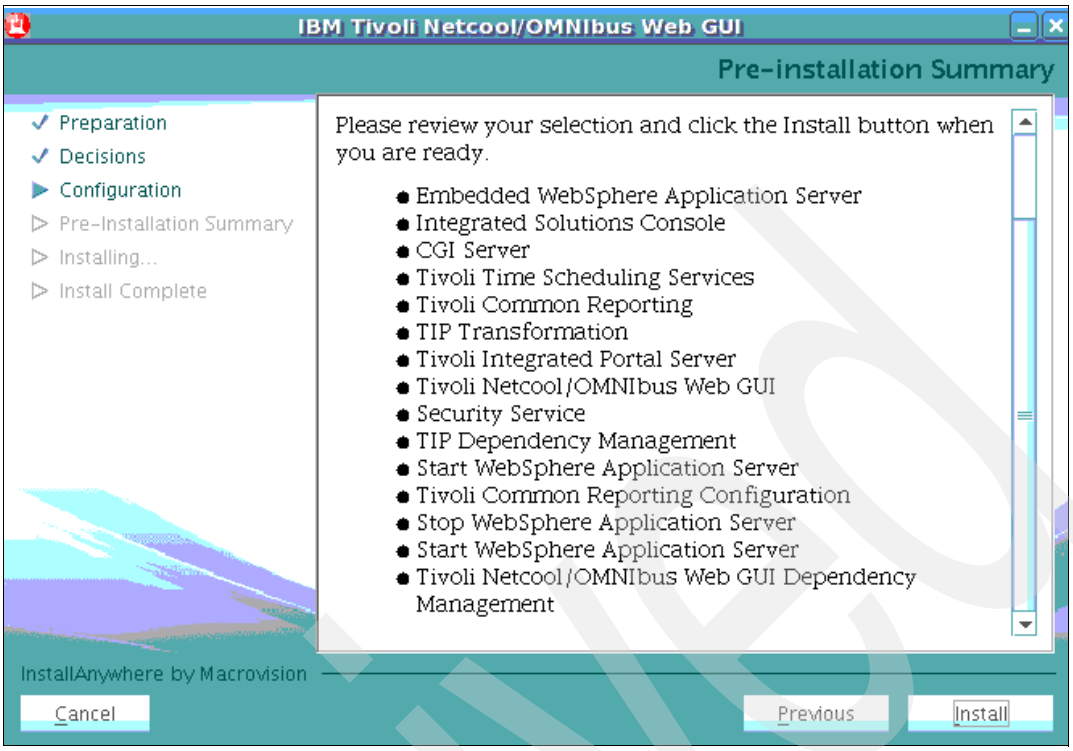


Figure 7-3 WebGUI default installation

Install IBM Tivoli Network Manager IP Edition 3.8 with FP2

The steps are as follows:

1. WebSphere Application Server Update Installer is a prerequisite for installing IBM Tivoli Network Manager IP Edition 3.8. WebSphere Application Server Update Installer is bundled in the CD image of IBM Tivoli Network Manager. Extract the image and perform the default installation of WebSphere Application Server Update Installer.
2. Perform the Prerequisite checking before starting the IBM Tivoli Network Manager installation for Network Manager core components and Topology database. Fix any issues found. For DNS issues, verify that both IP address to name and name to IP address resolve correctly. The following checks are performed by the Prerequisite checker for the DNS:

   ```
   ping 127.0.0,1
   ping <systemname>
   ping <ip-address>
   nslookup <sysname> | grep <ip-address>
   nslookup <ip-address> | grep <ip-address>
   ipconfig -a | grep <ip-address>
   ```

3. Because IBM Tivoli Netcool/OMNIbus is already installed, to integrate with this existing installation, you must configure the IBM Tivoli Netcool/OMNIbus installation before installing the IBM Tivoli Network Manager.
4. To use FIPS Compliance, install the supported database before installing IBM Tivoli Network Manager.
5. Perform custom installation of IBM Tivoli Network Manager:
   a. Select the multi-server installation and choose to install Core components, Web Applications, and Topology database.

b. Reuse the existing Tivoli Integrated Portal installation and provide all the necessary information.

c. Ignore the comment for non-root installation and Network Manager components found.

The components listed in Figure 7-4 are installed.

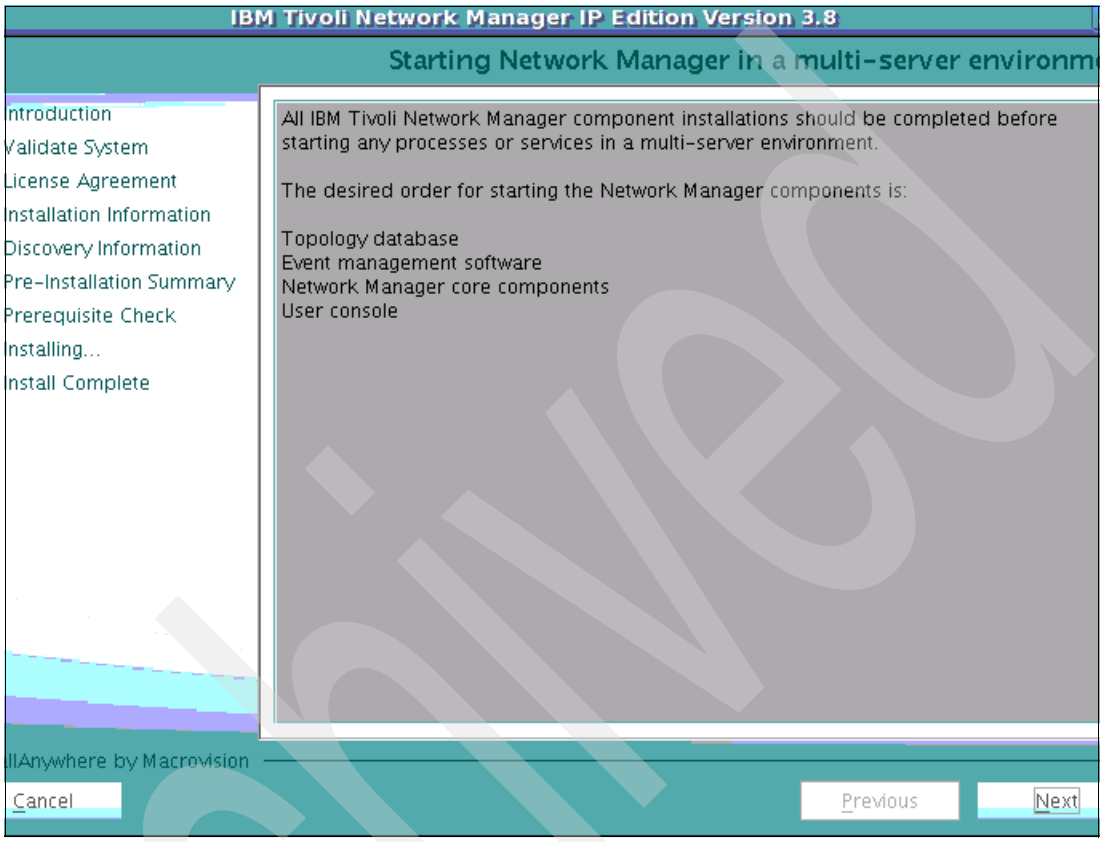


Figure 7-4 Components installed

6. Perform the post installation step for setting UID for root user because we have installed the product as non-root user to be able to integrate with IBM Tivoli Business Service Manager.
7. Perform the FP2 upgrade for IBM Tivoli Network Manager. Make sure NCHOME and TIPHOME environment variables are set before starting the upgrade. Perform all the post installation steps.

Install IBM Tivoli Business Service Manager 4.2.1 with FP1

The steps are as follows:

1. Perform the Advanced installation of IBM Tivoli Business Service Manager and choose to install Data server and Dashboard server. The Deploy Charts Services for Tivoli Monitoring option is disabled if it was already installed as part of IBM Tivoli Network Manager.
2. After the installation, update the IBM Tivoli Netcool/OMNIbus ObjectServer schema by running the import_schema script because we are installing to an existing ObjectServer.
3. Perform the installation and post installation of IBM Tivoli Business Service Manager Discovery Library Toolkit

Archived

4. Perform the installation and post installation of IBM Tivoli Business Service Manager 4.2.1 FP1.
5. Install the Discovery Library Toolkit service templates. Use the following command on UNIX for installing the templates:

```
cat /opt/IBM/tivoli/tbsm/install/BSM_Templates.radsh |
/opt/IBM/tivoli/tbsm/bin/rad_radshell
```

7.4.2 Creating a custom dashboard view for network management

In this section, we demonstrate how to build a custom dashboard showing the Service Tree, Active Event List, and Hop View of the network.

Note: Log in as user with IBM Tivoli Business Service Manager roles and IBM Tivoli Network Manager roles assigned.

This step assumes that the network discovery has been completed.

Perform the following steps:

1. Log in to the Tivoli Integrated Portal and navigate to **Administration** → **Service Configuration**.
2. Create the new template as shown in Figure 7-5.



Figure 7-5 New template

3. Create the incoming status rule, as shown in Figure 7-6 on page 177.

Archived

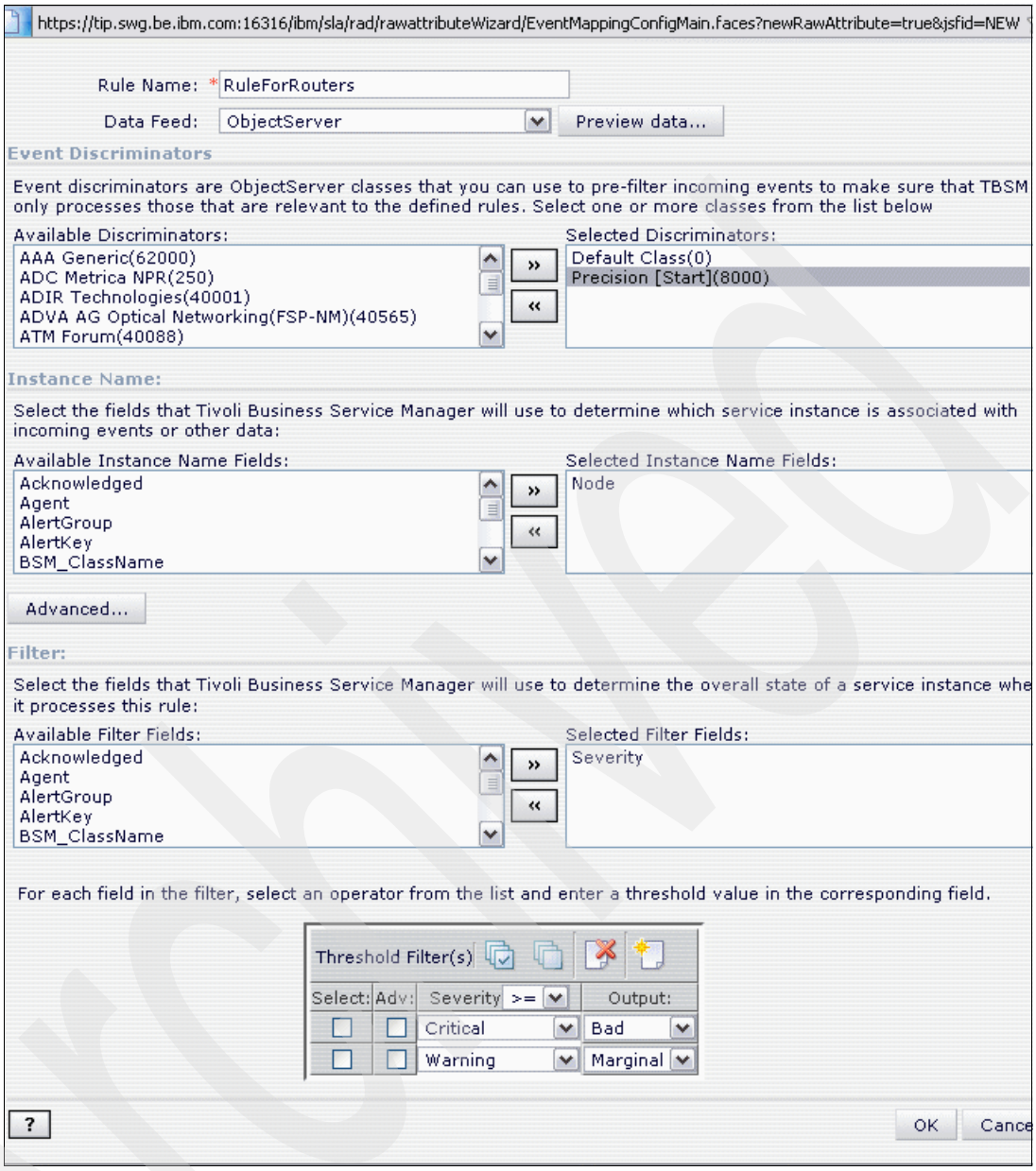


Figure 7-6 Incoming status rule

4. Save the template and create a new Service instance. Use the new template that was created for this purpose. See Figure 7-7 on page 178.

Archived

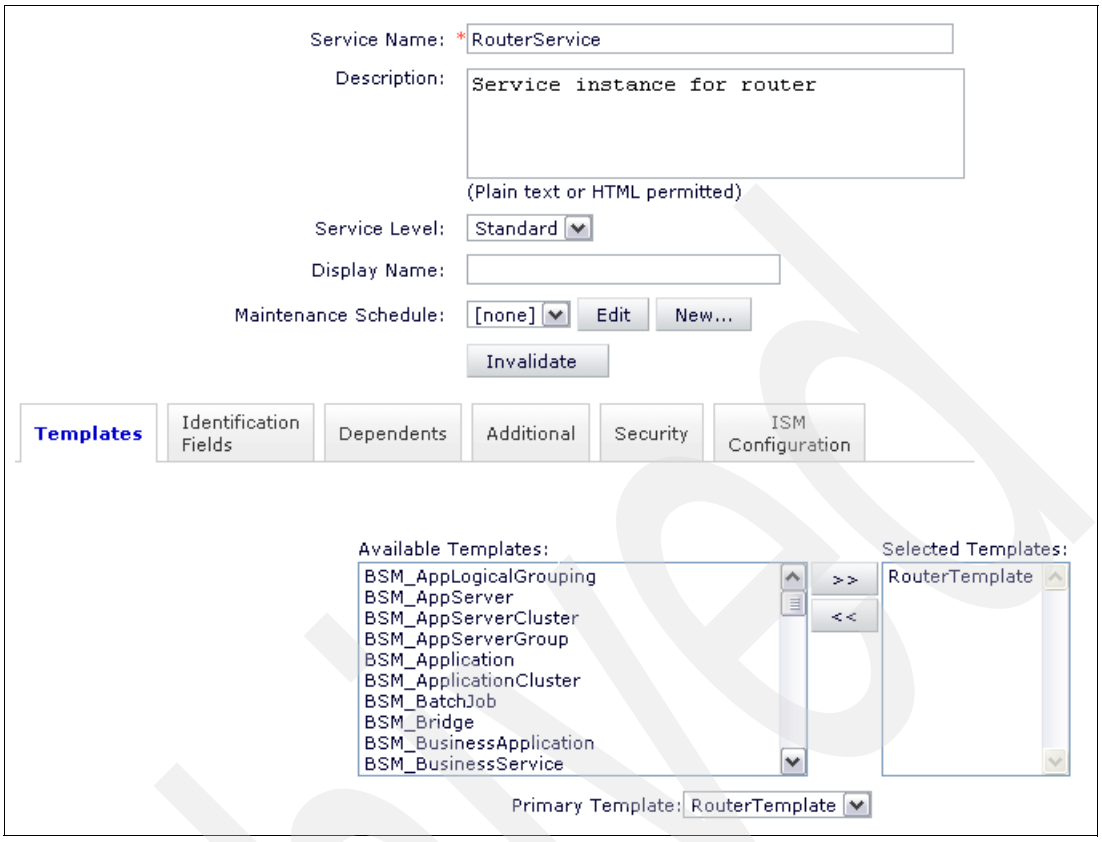


Figure 7-7 RouterService

5. On the Identification Fields tab, enter the value for the node that with which you want this instance to be identified. See Figure 7-8 on page 179.

Archived

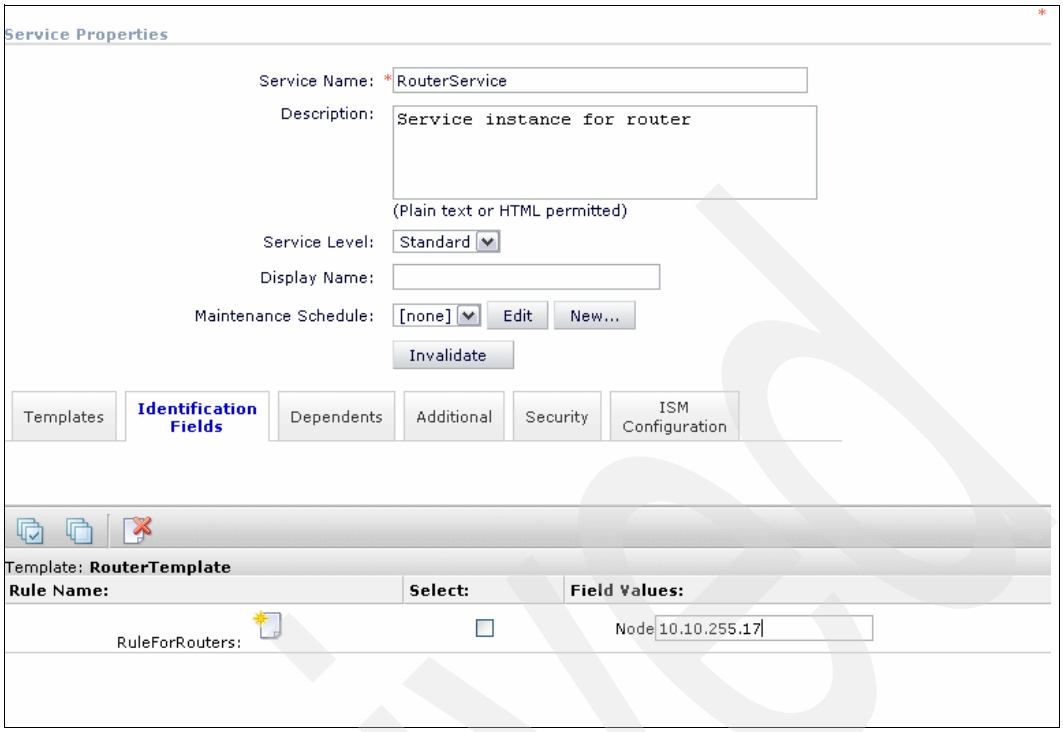


Figure 7-8 Identification fields tab

6. You can create the whole tree structure for your network router components manually, but the service model can also be imported from Tivoli Application Dependency Discovery Manager. Figure 7-9 on page 180 shows the service instance created by selecting the SCR_Network_Routers imported service.

> **Tip:** You can also import network routers from IBM Tivoli Network Manager IP Edition without using TADDM in this scenario.

Archived

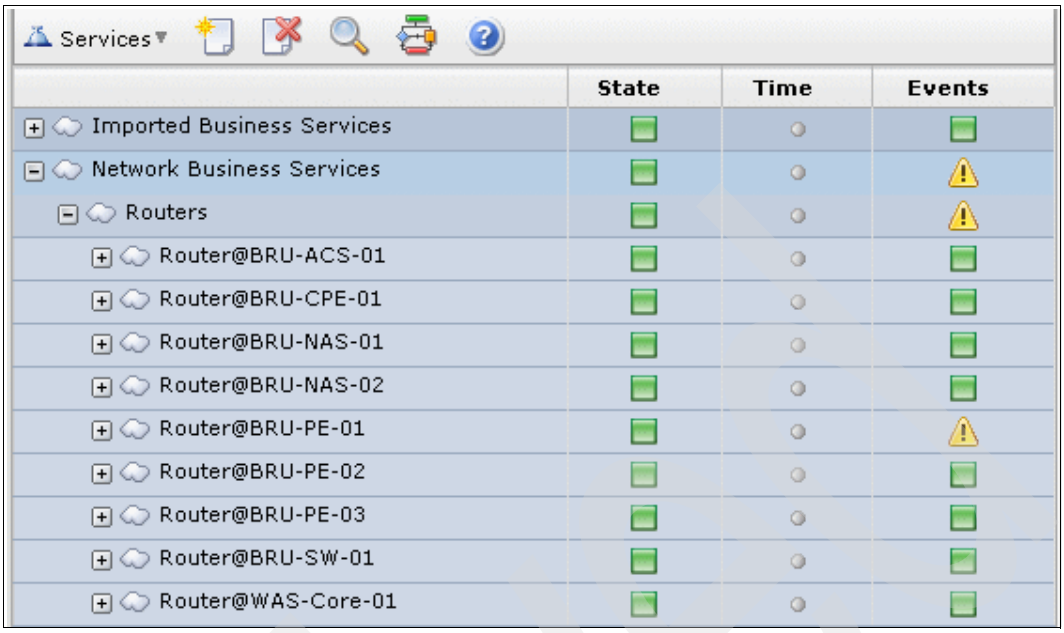


Figure 7-9 Service instance

7. Create a custom dashboard view by navigating to **Settings** → **Page Management**. Click **New Page**. See Figure 7-10.

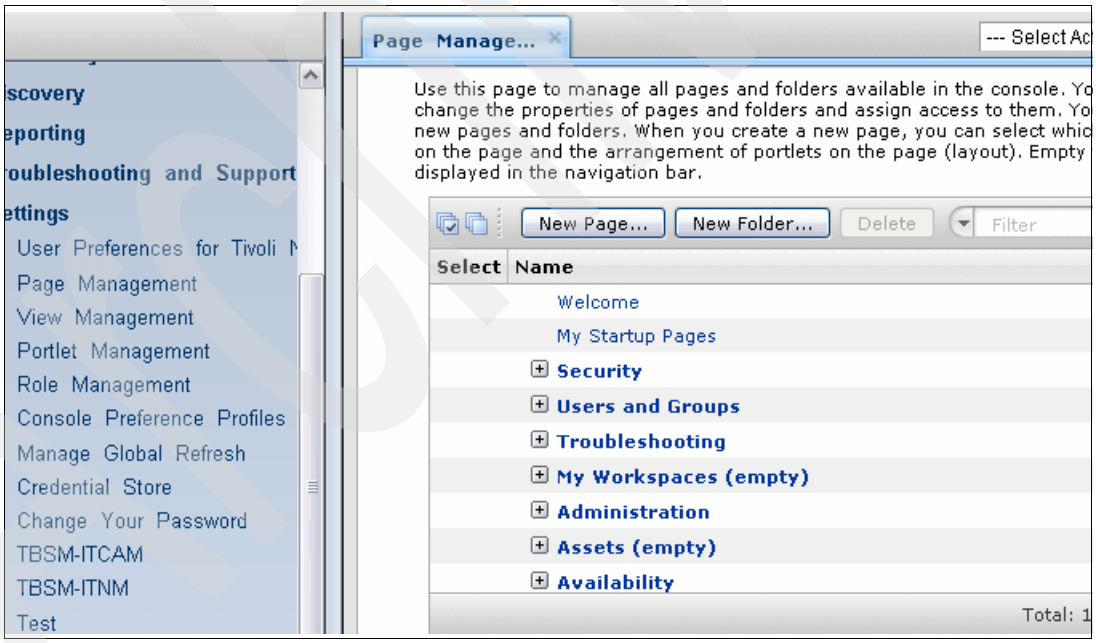


Figure 7-10 New Page

8. Split the page into three parts: first a horizontal split and then a vertical split. Select **Service Tree** portlet, **Active Event List** portlet, and **Hop View** portlet, as shown in Figure 7-11 on page 181.

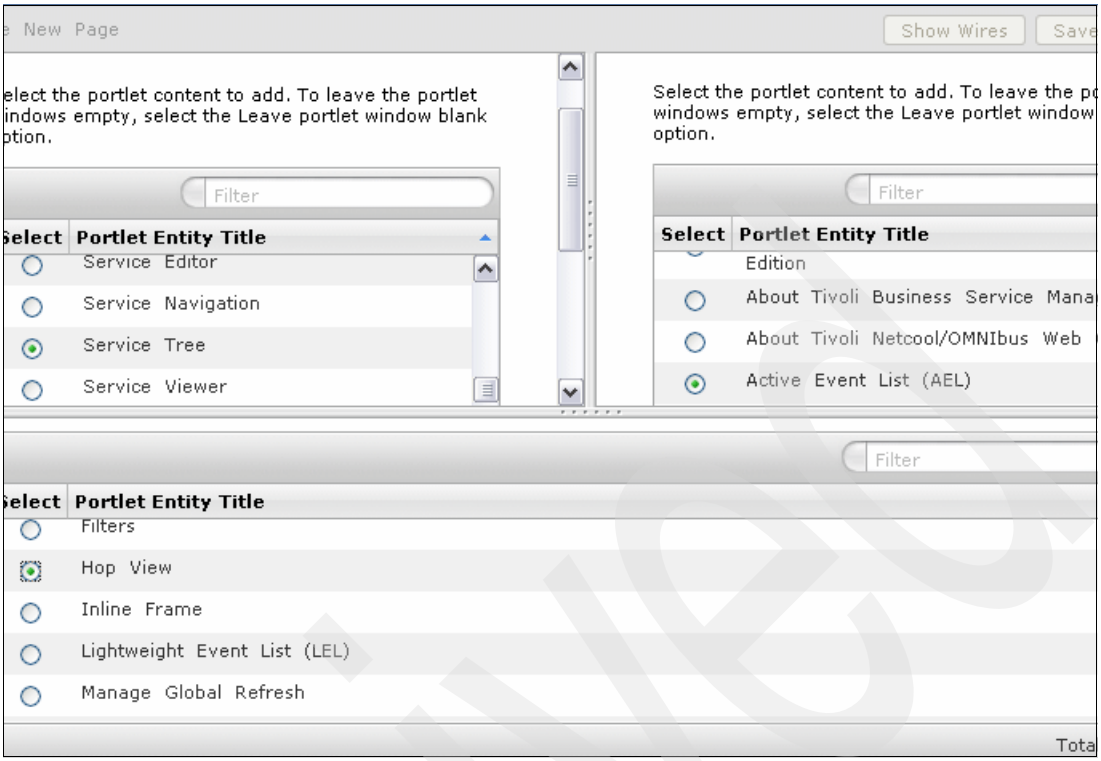


Figure 7-11 Select portlets

9. Save the page.

> **Note:** The page must be saved first before creating wires because the Active Event List portlet will not be a selectable option as one of the portlets.

10. Select **Edit Action** from the Select Action drop-down list at the top of the page. Click **Show Wires** and create a new wire between Service Tree and Active Event List (AEL).

    Select the source event for the wire as **Service Tree** → **NodeClickedOn**. See Figure 7-12 on page 182.

Archive

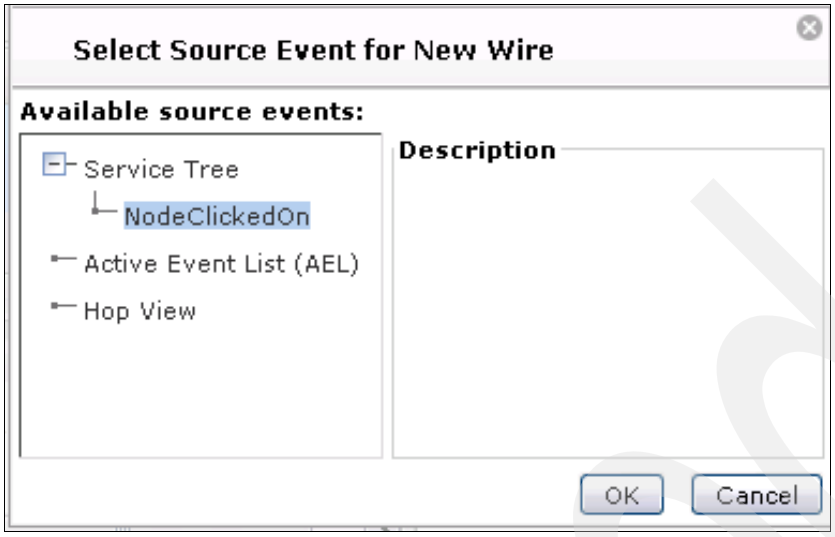


Figure 7-12 NodeClickedOn source

11. On the same page, select **Active Event List (AEL)** as the target portlet for the wire. See Figure 7-13.

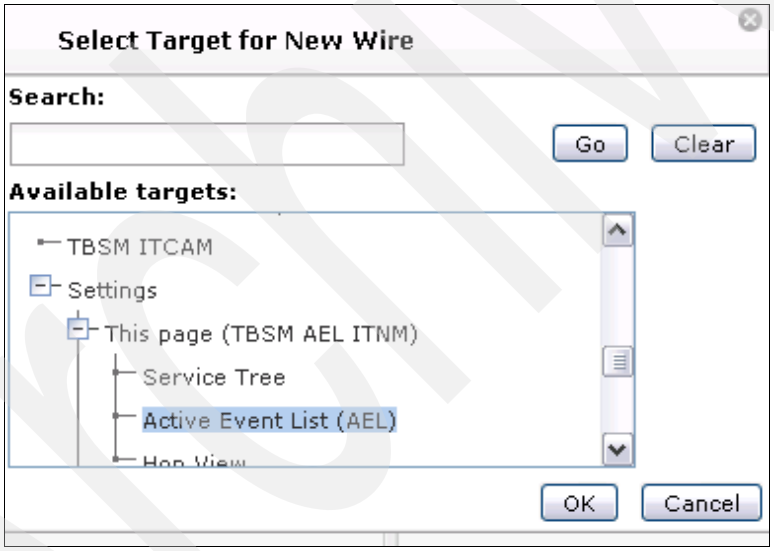


Figure 7-13 Portlet target

12. Select **None for Transformation** and save the page.

> **Note:** Although a wire can be created between Service Tree and Hop View, it does not work because Hop View does not perform any action for the NodeClickedon Event. Service Tree portlet communicates with Hop View portlet through Active Event List (AEL) portlet.

13. Set the title and context for Service Tree portlet by clicking **Edit Preferences** on the portlet.

Archived

14. Enter the title, select the starting instance (Figure 7-14), and save the portlet.

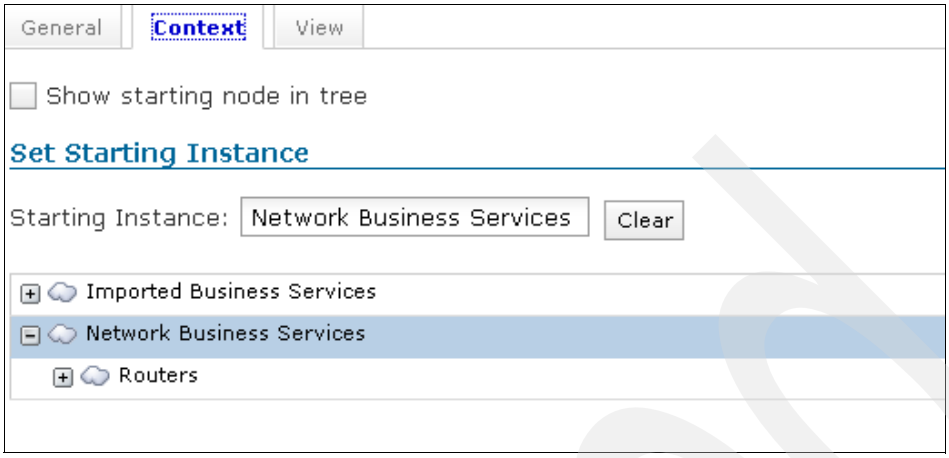


Figure 7-14 Context tab

15. Click **Edit Preferences** for Active Event List portlet. Enable the single-click action for the event list by selecting **BroadcastTopologyContext**. See Figure 7-15. On single-click, in the events, shows the Hop view of the corresponding node. Save the portlet.

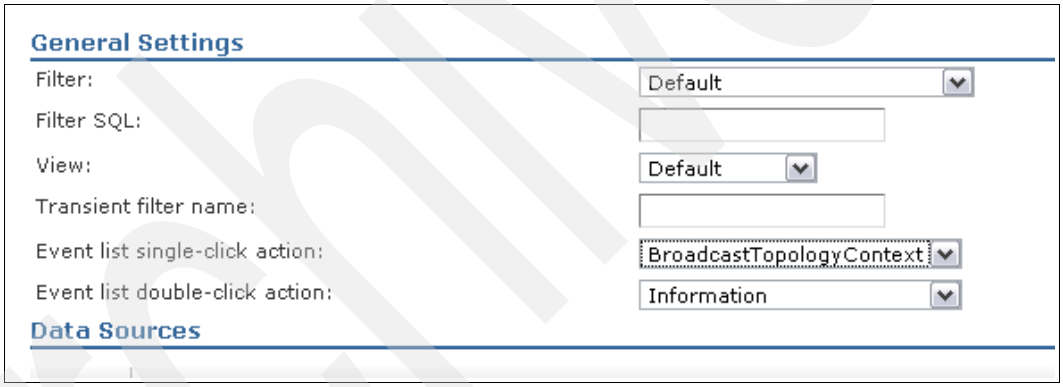


Figure 7-15 General settings

7.4.3 Integrating products to receive transactions data

In this section, we show how to configure IBM Tivoli Netcool/OMNIbus and IBM Tivoli Business Service Manager to receive IBM Tivoli Composite Application Manager for Transactions data.

Notes:

- This procedure assumes that IBM Tivoli Monitoring, IBM Tivoli Composite Application Manager for Transactions, and the data collector agents are installed. It also assumes that IBM Tivoli Composite Application Manager for Transactions is configured to collect data and the historical configuration for these transactions is enabled.
- Another capability for TADDM and IBM Tivoli Composite Application Manager for Transactions integration is for TADDM to tell you which IBM Tivoli Composite Application Manager for Transactions tracking agents should go.

Archived

The steps are as follows:

1. IBM Tivoli Netcool/OMNIbus must be configured for data from IBM Tivoli Composite Application Manager for Transactions. This step is necessary so you can perform the following tasks:
   a. Set the BSM_Identity so that both the defined services in the Discovery Library Adapter and the incoming alerts can be identified.
   b. Add specific columns related to IBM Tivoli Composite Application Manager for Transactions to `alerts.status` table.
2. The `ITCAMT072010_OMNIbus.[zip|tar.gz]` file contains the executable and `.baroc` files that must be run on the OMNIbus server. Execute omnibusUpdater script to process the `kt3`, `kt4`, `kt5`, and `kt6` baroc files:

   ```
   omnibusUpdater.sh -b kt3.baroc kt4.baroc kt5.baroc kt6.baroc -u root
   ```

3. Forward situations from IBM Tivoli Composite Application Manager for Transactions to IBM Tivoli Netcool/OMNIbus using IBM Tivoli Monitoring Tivoli Event Integration Facility.

   Important: Steps 4 and 5 must be executed as the non-root user that was used to install Tivoli Business Service Manager. By default, this user must have permission for the `/var/ibm/` IBM common logging directory.

4. Create the IBM Tivoli Business Service Manager templates that are specific to IBM Tivoli Composite Application Manager for Transactions, policies, and service trees by running the executable in `ITCAMT072010_TBSM.[zip|tar.gz]` file. This step is performed for IBM Tivoli Composite Application Manager for Transactions topology discovery to automatically generate Service Models in Tivoli Business Service Manager. It must be run on the Tivoli Business Service Manager server:

   ```
   tbsmconfig.sh -install -host itm.ibm.com -user itmuser -password mypwd -port
   50000 -tdw WAREHOUS -type DB2 -driver 4
   ```

   Notes:

   - Inspect the following files for the success or failure of the **`tbsmconfig`** script (Use the `tbsmconfig-logging.properties` file to enable tracing for this command):
     - On UNIX:

       ```
       /var/ibm/tivoli/common/BWM/logs/trace-tbsmconfig.log
       ```

     - On Windows:

       ```
       C:\Program Files\IBM\tivoli\common\BWM\logs\trace-tbsmconfig.log
       ```

   - Inspect the following files for the success of failure of **`dlAdapter`** script (Use the `dla-logging.properties` file to enable tracing for this command):
     - On UNIX:

       ```
       /var/ibm/tivoli/common/BMW/logs/trace-dla.log
       ```

     - On Windows:

       ```
       C:\Program Files\IBM\tivoli\common\BWM\logs\trace-dla.log
       ```

5. Create the Discovery Library Adapter books against the IBM Tivoli Composite Application Manager for Transactions data inside the Tivoli Data Warehouse. The executables for this task are included in the `ITCAMT072010_DLA.[zip|tar.gz]` file. Two files, `dlAdapter.sh` and `dla.config.properties`, must be modified before running the command. Instructions are in the `readme.txt` file, which is included in the `.zip` or `.tar.gz` file.

Set the output directory of the Discovery Library Adapter book in the properties file to the location of DL_Filesystem so that the Discovery Library Adapter picks the book for processing (`dlAdapater.sh`).

6. Restart the Discovery Library Adapter:

```
/opt/IBM/tivoli/tbsm/XMLtoolkit/bin/tbsmxmltk stop
/opt/IBM/tivoli/tbsm/XMLtoolkit/bin/tbsmxmltk start
```

7. Check XMLToolkit logs for message about successful processing of the books.

7.4.4 Creating a custom dashboard view for transaction management

To build a custom dashboard for transaction management, perform the following steps:

1. Log in to Tivoli Integrated Portal console, and navigate to **Settings** → **Page Management** to create a New Page.
2. Split the page into three parts: split horizontally first and then vertically. Then, select **Service Tree**, **Service Viewer**, and **Charting** as the portlets. See Figure 7-16.

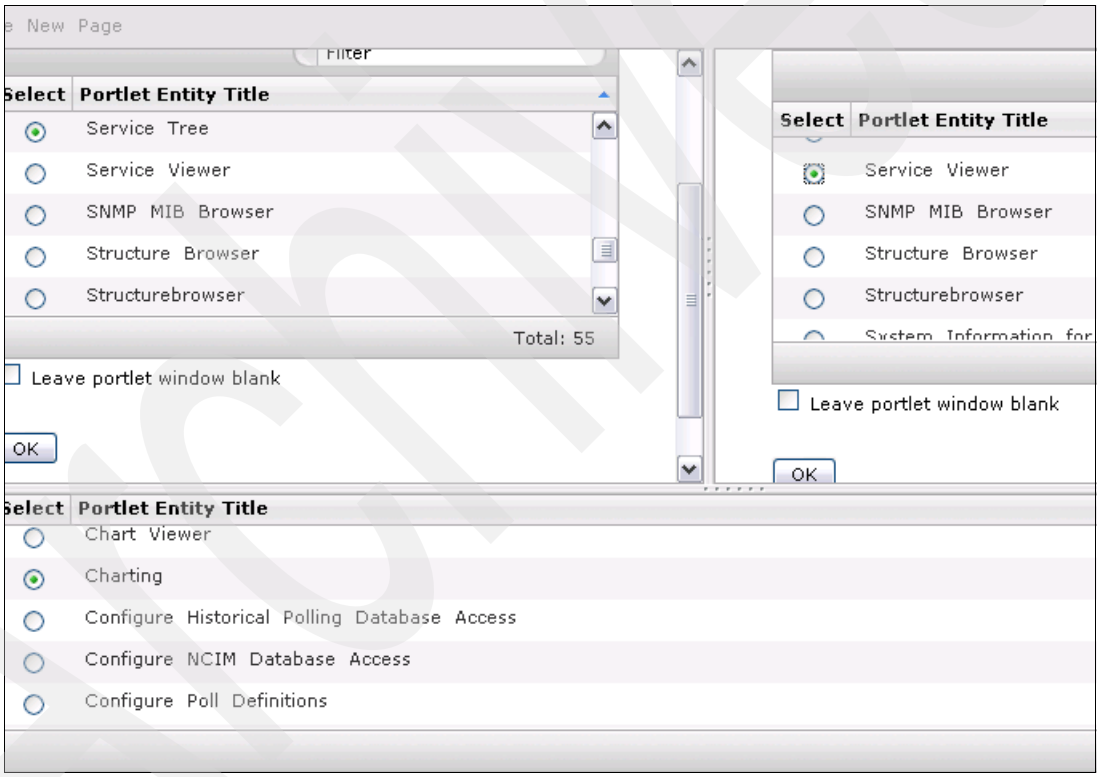


Figure 7-16 Selecting portlets

3. Save the page.
4. On Service Tree portlet (Figure 7-17 on page 186), edit preferences to set the title and context.

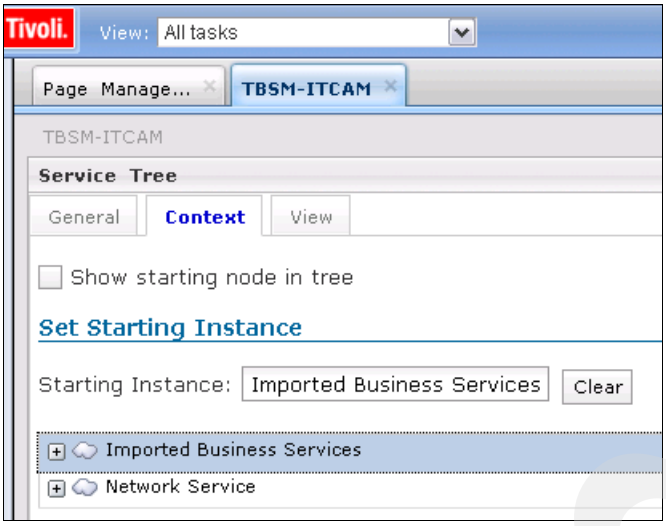


Figure 7-17 TBSM-ITCAM tab

Note: Imported Business Services are created when you import the Discovery Library Adapter book created using **`dlAdapter`** command

5. On the charting portlet, select the **Tivoli Charts** for the type of the chart, and click **Next**. See Figure 7-18.

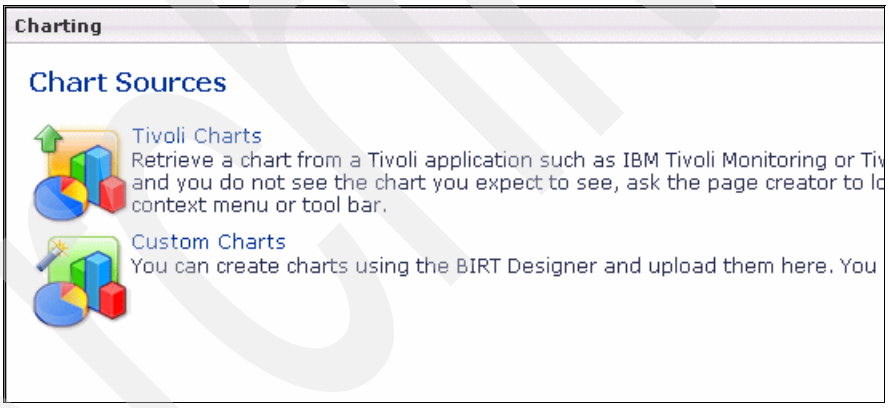


Figure 7-18 Tivoli Charts

6. Select **TBSMChartService** from the Chart Sources list and click **Next**. See Figure 7-19 on page 187.

Archived

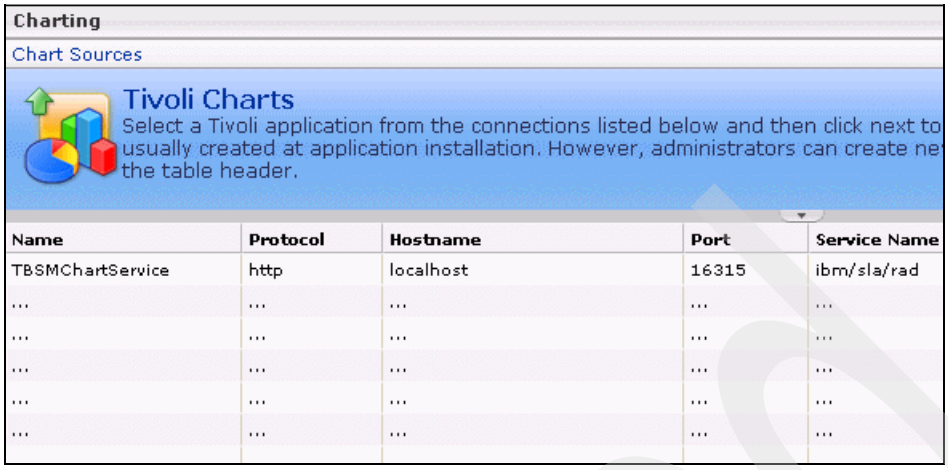


Figure 7-19 Tivoli Charts

7. Select the **CAM_Performance_Trend** chart from the list of charts (Figure 7-20) and then click **Finish**.

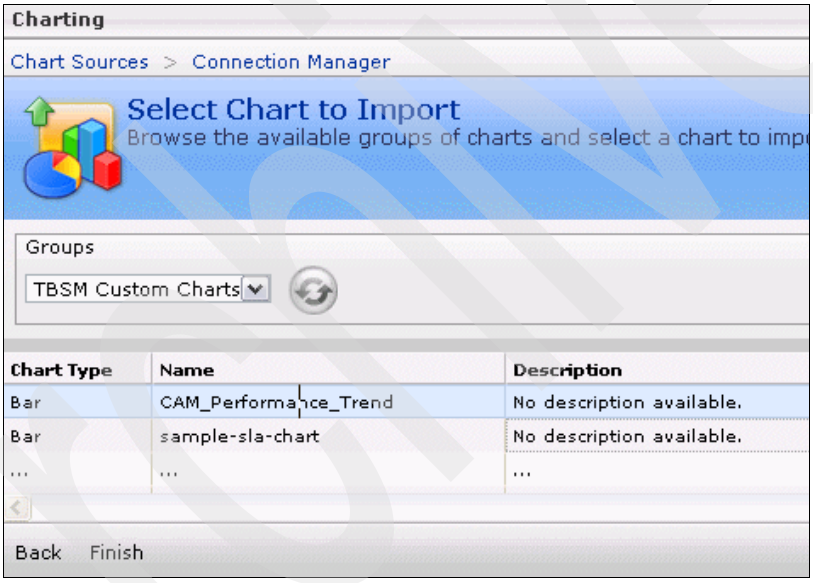


Figure 7-20 CAM_Performance_Trend chart

7.5 Sample walk-through

After performing all prerequisite tasks in 7.4, “Scenario setup” on page 171, the integration allows the execution of the following scenarios:

- Scenario for transaction management
- Scenario for network management

7.5.1 Scenario for transaction management

The following sample scenario is for transaction management:

1. This scenario starts with an operator viewing the custom dashboard for transaction management created in 7.4.4, “Creating a custom dashboard view for transaction management” on page 185. See Figure 7-21.



Figure 7-21 Scenario for transaction management

2. The operator sees that the state for one of the nodes is “critical” (red circle with x through it indicates critical).It is shown in the Service Tree and Service Viewer portlet. The operator can also see the performance trend for this application on the Performance Trend chart. From the chart, the operator learns that all the requests are failed requests. The operator right-clicks on the failed node to see the events affecting the node.See Figure 7-22 on page 189.

Archive

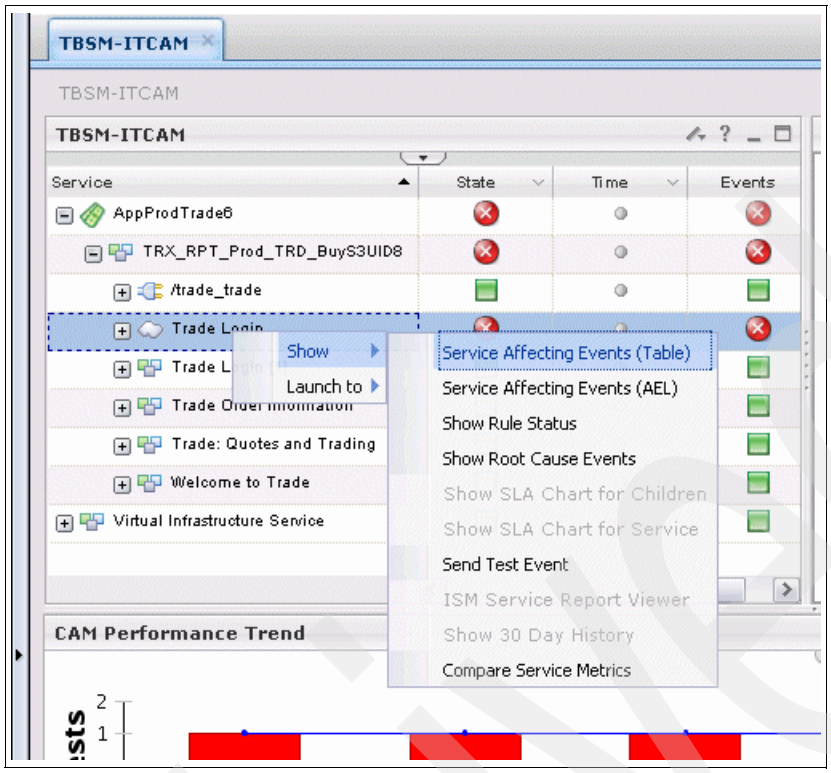


Figure 7-22 Right click the failed node

Figure 7-23 shows the event.

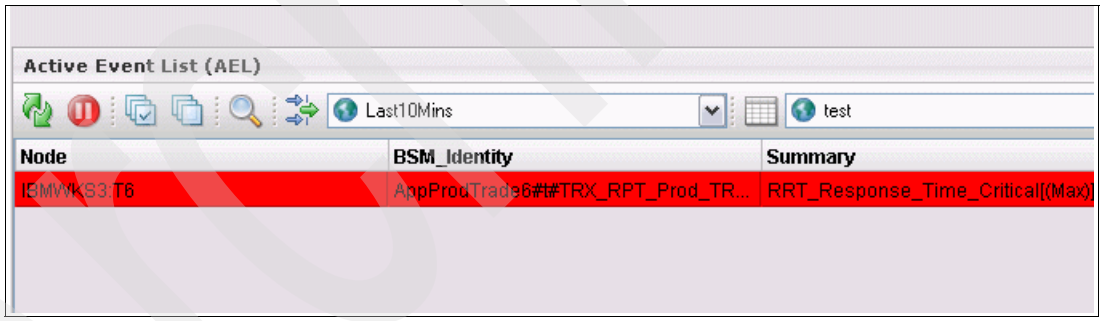


Figure 7-23 Affecting events view

3. The Response Time Critical event is listed in the AEL portlet. The operator can double-click the event to view the detailed information. See Figure 7-24 on page 190.

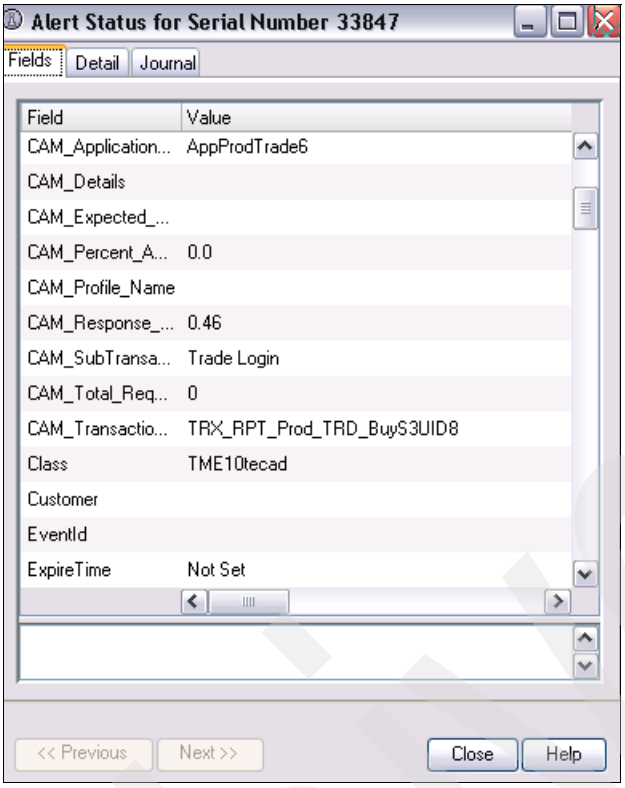


Figure 7-24 Response Time Critical event

4. The operator switches back to the custom dashboard view and right-clicks on the failed node to launch the Application Management console for further analysis of the issue. See Figure 7-25 on page 191.

Archived

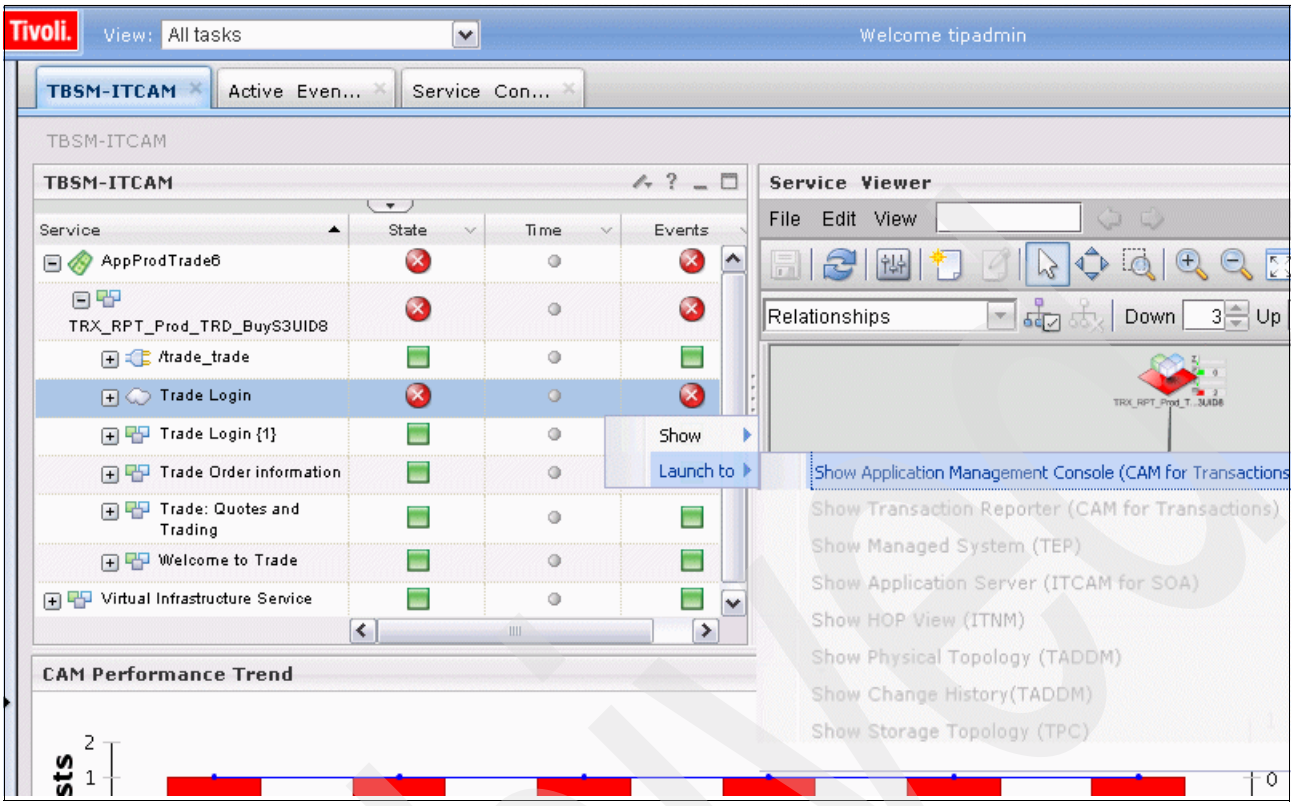


Figure 7-25 Application Management console

The Application Management console, which is part of the Tivoli Enterprise portal, shows the transaction status and server status. See Figure 7-26.

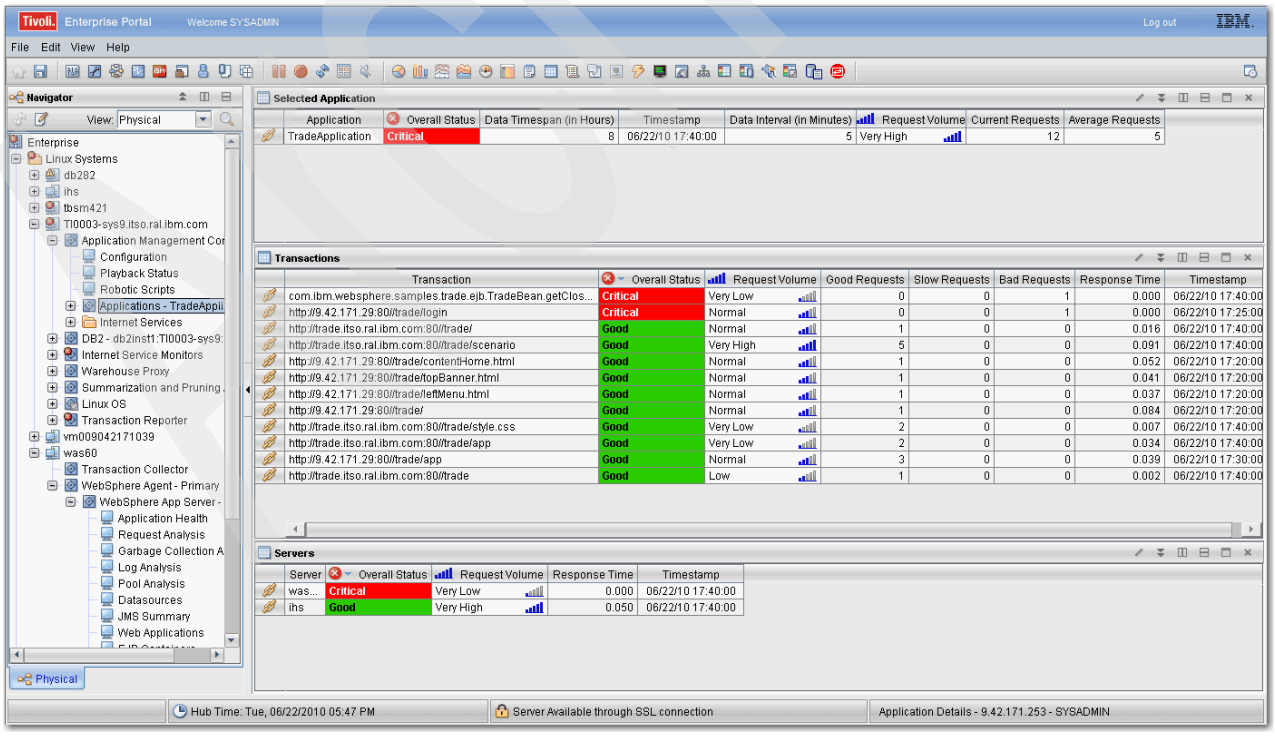


Figure 7-26 Application Management console

5. The WebSphere Application Server is showing a Critical status. The operator right-clicks on the server entry and clicks the **WebSphere Application Server** custom link. See Figure 7-27.

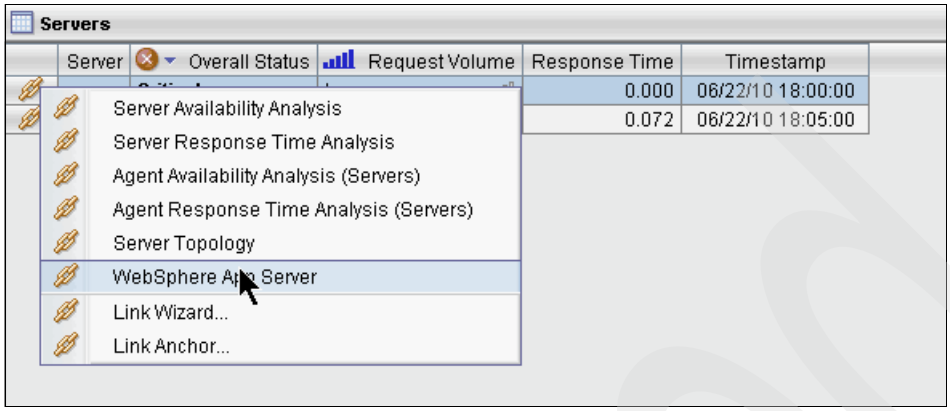


Figure 7-27 WebSphere Application Server custom link

This takes the operator to the WebSphere Application Server Resources console, which shows that a critical situation is generated for the thread pool. See Figure 7-28.

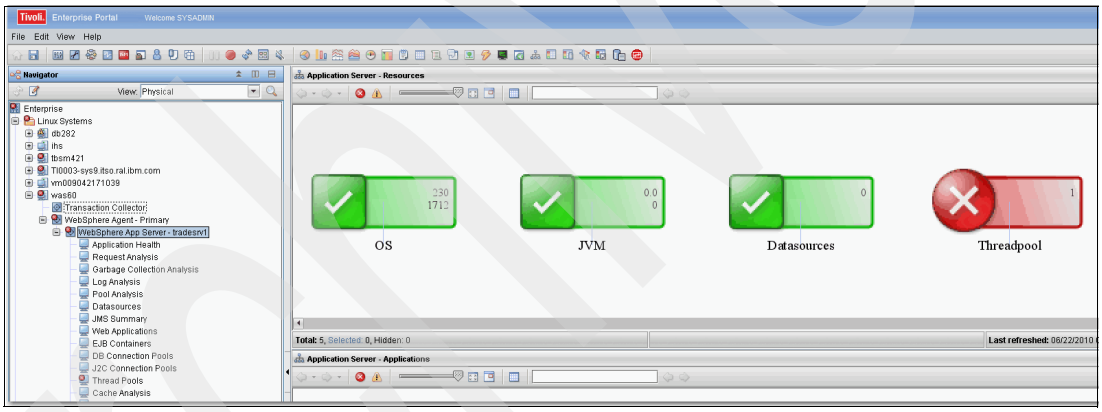


Figure 7-28 WebSphere Application Server Resources console

6. The operator clicks **Thread pools** in the tree to see the detailed information. See Figure 7-29 on page 193.

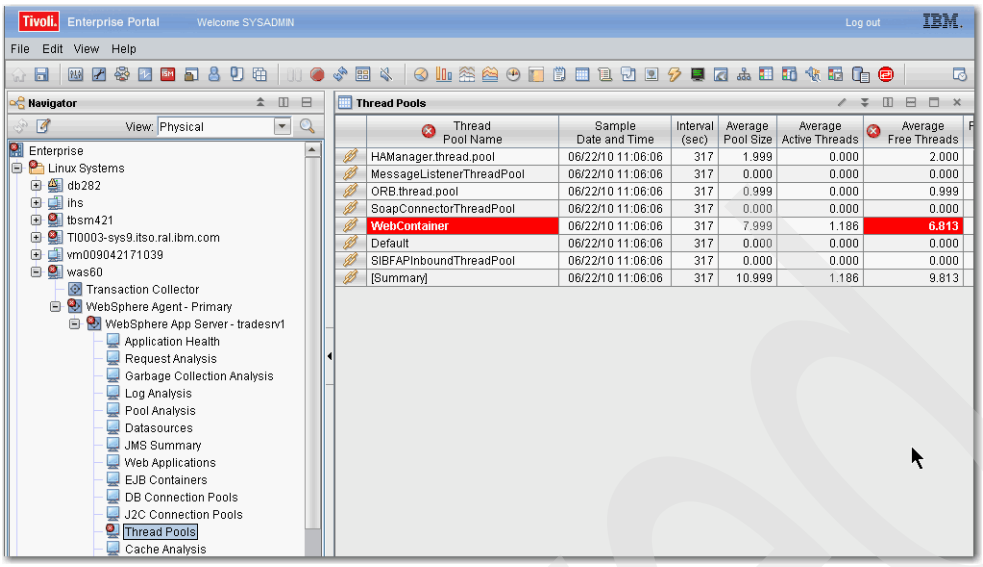


Figure 7-29 Thread pools

7. The operator sees that the WebContainer Thread Pool has crossed the threshold set for Average Free Threads. The operator can also move the mouse pointer over the Thread Pools in the tree to identify the root cause of the issue. See Figure 7-30.

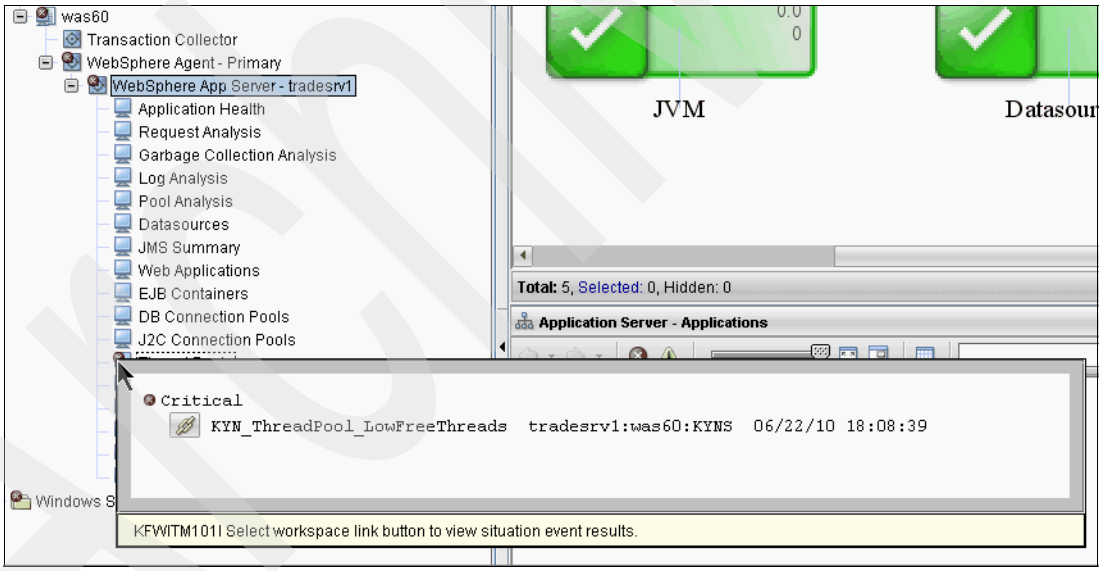


Figure 7-30 Identify the root cause of the issue

8. The operator now can conclude that the transaction is failing with critical response time because of the Webcontainer thread pool issues. The operator can discuss the issue with the SME who then decide how to resolve this problem.

7.5.2 Scenario for network management

The following sample scenario is for network management:

1. This scenario, depicted in Figure 7-31, starts with an operator viewing the custom dashboard for network management, created in 7.4.2, “Creating a custom dashboard view for network management” on page 176.

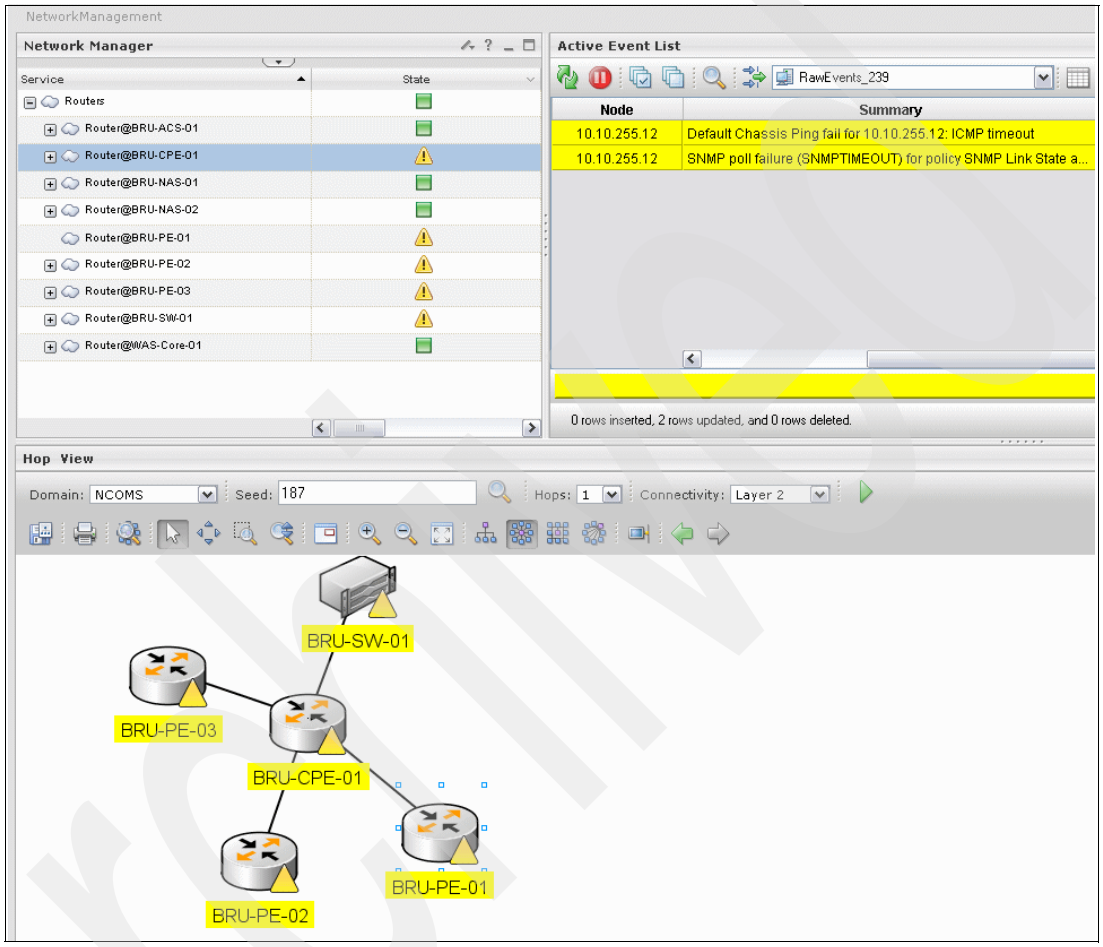


Figure 7-31 Scenario for network management

2. The operator clicks the node in the Service Tree, which shows all the related events in the Active Event List portlet. Clicking the **Active Event List** portlet opens the node in the IBM Tivoli Network Manager Hop View.
3. The operator double-clicks the node in Hop View to open the Structure View (Figure 7-32 on page 195) for that node in a new tab.

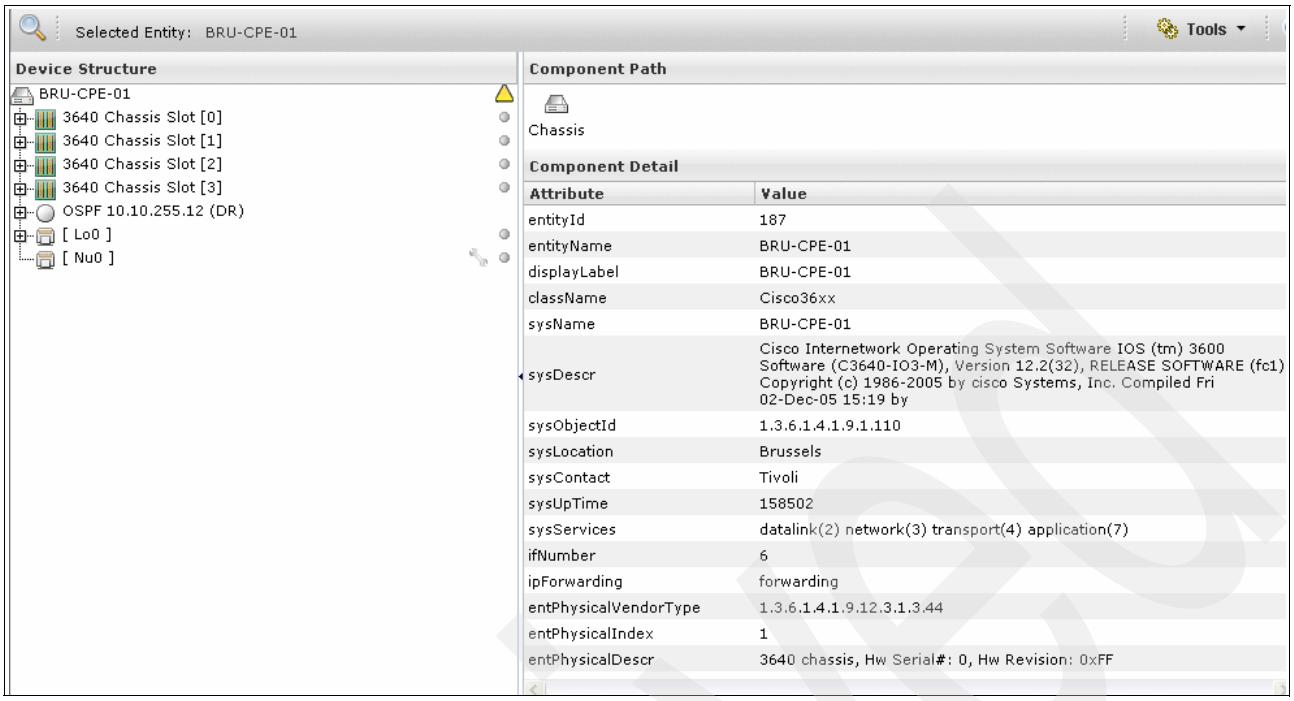


Figure 7-32 Structure View

The operator can right-click the node and use various diagnostic tools that are available in the menu to identify the problem and resolve it. See Figure 7-33 on page 196.

Archive

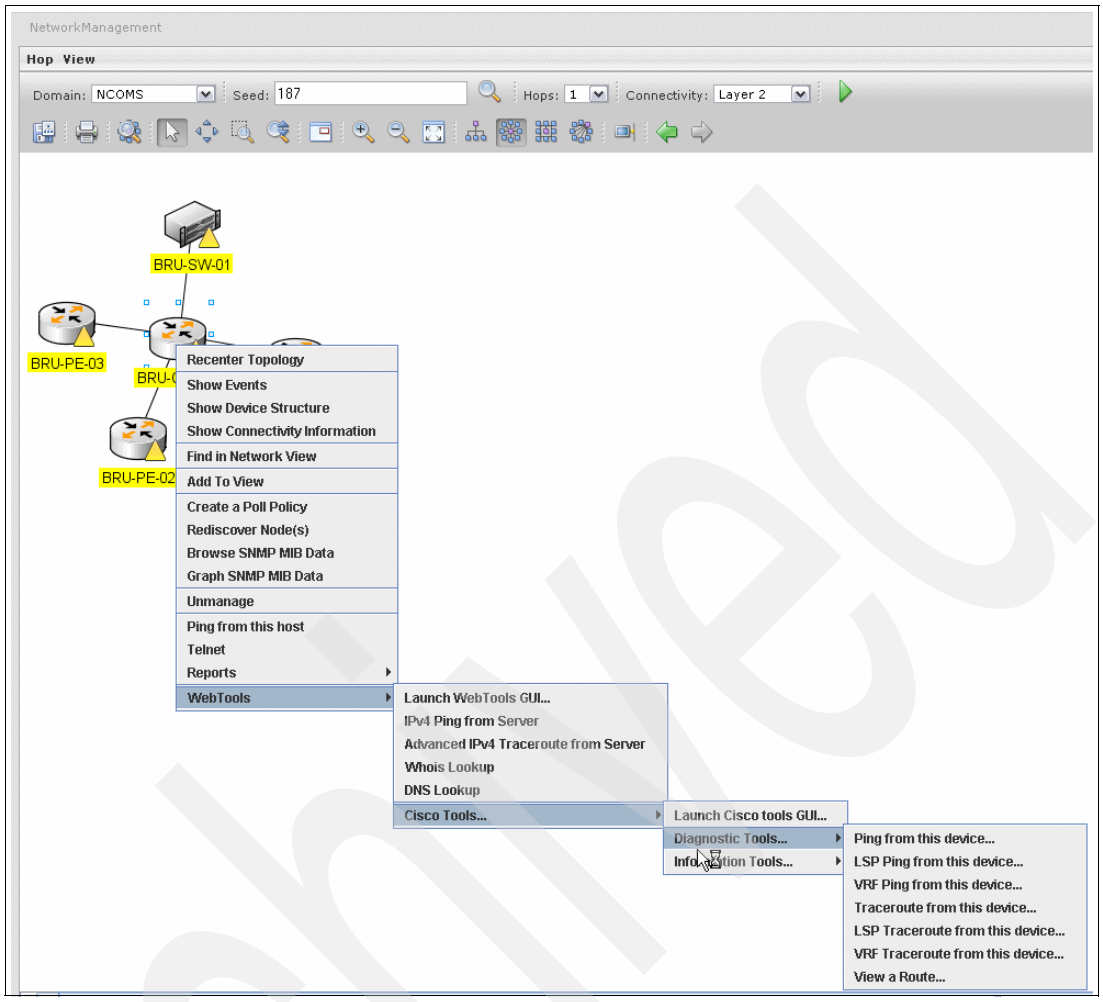


Figure 7-33 Diagnostic Tools

For more information: IBM Tivoli Network Manager and IBM Tivoli Netcool/OMNIbus are tightly integrated products. See *Integration Guide for IBM Tivoli Netcool/OMNIbus, IBM Tivoli Network Manager and IBM Tivoli Netcool Configuration Manager*, SG24-7893 for details about integration capabilities of these products, including GUI, reporting, and launch-in-context level integrations. It also covers integration scenarios for IBM Tivoli Netcool Configuration Manager, which is a product in the IBM network management portfolio.

Archived

7.6 Summary

Figure 7-34 shows the summary of this integration.

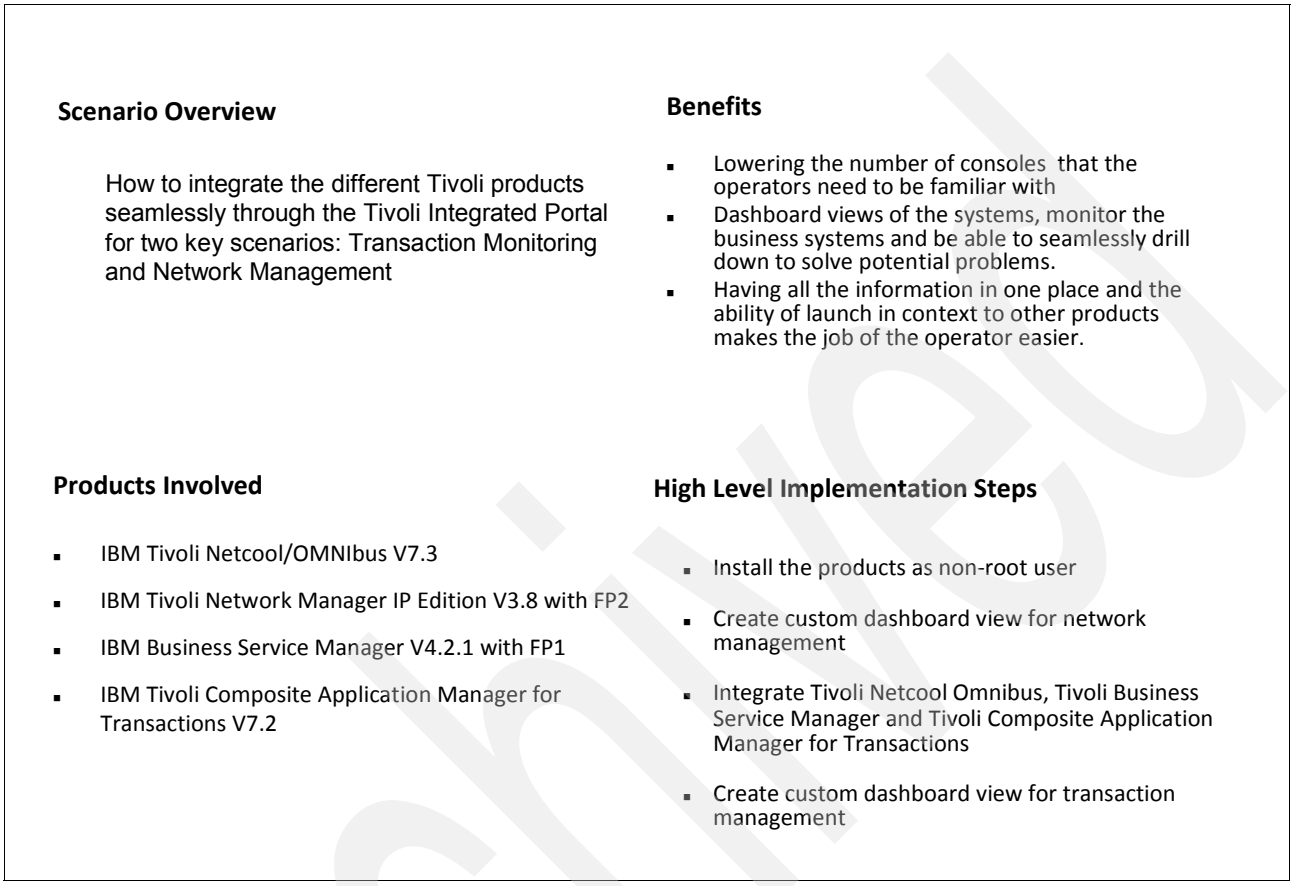


Figure 7-34 Quick summary

Archived

Archived

8

Tivoli reporting integration: IBM Tivoli Monitoring and TADDM

This chapter describes integrating IBM Tivoli Common Reporting for Asset and Performance Management V1.3 with IBM Tivoli Monitoring and TADDM to show you how to create reports using data collected by various Tivoli products. IBM Tivoli Common Reporting for Asset and Performance Management V1.3 is contains embedded IBM Cognos 8 technology.

Important: The framework model used in these scenarios can be downloaded from the ITSO FTP site. For downloading instructions, see Appendix A, “Additional material” on page 803. This framework model is included on an “as-is” basis and not officially supported. It is provided with this book to give you a preview of the planned product functionality for Tivoli Common Reporting integration.

This chapter contains the following topics:

- 8.1, “Scenario overview” on page 200
- 8.2, “Products involved” on page 200
- 8.3, “Benefits” on page 201
- 8.4, “Architecture of the integration” on page 202
- 8.5, “Implementation steps” on page 205
- 8.6, “Scenario walk-through” on page 237
- 8.7, “Online resources” on page 243
- 8.8, “Summary” on page 244

© Copyright IBM Corp. 2011. All rights reserved.

8.1 Scenario overview

Tivoli products gather large numbers of data in their own databases. Each product or solution maintains a separate database instance, which contains its own schema that has the purpose to fulfill requirements of the product for storage of the data. Although delivering a value for the user based on the data gathered within the one product is easy, the challenge is to deliver additional value based on the data taken from multiple products (data sources). The scenario in this chapter describes this kind of integration. Data gathered by two products are integrated through a reporting engine. These products are IBM Tivoli Monitoring data, gathered by operating system agents, and IBM Tivoli Application Dependency Discovery Manager data.

By having cross-product data virtually integrated and having access to them from one location, a user is able to find the answers to additional questions from IT management area and make more confident business decisions. Cross-product data integration provides additional facts that cannot be derived from a database of a single product. The Tivoli portfolio contains a reporting product, IBM Tivoli Common Reporting for Asset and Performance Management, which helps build a cross-product reporting solution.

Manual browsing through the database structure is a difficult task for someone who does not know how the product is implemented and what is the mechanism for storing the data. IBM Tivoli Common Reporting for Asset and Performance Management can cover a database structure with a user interface in which reporting against data gathered by various products becomes easier.

8.2 Products involved

Cross-product reporting integration is performed on two levels: product database level and reporting engine level. IBM Tivoli Common Reporting for Asset and Performance Management server is the central point of the solution. The server is connected to the separate databases that are used by the Tivoli products. This way creates opportunity to access data gathered by separate Tivoli products from one, central location. Cognos BI Reporting engine, which is running behind the IBM Tivoli Common Reporting for Asset and Performance Management server, provides mechanisms to join the data from separate data sources and, based on that information, generate reports.

The scenario in this chapter shows a sample cross-product reporting solution that is based on two Tivoli products:

- IBM Tivoli Monitoring 6.2.2 Fix Pack 2
- IBM Tivoli Application Dependency Discovery Manager 7.2

The element that connects those two products is IBM Tivoli Common Reporting for Asset and Performance Management server (also known as IBM Tivoli Common Reporting 1.3). It includes embedded IBM Cognos 8 technology. This reporting solution is available for free at the IBM Tivoli Monitoring version 6.2.2 Fix Pack 2 users.

8.3 Benefits

The following list summarizes general benefits of cross-product reporting solution and IBM Tivoli Common Reporting for Asset and Performance Management platform:

- Revealing new facts. Cross-products reporting reveals information that already exists in separate databases. However, without suitable linkage the information is meaningless or hardly accessible for the product's users. It is not possible to deliver one product that addresses all the questions related to IT infrastructure that are being asked by the users. However, based on the data and reporting integration, we can get the answers to more questions by using the products that we already have. Data level integration "glues" products together as a component and delivers additional value through the reports.
- Integration of various products. IBM Tivoli Common Reporting for Asset and Performance Management provides a vehicle on which products developed in separate technologies can be easily integrated on the databases level. Cross-product solution enables users to rapidly ask questions of multiple data sources.
- No database skills are required for business users. Cognos models facilitate access to information that is gathered by Tivoli products. Users without SQL skills and knowledge of the structure of the product's database can easily get the data and analyze it, based on individual needs. Cognos models provide understandable names for database tables, views, and columns.
- Single access point for multiple data sources. Tivoli Common Reporting server can be used as a single entry point to all the IT data that is gathered in the company.
- Performance analysis by application. This integration can help you understand, for a period of time, when there is a performance problem and also see other relevant metrics for the other application components.

Sample analyses can be performed by means of the IBM Tivoli Monitoring and TADDM integrated reporting solution, summarized in the following list:

- Show CPU, memory, disk usage by a manufacturer or a model of the server. You may select computer systems discovered by TADDM, group them by a manufacturer and by model, join through the common dimension with IBM Tivoli Monitoring data. The computer system is the join-point. In addition, you may display performance metrics (such as average CPU utilization), aggregated for groups introduced in TADDM selection.
- Show CPU, memory, disk usage by an operating system type. Group computer system data in any way based on the information gathered in TADDM, for example by operating system name and version. Check aggregated average CPU utilization for this grouping.
- Check the resources utilization for application servers and databases. You may select application server objects discovered by TADDM. These are web and application servers, databases, custom servers, and so on. Filter the applications from your area of interest, for example DB2 databases instances that you are administering. Select computer systems on which these databases are running. Join the servers through common dimension with monitoring data. Again computer system attributes such as host name are the join-point between two products. Display average disk utilization for machines and indirectly for databases selected from TADDM. Find the database instance which will hit not enough disk space issue before the problem occurs. Perform the same steps for your web servers and check average CPU and memory utilization of the machines on which your servers are running.
- Show CPU utilization and energy consumption. Compare systems from different server makers and manufacturers in terms of CPU utilization and energy consumption.

8.4 Architecture of the integration

Figure 8-1 presents all the components that are used in the integration scenario. The main components are:

- Tivoli Data Warehouse, which is used by the IBM Tivoli Monitoring solution
- Database instance, which works in the background of Tivoli Application Dependency Discovery Manager
- IBM Tivoli Common Reporting for Asset and Performance Management server (IBM Tivoli Common Reporting 1.3).

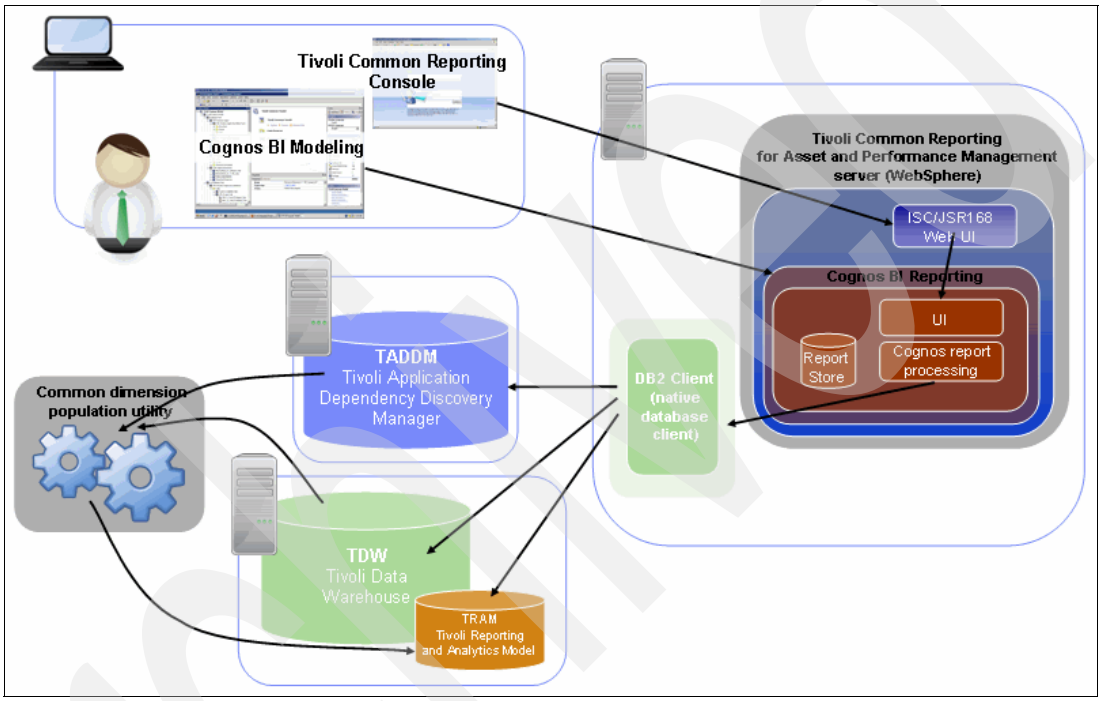


Figure 8-1 Integration architecture

The scenario assumes that IBM Tivoli Monitoring, Tivoli Application Dependency Discovery Manager, and IBM Tivoli Common Reporting server are already installed. All additional steps that are required to set up the integration are covered in this chapter.

Tivoli Data Warehouse contains additional schema called IBM_TRAM (IBM Tivoli Reporting and Analytics Model), which contains dimensions data (time dimension table and ComputerSystem dimension), common data between IBM Tivoli Monitoring operating system agents and TADDM application. The IBM Tivoli Common Reporting for Asset and Performance Management server is installed on a separate machine. It is connected to the databases of the products through the database native client (DB2 client).

JDBC connections are not supported in Cognos BI Reporting engine, which is used internally by IBM Tivoli Common Reporting for Asset and Performance Management sever. DB2 client has those two databases (IBM Tivoli Monitoring and TADDM) cataloged. Cognos engine uses the connections from that catalog. Additional tools are responsible for populating and maintaining common dimension tables. Cross-product model, which is deployed on Cognos BI Reporting server, is an abstraction layer on top of the databases schema. This entry point is for the reporting in Cognos. The user builds reports or uses an ad-hoc reporting feature based on that model.

Cognos engine operates on the model that contains objects from both Tivoli Data Warehouse and TADDM databases. Joining between those databases is implemented in the IBM_TRAM schema, particularly in this case, in ComputerSystem table. This table contains computer systems that are common for both products. The computer systems are both monitored by IBM Tivoli Monitoring operating system agents and discovered by TADDM.

The ComputerSystem table with common data is populated by an additional tool that was created for this integration scenario. At this time, products that are involved in the integration do not provide capabilities to automatically populate common dimensions. The additional tool is an external Java program that connects to three databases (Tivoli Data Warehouse, TADDM, and database on which IBM_TRAM schema is installed). The program finds the common computer systems (machines) between Tivoli Data Warehouse and TADDM and stores the results in the IBM_TRAM.ComputerSystem table. The utility must be run periodically to refresh joint point, which is the ComputerSystem table.

Figure 8-2 shows a sample report that was generated, based on data taken from two products: TADDM and IBM Tivoli Monitoring. Detailed information about the hosts configuration (operating system name, manufacturer and model of the hardware, CPU type) come from TADDM database. Average percentage CPU utilization is received from Tivoli Data Warehouse view. The utilization value is additionally aggregated by Cognos BI Reporting engine: the average value is computed for all days that the data is stored in the warehouse. The data is integrated through the IBM_TRAM schema. Matching is done for the host names of the computer systems.

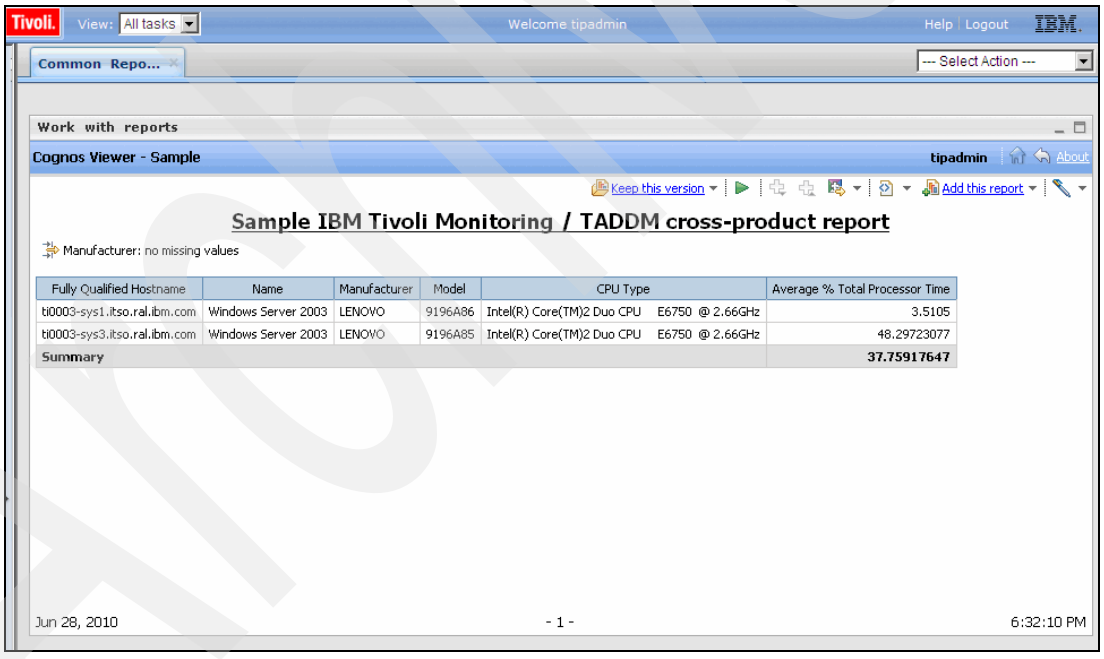


Figure 8-2 Sample cross-product report

Note: This report is an ad-hoc report. It was created in the Query Studio component, *taking less than one minute to select the columns and join them together*. Tables and views were glued by Cognos BI Reporting engine according to the rules defined in the model. Figure 8-3 on page 204 presents a part of the cross-product reporting model, which is provided as an additional download material for this chapter. Model objects can be grouped into three categories: TADDM domain, IBM Tivoli Monitoring domain, common domain (IBM_TRAM).

How does the cross-product join work? Elements from various products are connected to common dimensions. In our case, common dimension is the IBM_TRAM.ComputerSystem table.

Cross-product-join behind the report can be converted to the following SQL conditions:

- `[TADDM].BB_COMPUTERSYSTEM40_V.FQDN_C = [IBM_TRAM].ComputerSystem.FQDN`
- `[IBM_TRAM].ComputerSystem.ManagedSystemName = [TDW].ManagedSystem.DisplayName`
- `[TDW].ManagedSystem.ManagedSystemName = [TDW].NT_System_DV.Server_Name`

BB_COMPUTERSYSTEM40_V view from TADDM has the relationship with ComputerSystem table from IBM_TRAM schema. MANAGEDSYSTEM table from Tivoli Data Warehouse also has a relationship with ComputerSystem table from IBM_TRAM schema. All views from IBM Tivoli Monitoring with metrics (such as NT_System_DV, NT_System_HV, System_DV, System_HV, KLZ_CPU_HV) have relationships with MANAGEDSYSTEM table.

Additional relationships between metrics views and TIME_DIMENSION table (for example `[TDW].NT_System_DV.WRITETIME = [IBM_TRAM].TIME_DIMENSION.CANDLETIMESTAMP`) introduce the possibility of filtering data by custom date ranges, executing the reports for a specific period of time only.

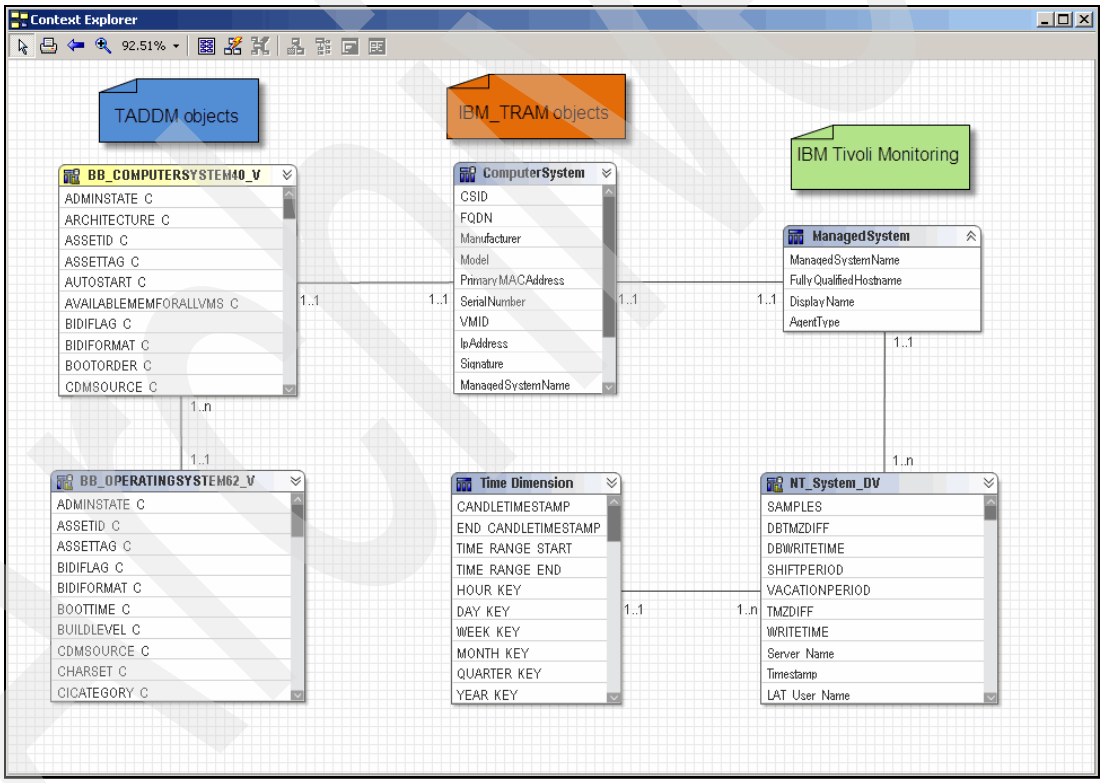


Figure 8-3 Cross-product model fragment

Figure 8-4 on page 205 shows the origination of each column in the report by mapping them on the query subjects from model. In certain cases, the full path to database query subjects is presented. The diagram was generated by Cognos BI Reporting engine.

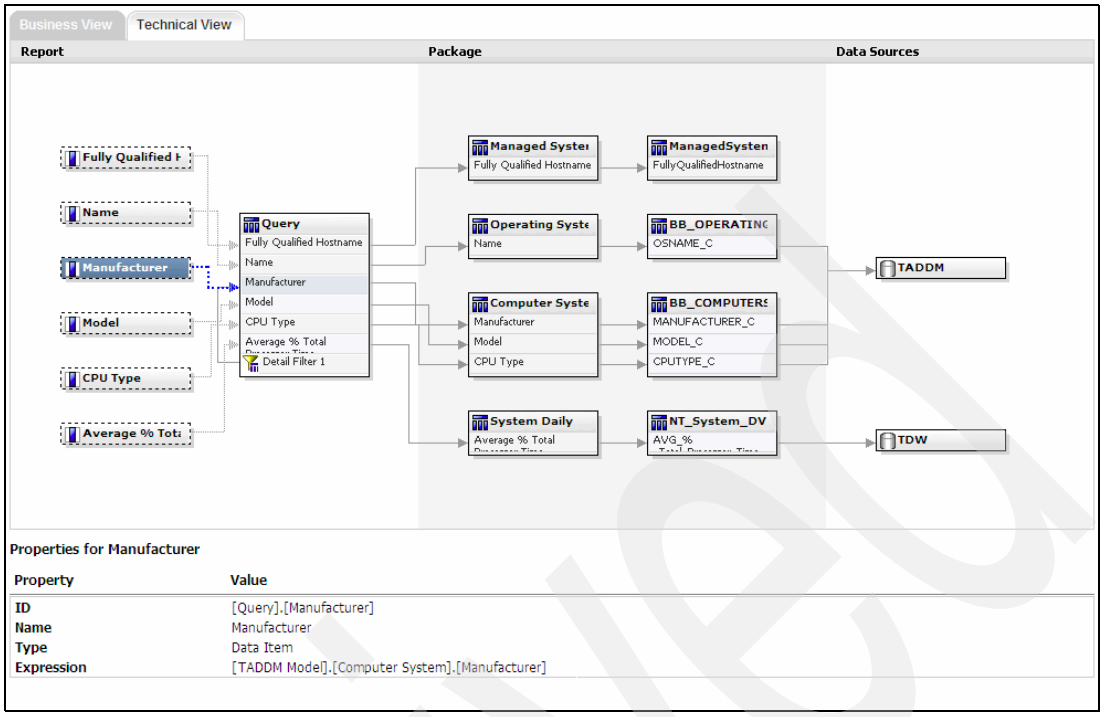


Figure 8-4 Database views and tables behind the sample report

8.5 Implementation steps

This scenario assumes that IBM Tivoli Monitoring and TADDM products are already installed and configured. There is no need to perform additional configuration steps on TADDM server. The only prerequisite is to have several servers that are monitored by IBM Tivoli Monitoring agents and were discovered by TADDM. This means, information about the same machine must exist in the Tivoli Data Warehouse and TADDM database.

In case of IBM Tivoli Monitoring, enabling historical data collection for some attributes is required. For information about which attributes to configure, see 8.5.2, “Configuration of the products” on page 209.

The scenario also assumes that IBM Tivoli Common Reporting for Asset and Performance Management server is already installed. The server can be installed on an additional machine, but it can also be running on one of the machines on which other products participating in the scenario are installed. IBM Tivoli Common Reporting for Asset and Performance Management installation is covered in the Information Center for the product:

http://publib.boulder.ibm.com/infocenter/tivihelp/v3r1/topic/com.ibm.tivoli.tcr_cog.doc/ttcr_cog_install.html

For this integration, a sufficient approach is to select “Single-computer installation” scenario (described in the document at the website) and proceed with basic setup (without “BIRT-based Tivoli Common Reporting engine” and LDAP setup).

Scenario implementation consists of the following steps, which are detailed in this section:

1. Database connections setup
2. Cross-product database structures creation
3. Common dimensions population with data
4. Cross-product model installation and publication

8.5.1 Database native client

In addition to the IBM Tivoli Common Reporting for Asset and Performance Management server on the same machine database, native client should be installed and configured with remote databases information. IBM Tivoli Common Reporting 1.3 introduces Cognos BI Reporting engine; this integration scenario is based on that reporting engine. Cognos does not support JDBC connections, which were used in the previous version of IBM Tivoli Common Reporting by BIRT reporting engine. It uses native connection drivers such as DB2 client.

To set up a data source for DB2 in Cognos BI Reporting, you must install the DB2 client application on the machine where the IBM Tivoli Common Reporting for Asset and Performance Management server is installed. Another requirement is to set up (catalog) a connection to the remote databases within that DB2 client. The installation is not required if IBM Tivoli Common Reporting 1.3 server is installed on the same machine on which Tivoli Data Warehouse or TADDM databases are running. In this case, one of the databases is local and it is already cataloged. It is only required to catalog a connection to the second remote database.

This section explains the DB2 Client installation and configuration. We will be acting in the environment when Tivoli Data Warehouse and TADDM DB2 databases are remotely accessible for the IBM Tivoli Common Reporting 1.3 server machine.

Perform the following steps to set up the native connections:

1. Download the DB2 client application package. Select the version that matches your database. Download the application from either of the following websites:
   - IBM support and download site
     - DB2 9.1:

       `http://www.ibm.com/support/docview.wss?rs=71&uid=swg21255572`
     - DB2 9.5:

       `http://www.ibm.com/support/docview.wss?rs=71&uid=swg21287889`

       For example, the IBM Data Server Client for Windows file:

       `ftp://ftp.software.ibm.com/ps/products/db2/fixes2/english-us/db2winIA32v95/fixpack/FP4_WR21449/v9.5fp4_nt32_client.exe`
   - IBM Fix Central site

     `http://www.ibm.com/support/fixcentral`

     i. Select group Information Management, product DB2, correct database version and platform.

     ii. Browse for the client application

     For example:

     DB2-nt32-client-9.5.400.576-FP004

     DB2 9.5 Fix Pack 4 for Windows/x86-32 (32 bit), IBM Data Server Client

2. Install the client application.

   Note: On UNIX or Linux machines it is necessary to create manually additional operating system user for DB2 administration purposes and create DB2 instance associated with that user. On Windows machine installation program will do all the required work for you.

   Instance creation is required even though a real database is not going to be created.

   For Windows platform, the installation wizard guides you through the installation process.

   For UNIX or Linux platform, use the following steps to add the user and create a new DB2 instance:

   a. Install the client application by invoking **db2_install** script first.

   b. As a root user, create a new operating system user (Example 8-1).

   Example 8-1 Creating a new user for database administration

```
nc043243:~ # groupadd db2grp1
nc043243:~ # useradd -d /home/db2inst1 -g db2grp1 db2inst1
nc043243:~ # passwd db2inst1
Changing password for db2inst1.
New Password:
Reenter New Password:
Password changed.
```

   c. As a root user create a new database instance - issue **db2icrt** command. Associate the instance with the user created (Example 8-2).

   Example 8-2 Creating DB2 instance

```
nc043243:~ # cd /opt/ibm/db2/V9.5/instance/
nc043243:/opt/ibm/db2/V9.5/instance # ./db2icrt -u db2inst1 db2inst1
DBI1070I  Program db2icrt completed successfully.
```

3. Add remote databases to the DB2 client catalog:

   a. Open the command line for DB2 (see Figure 8-5 on page 208):

      On Windows: **Start** → **All Programs** → **IBM DB2** → **DB2COPY1 (Default)** → **Command Line Tools** → **Command Window**

Archived

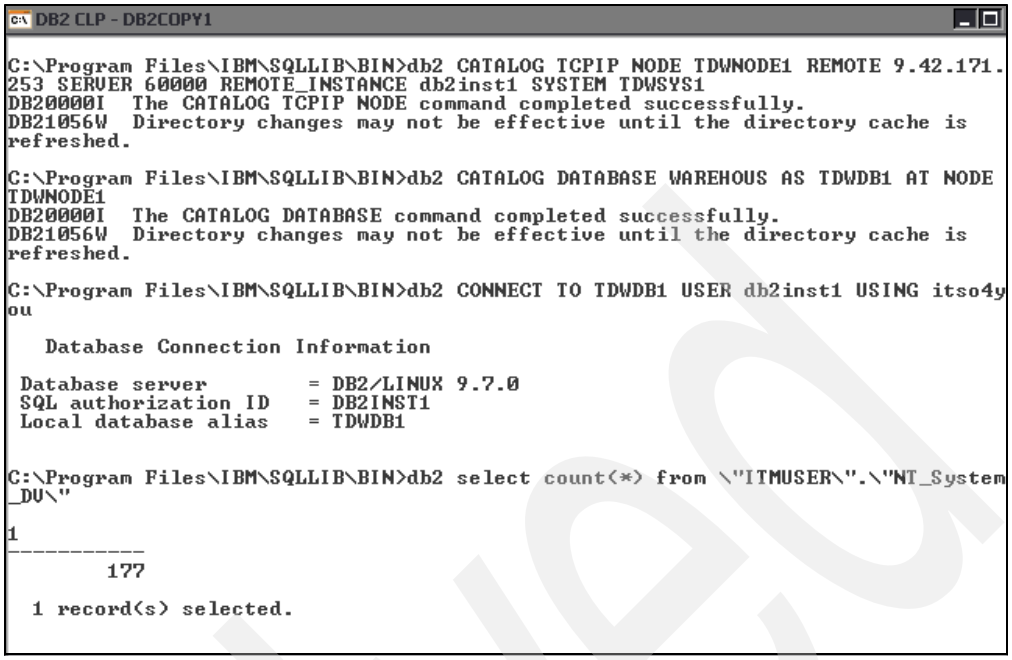


Figure 8-5 Sample DB2 client command-line window

On UNIX or Linux, log in as the database user (for example execute **su db2inst1**).

b. Catalog the database instance node by executing the following command:

```
db2 CATALOG TCPIP NODE <node-name> REMOTE <hostname-or-IP-address> SERVER
<service-name-or-port-number> REMOTE_INSTANCE <instance-name> SYSTEM
<system-name>
```

c. Catalog the database by executing the following command

```
db2 CATALOG DATABASE <database-name> AS <alias> AT NODE <node-name>
```

d. Test connection to the database by issuing a sample select SQL query. For Tivoli Data Warehouse and TADDM databases, see Example 8-3 and Example 8-4 on page 209.

e. Repeat the commands (step 3 on page 207) for Tivoli Data Warehouse database and for TADDM database. Remember the aliases under which you have cataloged your databases connections because you will use them later in configuration for IBM Tivoli Common Reporting for Asset and Performance Management server.

Example 8-3 Adding Tivoli Data Warehouse to the client's catalog

```
C:\Program Files\IBM\SQLLIB\BIN>db2 CATALOG TCPIP NODE TDWNODE1 REMOTE
9.42.171.253 SERVER 60000 REMOTE_INSTANCE db2inst1 SYSTEM TDWSYS1
DB20000I  The CATALOG TCPIP NODE command completed successfully.
DB21056W  Directory changes may not be effective until the directory cache
is refreshed.

C:\Program Files\IBM\SQLLIB\BIN>db2 CATALOG DATABASE WAREHOUS AS TDWDB1 AT
NODE TDWNODE1
DB20000I  The CATALOG DATABASE command completed successfully.
DB21056W  Directory changes may not be effective until the directory cache
is refreshed.

C:\Program Files\IBM\SQLLIB\BIN>db2 CONNECT TO TDWDB1 USER db2inst1 USING
itso4you
```

```
   Database Connection Information

 Database server        = DB2/LINUX 9.7.0
 SQL authorization ID   = DB2INST1
 Local database alias   = TDWDB1

C:\Program Files\IBM\SQLLIB\BIN>db2 select count(*) from
\"ITMUSER\".\"NT_System_DV\"

1
-----------
        177

  1 record(s) selected.
```

Example 8-4 shows adding TADDM database to the catalog

Example 8-4 Adding TADDM database to the client's catalog

```
C:\Program Files\IBM\SQLLIB\BIN>db2 CATALOG TCPIP NODE TADDMN1 REMOTE
9.42.171.157 SERVER 50000 REMOTE_INSTANCE db2inst1 SYSTEM TADDMS1
DB20000I  The CATALOG TCPIP NODE command completed successfully.
DB21056W  Directory changes may not be effective until the directory cache
is refreshed.

C:\Program Files\IBM\SQLLIB\BIN>db2 CATALOG DATABASE DEMO AS TADDMDB1 AT
NODE TADDMN1
DB20000I  The CATALOG DATABASE command completed successfully.
DB21056W  Directory changes may not be effective until the directory cache
is refreshed.

C:\Program Files\IBM\SQLLIB\BIN>db2 CONNECT TO TADDMDB1 USER db2inst1 USING
smartway

   Database Connection Information

 Database server        = DB2/LINUX 9.5.0
 SQL authorization ID   = DB2INST1
 Local database alias   = TADDMDB1

C:\Program Files\IBM\SQLLIB\BIN>db2 select count(*) from
BB_COMPUTERSYSTEM40_V

1
-----------
       2790

  1 record(s) selected.
```

8.5.2 Configuration of the products

To implement this integration scenario, we do not have to introduce significant configuration changes in the products. The common data must be for the machines that are discovered by TADDM and monitored by Tivoli Monitoring. For example, if there is data for a machine that is

Archived

discovered by TADDM, but this machine is not monitored by Tivoli Monitoring, the data about this machine is not taken into account for this scenario.

For the TADDM application, you do not have to perform extra steps except for obtaining access information to the database. The list of views that are required by the solution should have the following data (as a result of TADDM discovery scans):

- BB_COMPUTERSYSTEM40_V
- BB_COMPUTERSYSTACES_76E84D2AJ
- BB_IPINTERFACE96_V
- BB_IPADDRESS73_V
- BB_L2INTERFACE41_V

For IBM Tivoli Monitoring, configuring historical data collection for some attributes is necessary. By enabling data collections for attribute groups special tables and views are created in Tivoli Data Warehouse. The views that are used to set up the scenario are in the following list. Not all views are required, but at least one set of views must be for one agent (Windows, Linux or UNIX). You do not need to have all the agent types in your environment.

- Linux Agent
  - Linux_Machine_Information_DV
  - Linux_IP_Address_DV
- UNIX Agent
  - Machine_Information_DV
  - UNIX_IP_Address_DV
  - System_DV
- Windows Agent
  - NT_Computer_Information_DV
  - NT_System_DV

Instructions for enabling attribute groups for historical data collection is in the Information Center for IBM Tivoli Monitoring:

http://publib.boulder.ibm.com/infocenter/tivihelp/v15r1/index.jsp?topic=/com.ibm.itm.doc_6.2.2fp2/tcr_reports_dimensionresource.htm

8.5.3 Databases setup

The integration scenario described in this chapter is based on two databases:

- Tivoli Data Warehouse database, which is used by the IBM Tivoli Monitoring solution,
- Tivoli Application Dependency Discovery Manager database.

Configuration steps described in this section are based on IBM DB2 databases only. However similar steps can be applied to other databases (such as Oracle or MSSQL) that are supported by IBM Tivoli Monitoring and by IBM Tivoli Application Dependency Discovery Manager.

Tivoli Data Warehouse setup

Create and populate data for the MANAGEDSYSTEM table within IBM Tivoli Monitoring data warehouse schema (by default ITMUSER schema). This additional table is introduced for IBM Tivoli Monitoring operating system agents model and reports package for IBM Tivoli Common Reporting for Asset and Performance Management. The table contains a list of all machines

that are monitored by types of operating system agents (Windows, Linux, and UNIX). The table is a single access point to data gathered by three types of agents.

Follow the instructions provided with IBM Tivoli Monitoring 6.2.2 Fix Pack 2 to create and populate additional table:

http://publib.boulder.ibm.com/infocenter/tivihelp/v15r1/topic/com.ibm.itm.doc_6.2.2fp2/tcr_reports_dimensionresource.htm

We present sample steps of the instructions from IBM Tivoli Monitoring Information Center. The example is executed on a Windows machine and it is based on the DB2 client (installation and configuration is explained in 8.5.1, “Database native client” on page 206). DB2 command window is used to connect to the remote database and execute all the necessary steps.

Perform the following steps:

1. Download IBM Tivoli Monitoring 6.2.2 Fix Pack 2 Agent Reports package and extract it on the machine on which you have DB2 client and IBM Tivoli Common Reporting for Asset and Performance Management server installed.

   Download the package from the IBM support and downloads site:

   http://www.ibm.com/support/docview.wss?uid=swg24025630

   The package is required only for the purpose of obtaining the following file:

   `6.2.2-TIV-ITM_TMV-AgentReports-FP0002.tar`

   The agent reports package can be also obtained from IBM Passport Advantage® site. The package name is: IBM Tivoli Monitoring v6.2.2 Fix Pack 2 Agent Reports, Multiplatform, English (part number CZH3MEN).

   The package contains all the necessary scripts that are used in this procedure. The scripts can be found in the `db_scripts\db2` subdirectory of the archive.

2. Open DB2 command line by clicking **Start** → **All Programs** → **IBM DB2** → **DB2COPY1 (Default)** → **Command Line Tools** → **Command Window**. See Figure 8-6.

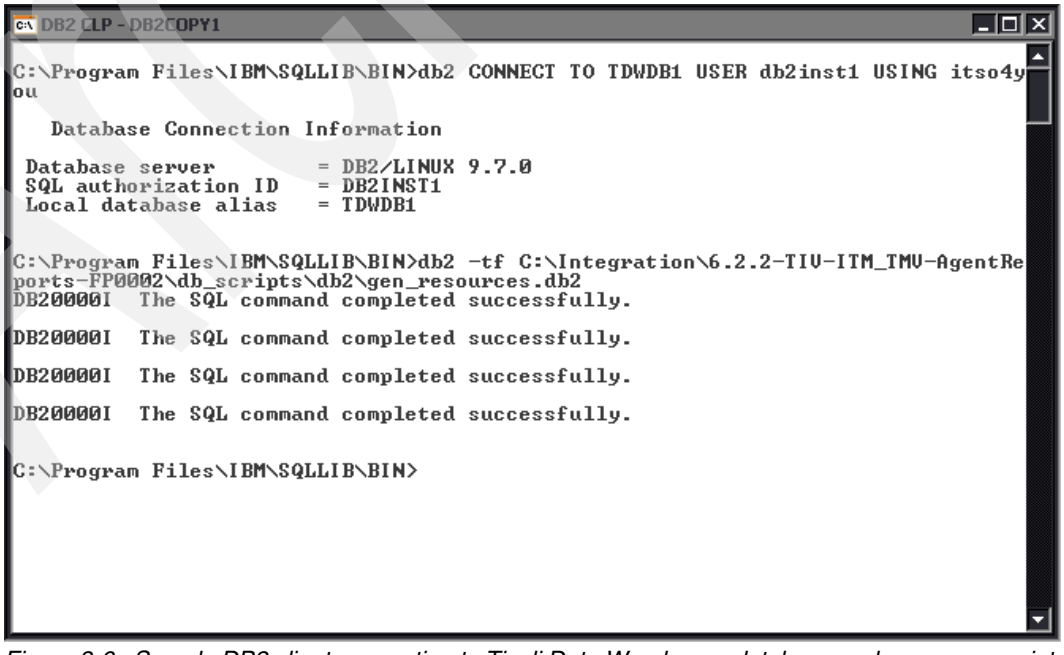


Figure 8-6 Sample DB2 client connection to Tivoli Data Warehouse database and resources script execution

3. Connect to the Tivoli Data Warehouse database by executing the following command (Example 8-5) in the window. Be sure you have administrative access rights to create new table within the ITMUSER schema:

```
db2 CONNECT TO <database-alias> USER <user-name> USING <password>
```

Example 8-5 Establishing connection to Tivoli Data Warehouse database

```
C:\Program Files\IBM\SQLLIB\BIN>db2 CONNECT TO TDWDB1 USER db2inst1 USING itso4you

   Database Connection Information

 Database server        = DB2/LINUX 9.7.0
 SQL authorization ID   = DB2INST1
 Local database alias   = TDWDB1

C:\Program Files\IBM\SQLLIB\BIN>
```

4. Create a MANAGEDSYSTEM table (Example 8-6) by running the `gen_resources.db2` script:

```
db2 -tf gen_resources.db2
```

If you have other than the default schema name in Tivoli Data Warehouse database (different from ITMUSER), edit the script and update it with your schema name. Replace all 'ITMUSER' occurrences in the script with the correct value.

Example 8-6 Creating MANAGEDSYSTEM table

```
C:\Program Files\IBM\SQLLIB\BIN>db2 -tf
C:\Integration\6.2.2-TIV-ITM_TMV-AgentReports-FP0002\db_scripts\db2\gen_resources.db2
DB20000I  The SQL command completed successfully.
...
DB20000I  The SQL command completed successfully.

C:\Program Files\IBM\SQLLIB\BIN>
```

5. Create the stored procedure that will populate the MANAGEDSYSTEM table (see Example 8-7). Execute the `populate_resources.db2` script:

```
db2 -td@ -vf populate_resources.db2
```

If you have other than the default schema name in Tivoli Data Warehouse database (different from ITMUSER), edit the script and update it with your schema name. Replace all 'ITMUSER' occurrences in the script with the correct value.

Example 8-7 Creating population procedure

```
C:\Program Files\IBM\SQLLIB\BIN>db2 -td@ -vf
C:\Integration\6.2.2-TIV-ITM_TMV-AgentReports-FP0002\db_scripts\db2\populate_resources.d
b2
create procedure ITMUSER.POPULATE_OSAGENT(IN MSNCOL VARCHAR(50), IN FQHNCOL VARCHAR(50),
IN DFQHNCOL VARCHAR(50), IN TABLENAME VARCHAR(50), IN OSTYPE VARCHAR(32))
LANGUAGE SQL
BEGIN
...
END

DB20000I  The SQL command completed successfully.

C:\Program Files\IBM\SQLLIB\BIN>
```

Archived

6. Call the stored procedure to populate the table with machines information taken from the various agents views. See Example 8-8. Use the schema name appropriate to your database instance (by default it is ITMUSER):

```
db2 call ITMUSER.POPULATE_OSAGENTS();
```

Example 8-8 MANAGEDSYSTEM table population

```
C:\Program Files\IBM\SQLLIB\BIN>db2 call ITMUSER.POPULATE_OSAGENTS();

  Return Status = 0

C:\Program Files\IBM\SQLLIB\BIN>
```

Now, a MANAGEDSYSTEM table is created and populated with the entries for each monitored machine (Figure 8-7).

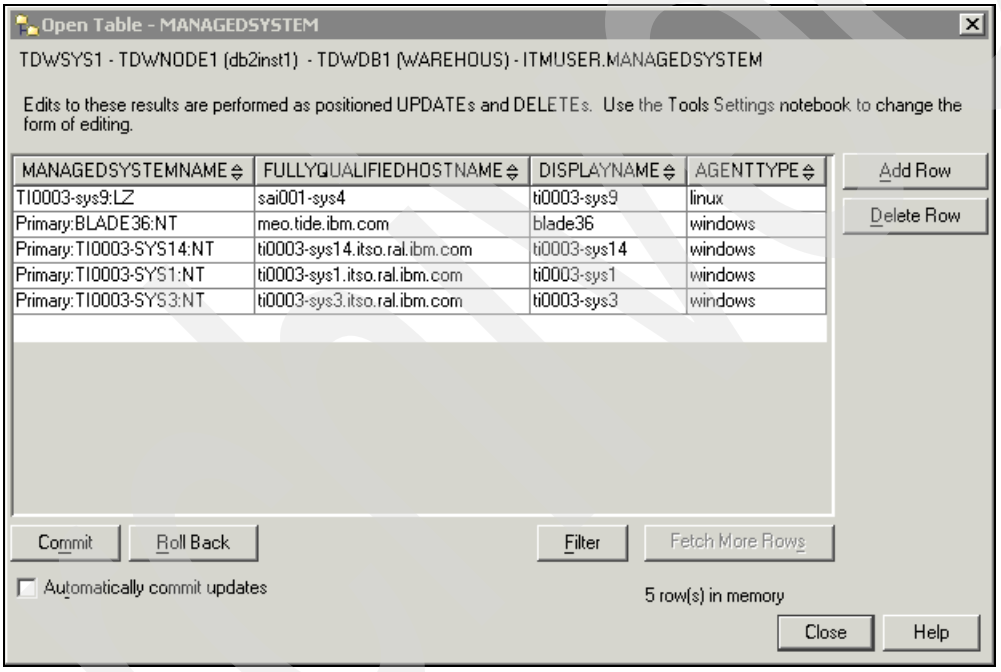


Figure 8-7 Sample MANAGEDSYSTEM table contents

Tivoli Reporting and Analytics Model database/schema setup

Set up IBM_TRAM (IBM Tivoli Reporting and Analytics Model) schema and populate TimeDimension table and ComputerSystem table.

The schema can be created in one of the existing databases, Tivoli Data Warehouse or TADDM, or it can also be a separate DB2 database instance. However, the preference is to create schema in the Tivoli Data Warehouse.

Two main tables are in the IBM_TRAM schema:

- TimeDimension with timestamps generated according to specified period of time (for example 1 June 2010 - 31 December 2010, every 5 minutes).
- ComputerSystem that will hold the information about the machines that are common for Tivoli Data Warehouse and IBM Tivoli Application Dependency Discovery Manager. This table will be populated by the utility described later in this section.

Follow the instructions provided with IBM Tivoli Monitoring 6.2.2 Fix Pack 2 to create and populate additional schema. See the following web address for instructions:

http://publib.boulder.ibm.com/infocenter/tivihelp/v15r1/topic/com.ibm.itm.doc_6.2.2fp2/tcr_reports_dimensionsshared.htm

We present sample steps of the instructions for DB2 database. The example is based on the same environment as in "Tivoli Data Warehouse setup" on page 210. The scripts that are used during the configuration are from the same IBM Tivoli Monitoring agent reports package downloaded previously.

Perform the following steps:

1. Open a DB2 command line by clicking **Start** → **All Programs** → **IBM DB2** → **DB2COPY1 (Default)** → **Command Line Tools** → **Command Window**. See Figure 8-8.

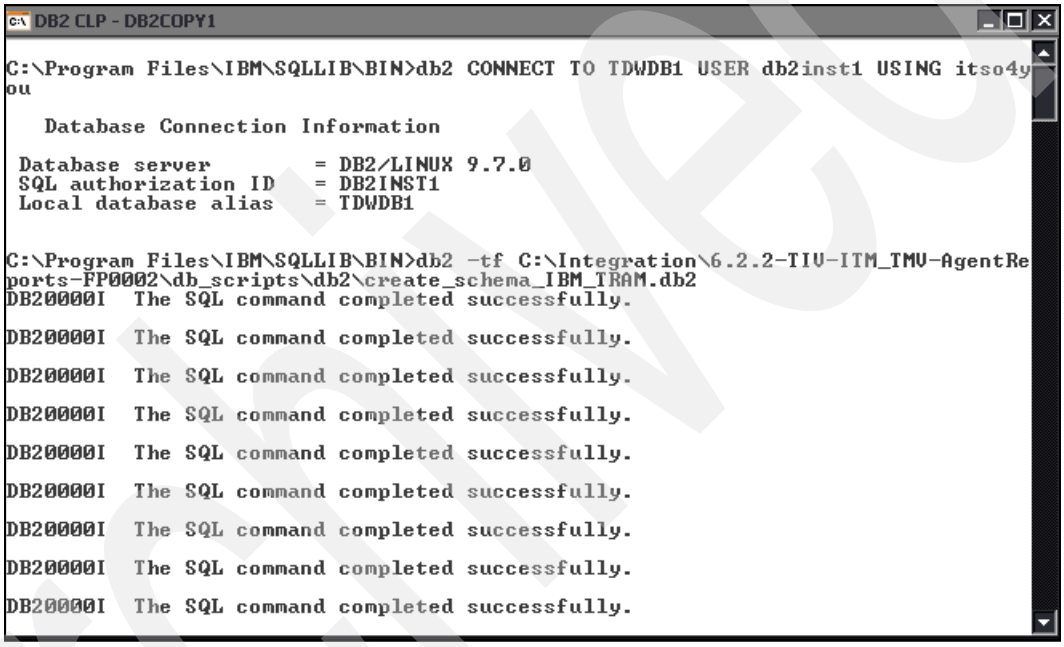


Figure 8-8 Sample DB2 client connection and IBM_TRAM.db2 script execution

2. Connect to the Tivoli Data Warehouse database by executing the following command in the window. You must have administrative access rights to create new schema, tables and stored procedure in the Tivoli Data Warehouse database.

   ```
   db2 CONNECT TO <database-alias> USER <user-name> USING <password>
   ```

   See Example 8-9.

Example 8-9 Establishing a connection to Tivoli Data Warehouse database

```
C:\Program Files\IBM\SQLLIB\BIN>db2 CONNECT TO TDWDB1 USER db2inst1 USING itso4you

   Database Connection Information

 Database server        = DB2/LINUX 9.7.0
 SQL authorization ID   = DB2INST1
 Local database alias   = TDWDB1

C:\Program Files\IBM\SQLLIB\BIN>
```

Archived

3. Create IBM_TRAM schema and tables by running the `create_schema_IBM_TRAM.db2` script (see Example 8-10):

   ```
   db2 -tf create_schema_IBM_TRAM.db2
   ```

   Example 8-10 Creating IBM_TRAM schema and tables

   ```
   C:\Program Files\IBM\SQLLIB\BIN>db2 -tf
   C:\Integration\6.2.2-TIV-ITM_TMV-AgentReports-FP0002\db_scripts\db2\create_sche
   ma_IBM_TRAM.db2
   DB20000I  The SQL command completed successfully.

   DB20000I  The SQL command completed successfully.

   ...
   C:\Program Files\IBM\SQLLIB\BIN>
   ```

4. Create a stored procedure that will be used to populate TIME_DIMENSION table. Run `gen_time_dim_granularity_min.db2` script (see Example 8-11):

   ```
   db2 -td@ -vf gen_time_dim_granularity_min.db2
   ```

   Example 8-11 Creating population procedure

   ```
   C:\Program Files\IBM\SQLLIB\BIN>db2 -td@ -vf
   C:\Integration\6.2.2-TIV-ITM_TMV-AgentReports-FP0002\db_scripts\db2\gen_time_di
   m_granularity_min.db2
   drop procedure IBM_TRAM.CREATE_TIME_DIMENSION
   DB21034E  The command was processed as an SQL statement because it was not a
   valid Command Line Processor command.  During SQL processing it returned:
   SQL0204N  "IBM_TRAM.CREATE_TIME_DIMENSION" is an undefined name.
   SQLSTATE=42704

   create procedure IBM_TRAM.CREATE_TIME_DIMENSION(IN startDate VARCHAR(32), IN
   end Date VARCHAR(32), IN granularity SMALLINT)
   LANGUAGE SQL
   BEGIN
   ...
   END

   DB20000I  The SQL command completed successfully.

   C:\Program Files\IBM\SQLLIB\BIN>
   ```

5. Populate the TIME_DIMENSION table by calling the stored procedure created in the previous step. The procedure requires three arguments:

   ```
   db2 call IBM_TRAM.CREATE_TIME_DIMENSION('start_date', 'end_date',
   granularity_of_data);
   ```

   For example, the command in Example 8-12 on page 216 generates timestamps from 15 May 2010 - 05/15/2010 to 1 August 2010, for 1 minute granularity. You can also generate more months in the range, depending on your needs.

Archived

Example 8-12 TIME_DIMENSION table population

```
C:\Program Files\IBM\SQLLIB\BIN>db2 call
IBM_TRAM.CREATE_TIME_DIMENSION('2010-05-15-00.00.00.000000',
'2010-08-01-00.00.00.000000', 1);

  Return Status = 0

C:\Program Files\IBM\SQLLIB\BIN>
```

Now, you have IBM_TRAM schema created in your Tivoli Data Warehouse. See Figure 8-9. The schema contains five tables:

- TIME_DIMENSION: This table contains generated timestamps. Generation is done by the stored procedure, within a specified date range and granularity. Timestamps are converted to separate keys at the generation time and stored in separate columns.
- WEEKDAY_LOOKUP: This table is of globalized weekday names for Time Dimension.
- MONTH_LOOKUP: This table has globalized month names for Time Dimension.
- TIMEZONE_DIMENSION: This table stores information about the timezone of the collected data.
- “ComputerSystem”: This table is dimension for common machines between separate Tivoli products (in this scenario between IBM Tivoli Monitoring and TADDM).

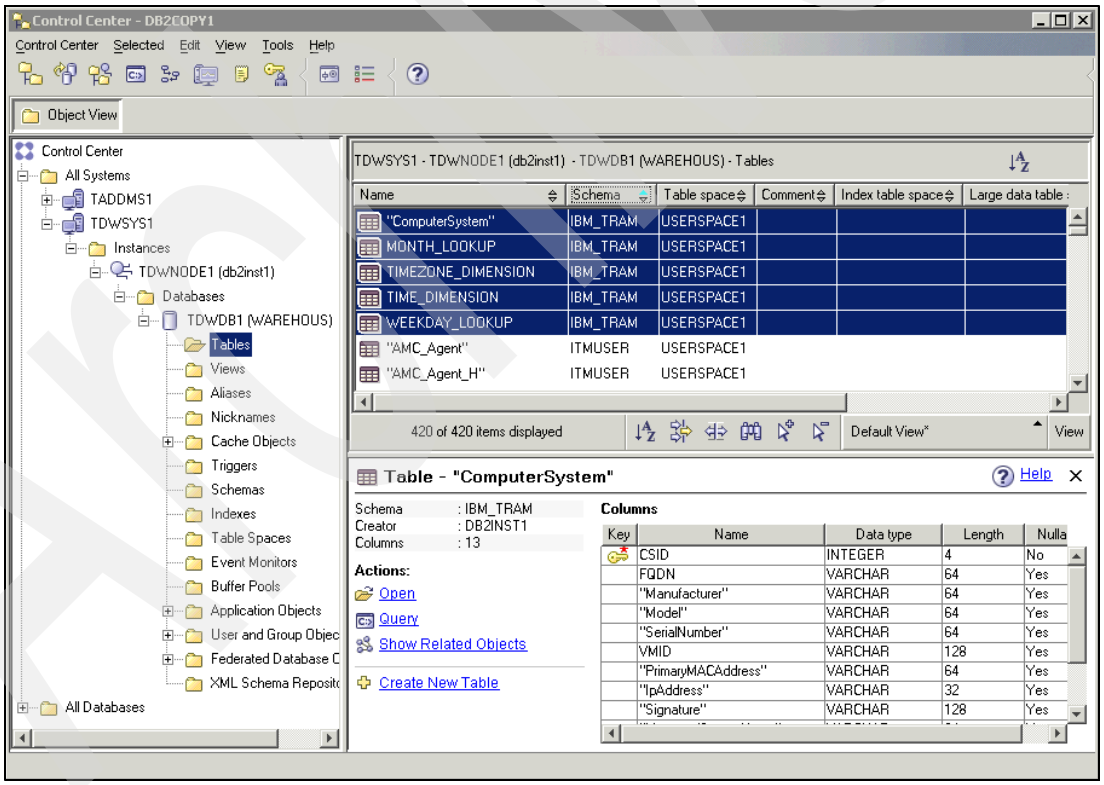


Figure 8-9 IBM_TRAM schema tables

ComputerSystem table population

In this step, the `IBM_TRAM.”ComputerSystem”` table will be populated with common data taken from Tivoli Data Warehouse and TADDM. A simple Java utility program is provided for this integration scenario; it performs all the necessary work. The program connects to Tivoli Data

Warehouse and TADDM databases through the JDBC connections, selects the records about the machines monitored by IBM Tivoli Monitoring and discovered by TADDM, and then merges them in the “ComputerSystem” table that is defined within the IBM_TRAM schema. See Figure 8-10.

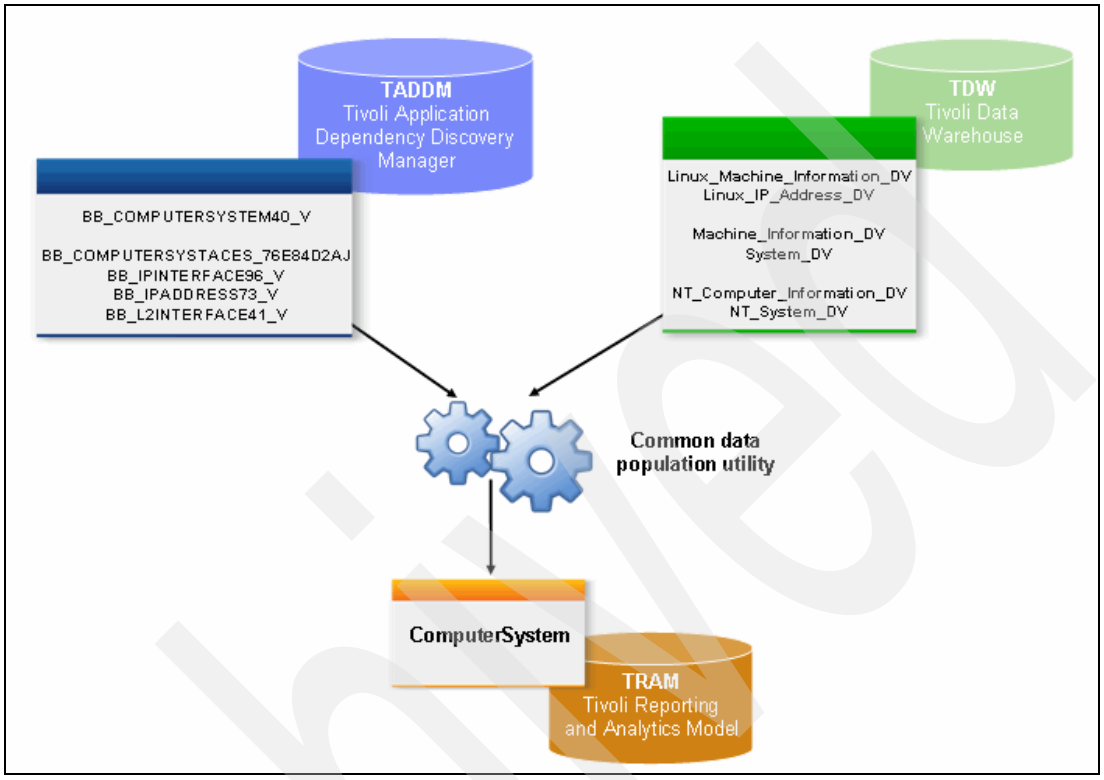


Figure 8-10 Common data population utility

The following high-level steps are performed by the utility:

1. Creates computer systems list monitored by IBM Tivoli Monitoring. Three separate queries must be executed because agents for separate operating system types (Windows, Linux and UNIX) store data in separate tables:
   a. Gets all Windows computer systems from Tivoli Data Warehouse.
   b. Gets all Linux computer systems from Tivoli Data Warehouse.
   c. Gets all UNIX computer systems from Tivoli Data Warehouse.
2. Creates computer systems list discovered by TADDM.
3. Iterates through the IBM Tivoli Monitoring list created in step 1, creates the result list.
   a. If that TADDM list that is created in step 2 contains a computer system (host names are equal or IP addresses are equal or serial numbers are equal):
      i. Adds the machine to the result list.
      ii. Updates the result computer system information with TADDM data (they tend to be more accurate in the area of configuration).
   b. If TADDM list does not contain computer system, skips the machine.
4. Removes all the rows from the IBM_TRAM.ComputerSystem table.
5. Inserts all the elements from the result list into the IBM_TRAM.ComputerSystem table.

Important: Tivoli Data Warehouse data and TADDM data are not modified by this utility. The only table that is cleaned and populated with the new data is `IBM_TRAM.ComputerSystem` table. Access to the tables from Tivoli Data Warehouse and TADDM systems can be considered as read-only access type.

Tables in this section summarize columns mapping between IBM_TRAM.ComputerSystem table and Tivoli Data Warehouse and TADDM views and columns.

Table 8-1 shows Windows agent columns mapping.

Table 8-1 Windows agent columns mapping on ComputerSystem table

IBM_TRAM.ComputerSystem table (common dimension)	IBM Tivoli Monitoring Windows Agent views and columns
CSID	Automatically generated
FQDN	NT_Computer_Information_DV. LAT_Computer_Domain_Name
Manufacturer	NT_Computer_Information_DV. LAT_Computer_Vendor
Model	NT_Computer_Information_DV. LAT_Computer_Name
SerialNumber	NT_Computer_Information_DV. LAT_Computer_ID_Number
VMID	Not available
PrimaryMACAddress	Not available
IpAddress	NT_System_DV. LAT_Network_Address
Signature	Not available
ManagedSystemName	NT_Computer_Information_DV. System_Name
Group	Not available
TIME_RANGE_START	Out of the scope of this integration
TIME_RANGE_END	Out of the scope of this integration

Windows computer systems data is taken from two Tivoli Data Warehouse views:

- NT_Computer_Information_DV
- NT_System_DV

Join operation between these two views is created through the following columns:

- NT_Computer_Information_DV.System_Name
- NT_System_DV.Server_Name

The last entry (maximum value from the WRITETIME column) at the time when the tool is executed is taken for each machine that is stored in the database.

Table 8-2 on page 219 lists Linux agent columns mapping.

Archived

Table 8-2 Linux agent columns mapping on ComputerSystem table

IBM_TRAM.ComputerSystem table (common dimension)	IBM Tivoli Monitoring Linux Agent views and columns
CSID	Automatically generated
FQDN	Linux_IP_Address_DV.LAT_DNS_Name
Manufacturer	Linux_Machine_Information_DV. LAT_Hardware_Brand
Model	Linux_Machine_Information_DV. LAT_Hardware_Model
SerialNumber	Linux_Machine_Information_DV. LAT_Machine_Serial
VMID	Not available
PrimaryMACAddress	Not available
IpAddress	Linux_IP_Address_DV.IP_Address
Signature	Not available
ManagedSystemName	Linux_Machine_Information_DV. System_Name
Group	Not available
TIME_RANGE_START	Out of the scope of this integration
TIME_RANGE_END	Out of the scope of this integration

Linux computer systems data is taken from two Tivoli Data Warehouse views:

- Linux_Machine_Information_DV
- Linux_IP_Address_DV

Join operation between these two views is created through the System_Name columns. Similar to Windows agent, the last entry (maximum value from the WRITETIME column) for each machine is taken. If a machine has several IP addresses configured, one IPv4 address is taken (excluding local host addresses).

Table 8-3 shows UNIX agent columns mapping.

Table 8-3 UNIX agent columns mapping on ComputerSystem table

IBM_TRAM.ComputerSystem table (common dimension)	IBM Tivoli Monitoring UNIX Agent views and columns
CSID	Automatically generated
FQDN	Machine_Information_DV.LAT_Hostname
Manufacturer	Machine_Information_DV.LAT_Hardware_Manufacturer
Model	Machine_Information_DV.LAT_Hardware_Model
SerialNumber	Machine_Information_DV.LAT_Machine_Serial
VMID	Not available
PrimaryMACAddress	Not available
IpAddress	System_DV.LAT_Net_Address
Signature	Not available

IBM_TRAM.ComputerSystem table (common dimension)	IBM Tivoli Monitoring UNIX Agent views and columns
ManagedSystemName	Machine_Information_DV.System_Name
Group	Not available
TIME_RANGE_START	Out of the scope of this integration
TIME_RANGE_END	Out of the scope of this integration

UNIX computer systems data is taken from two Tivoli Data Warehouse views:

- Machine_Information_DV
- System_DV

Join operation between these two views is created through the System_Name column. Again, in this case, only the last entry (maximum value from the WRITETIME column) for each of the machines is selected.

Table 8-4 shows TADDM database columns mapping.

Table 8-4 TADDM database columns mapping on ComputerSystem table

IBM_TRAM.ComputerSystem table (common dimension)	TADDM views and columns
CSID	Automatically generated
FQDN	BB_COMPUTERSYSTEM40_V.FQDN_C
Manufacturer	BB_COMPUTERSYSTEM40_V.MANUFACTURER_C
Model	BB_COMPUTERSYSTEM40_V.MODEL_C
SerialNumber	BB_COMPUTERSYSTEM40_V.SERIALNUMBER_C
VMID	BB_COMPUTERSYSTEM40_V.VMID_C
PrimaryMACAddress	BB_L2INTERFACE41_V.HWADDRESS_C
IpAddress	BB_IPADDRESS73_V.STRINGNOTATION_C
Signature	BB_COMPUTERSYSTEM40_V.SIGNATURE_C
ManagedSystemName	BB_COMPUTERSYSTEM40_V.FQDN_C
Group	Not available
TIME_RANGE_START	Out of the scope of this integration
TIME_RANGE_END	Out of the scope of this integration

TADDM database may also contain historical data, for example information about the computer system configuration, which was discovered in the past. The same computer system identified, for example by the fully qualified domain name (FQDN) that is assigned to different machine models. This situation can occur, for example, with hardware replacement. Because of this reason, only the last entry for given FQDN is selected from computer systems view (maximum value from the LASTMODIFIEDTIME_C column per computer system).

Computer system may contain several network interfaces which implicate several IP addresses and MAC addresses. This information is stored by TADDM and obtained by the data synchronization utility based on the following joins:

- BB_COMPUTERSYSTEM40_V.PK_C = BB_COMPUTERSYSTACES_76E84D2AJ.PK__JDOID_C
- BB_COMPUTERSYSTACES_76E84D2AJ.PK__IPINTERFACES_C = BB_IPINTERFACE96_V.PK_C
- BB_IPINTERFACE96_V.PK__IPADDRESS_C = BB_IPADDRESS73_V.PK_C
- BB_IPINTERFACE96_V.PK__L2INTERFACE_C = BB_L2INTERFACE41_V.PK_C

To run the data population utility perform the following steps:

1. Extract the `.zip` package (listed for this chapter in Appendix A, "Additional material" on page 803) into a convenient location in the file system. The tool is in the `Cross_Product_Setup` subdirectory.
2. Update `config.properties` file with connections details to your databases. You must provide connections to three databases:
   - Database with the IBM_TRAM schema
   - Tivoli Data Warehouse database
   - TADDM database

   The tool uses JDBC connections, not the native connections, therefore reusing DB2 client configuration that was prepared in the previous steps is not possible.

   The `config.properties` file contains a sample configuration. Specify IP address, port on which the database process is listening, database name, schema name, and access credentials suitable for your environment. In Example 8-13, IBM_TRAM schema was created inside Tivoli Data Warehouse so the connections information is the same except for the schema name value.

Example 8-13 Sample JDBC configuration

```
########################################
# IBM_TRAM
########################################
tram.server=9.42.171.253
tram.port=60000
tram.user=db2inst1
tram.password=itso4you
tram.db=WAREHOUS
tram.schema=IBM_TRAM
########################################
# IBM Tivoli Monitoring - Tivoli Data Warehouse
########################################
tdw.server=9.42.171.253
tdw.port=60000
tdw.user=db2inst1
tdw.password=itso4you
tdw.db=WAREHOUS
tdw.schema=ITMUSER
########################################
# TADDM
########################################
taddm.server=9.42.171.157
taddm.port=50000
taddm.user=db2inst1
taddm.password=smartway
taddm.db=DEMO
taddm.schema=DB2INST1
```

3. Update the `run.bat` script. Replace the path to the Java Virtual Machine executable with your settings. By default, the tool is configured to run on Java provided with IBM Tivoli Common Reporting for Asset and Performance Management server. Replace the following string with path to java.exe program from your environment

   ```
   "C:\IBM\tivoli\tip\java\bin\java"
   ```

   If you do not have a JRE installed, you can use one that is provided with IBM Tivoli Common Reporting or with DB2 Client:

   "`C:\Program Files\IBM\SQLLIB\java\jdk\jre\bin\java.exe`**"**

4. Execute the `run.bat` script. The script inserts rows into IBM_TRAM.ComputerSystem table according to the algorithm described in "ComputerSystem table population" on page 216.

   Figure 8-11 shows an error reported for the UNIX agent query. The error is because no UNIX machines are in the environment so no UNIX database tables are in the warehouse.

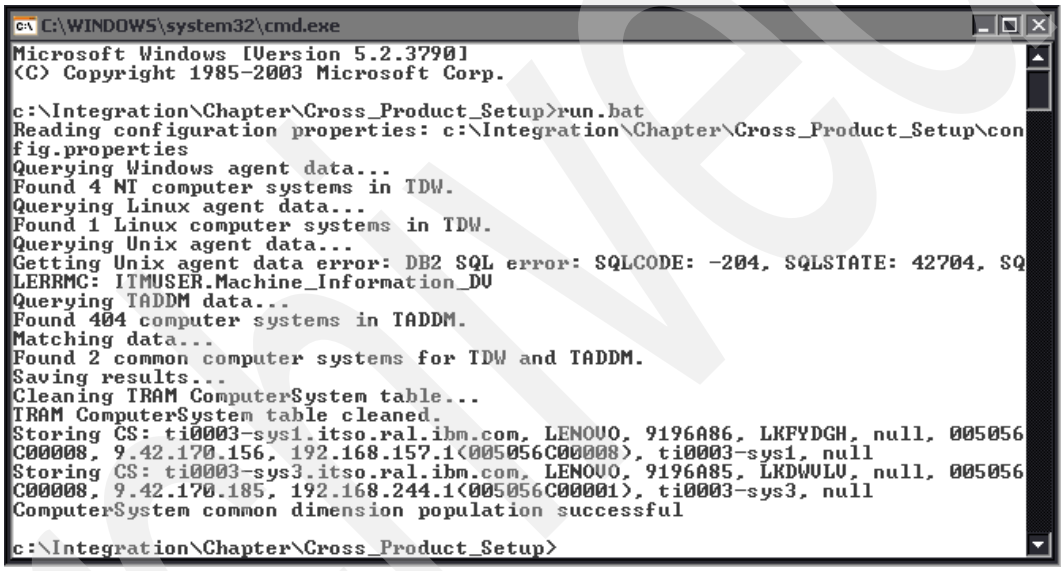


Figure 8-11 Sample population tool execution

If you are running the population utility on a Linux or UNIX platform you can execute the Java class manually (see Example 8-14) by specifying a similar command as in the `run.bat` script. The path to Java Virtual Machine is also required.

Example 8-14 Executing the Java class manually

```
[root@tsekman2 Synchronize]# /opt/IBM/tivoli/tip/java/bin/java -classpath
lib/db2jcc.jar:lib/db2jcc_license_cu.jar:lib/synchronizer.jar
com.ibm.tivoli.tcr.tram.synchronizer.PopulateData config.properties
```

8.5.4 Data sources definition in IBM Tivoli Common Reporting

Now you have products databases prepared for cross-product reporting sample. You also have the DB2 client installed on your IBM Tivoli Common Reporting for Asset and Performance Management server machine and cataloged connections to your databases. It is time to start using reporting utility. First we need to define three data sources in the Cognos BI Reporting engine.

Sample cross-product model attached to the book is based on the following three data sources:

- IBM_TRAM: Connection to the database in which IBM_TRAM schema was created.

  In our case it will be the same connection as TDW. Even in this case, a separate data source must be defined.
- TDW: Connection to the Tivoli Data Warehouse database
- TADDM: Connection to the TADDM database

Using the same names for the data sources definitions is a good idea; modifying the sample model is not necessary. However, small updates in the area of schema names still might be required. We describe the details in a subsequent sections.

To connect IBM Tivoli Common Reporting for Asset and Performance Management to a database, create a data source definition within the Cognos reporting engine. The definition is based on the native or ODBC connection. Cognos does not support light JDBC connections.

To show a sample configuration, we use connections that we have defined in the section 8.5.1, “Database native client” on page 206. We will create data sources in Cognos BI Reporting engine based on databases’ aliases cataloged in that section.

1. Log in to the IBM Tivoli Common Reporting web interface. The application is available by default under the following addresses:
   - http://myserver.mycompany.com:16310
   - https://myserver.mycompany.com:16316/ibm/console

   Connection on non-secure port 16310 is redirected to a secure one 16316.

   Enter the user name and password. Choose the user that has the administrative privileges (by default tipadmin).
2. In the navigation area on the left side of the window, click **Reporting** → **Common Reporting**. The IBM Tivoli Common Reporting for Asset and Performance Management report navigation pane opens, IBM Cognos Connection application. See Figure 8-12 on page 224.

Archived

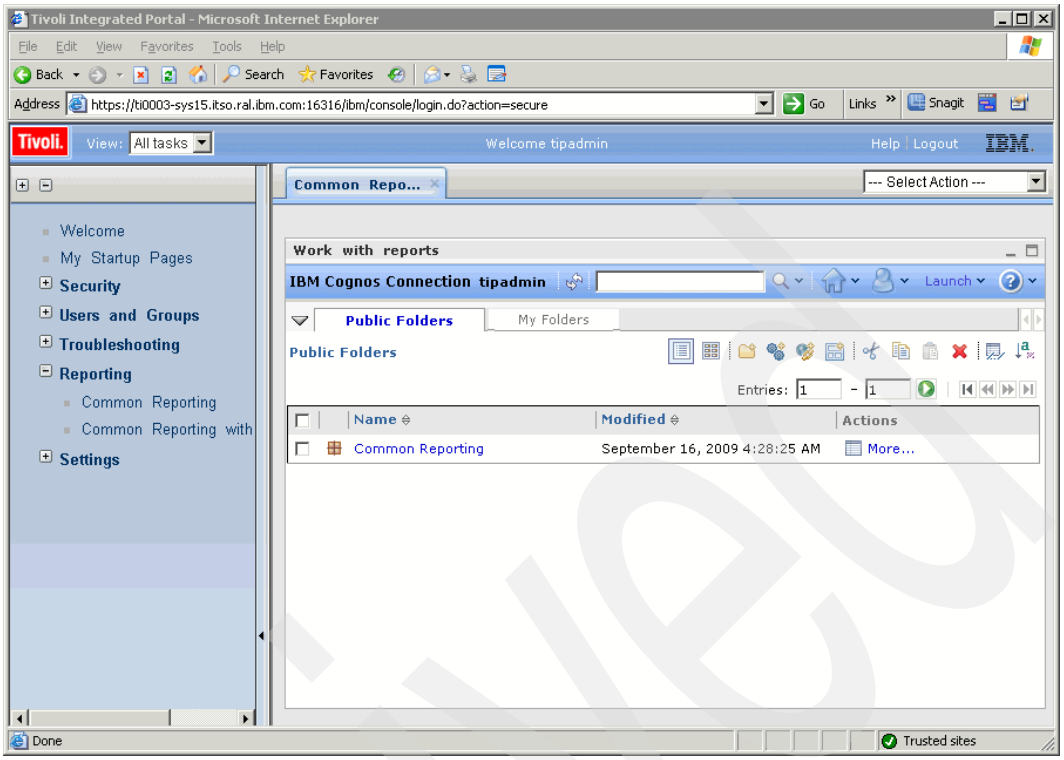


Figure 8-12 IBM Tivoli Common Reporting for Asset and Performance Management navigation pane

3. At the top right corner of the portlet click **Launch** → **IBM Cognos Administration**. See Figure 8-13.

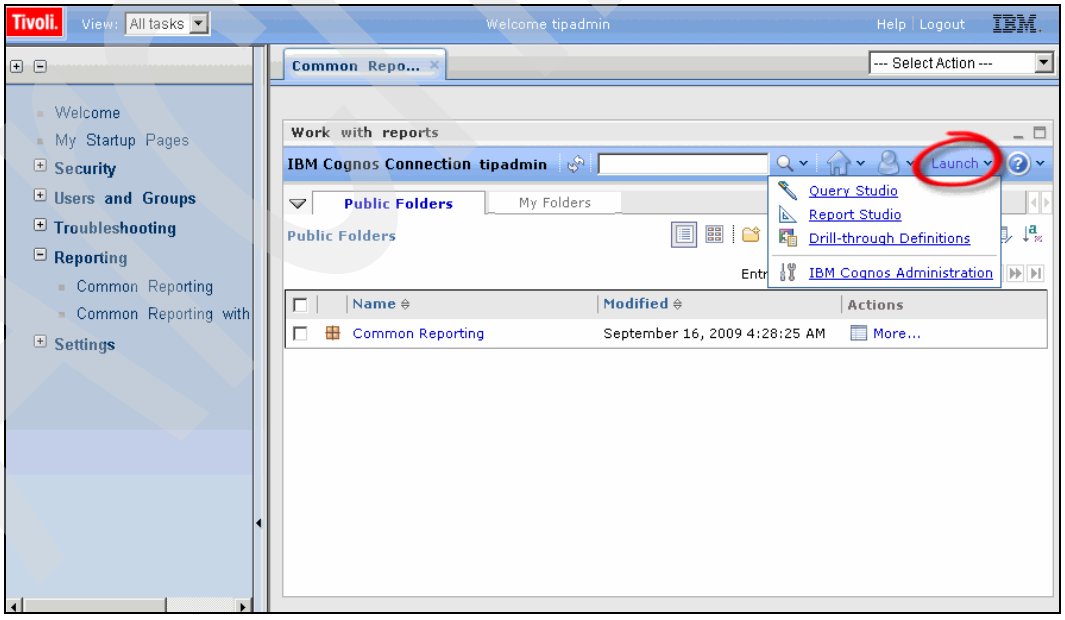


Figure 8-13 Switching between Cognos applications

4. In the IBM Cognos Administration portlet, click the **Configuration** tab. All the data sources defined within your instance are listed. See Figure 8-14 on page 225. By default, only one

connection is created during the IBM Tivoli Common Reporting 1.3 server installation. That connection is used by MOSWOS (Manage Our Stuff With Our Stuff) report.

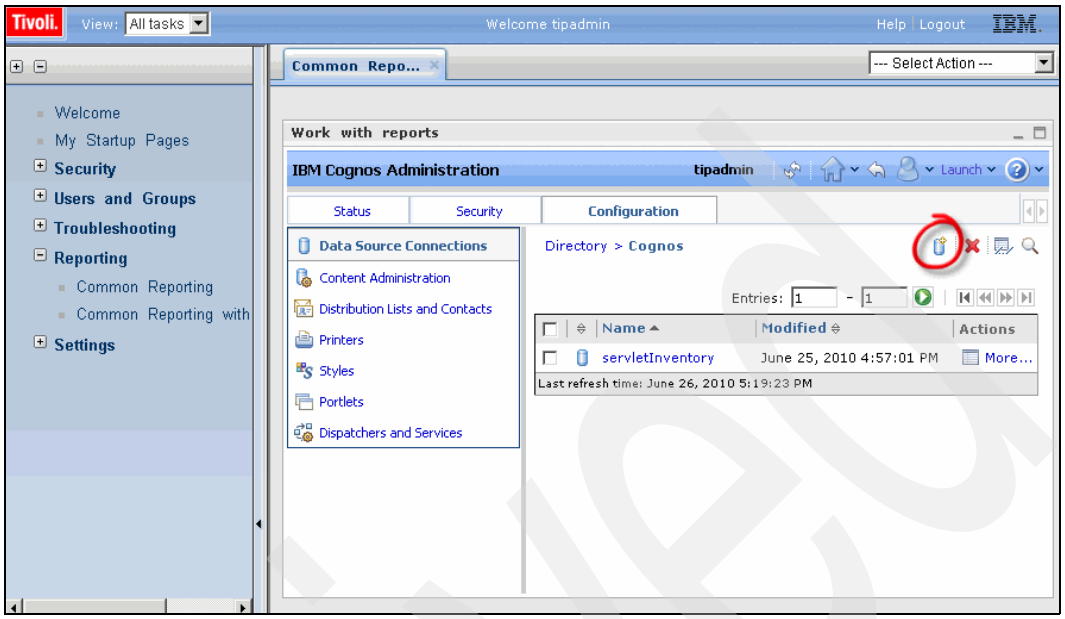


Figure 8-14 Data sources administration portlet

5. To define a data source, click the **New Data Source** icon, which is located in the top right corner of the administration portlet. Follow the New Data Source wizard. Specify the following values:

   – Name: `TDW` (See Figure 8-15.)

     The sample model for this chapter is based on that data source connection name. Therefore, define the same name (TDW).

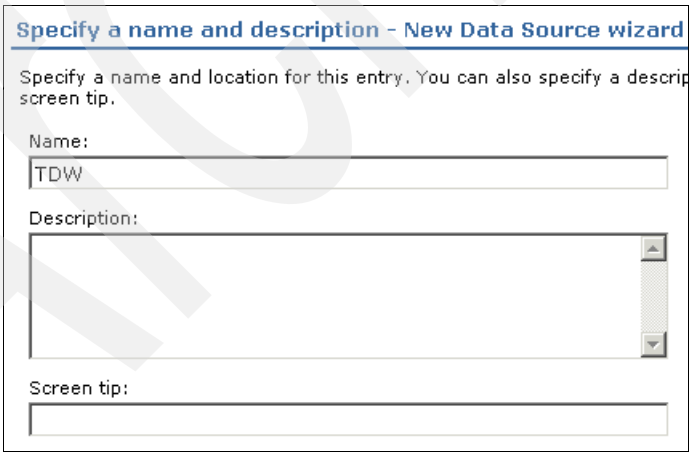


Figure 8-15 Enter data source name

   – Type: `DB2` (See Figure 8-16 on page 226.)

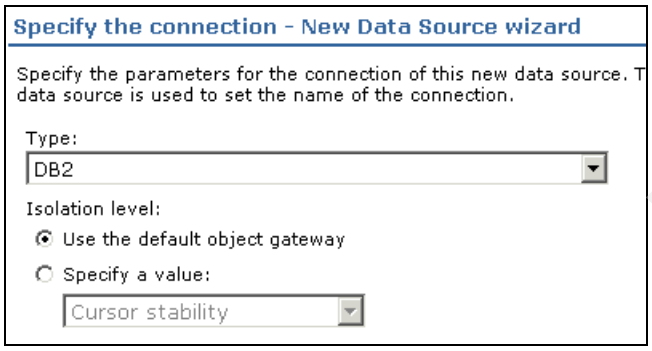


Figure 8-16 Specify connection type

– DB2 database name: `TDWDB1` (See Figure 8-17.)

  Use the database alias under which you have cataloged your Tivoli Data Warehouse database.

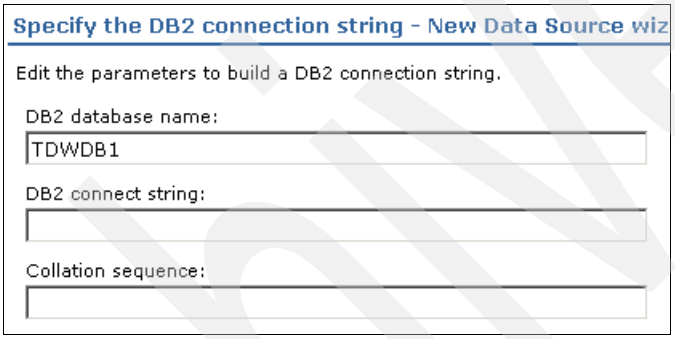


Figure 8-17 Enter database alias from the DB2 client catalog

Select **Signons** and **Password**, and enter the **User ID** and **Password** to your Tivoli Data Warehouse database. See Figure 8-18.

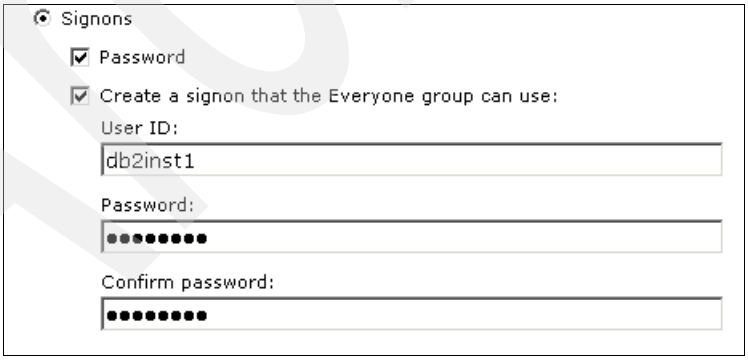


Figure 8-18 Enter database credentials

6. Click **Finish** to save the connection definition and return to the administration portlet.
7. Click the name of the newly created data source. We will test the connection.

Archived

8. Select **Test the connection** icon on the right side of the connection entry. See Figure 8-19.

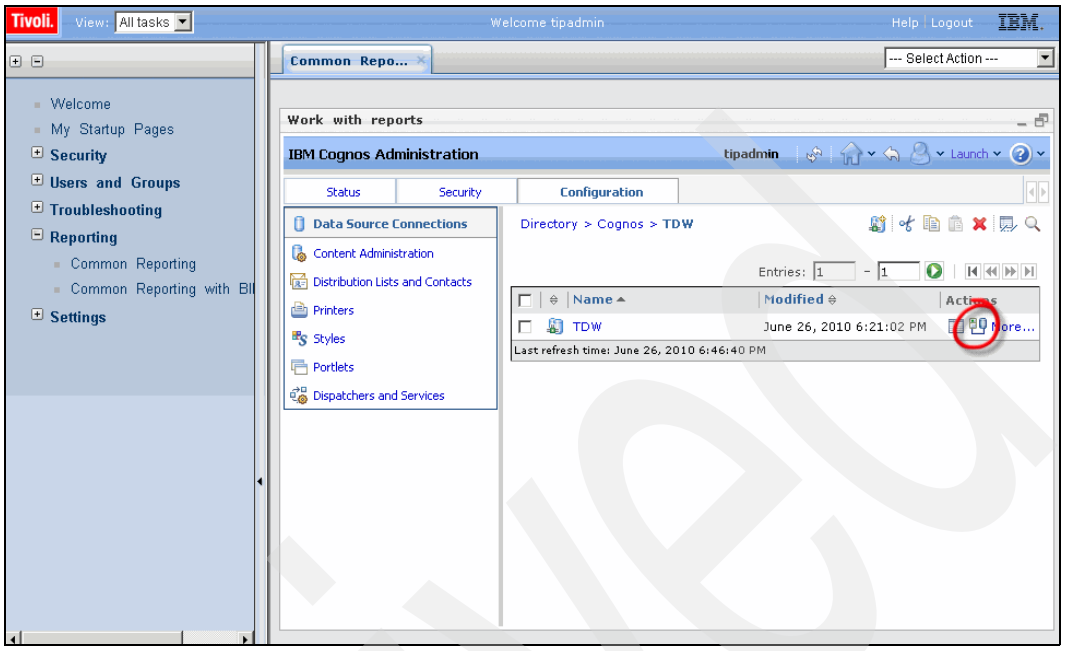


Figure 8-19 Testing the connection

9. Click **Test** in the Test the connection panel.

   Cognos engine tests the connection and returns the result, as shown in Figure 8-20 on page 228.

> **Note:** On Linux or UNIX platforms you may experience difficulties with the data sources tests. You might see the errors related with native libraries loading. IBM Tivoli Common Reporting for Asset and Performance Management must be able to find the native DB2 libraries. To be sure, verify whether the DB2 `lib32` directory exists before starting the IBM Tivoli Common Reporting for Asset and Performance Management server. Configure the system library path to point to the database client library directory by modifying the following environment variable:
>
> - LIBPATH on AIX
> - SHLIB_PATH on HP-UX
> - LD_LIBRARY_PATH on Linux or Solaris
>
> An example is as follows:
>
> ```
> export LD_LIBRARY_PATH=/opt/ibm/db2/V9.5/lib32:$LD_LIBRARY_PATH
> ```

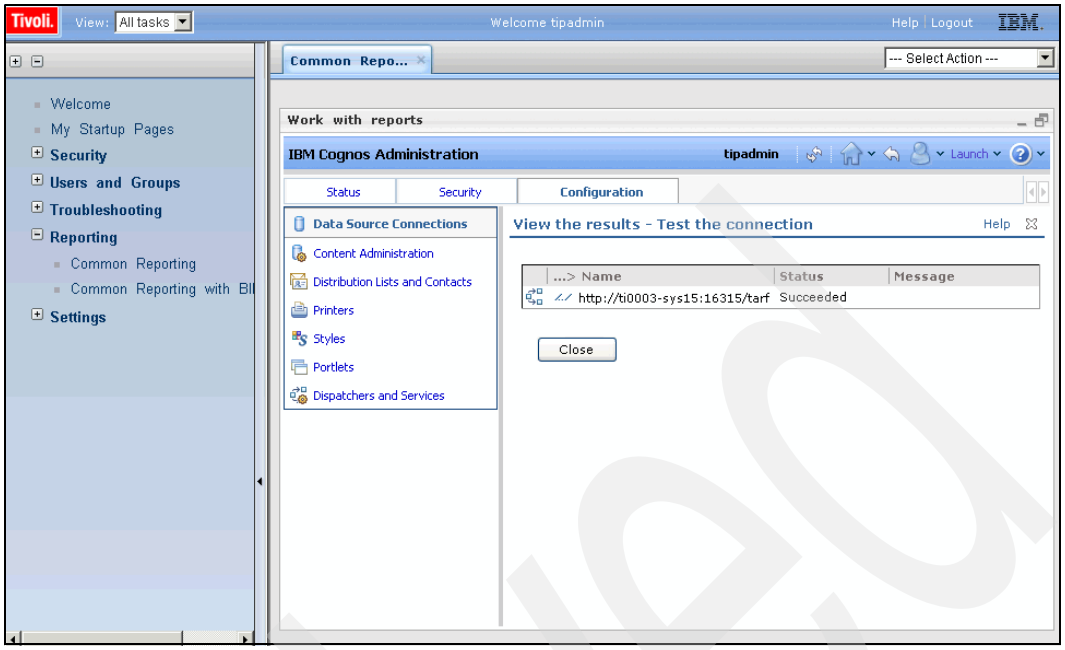


Figure 8-20 Successful connection test

Follow the step 4 on page 224 through step 9 on page 227 to define two remaining data sources: IBM_TRAM and TADDM. In our example, the IBM_TRAM data source is defined in the same way as TDW, using the same database alias. The reason is because IBM_TRAM schema was created within Tivoli Data Warehouse database, not in the separate database instance. Nevertheless it is important to define IBM_TRAM as a separate connection because the sample cross-product model is based on this data source.

Now, your three data sources are defined and tested. See Figure 8-21.

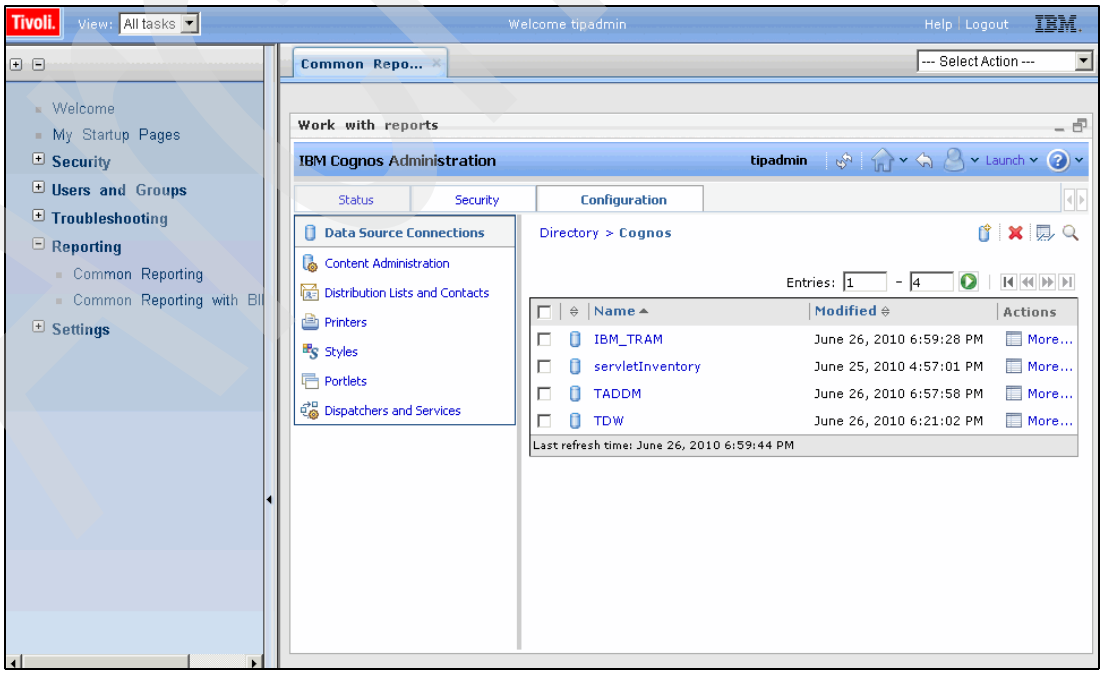


Figure 8-21 Required data sources

For more information about the data sources definition and how to set up data sources for various database vendors, see the IBM Tivoli Common Reporting for Asset and Performance Management documentation:

http://publib.boulder.ibm.com/infocenter/tivihelp/v3r1/topic/com.ibm.tivoli.tcr_cog.doc/ttcr_cog_config_db.html

8.5.5 Cross-product model update

The `.zip` package that is listed as an additional material for this chapter (see Appendix A, "Additional material" on page 803)contains a sample cross-product model that can be used to generate Cognos reports. The model is delivered in two forms: published version and source file. If you have default schema names in your IBM Tivoli Monitoring and TADDM databases, respectively ITMUSER and DB2INST1, you may skip this section and go straight to the next step to import a published version of the model into your IBM Tivoli Common Reporting 1.3 server (8.5.6, "Package import" on page 235). Otherwise, you must modify the model source by updating it with the schema names from your databases. After the modification you will be able to publish the model on your reporting server directly without the necessity of the package import.

However, performing this step is a good practice, even if your schema names match the values from the ready-package. By performing this step, you will set up Cognos BI Modelling tool (Framework Manager) and learn how to modify the model, customizing it for your needs.

What publishing the model means: The source of the model is written in the XML file that is maintained by the Cognos BI Modeling tool: Framework Manager. The tool is attached (connected) to the IBM Tivoli Common Reporting for Asset and Performance Management server (more precisely to the Cognos BI Reporting engine). This connection Framework Manager is able to get the information about the data sources that are defined in Cognos BI Reporting engine and also store the information about objects from the model into the Cognos content store. By storing selected objects (query subjects, query items, relationships, and so on) in the content store Framework Manager, they become more available for the reporting engine users. This process is called *model publication*. After the model is published, it can be exported from the Cognos BI Reporting engine into the packaged version (different from the source) and imported into another Cognos BI Reporting engine instance.

Framework Manager allows necessary configuration updates. It is the utility in which you can model a business domain, an abstraction layer that is based on the objects from the database. The model covers database structure and facilitates report creation in IBM Tivoli Common Reporting for Asset and Performance Management (especially ad-hoc reports).

More information about the modeling tool is in the *Framework Manager User Guide*, available at the following location:

http://publib.boulder.ibm.com/infocenter/c8bi/v8r4m0/topic/com.ibm.swg.im.cognos.ug_fm.8.4.0.doc/ug_fm.html

Framework Manager configuration

To update and publish the cross-product reporting model you need the additional Cognos BI Modeling tool: Framework Manager. It is a stand-alone application, development utility, which can be installed only on the Windows platform.

The program is available on Windows installation media of IBM Tivoli Common Reporting for Asset and Performance Management (part number CZ6SWML). The installer executable

(`issetup.exe`) file is in the `CognosModeling\win32` subdirectory. After the installation (Figure 8-22) be sure to attach the Framework Manager instance to the IBM Tivoli Common Reporting 1.3 server by specifying suitable URIs to Cognos BI Reporting engine. Follow the instruction at the IBM Tivoli Common Reporting for Asset and Performance Management information center to install and configure Framework Manager:

`http://publib.boulder.ibm.com/infocenter/tivihelp/v3r1/topic/com.ibm.tivoli.tcr_cog.doc/ttcr_cog_configure_frameworkmgr.html`

Note: If you have IBM Tivoli Common Reporting 1.3 server running on a Windows machine, you may install Framework Manager on the same machine, to the directory of the Cognos engine that is installed and used by the IBM Tivoli Common Reporting 1.3 server. The advantage of this solution is that Framework Manager requires no additional configuration and is ready to use after the installation. To achieve this state, be sure to specify the correct destination directory during the modeling tool installation wizard. If your IBM Tivoli Common Reporting instance is installed in `C:\IBM\tivoli\tip` directory (the default location), specify the following destination directory for Framework Manager:

`C:\IBM\tivoli\tip\products\tcr\Cognos\c8`

Ignore the warnings about replacing existing files. You will have to stop IBM Tivoli Common Reporting for Asset and Performance Management server for the installation time.

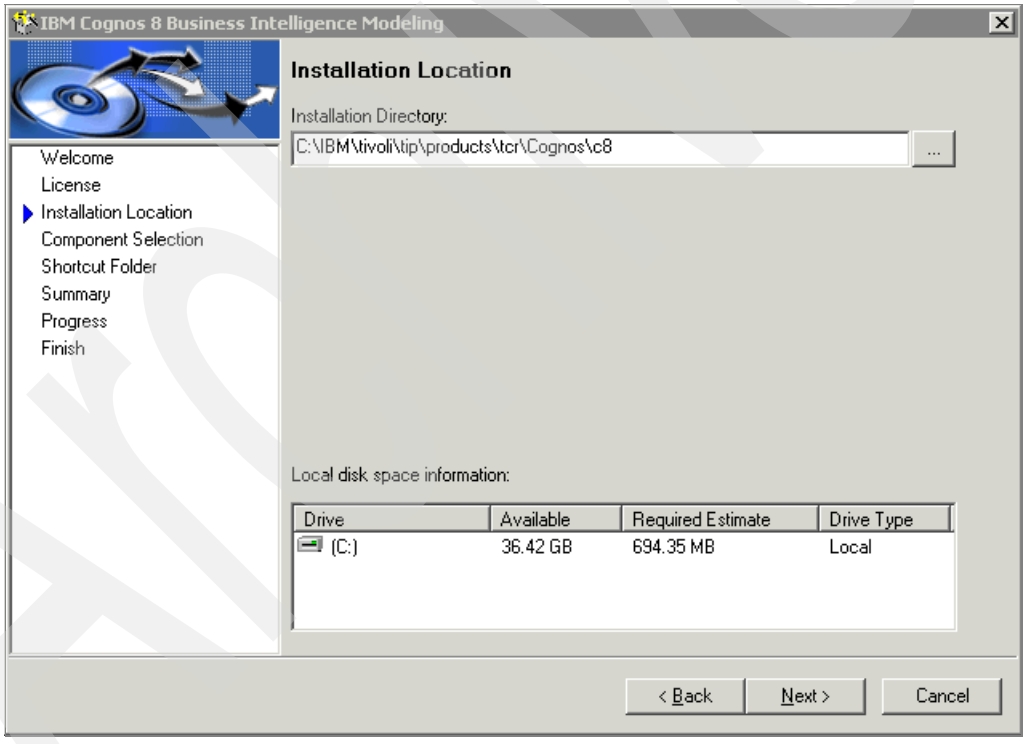


Figure 8-22 Framework Manager destination directory

After the installation and configuration, you may start the Framework Manager by clicking either of the following selection paths (if you have IBM Tivoli Common Reporting 1.3 server installed on the same machine):

- **Start → All Programs → Tivoli Common Reporting → Framework Manager**
- **Start → All Programs → IBM Cognos 8 → Framework Manager**

Working with Cognos model

Now you are ready to work with the Cognos models. Extract the `.zip` package (listed as an additional material for this chapter in Appendix A, “Additional material” on page 803) into a convenient location in the file system. The model is located in the `Cross_Product_Model` subdirectory.

1. Select **Open a project** from the Framework Manager welcome window and browse to the `Cross_Product_Model` directory. Select the **Cross_Products_Model_v3.cpf** file and click **Open**. See Figure 8-23.

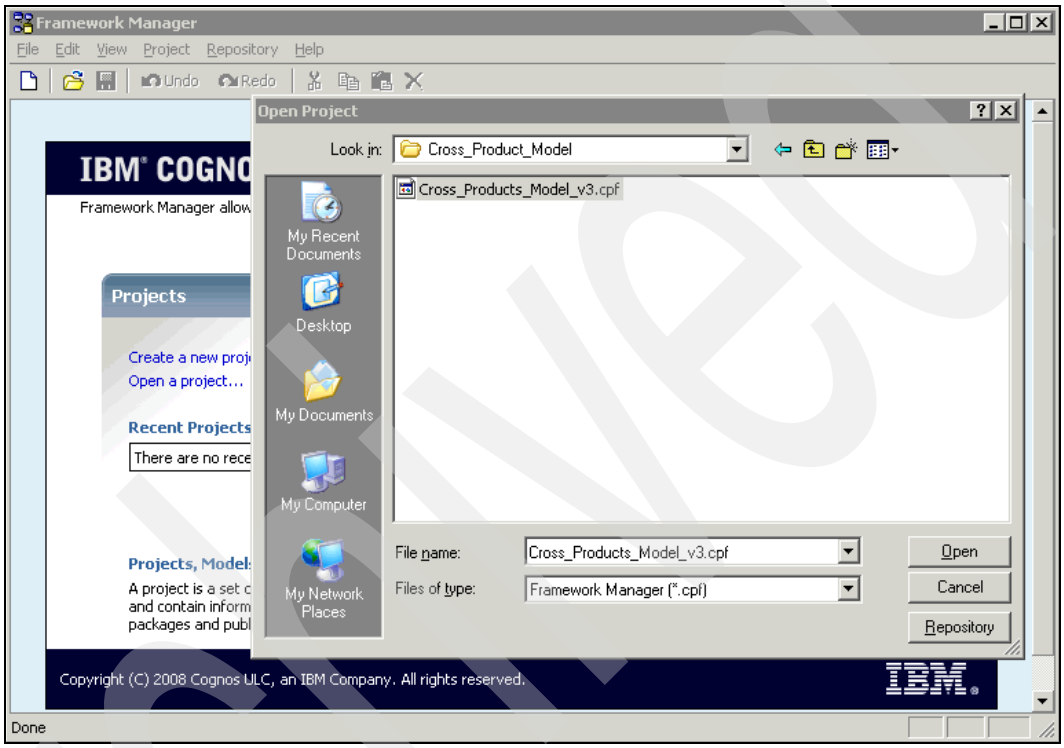


Figure 8-23 Opening Cognos model project

2. Framework Manager tries to establish the connection with IBM Tivoli Common Reporting for Asset and Performance Management server (Cognos BI Reporting engine). Framework Manager must be able to get access to data sources that are defined in Cognos. The connection is performed over HTTP protocol so you must accept the certificate and provide administrator credentials (tipadmin user name and password), the same account that you are using to log in to the IBM Tivoli Common Reporting console. In Framework Manager, you have to authenticate twice because there is no support for single sign-on functionality.

   Framework Manager does not accept cookies, therefore provide the same user name and password twice:

   – The first time is to access WebSphere application container. See Figure 8-24 on page 232.



Figure 8-24 First login window in Framework Manager

– The second time is to access Cognos application deployed inside that container. See Figure 8-25.

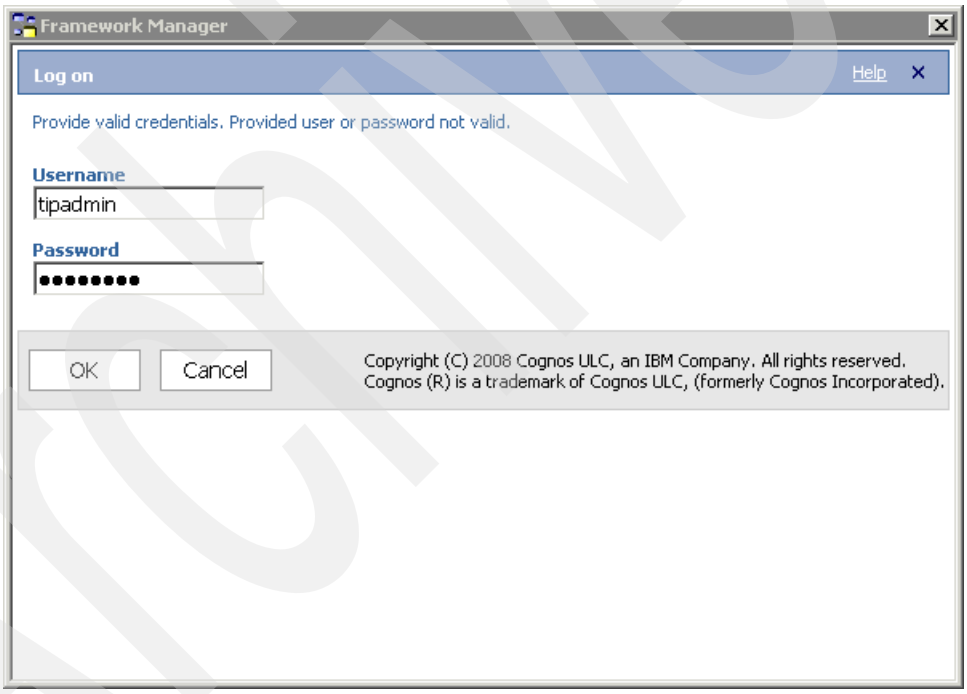


Figure 8-25 Second login window in Framework Manager

3. After connection, establishing Framework Manager loads the model. Examine the various namespaces and query subjects, which are available in the model. See Figure 8-26 on page 233.

Archived

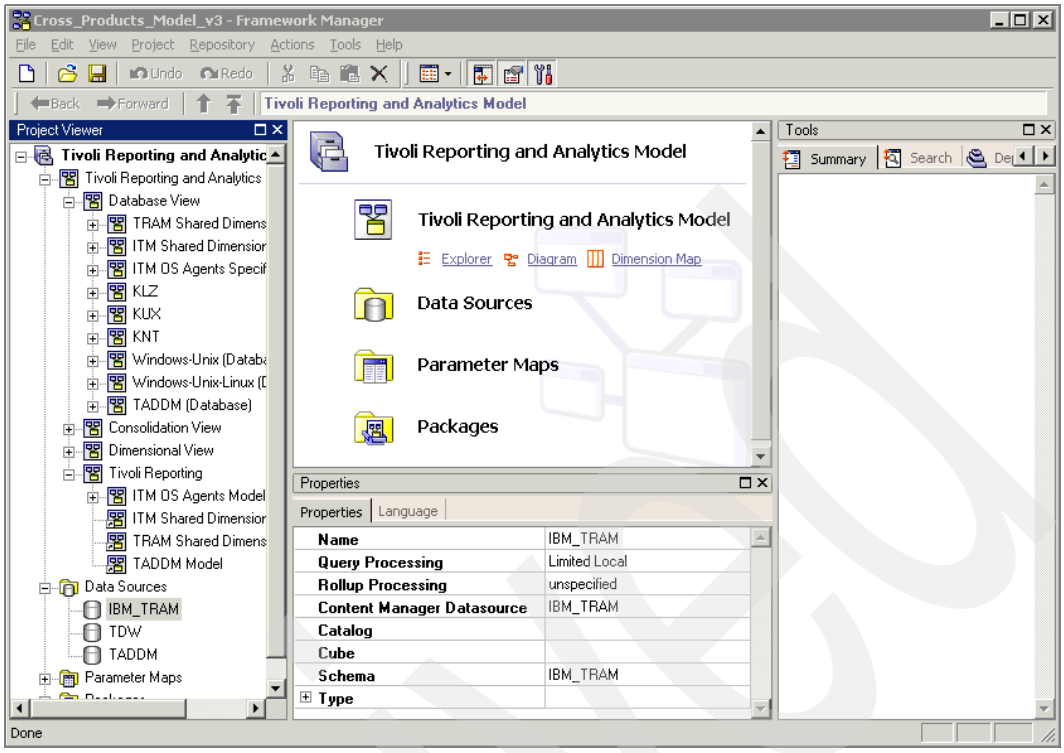


Figure 8-26 Namespaces and query subjects which are available in the model

4. Update the schema names in the data sources definitions:
   a. Select a data source node from the model tree. See Figure 8-27.

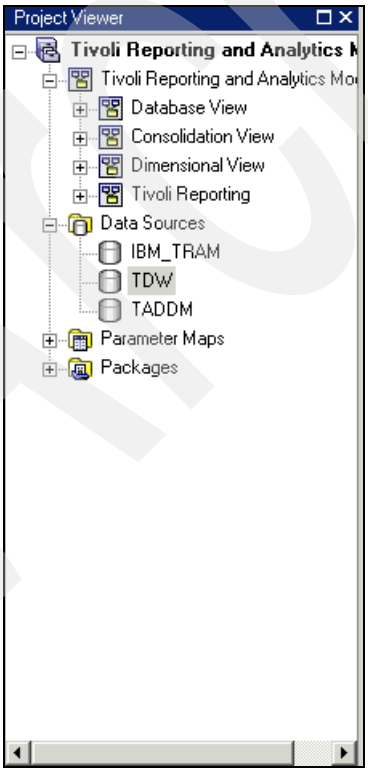


Figure 8-27 Data Sources model node

b. Specify your schema name for TDW and TADDM data sources. See Figure 8-28.

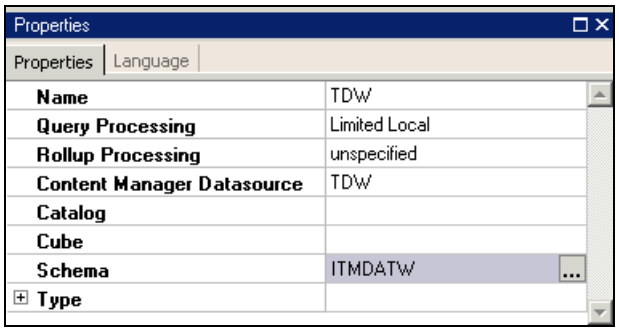


Figure 8-28 Schema name property

5. Optional: you can introduce additional changes to the model. For example add query subjects for the tables or views you are interested in. For this task you will need to be familiar with the structure of product's database.
6. Save the model changes.
7. Publish the package:
   a. Right-click the **Tivoli Reporting Package** tree node and select **Publish Packages**. See Figure 8-29.

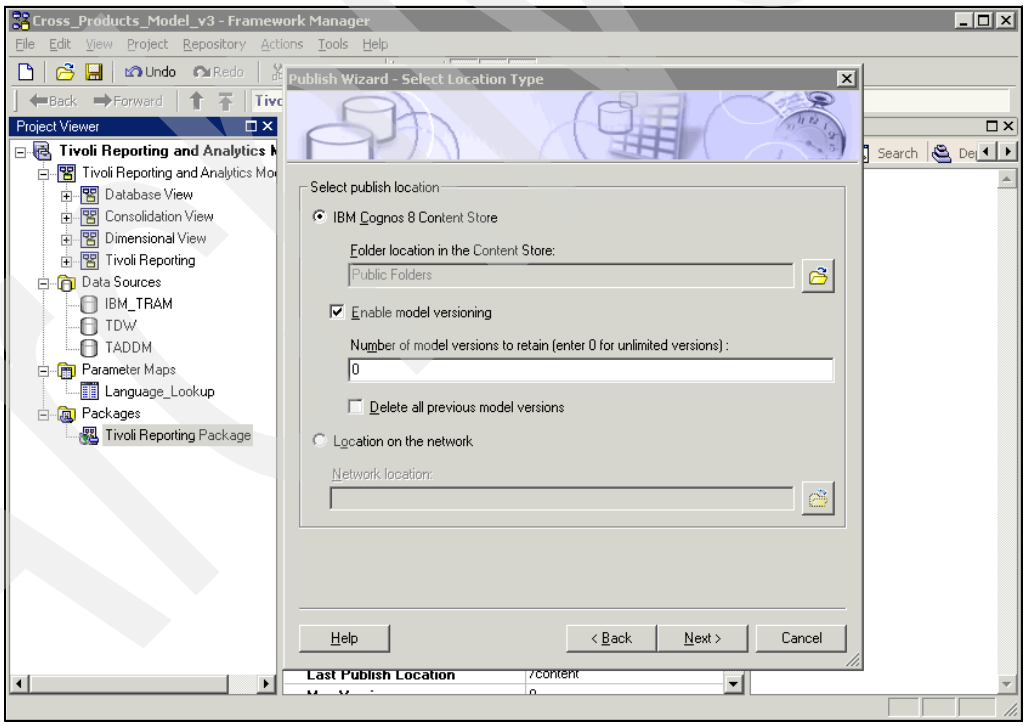


Figure 8-29 Model publication wizard

   b. Go through the publication wizard with default settings by clicking **Next**.
   c. Click **Publish** to publish the model. Ignore the warnings in the summary window at this time. See Figure 8-30 on page 235.

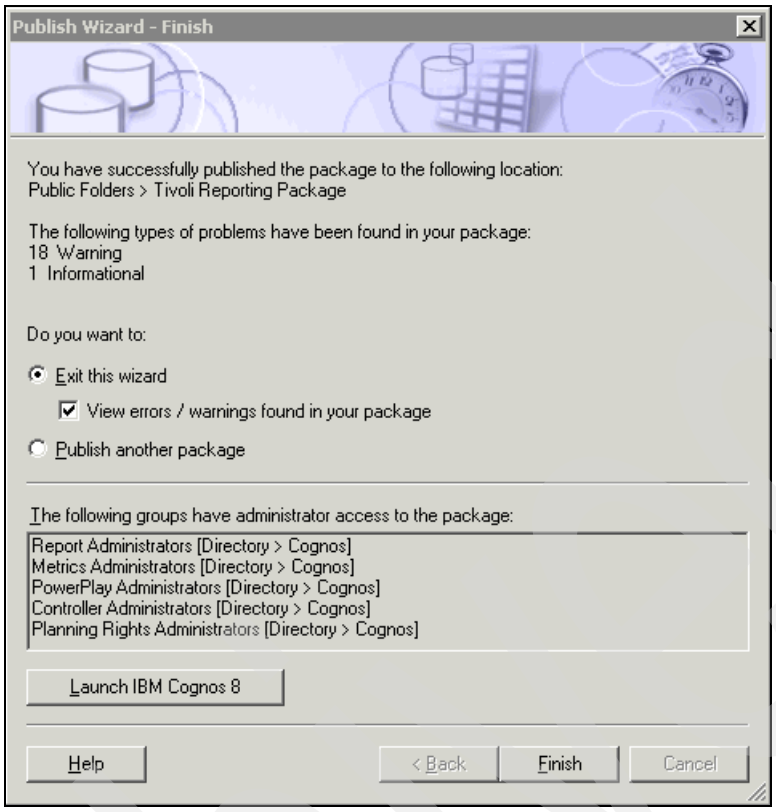


Figure 8-30 Model publication summary

After successful publication your model becomes available in IBM Tivoli Common Reporting for Asset and Performance Management web application.

8.5.6 Package import

The ready-to-use cross-product sample package is listed as an additional material for this chapter (see Appendix A, “Additional material” on page 803). You may import that package into your IBM Tivoli Common Reporting for Asset and Performance server without any modifications if you have the default schema names for Tivoli Data Warehouse (ITMUSER) and TADDM (DB2INST1) databases. Otherwise, follow the steps in 8.5.5, “Cross-product model update” on page 229 to update the model and publish it on the server. The result of both paths (publication and import) is the same: the reporting package available on IBM Tivoli Common Reporting for Asset and Performance Management server, ready for reports generation.

To import the package, perform the following steps:

1. Extract the `.zip` package for this chapter (see Appendix A, “Additional material” on page 803) into the convenient location in the file system and copy `TivoliReportingPackage.zip` into the Cognos BI Reporting engine deployment folder:

   `<TCR_installation_directory>products\tcr\Cognos\c8\deployment.`

2. Log in to the IBM Tivoli Common Reporting for Asset and Performance Management console and go to Common Reporting.
3. In the Work with reports window on the right, choose the **IBM Cognos Administration** from the Launch drop-down list.

4. In the IBM Cognos Administration portlet select **Configuration** tab and open the **Content Administration** section.
5. Start a new package import wizard by clicking the **New Import** icon, which is located in the top-right corner of the administration portlet. See Figure 8-31.

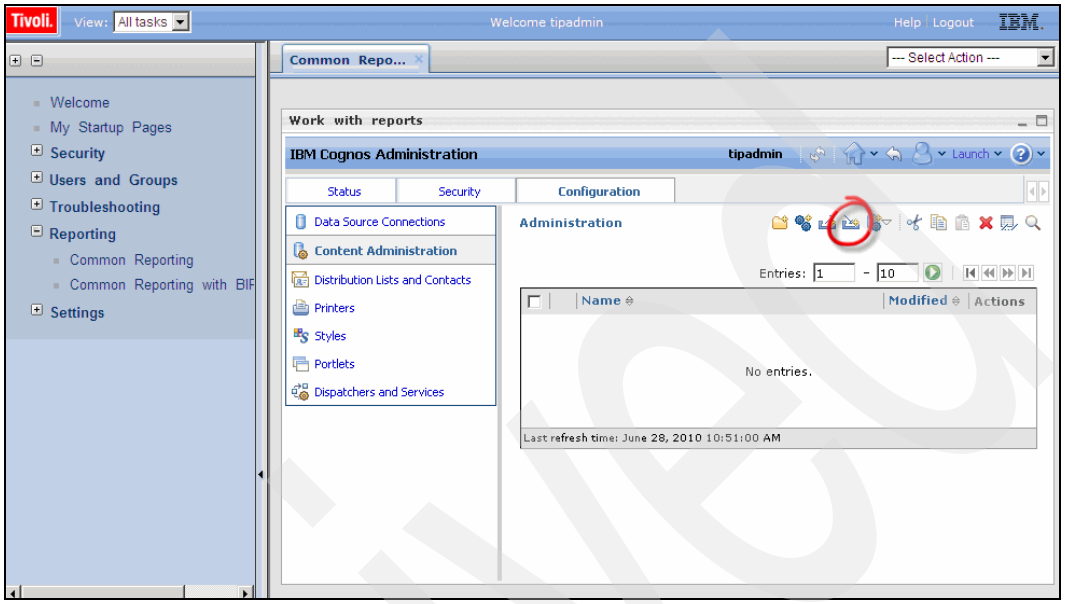


Figure 8-31 Importing report package

6. Follow the wizard to import the package:
   a. Select **TivoliReportingPackage** from the list of available packages. The list contains all the `.zip` files from the deployment directory.
   b. Keep the import name default.
   c. Select which element should be imported from the package. This important step is frequently omitted. Select the **Tivoli Reporting Package** check box. See Figure 8-32.

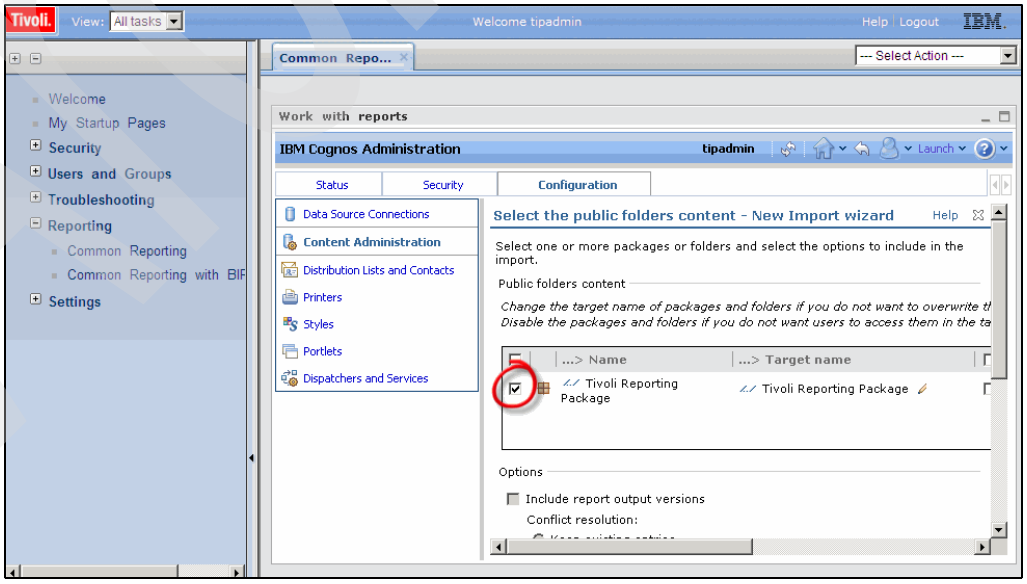


Figure 8-32 Package entry selection

d. Follow the remaining wizard steps, accepting the default settings.

e. Save and run after the import task is defined. Click **Finish** and then click **Run**.

You have now defined import task and executed it immediately at the end of the import wizard. Cross-product reporting package is available in IBM Tivoli Common Reporting for Asset and Performance Management console.

8.6 Scenario walk-through

You may now test the integration scenario. This section also explains how to create sample cross-product ad-hoc reports.

8.6.1 Integration test

To test the integration scenario perform the following steps:

1. Log in to the IBM Tivoli Common Reporting for Asset and Performance Management web application and open Common Reporting portlet: IBM Cognos Connection application. You see the new Tivoli Reporting Package (Figure 8-33). This package was published or imported to the Cognos BI Reporting engine in previous steps.

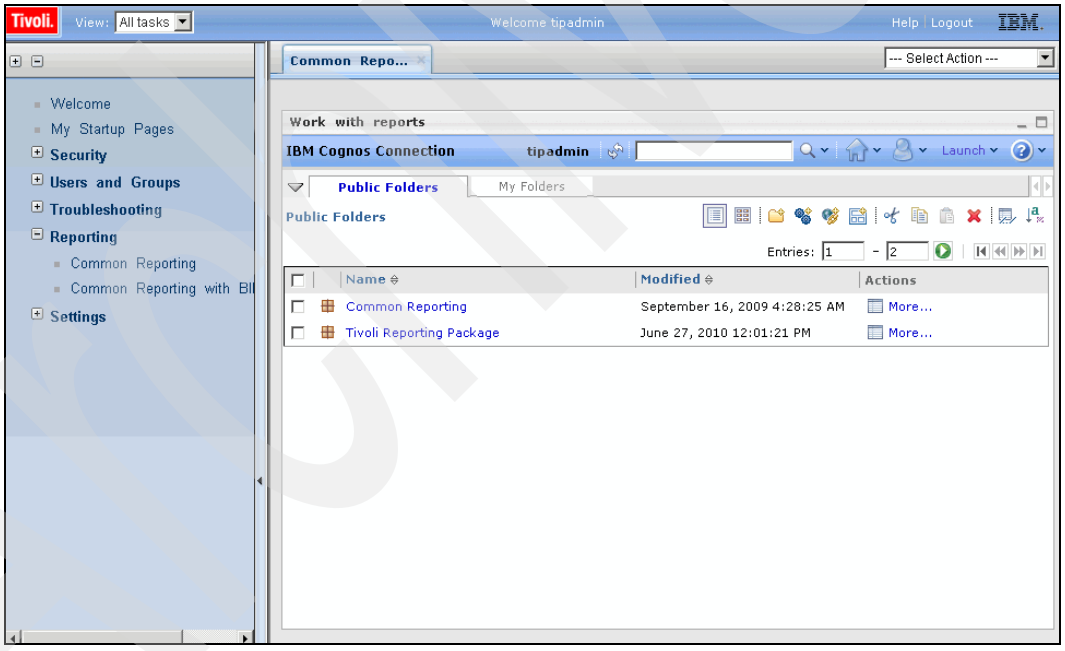


Figure 8-33 Tivoli Reporting Package - cross-product model

2. Click **Tivoli Reporting Package** to open it. The package is empty; it has no advanced or ad-hoc reports at this time. Only the model is available, but it is not visible in IBM Cognos Connection application.

3. To test the model in action, open Query Studio application. At the top right corner of the portlet click **Launch** → **Query Studio**. See Figure 8-34 on page 238.

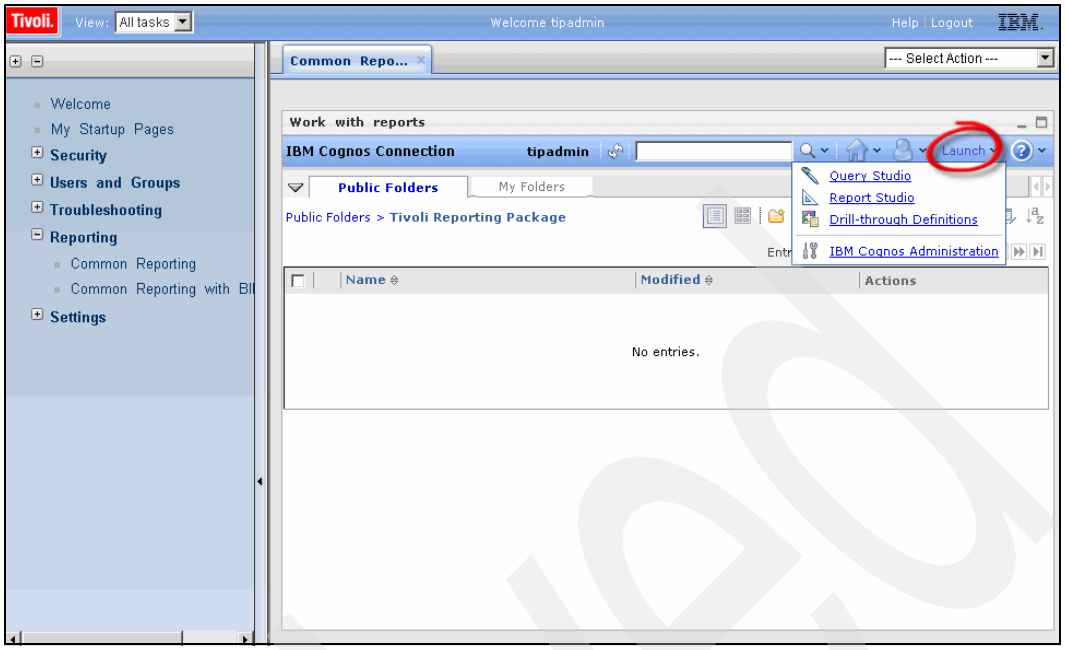


Figure 8-34 Launching Query Studio for cross-product package

Query Studio application can run a report that is based on one query composed from the elements taken from domain model. The query result is presented in a table. Additionally chart can be added. The main area of the Query Studio is the result table into which you drag elements from the model tree, available on the left side of the application window. See Figure 8-35 on page 239.

The cross-product model contains the following branches:

- ITM OS Agents Model: Query subjects based on views from Tivoli Data Warehouse,
- ITM Shared Dimensions: Query subjects based on MANAGEDSYSTEM table, created and populated in the previous sections,
- TRAM Shared Dimensions: Query subjects based on the IBM_TRAM schema, created and populated in the previous sections,
- TADDM Model: Query subjects based on views from TADDM database

Archived

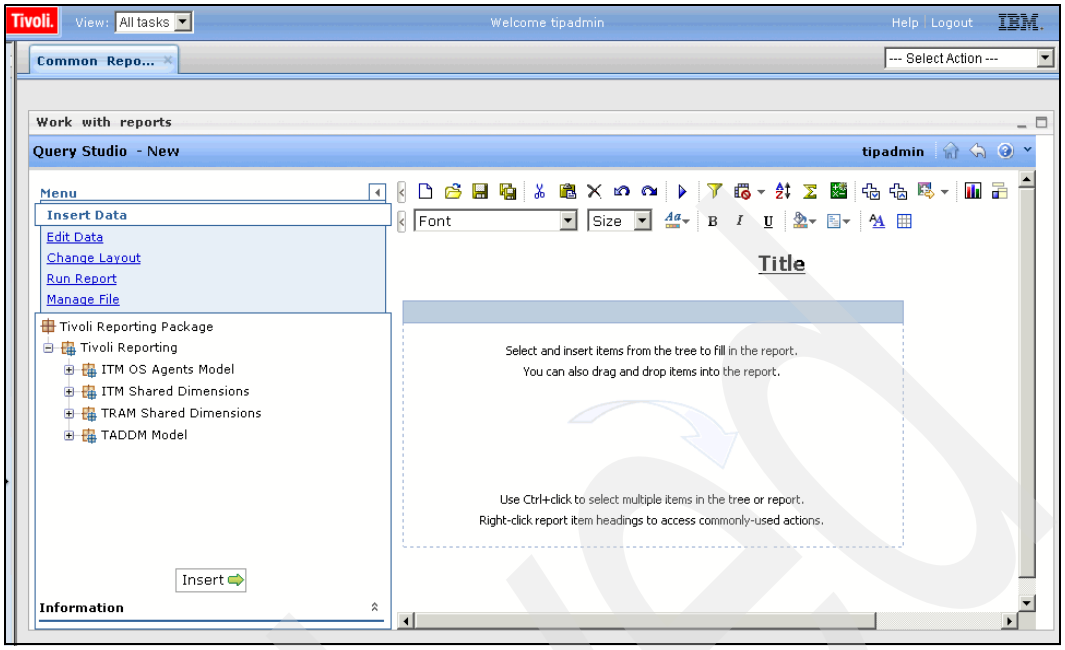


Figure 8-35 Cross-product model in Query Studio

4. Drag the following query items from the model:
   a. Tivoli Reporting / TADDM Model / Computer System / Manufacturer
   b. Tivoli Reporting / TADDM Model / Computer System / Model
   c. Tivoli Reporting / ITM Share Dimensions / Managed System / Managed System Name
   d. Tivoli Reporting / ITM Share Dimensions / Managed System / Agent Type
   e. Tivoli Reporting / ITM OS Agents Model / Windows / Daily / System Daily / Average % Total Processor Time

   While you drag elements, Cognos creates the necessary joins for you, execute the queries, aggregate the data, and present the results. You see a table that is similar to the one in Figure 8-36 on page 240. Notice that daily data taken from System Daily query subject where it is automatically rolled up (aggregated). Average CPU utilization value is presented for the entire time range, gathered in Tivoli Data Warehouse. Therefore, we do not see average values for each day but average values for all days computed (by Cognos) per selected machine.

Archive

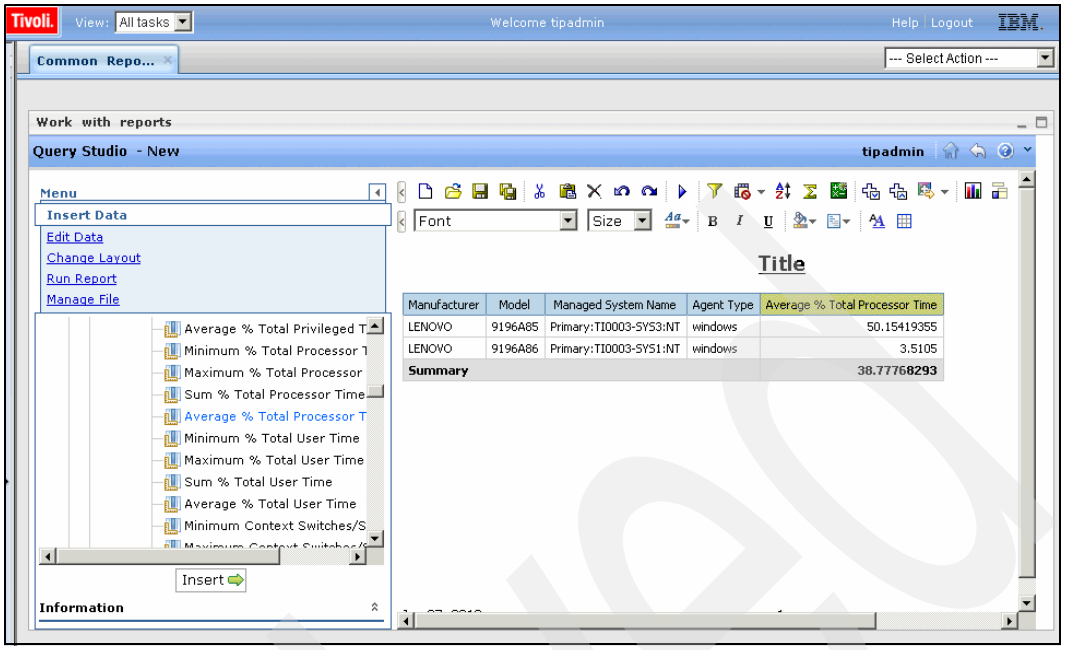


Figure 8-36 Sample cross-product report

What exactly was done by Cognos reporting engine behind this sample scenario? Three tables were joined by the engine based on the rules that are defined in the model. Two tables are taken from Tivoli Data Warehouse, and one is from TADDM database. Computer System view from TADDM was joined with Computer System table from shared dimension namespace, which subsequently was also joined with System Daily view from Tivoli Data Warehouse.

Cognos performed cross-database data merging outside the databases, on the IBM Tivoli Common Reporting for Asset and Performance Management server machine. In certain cases, this merge might have considerable impact on query execution performance and disk space requirements.

The user who prepares the report in Query Studio does not have to know what database structures are behind the model. All relationships, foreign and primary keys, views and tables are hidden behind the model. The model designer is responsible to deliver complex database structures in a clear form, understandable by a business domain analyst.

Figure 8-37 on page 241 presents lineage of the data that is used in the test report.

Archive

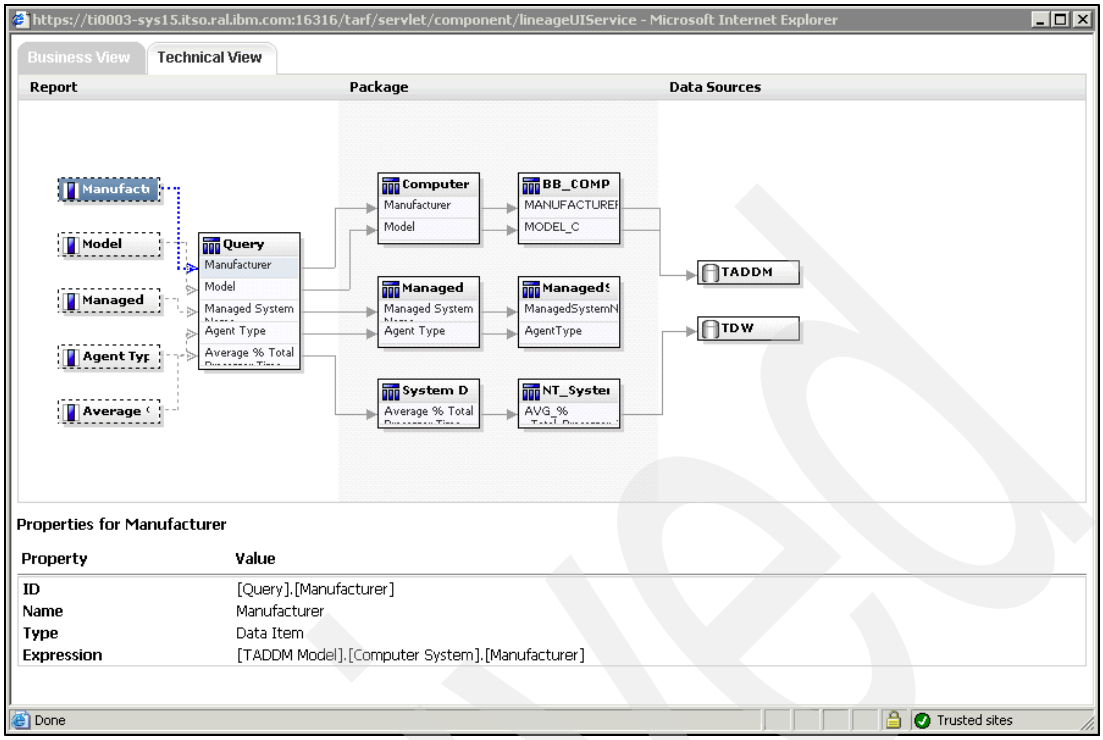


Figure 8-37 Cross-product query details

8.6.2 Aggregating resources utilization per machine model and manufacturer

Perform the following steps to create the report:

1. Log in to the IBM Tivoli Common Reporting for Asset and Performance Management console and open the Query Studio for Tivoli Reporting cross-product package. See Figure 8-38 on page 242.
2. Start with TADDM Computer System data.
3. Insert the `TADDM Model / Computer System / Manufacturer` query item.
4. Insert the `TADDM Model / Computer System / Model query` item.
5. Select **Manufacturer** column heading and click the **Group** icon from the Query Studio toolbar.

   The machines are now grouped by manufacturer name.
6. Add resource utilization metrics from IBM Tivoli Monitoring.

   For example, insert the `ITM OS Agents Model / Windows-Unix / Hourly / CPU hourly / AVG_%_CPU_Used` query item
7. The processor utilization values are now rolled up for each model and manufacturer. You may split the aggregation by adding the host name to the query.

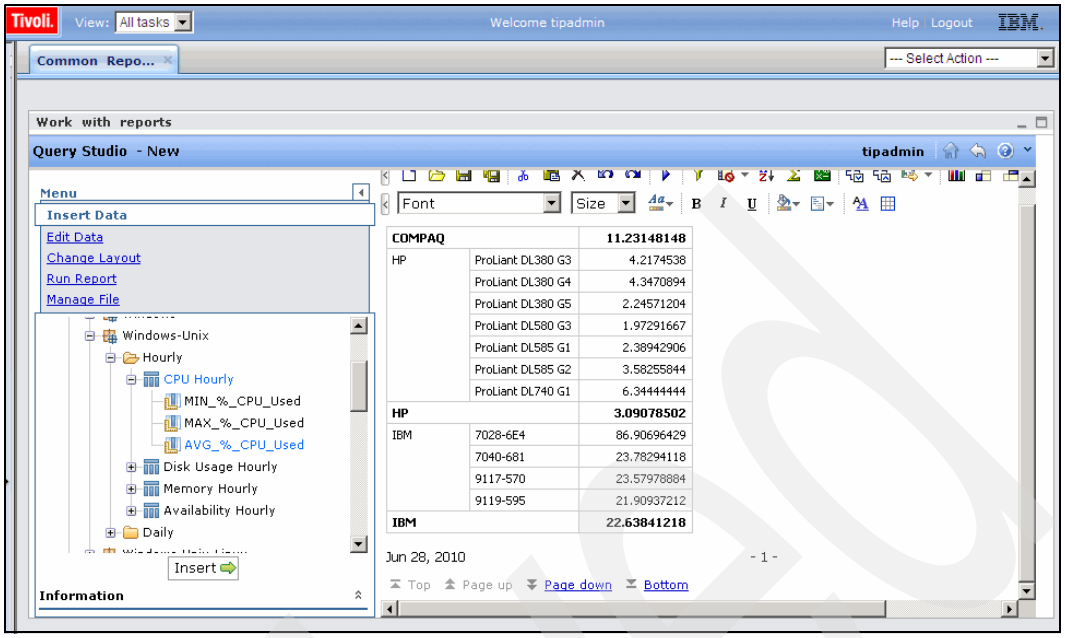


Figure 8-38 CPU average utilization for particular manufacturers and machine models

8.6.3 Aggregating resources utilization per operating system

Perform the following steps to create this report:

1. Log in to the IBM Tivoli Common Reporting for Asset and Performance Management console and open the Query Studio for Tivoli Reporting cross-product package.
2. Start with TADDM Operating system data.
3. Insert the `TADDM Model / Operating System / Name` query item.
4. Right-click the **Name** column heading and select **Filter**.
5. From the list of all operating systems. which were discovered in your environment, select operating systems that you want to check.
6. Insert the `TADDM Model / Operating System / Version` query item.
7. Select **Name** column heading and click the **Group** icon from the Query Studio toolbar.

   Your operating systems are now grouped by name.
8. Add resource utilization metrics from IBM Tivoli Monitoring.

   For example, insert the `ITM OS Agents Model / Windows-Unix / Hourly / CPU Hourly` query subject. It is possible to add a whole query subject to a report. All query items from the subject are added in that case.
9. The processor utilization data are now rolled up for each operating system version and name. You may split the aggregation by adding the host name to the query. See Figure 8-39 on page 243.

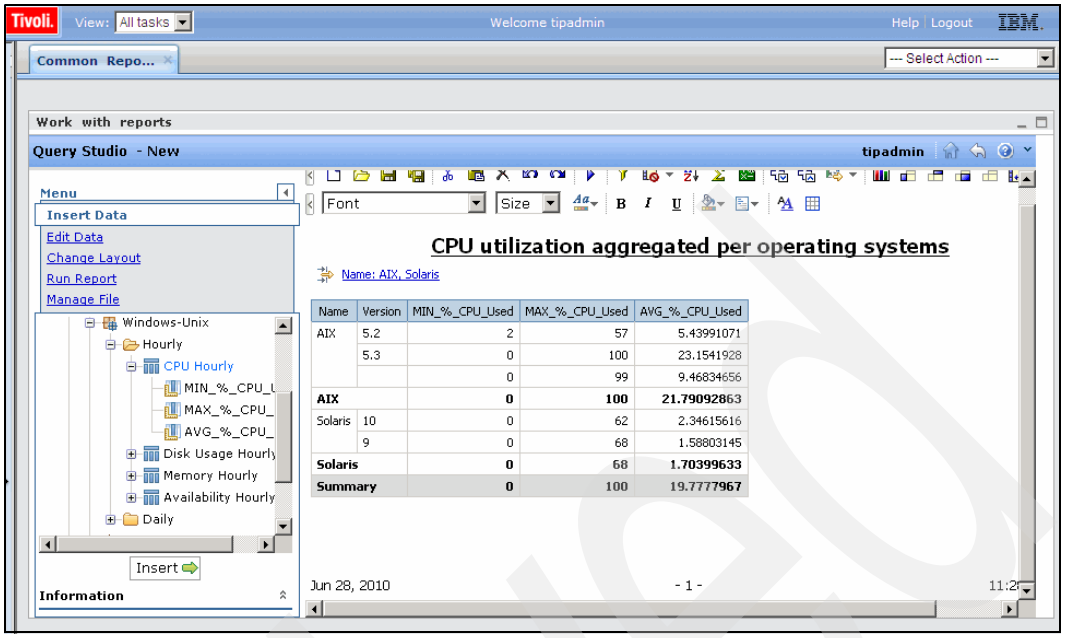


Figure 8-39 CPU utilization for particular operating systems

8.7 Online resources

See the following resources:

- IBM Tivoli Common Reporting for Asset and Performance Management 1.3 information center

  http://publib.boulder.ibm.com/infocenter/tivihelp/v3r1/topic/com.ibm.tivoli.tcr_cog.doc/tcr_welcome.html

- IBM Tivoli Monitoring 6.2.2 Fix Pack 2 information center

  http://publib.boulder.ibm.com/infocenter/tivihelp/v15r1/topic/com.ibm.itm.doc_6.2.2fp2/welcome.htm

- IBM Tivoli Application Dependency Discovery Manager 7.2 information center

  http://publib.boulder.ibm.com/infocenter/tivihelp/v10r1/topic/com.ibm.taddm.doc_7.2/welcome_page/welcome.html

Archive

8.8 Summary

Figure 8-40 shows the quick summary of this integration,

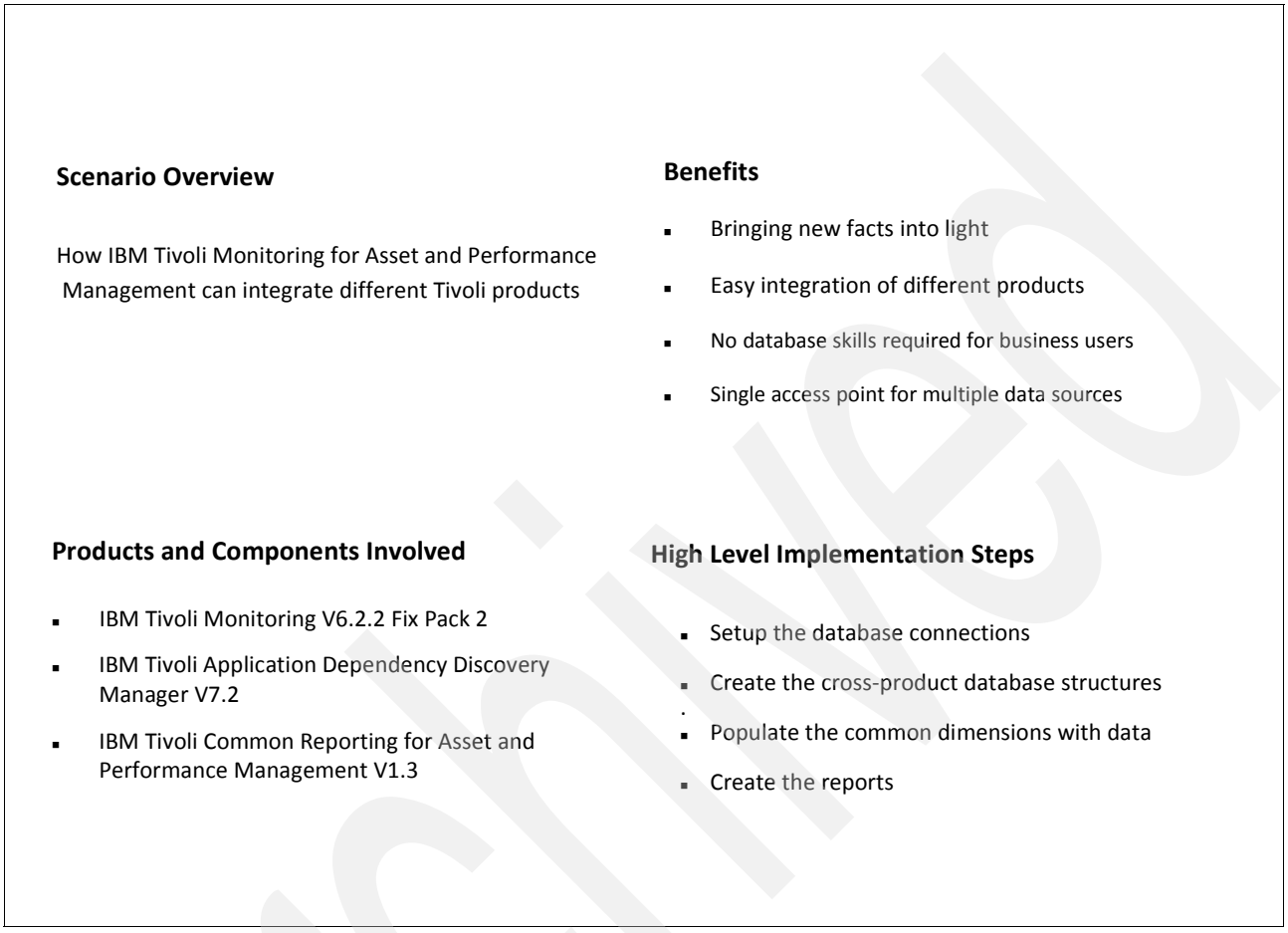


Figure 8-40 Quick summary

Archived

9

Tivoli reporting integration: IBM Tivoli Monitoring and IBM Tivoli Service Request Manager

Chapter 8, “Tivoli reporting integration: IBM Tivoli Monitoring and TADDM” on page 199, described how to create Tivoli Common Reporting (TCR) reports based on the Cognos technology from the data collected by IBM Tivoli Monitoring and TADDM.

This chapter expands on that information and describes how you can create reports by integrating data from IBM Tivoli Monitoring and IBM Tivoli Service Request Manager. The purpose of the chapter is to give you the information of the tools available so you can get started with using Tivoli Common Reporting V1.3.

This chapter contains the following topics:

- 9.1, “Scenario overview” on page 246
- 9.2, “Products involved” on page 248
- 9.3, “Benefits” on page 248
- 9.4, “Creating a framework model” on page 248
- 9.5, “Creating reports” on page 278
- 9.6, “Summary” on page 290

Important: The framework model that is used in these scenarios can be downloaded from the ITSO FTP site. For downloading instructions, see Appendix A, “Additional material” on page 803. This framework model is included on an “as-is” basis and is not officially supported. It is provided with this book to give you a preview of the planned product functionality for Tivoli Common Reporting integration.

© Copyright IBM Corp. 2011. All rights reserved.

9.1 Scenario overview

The three steps to create a new report with data from two or more Tivoli products in Tivoli Common Reporting 1.3 are as follows:

1. Locate and prepare data sources.
2. Build the framework model.
3. Build and publish the content.

This section describes those steps and lists other ways to create reports.

9.1.1 Locate and prepare data sources

First, locate and prepare the data sources and models. In our example, the data sources are the IBM Tivoli Monitoring data warehouse (WAREHOUS) and the Tivoli Service Request Manager database (MAXDB71).

In IBM Tivoli Monitoring, be sure that you have warehoused the data you want to report on. Our example creates a report about hourly metrics for CPU, disk, and memory so that you have to check that warehoused data is also summarized into hourly tables.

You also have to run several database scripts, which are delivered in IBM Tivoli Monitoring 6.2.2 Fix Pack 2. The database scripts and running the included procedures create and populate additional tables (TIME_DIMENSION and MANAGEDSYSTEM).

In Tivoli Service Request Manager, we use data from one view (INCIDENT) and one table (CI). CI contains all authorized configuration items, for example, you can report incidents or problems against these items. It is not the same as an actual CI (table ACTCI), which is where for instance you will upload what TADDM has discovered. To get an actual CI to become an authenticated CI, you must promote it to a CI in CCMDB. When you promote the CI, be sure that the CI contains information to enable you to connect it to the data in the IBM Tivoli Monitoring warehouse. Candidates for the join can be the host name or the fully qualified host name.

Finally, you must of course have certain incidents reported on the configuration items. These can be automatically generated, such as IBM Tivoli Monitoring situations creating events in Tivoli Netcool/OMNIbus, which in turn create a trouble ticket in Tivoli Service Request Manager.

For all data sources, you need to know the location and name of the database and have a user ID and password to be able to read the data. This user ID needs only read access.

9.1.2 Build the framework model

Second, when all these preparations are done, you can start to build your framework model. The framework model is the base for all queries and reports in Tivoli Common Reporting. This is where you define what data sources and tables the user will be able to use in the report. You define relationships between tables in the Tivoli Service Request Manager and the IBM Tivoli Monitoring database. You can also create simplified views of the data to help users find the metrics the user requires. Additionally, you can create calculations for metrics you know that the user is going to use frequently. For instance, if the IBM Tivoli Monitoring agent reports on kilobits per second (kb/s) and you want to use megabits per second (Mbps) in the report, you can create this calculation in the framework model.

Data models are created in Framework Manager, delivered with Tivoli Common Reporting, Download this together with Tivoli Common Reporting 1.3. Install it on your desktop and follow the instructions for setting up the connection to Tivoli Common Reporting 1.3 in the Tivoli Common Reporting documentation. See the following web address:

http://publib.boulder.ibm.com/infocenter/tivihelp/v3r1/index.jsp?topic=/com.ibm.tivoli.tcr.doc/ttcr_install.html

You must have at least a run-time database client installed on your desktop for the databases (DB2 and so on) that you will use for the framework modeling. You must also configure the database client so it can access the IBM Tivoli Monitoring warehouse and the Tivoli Service Request Manager database. You also must do this step on the Tivoli Common Reporting server. This steps helps you more easily create the same alias for each database on both your desktop and the Tivoli Common Reporting server.

In Framework Manager, you may of course start at the beginning by selecting the tables and views you need from the databases. Alternatively, you can use one or more existing framework models to merge and modify to create the package you need. For instance, we are going to use the TADDM framework model used in this book as a staring point because it has query subjects we want to re-use.

9.1.3 Build and publish the content

The third and final step that we describe in this book is to build and publish the content. The last step, which you do in Framework Manager, is to publish the package to Tivoli Common Reporting. You then continue the work in the Tivoli Common Reporting web portal and the Report Studio.

One additional step you can do in Tivoli Common Reporting is to deliver the information, which can mean scheduling and sending the report as an attachment to an email.

This book does not include these steps; see the Tivoli Common Reporting manuals for this step.

9.1.4 Other ways of creating reports

Before starting on our scenario, the various ways you can use Tivoli Common Reporting 1.3 for reporting are summarized as follows:

- Modify an existing report

  If you just want to modify an existing report, you may use Report Studio included in Tivoli Common Reporting 1.3.

- Create a new report

  If you want to create a new report using the data from a report package such as “IBM Tivoli Monitoring OS Agents Reports” you can use Report Studio included in Tivoli Common Reporting 1.3.

- Use ad-hoc reports

  If you want to run an ad-hoc query against the data in your data warehouse, you can use Query Studio included in Tivoli Common Reporting 1.3

- Use, Tivoli cross-product reporting

  If you want to create a new framework model combining metadata from two or more Tivoli products such as IBM Tivoli Monitoring and Tivoli Service Request Manager, then you need to use the Tivoli Common Reporting Framework Manager.

9.2 Products involved

The following products and product components are involved in this scenario:

- IBM Tivoli Common Reporting V1.3
- IBM Tivoli Monitoring V6.2.2
- IBM Tivoli Service Request Manager V7.2

9.3 Benefits

Benefits of this scenario are as follows:

- Simplified visualization of information
- Simplified delivery of information
- Simplified sharing of information
- A consistent reporting solution that is shared across the Tivoli portfolio
- Improved capability for proactive problem analysis

9.4 Creating a framework model

Perform the following steps:

1. Start the Tivoli Common Reporting Framework Manager.
2. Create a new project. A project is a set of files that define the metadata in one or more models. Models represent data source information and contain information from business requirements and performance enhancements. The framework modeler creates packages and publishes them to Tivoli Common Reporting.
3. Log in to Tivoli Common Reporting 1.3. This is necessary to use the data sources defined in Tivoli Common Reporting and later publish the package. See Figure 9-1.

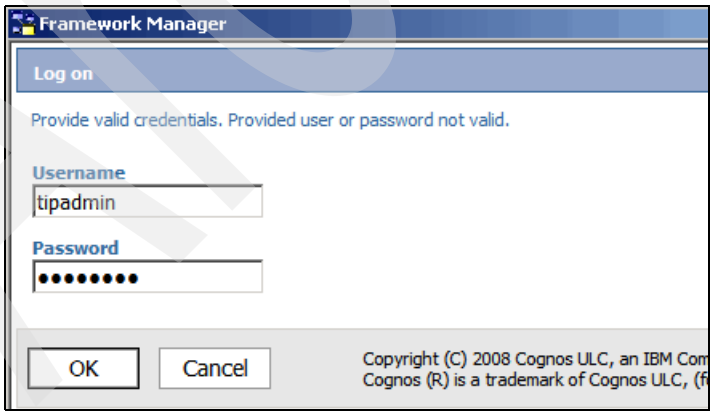


Figure 9-1 Framework Manager: TCR authentication

Archived

4. Select your language and select **IBM Cognos 8 Model** as the data source. See Figure 9-2.

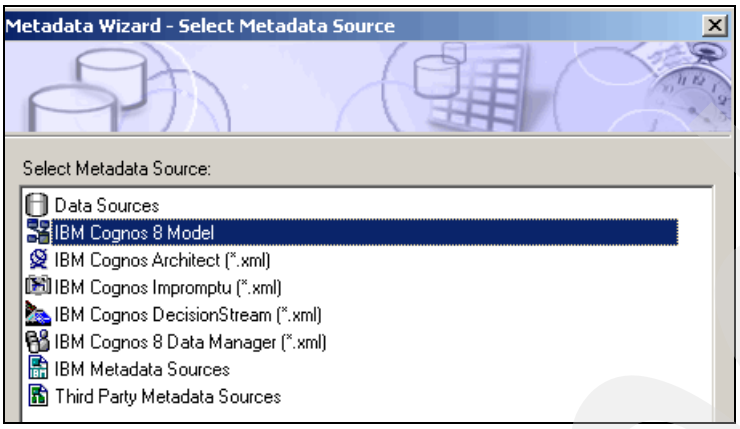


Figure 9-2 Framework Manager: Select data source

5. Navigate to where you extracted the TADDM framework model and select the `*.cpf` file (in this example `Cross_Products_Model_v2.cpf`). See Figure 9-3.

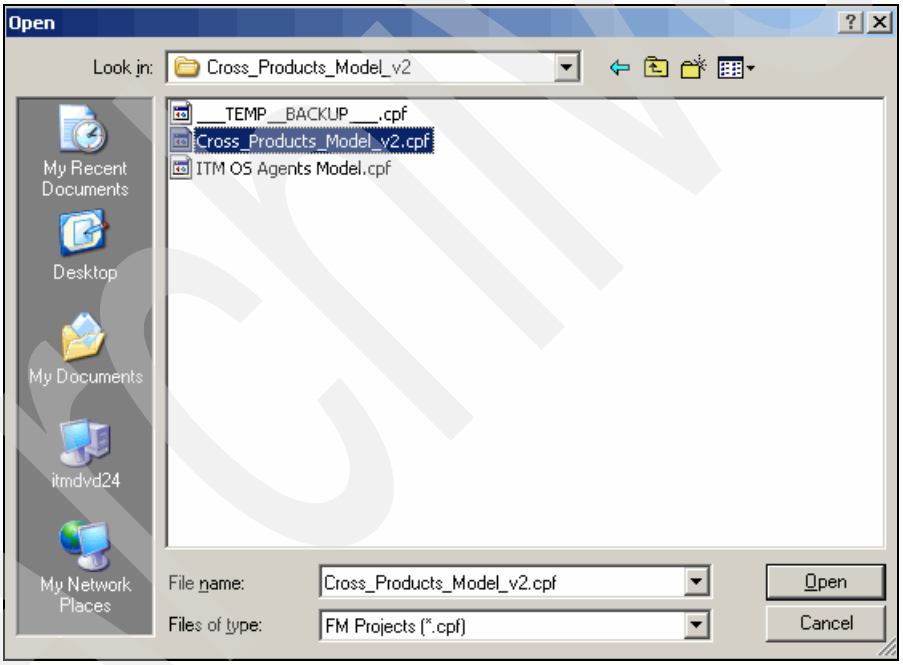


Figure 9-3 Framework Manager: Select TADDM project file

Archive

6. Select the objects you want to include in your framework model. Although you may select everything, we deselect the objects that are used for TADDM data because we do not use them. See Figure 9-4.

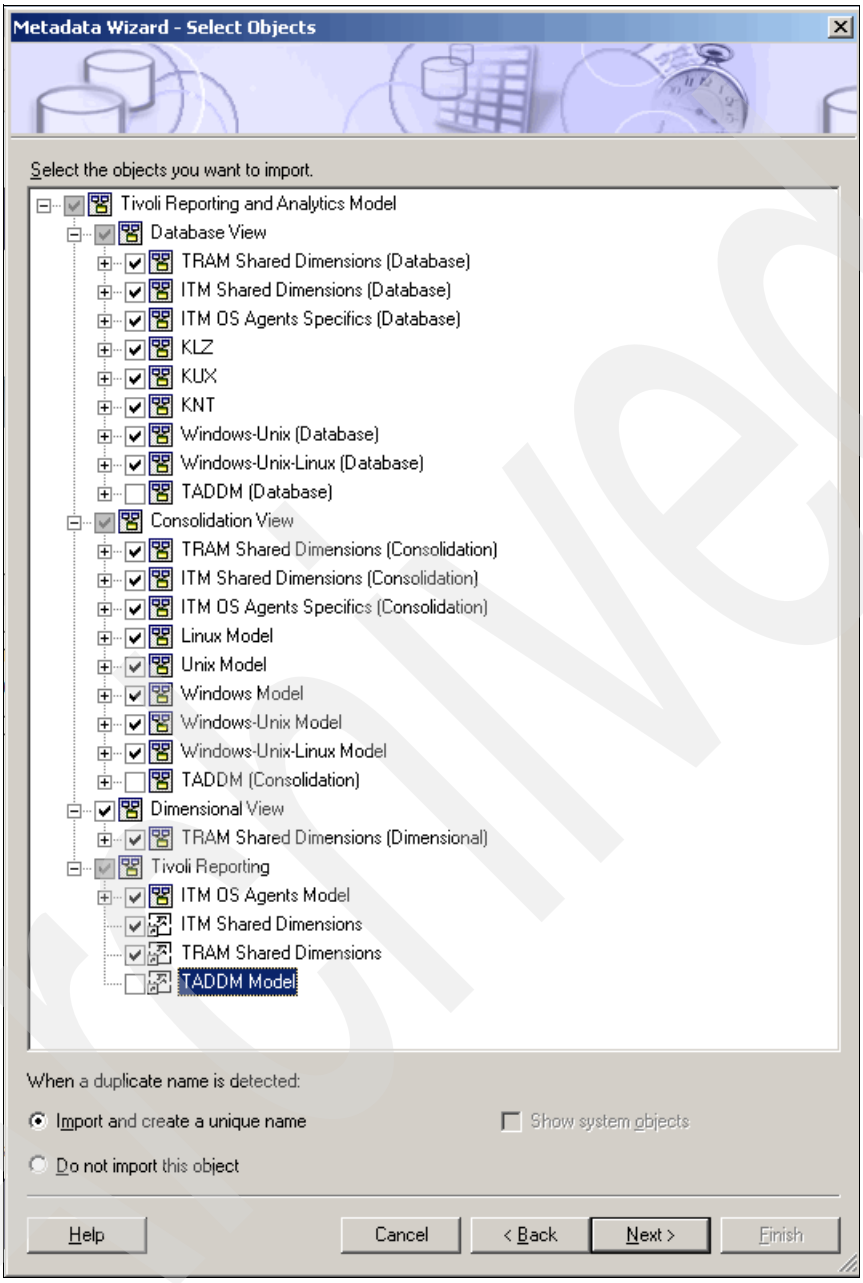


Figure 9-4 Framework Manager: Select objects

Archived

7. Click **Next**. The import starts. After the import finishes, the main window in Framework Manager opens. See Figure 9-5.

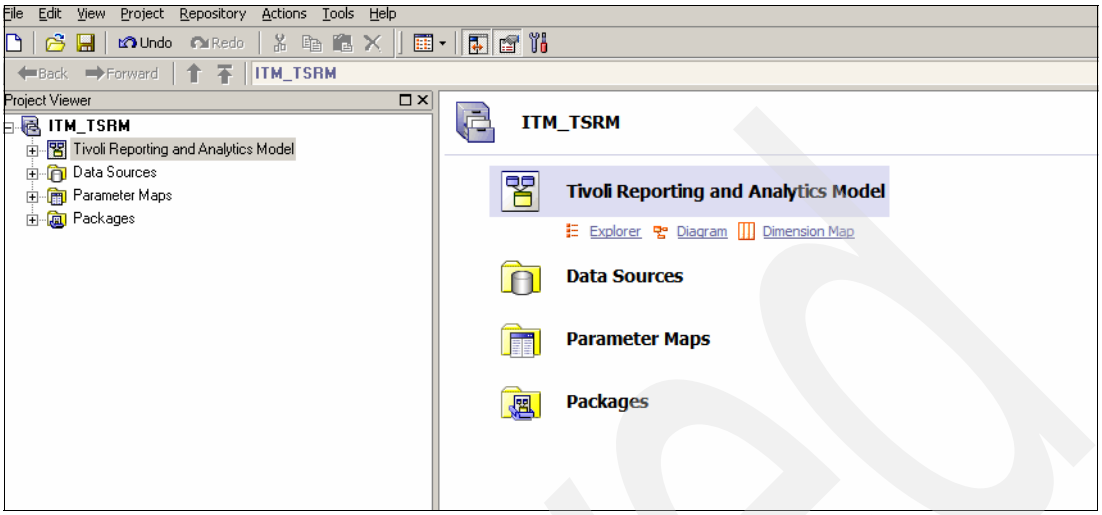


Figure 9-5 Framework Manager: Main windows

8. Expand **Data Sources** and determine whether you need to reconfigure. In our example, we change the schema name to ITMUSER because this schema is used in the IBM Tivoli Monitoring warehouse. See Figure 9-6.

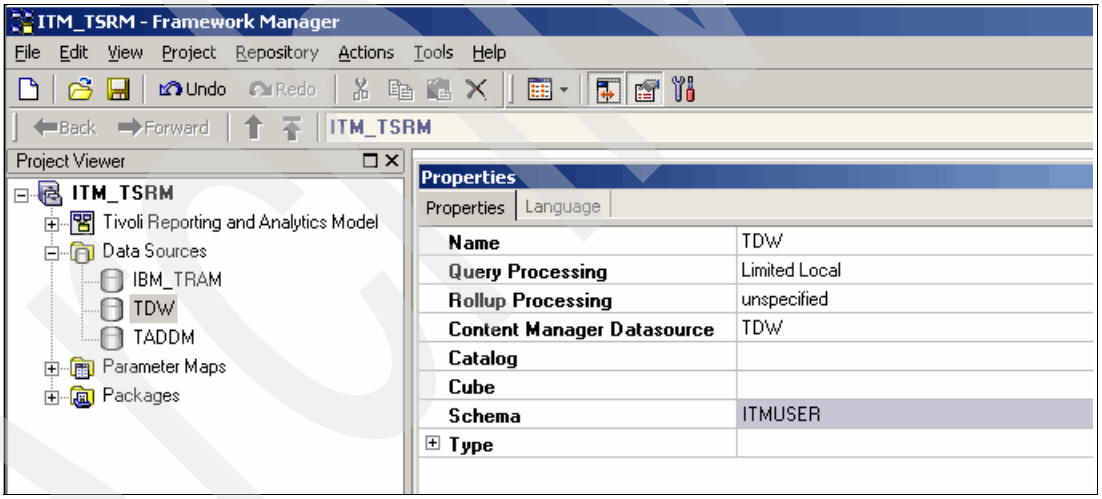


Figure 9-6 Framework Manager: TDW data source configuration

Do not change the name of the data source because all queries are using it. Verify that the Content Manager Datasource is set to what you have defined in Tivoli Common Reporting. In our example, we use TDW. Note that this is a Tivoli Common Reporting Connection alias to the database named WAREHOUSE and does *not* have to be the same name as the database. The Tivoli Common Reporting Connection also contains database user ID and password. See Figure 9-7 on page 252.

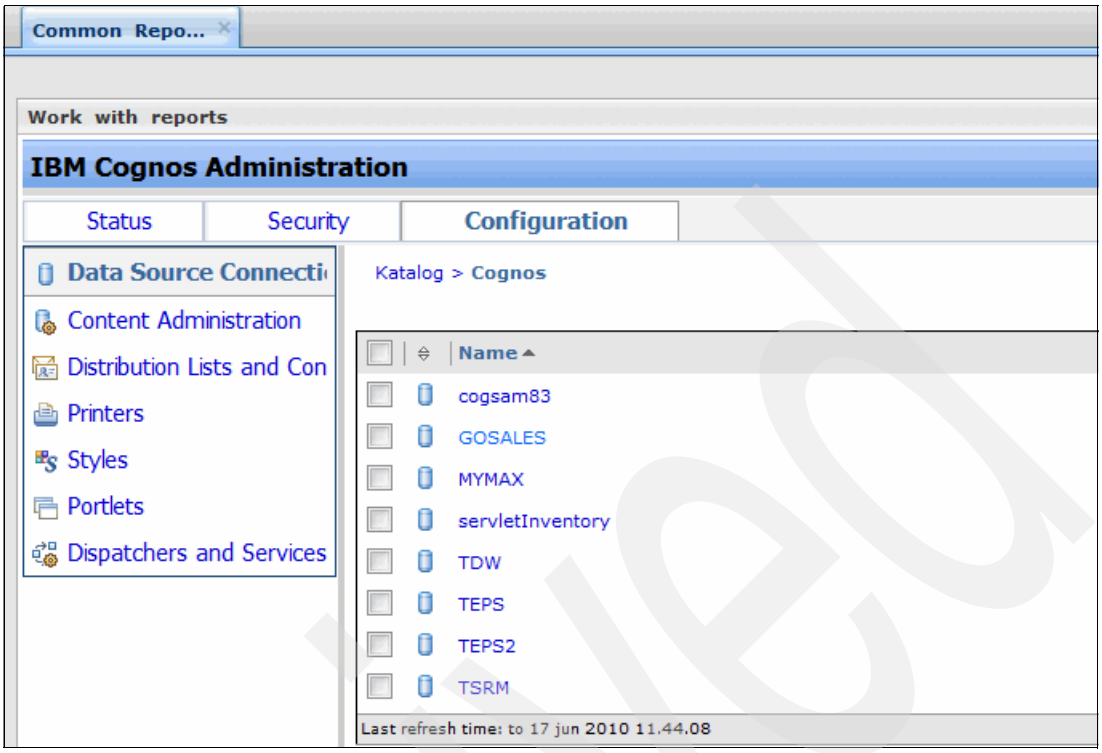


Figure 9-7 TCR: Data source connections

The IBM_TRAM (Tivoli Reporting and Analytics Model) data source is used as the source for the additional tables introduced in IBM Tivoli Monitoring 6.2.2 Fix Pack 2. In our setup, these are contained in the IBM Tivoli Monitoring warehouse database in schema IBM_TRAM. Therefore, the Content Manager Datasource must be set to TDW. See Figure 9-8.

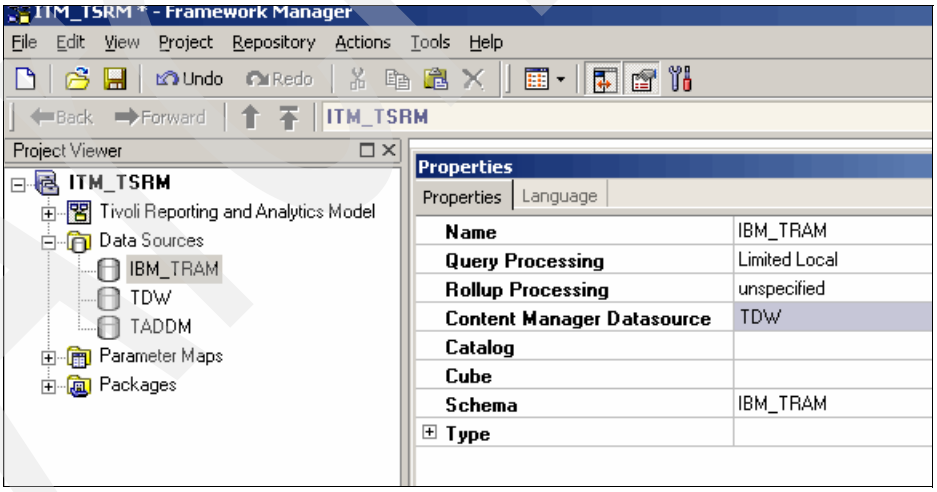


Figure 9-8 Framework Manager: IBM_TRAM data source configuration

9. Finally, because we will not be using any data from TADDM, we delete the data source.

After you confirm or change the data sources, a good practice is to test them by right-clicking and selecting **Test**. This testing ensures that you have the correct setup and permissions for the databases you want to read from.

You may expand the hierarchy and explore what is included in the framework model. Test all the query subjects to make sure that you have the tables that the framework model uses in the IBM Tivoli Monitoring warehouse database. You test a query subject by double-clicking it, selecting the **Test** tab, and then clicking **Test Sample**.

The query subjects to test are as follows:

1. Database View → TRAM Shared Dimension (Database) → Time Dimension
2. Database View → ITM OS Agents Specifics (Database) → Linux Managed Systems
3. Database View → ITM OS Agents Specifics (Database) → Windows Managed Systems
4. Database View → KLZ → CPU Hourly
5. Database View → KLZ → Disk Usage Hourly
6. Database View → KLZ → Memory Hourly
7. Database View → KNT → CPU Hourly
8. Database View → KNT → Disk Usage Hourly
9. Database View → KNT → Memory Hourly
10. Consolidation View → TRAM Shared Dimensions (Consolidation) → Time
11. Consolidation View → ITM Shared Dimensions (Consolidation) → Managed Systems
12. Consolidation View → ITM OS Agents Specifics (Consolidation) → Dimensions → Linux Managed Systems
13. Consolidation View → ITM OS Agents Specifics (Consolidation) → Dimensions → Windows Managed Systems

9.4.1 Creating a Database View

In our setup, we have only Linux and Windows servers, so we remove all references to the UNIX warehouse tables and create the query subjects we need only from Windows and Linux tables.

Although the model already contains a query subject for Linux and Windows hourly metrics, we want to combine them into one query subject. If we do this, we can use the same query in the report and we do not have to know that in the underlying data source, the metrics are actually stored in separate tables for Windows and Linux. You might have noticed that the model already has namespaces that are named, for instance, Windows-UNIX (Database). These namespaces contain unions between Windows and UNIX database tables and we want to do the same for Windows-Linux.

The steps are as follows:

1. Create a new namespace and name it Windows-Linux (Database). See Figure 9-9 on page 254.

Archived

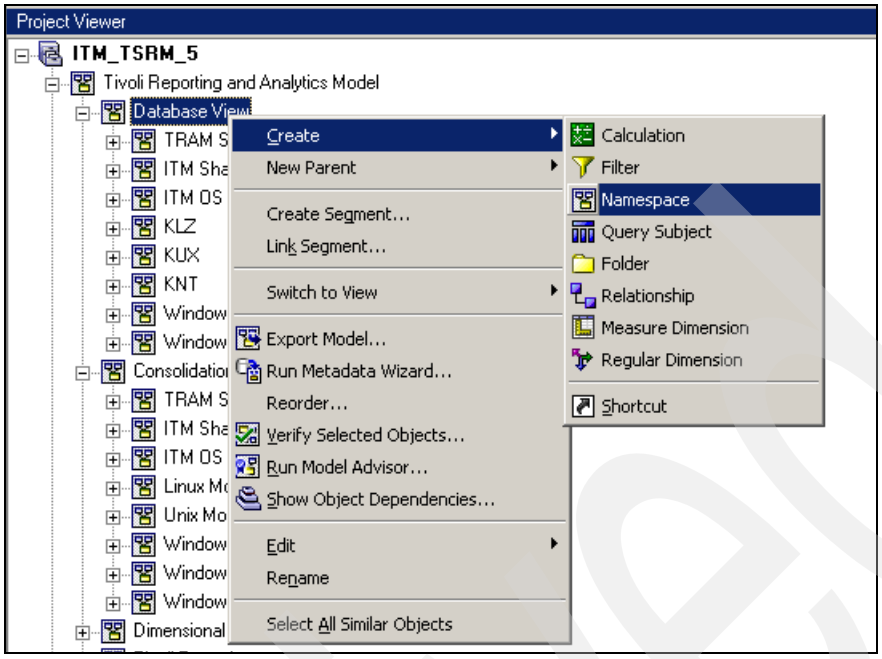


Figure 9-9 Framework Manager: Create Namespace

2. In this namespace, first create hourly query sets between the corresponding Windows and Linux database tables. For instance, to create Windows-Linux CPU Hourly, you create a query data set with a union of Database View → KNT → Hourly → CPU Hourly and Database View → KLZ → Hourly → CPU Hourly. To create a query, click **Actions** → **Define Query Set**. See Figure 9-10.

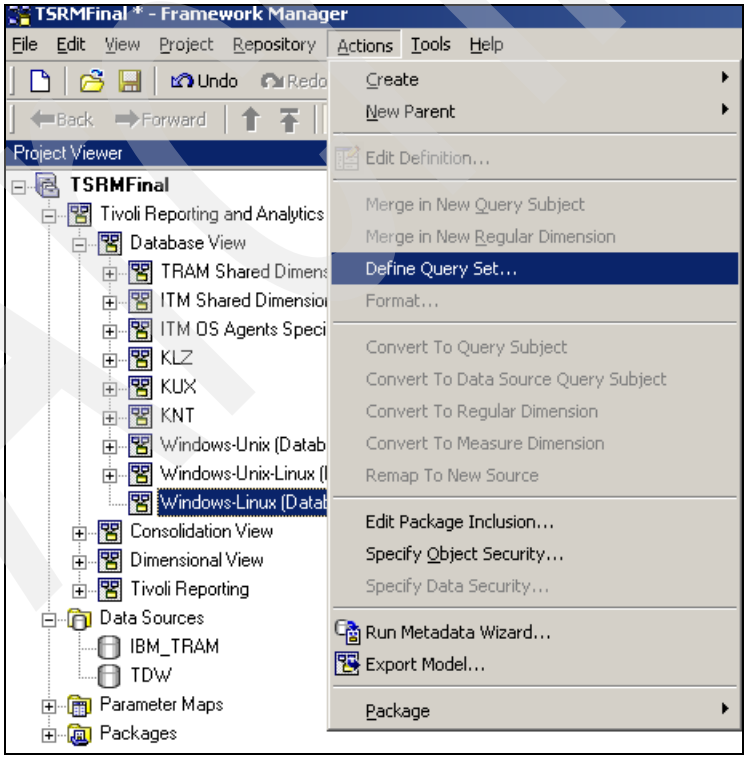


Figure 9-10 Framework Manager: Define Query Set

The New Query Set window opens (Figure 9-11).

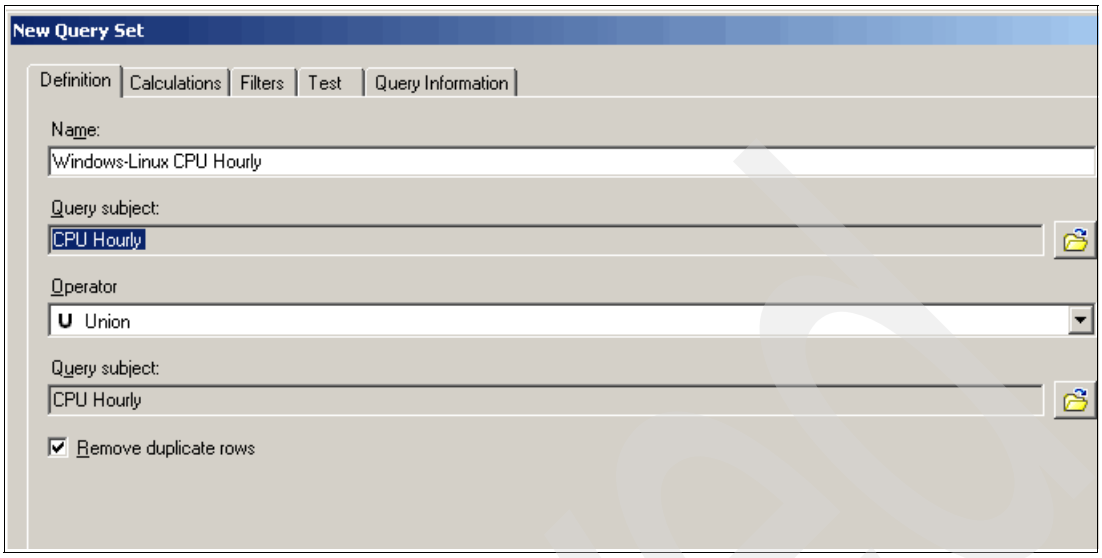


Figure 9-11 Framework Manager: Windows-Linux CPU Hourly Query Set

3. In the same manner, create the following definitions:
   - Windows-Linux Memory Hourly
   - Windows-Linux Disk Usage Hourly
4. Finally, create a folder named Hourly in the namespace and move the three query sets you just created to this folder. See Figure 9-12.

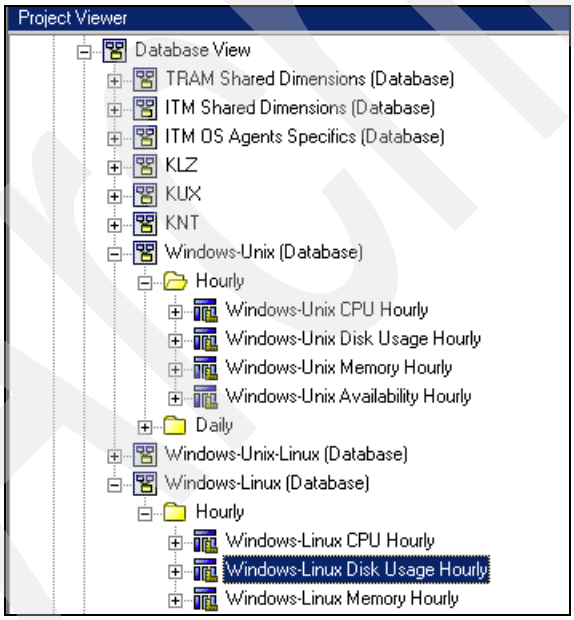


Figure 9-12 Framework Manager: Windows-Linux nameset and query sets

5. Now, create relationships between our new query sets and other query subjects. The query subjects to create relationships to are Time Dimension and ManagedSystem.

Archive

Create a relationship by selecting both **Time Dimension** and **Windows-Linux CPU Hourly,** right-clicking, and selecting **Create** → **Relationship**. See Figure 9-13.

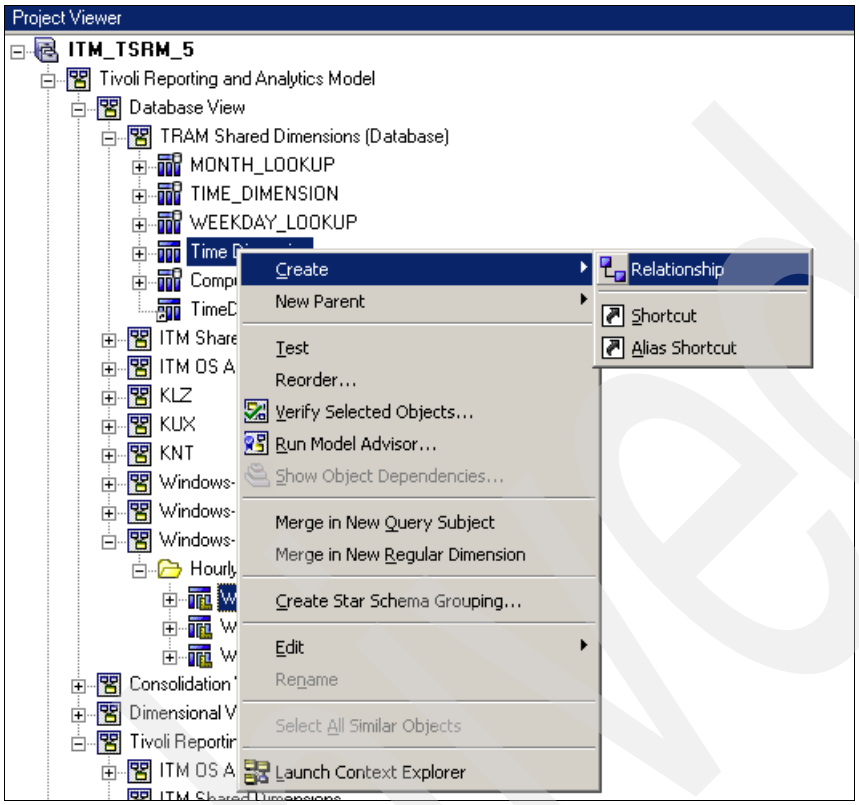


Figure 9-13 Framework Manager: Create relationship with Time Dimension

The Relationship Expression page opens (Figure 9-14 on page 257).

Archived

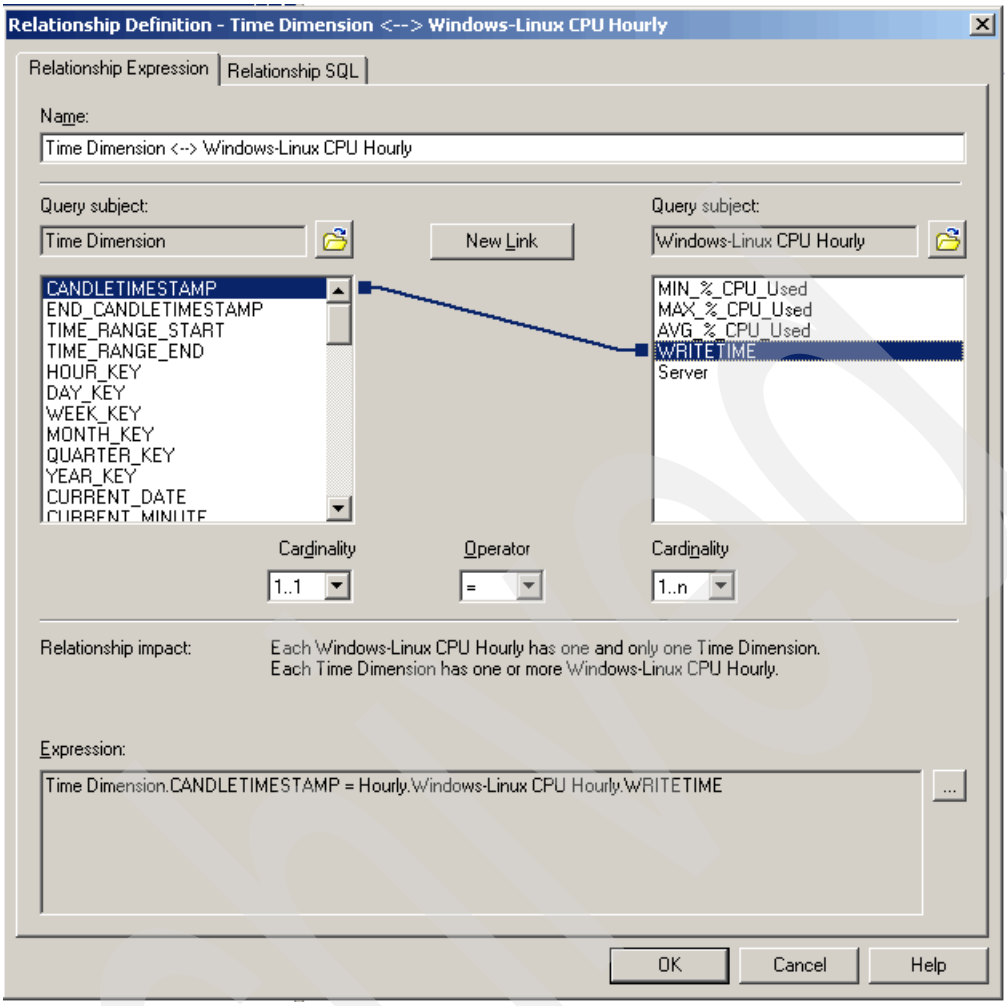


Figure 9-14 Framework Manager: Time Dimension Relationship definition

6. Verify that you have set the Relationship impact correctly. A record in the metrics query subject, Windows-Linux CPU Hourly will have only one corresponding record in the Time Dimension query subject. A record in the Time Dimension query subject will, however, have one or more corresponding records in the metrics query subject.

In addition, we create a relationship between Windows-Linux CPU Hourly and ManagedSystem. Make sure that you set the Relationship Impact correctly. See Figure 9-15 on page 258.

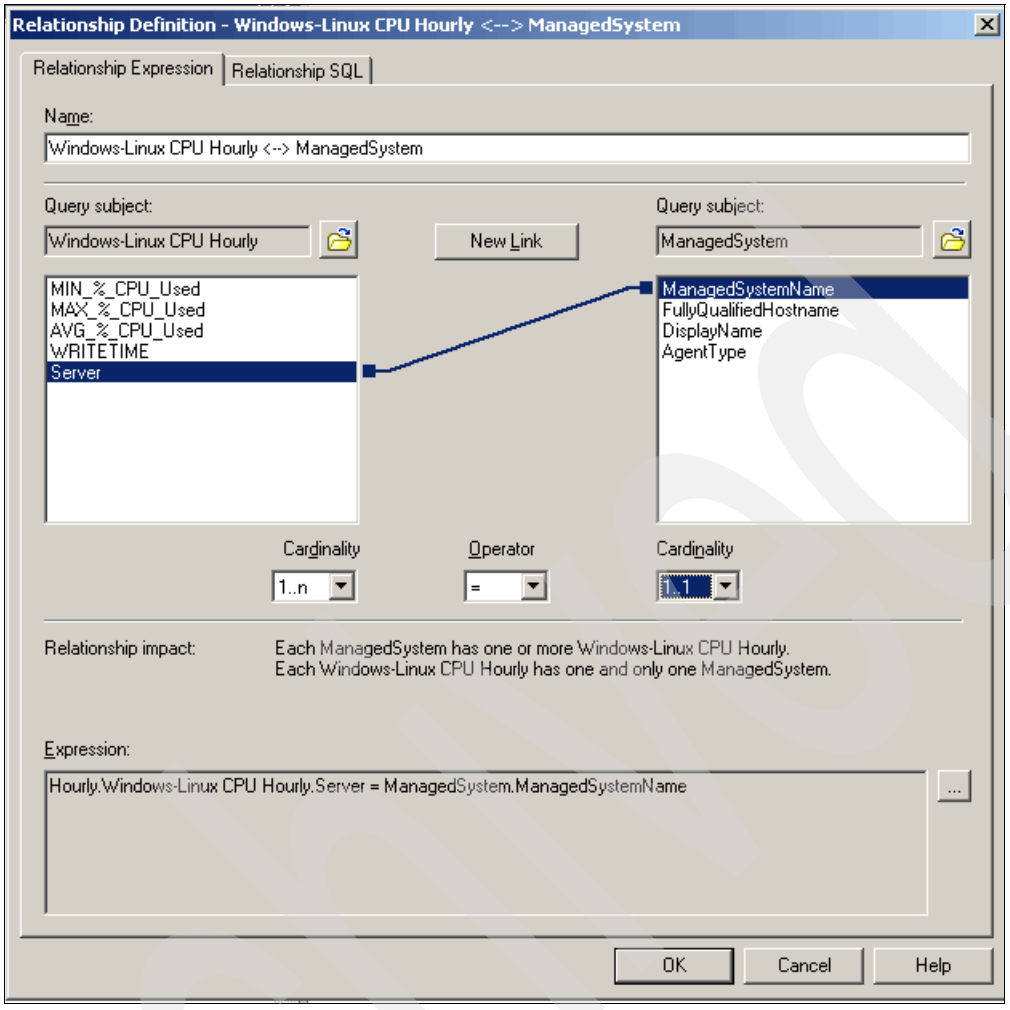


Figure 9-15 Framework Manager: ManagedSystem relationship

Repeat the instructions for the following items:

- Windows-Linux Disk Usage Hourly
- Windows-Linux Memory Hourly

9.4.2 Creating a Consolidation View

The steps are as follows:

1. Create a namespace named **Consolidation View** → **Windows-Linux Model**.
2. In this namespace, create a folder named Hourly.
3. Create a query subject named CPU Hourly in this namespace.
4. Indicate that the data you want to use to define the query subject is in the model (default setting). Then, select metrics from the query data set **Database View** → **Windows-Linux (Database)** → **Hourly** → **CPU Hourly**. See Figure 9-16 on page 259.

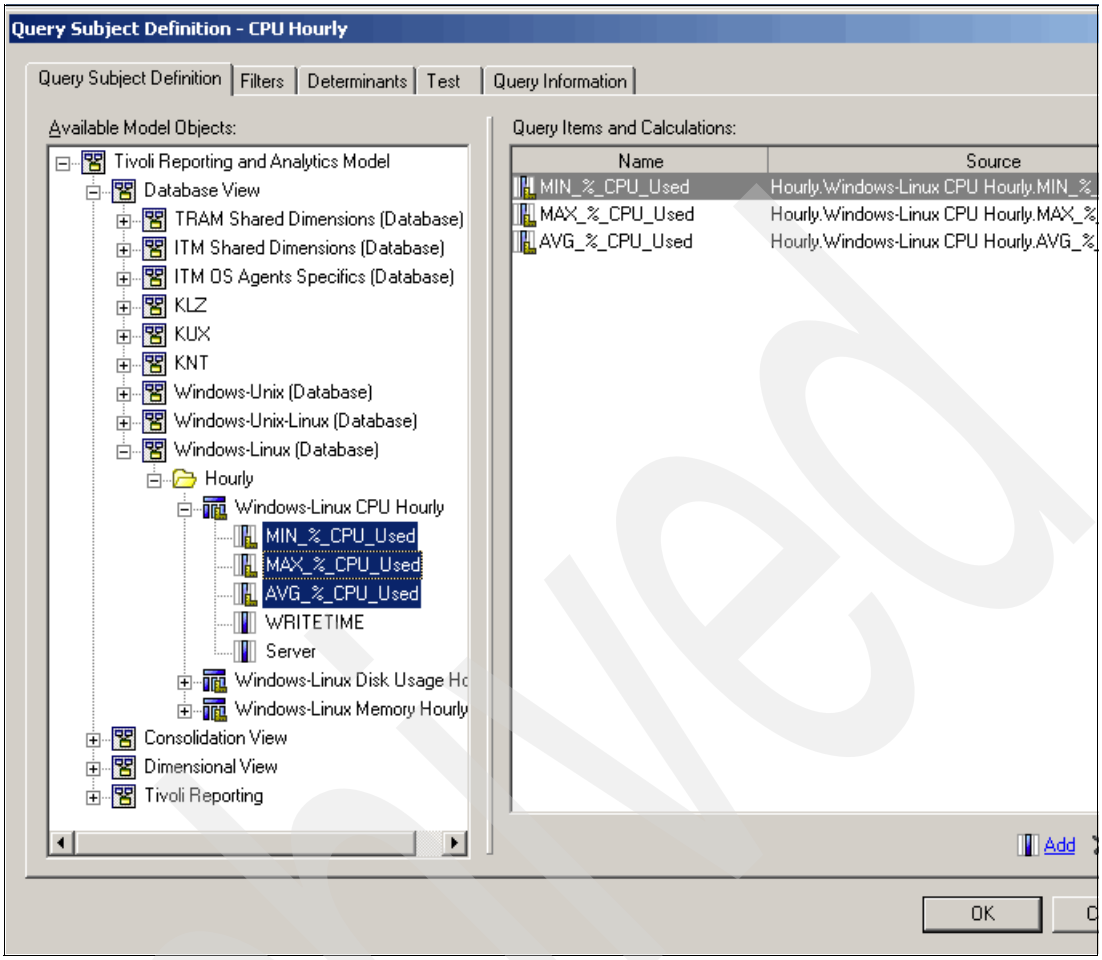


Figure 9-16 Framework Manager: Query Subject Definition - CPU Hourly

5. Repeat the steps to create the query subjects for the following items:
   - Disk Usage Hourly
   - Memory Hourly

   See Figure 9-17 on page 260.

Archived

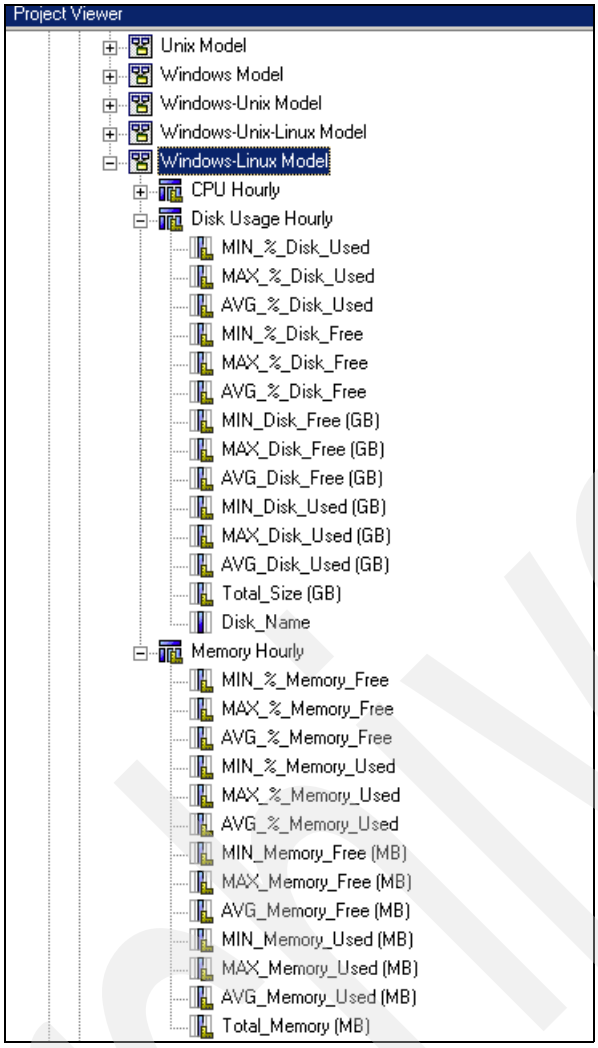


Figure 9-17 Framework Manager: Consolidation View Query Subjects

6. Also create a shortcut to the consolidation model in the **Tivoli Reporting → ITM OS Agents model** namespace. As a start, create a shortcut of the Windows-Linux model, move it to the ITM OS Agents namespace, and rename it to Windows-Linux. See Figure 9-18 on page 261 and Figure 9-19 on page 261.

Archived

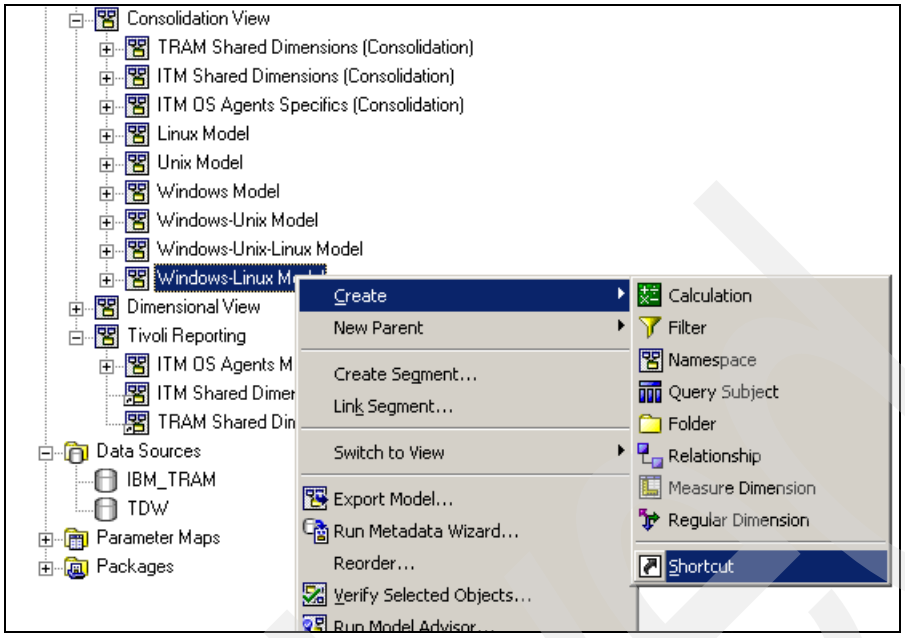


Figure 9-18 Framework Manager: Create shortcut

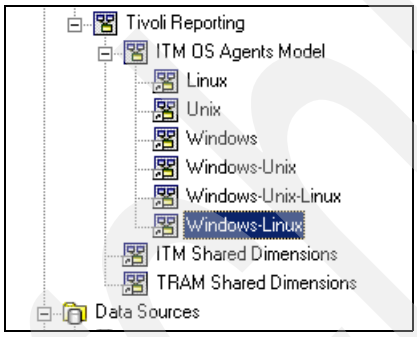


Figure 9-19 Framework Manager: Windows-Linux shortcut

7. When you have finished that, rename the model and the package to be published. See Figure 9-20.

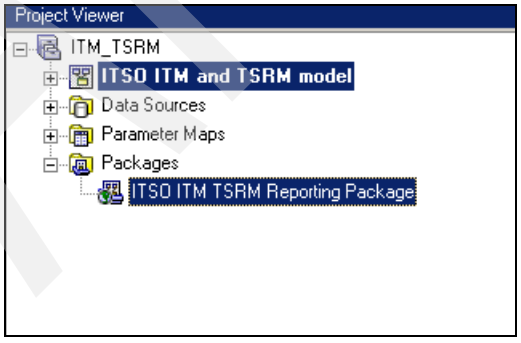


Figure 9-20 Framework Manager: Renamed model and package

Archived

9.4.3 Integrating Tivoli Service Request Manager

The next step is to import data structures from Tivoli Service Request Manager so we can use them later in our report:

1. Create a namespace in the Database View (Figure 9-21).

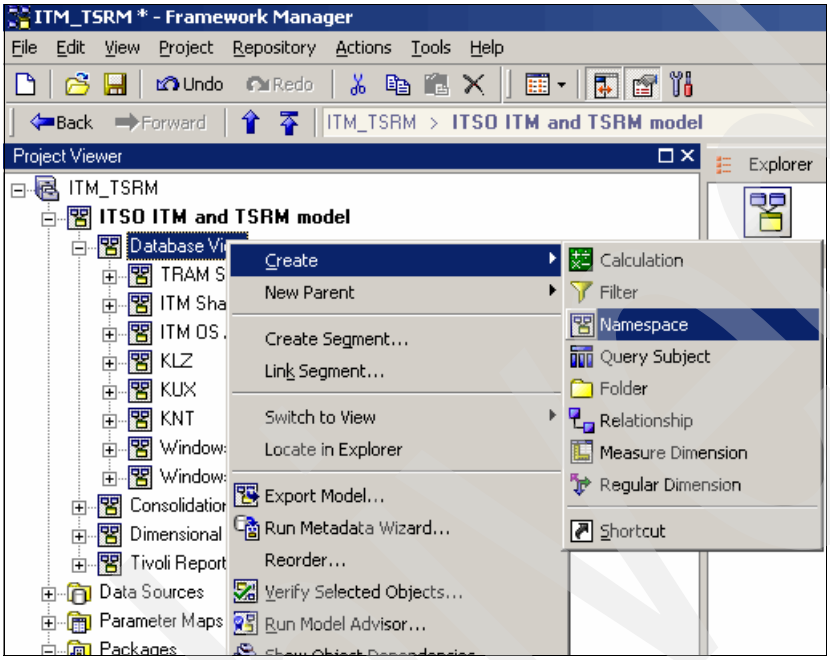


Figure 9-21 Framework Manager: Create Tivoli Service Request Manager Namespace

2. Right-click the namespace and select **Run metadata wizard** (Figure 9-22 on page 263).

Archive

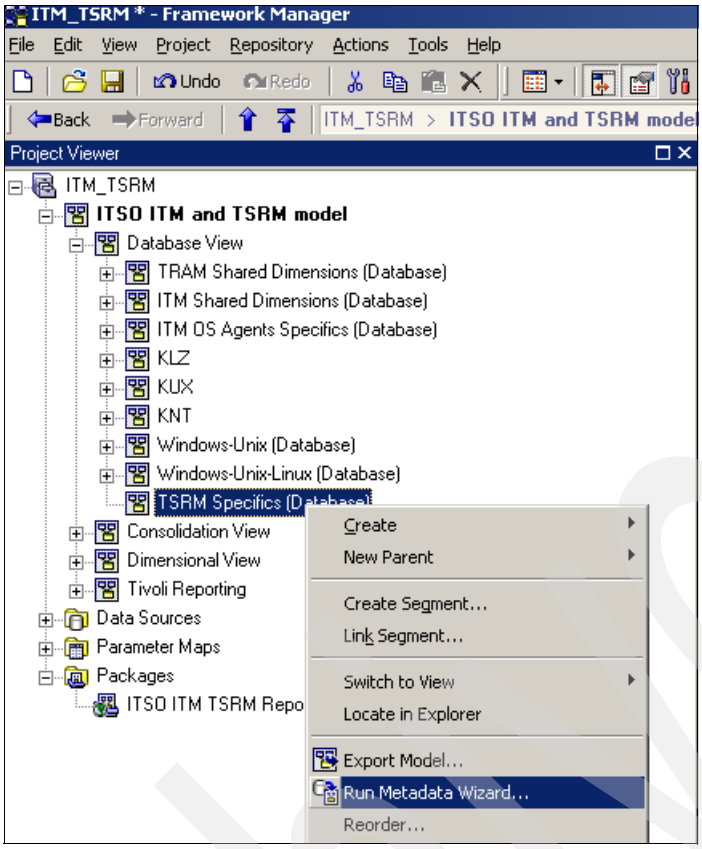


Figure 9-22 Framework Manager: Run Metadata Wizard

3. Select **Data Source** as Metadata Source. The next window, Figure 9-23, shows all the data sources that are currently configured in Tivoli Common Reporting.

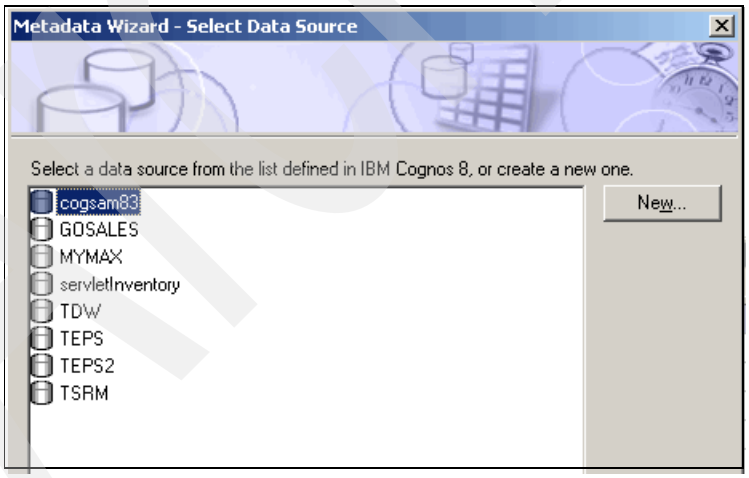


Figure 9-23 Framework Manager: Tivoli Common Reporting data sources

Archived

4. Create a new data source for Tivoli Service Request Manager:
   a. Click **New**. The New Data Source wizard opens (Figure 9-24).

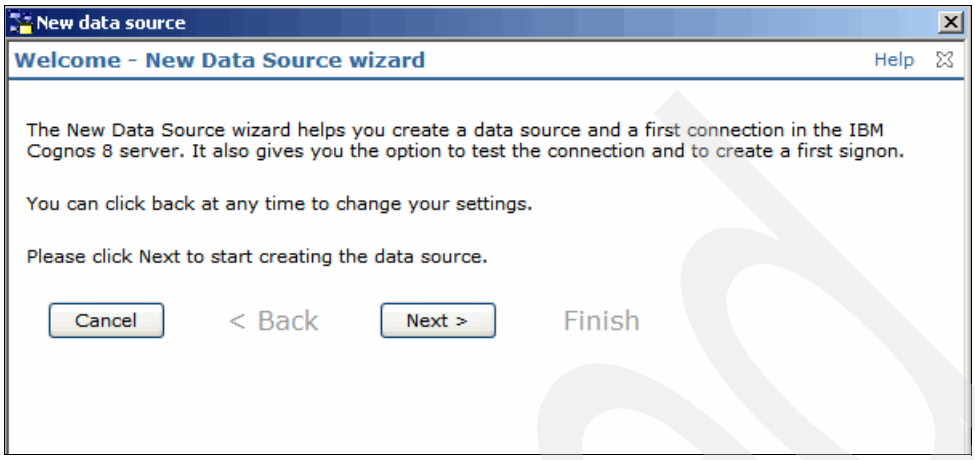


Figure 9-24 Framework Manager: New data source wizard

   b. Click **Next**. The next window opens (Figure 9-25).

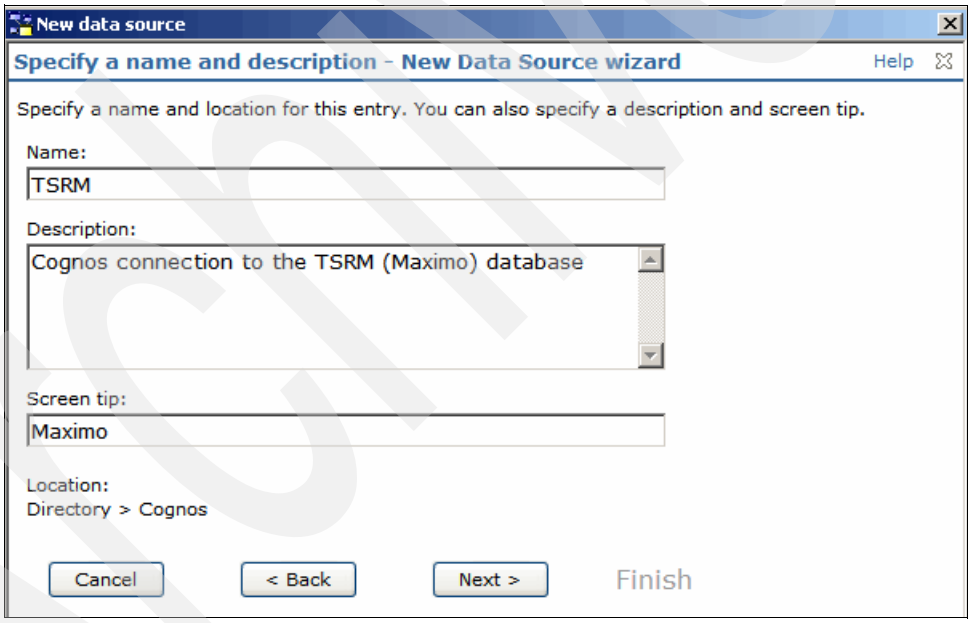


Figure 9-25 Framework Manager: Specify name and description for data source

Archived

c. Specify a name and description and click **Next**. The next window opens (Figure 9-26).

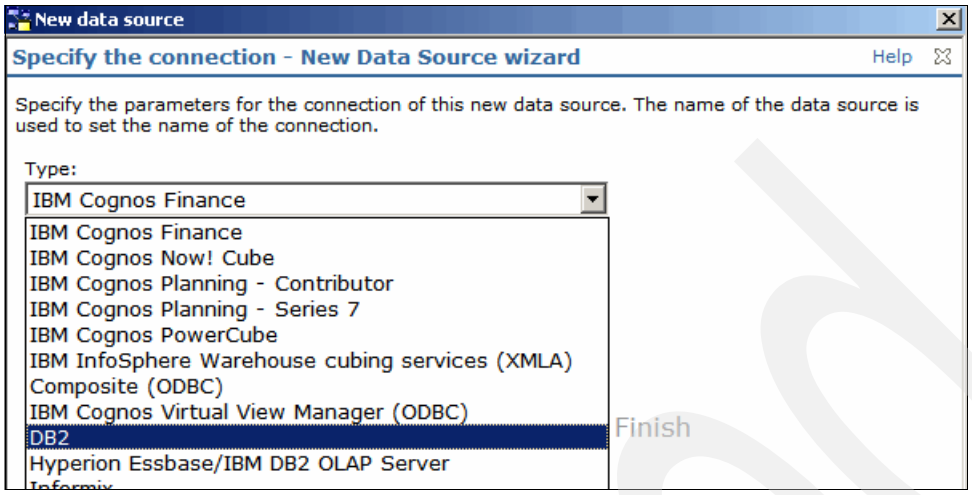


Figure 9-26 Framework Manager: Select type for data source

d. Select a type for the data source and click **Next** (Figure 9-27).



Figure 9-27 Framework Manager: DB2 as data source

5. The next window is where you configure the credentials for the database access. See Figure 9-28 on page 266. Set the DB2 Database name to the database name you have configured in the DB2 client on the Tivoli Common Reporting server. In our case, this is MAXDB71.

   Make sure that you have both boxes under Signons are selected and enter the user ID and password for connecting to the Tivoli Service Request Manager database.

Archived

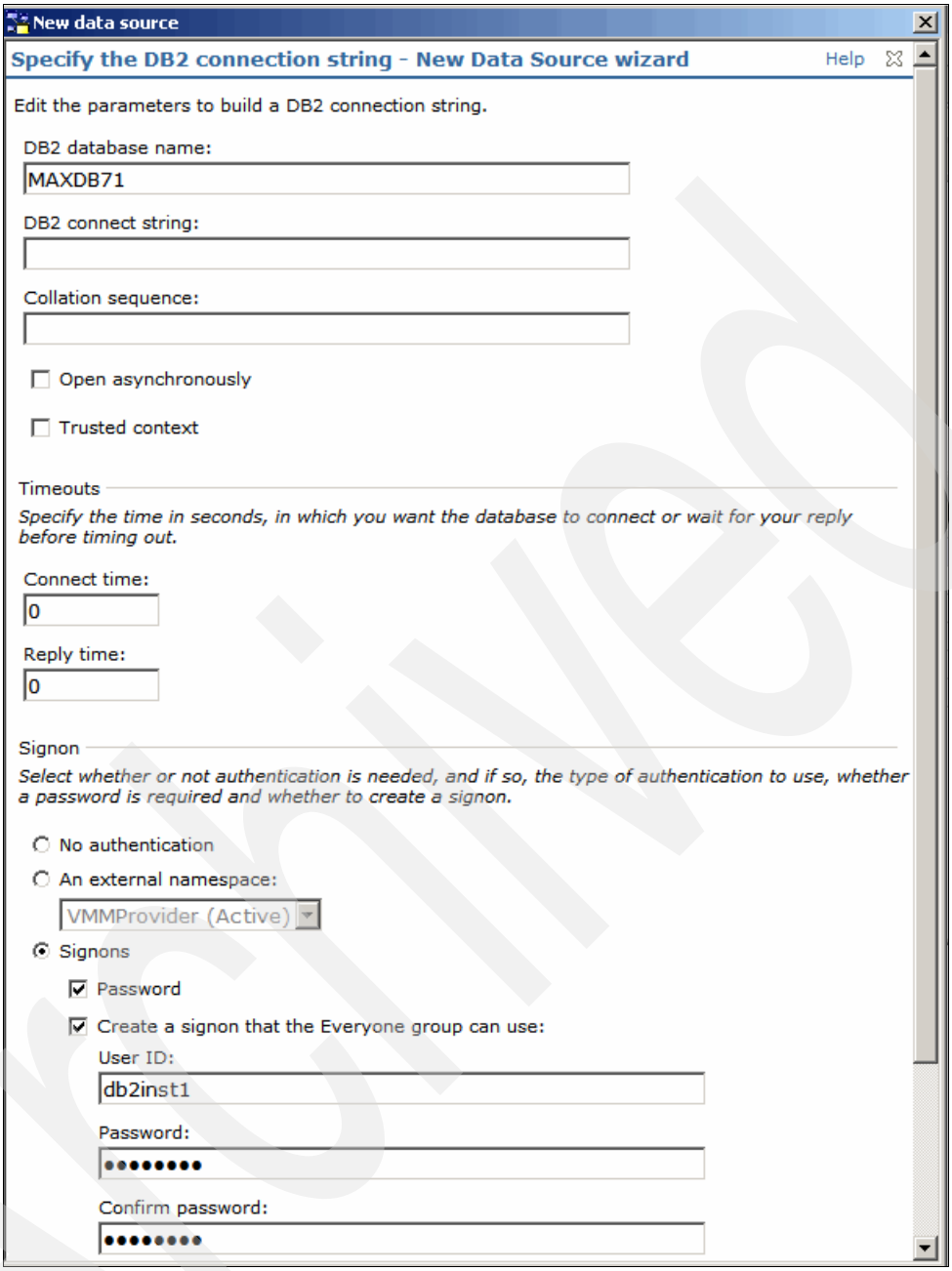


Figure 9-28 Framework Manager: Specify credentials for data source

6. In the same window, scroll until you see Test the connection. Verify that you have the right connection information and the correct user ID and password.
7. Click **Finish** when you return to the previous window.

Archived

8. Select the tables and views that you want to include in the framework model. We only select the CI table and the INCIDENT view. See Figure 9-29 and Figure 9-30 on page 268.

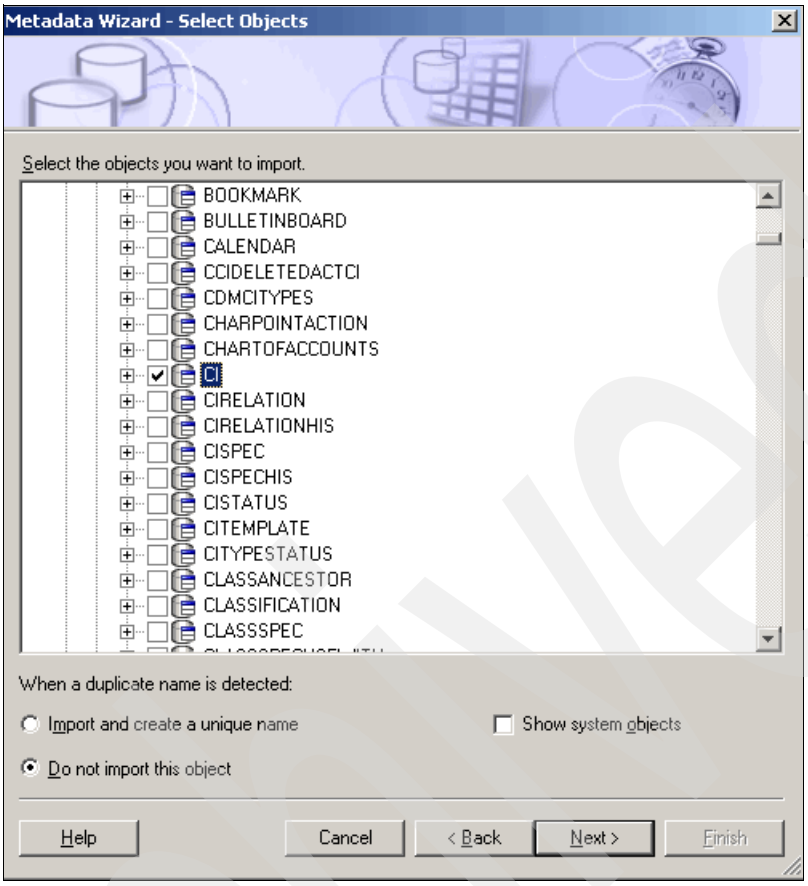


Figure 9-29 Framework Manager Metadata wizard: CI

Archived

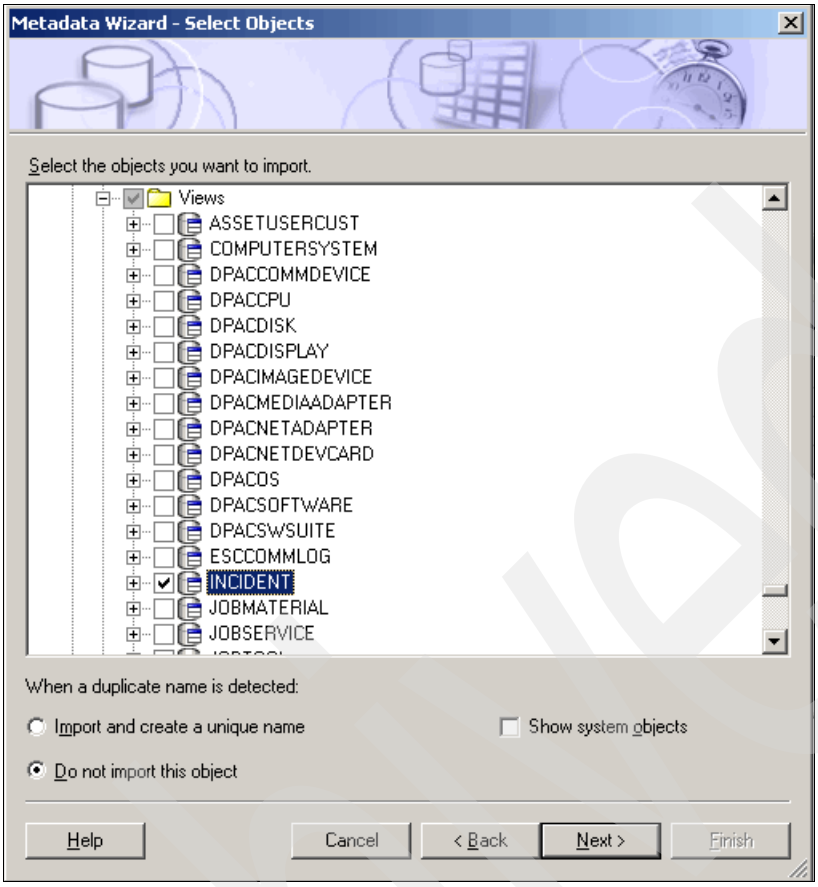


Figure 9-30 Framework Manager Metadata wizard: INCIDENT

9. Click **Next** after you have selected the objects.

Archive

10. Keep the default settings on the next window and click **Import**. See Figure 9-31.

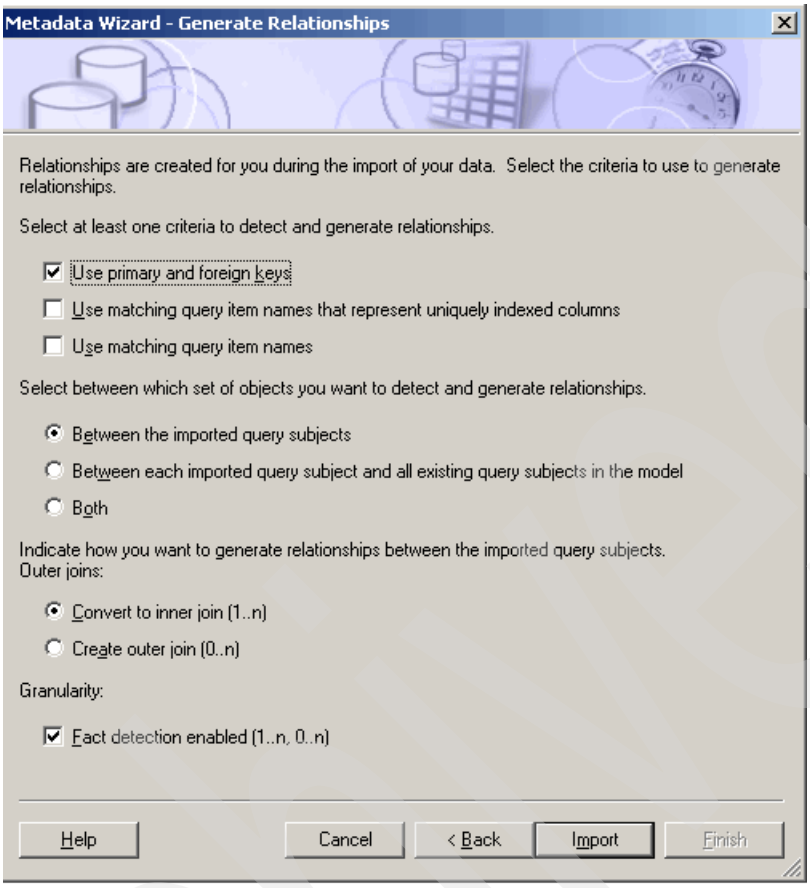


Figure 9-31 Framework Manager: Metadata wizard Import

11. After importing your tables in your framework model, verify the contents (see Figure 9-32 on page 270 and Figure 9-33 on page 270):
    a. Double-click the query subject such as INCIDENT
    b. Select the **Test** tab
    c. Click T**est Sample or Total Rows**.

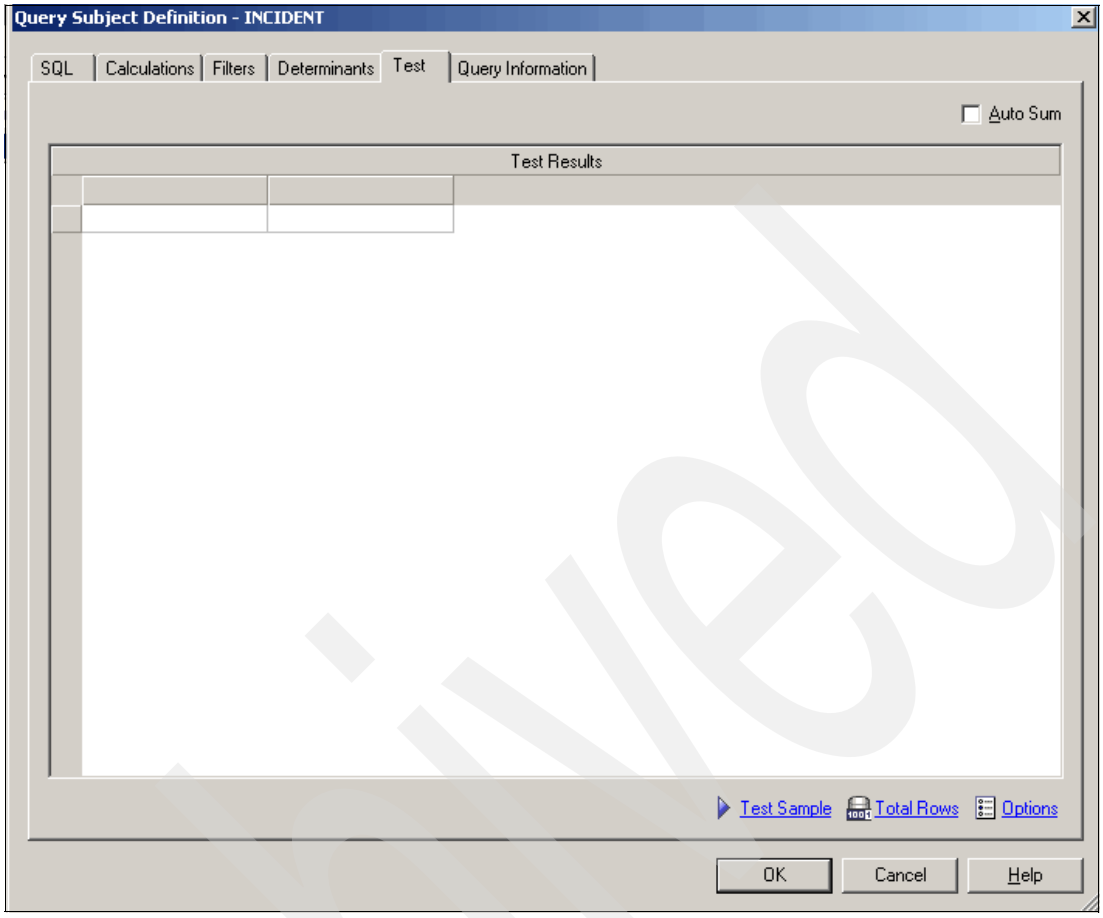


Figure 9-32 Framework Manager: Testing a query subject

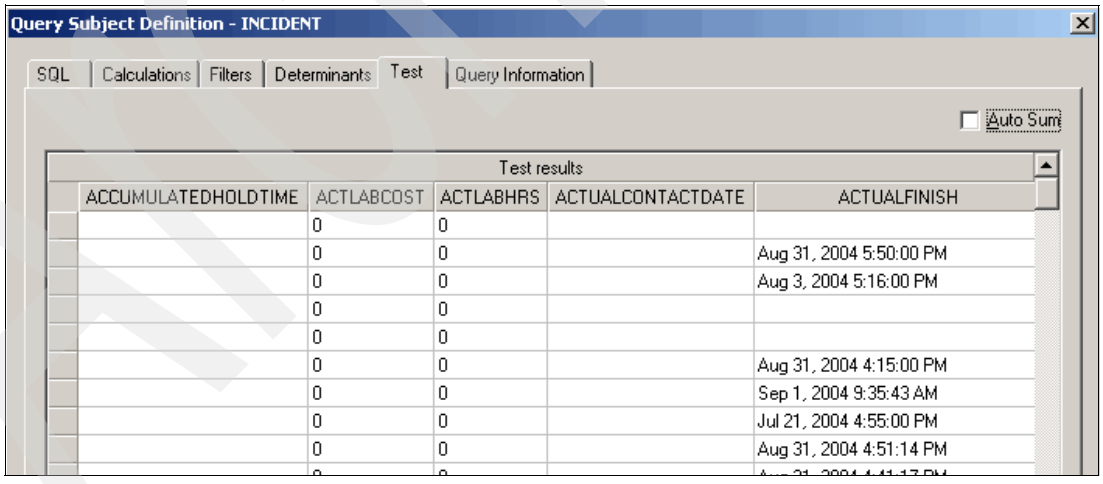


Figure 9-33 Framework Manager: Test Result INCIDENT

12. Create the relationship between CI and INCIDENT. Select both CI and INCIDENT, right-click this selection, and then select **Create** → **Relationship**. See Figure 9-34 on page 271.

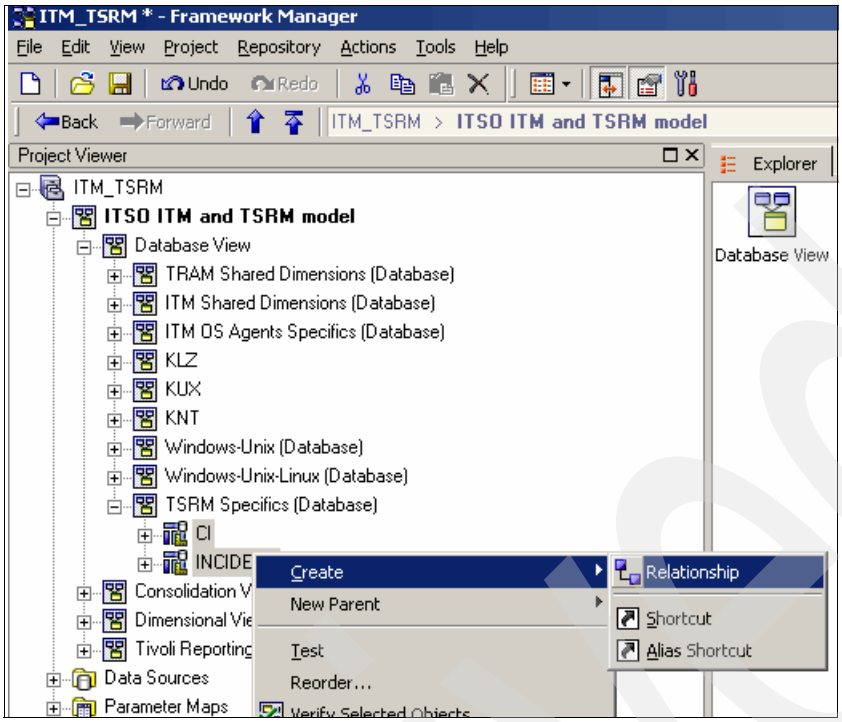


Figure 9-34 Framework Manager: Create relationship between CI and INCIDENT

13. Create a join between `CI.CINUM` and `INCIDENT.CINUM`. Verify that you have the correct cardinality (for example, “Each INCIDENT has one and only one CI” and “Each CI has one or more INCIDENT”). See Figure 9-35 on page 272.

Archive

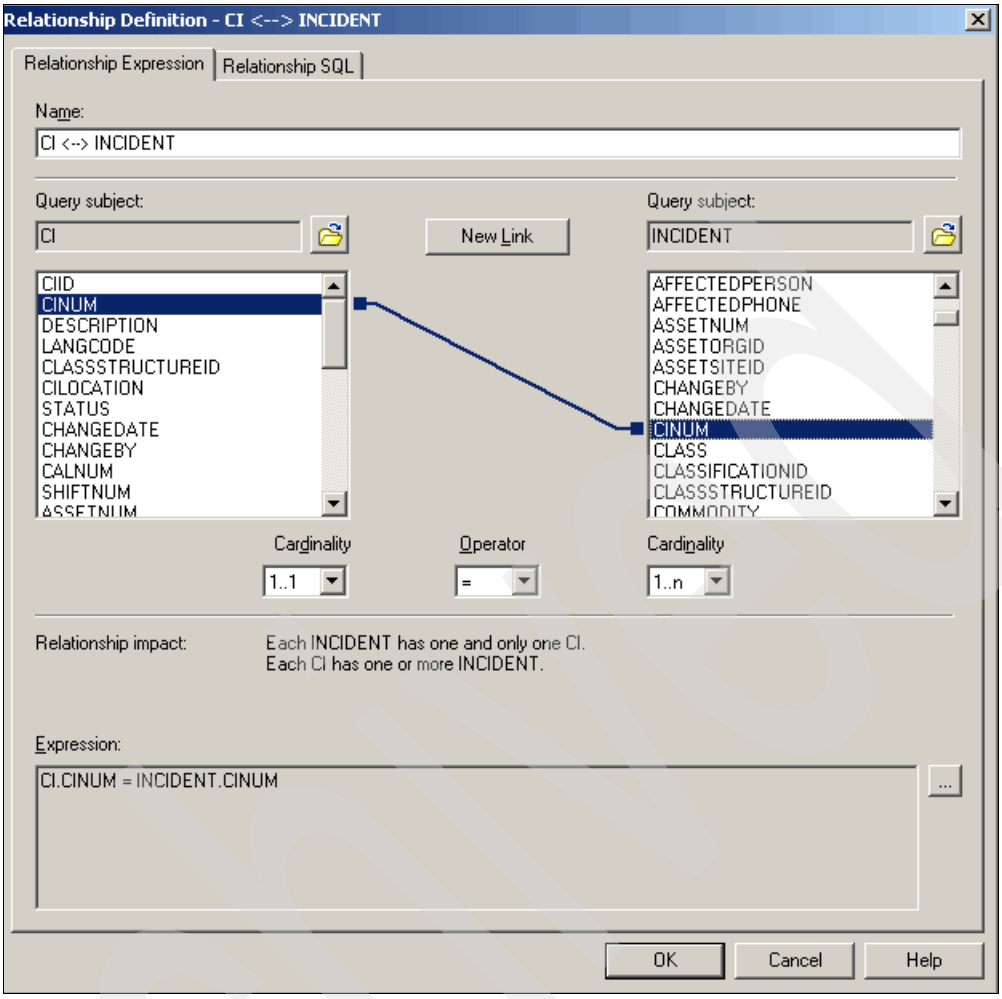


Figure 9-35 Framework Manager: Relationship Definition CI and INCIDENT

14. After creating the relationship, a good practice is to test it to be sure that you get the data you expect. To test, select both query subjects, right-click, and select **Test**. See Figure 9-36 on page 273.

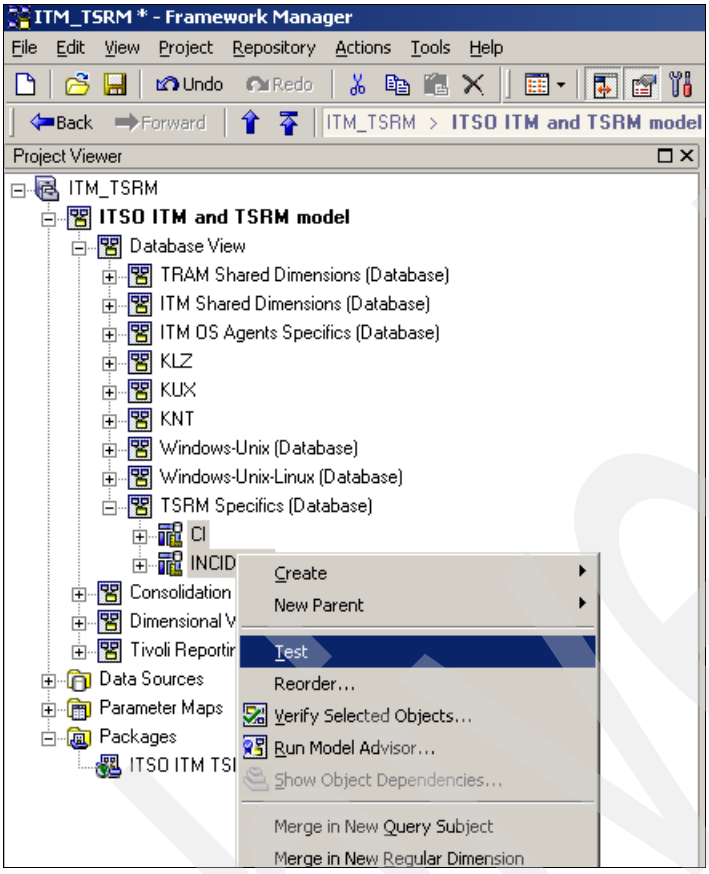
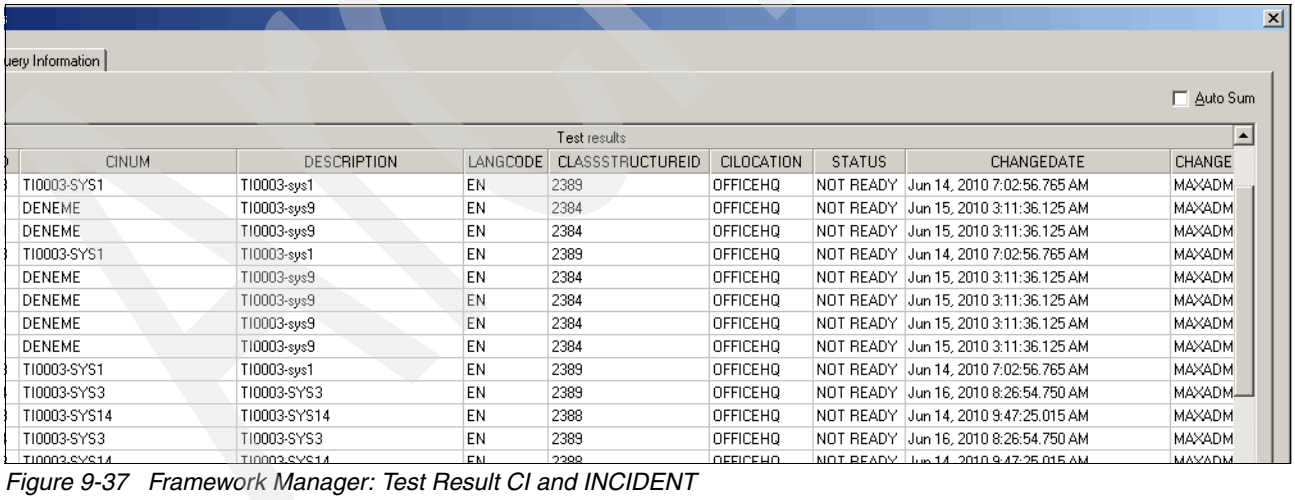


Figure 9-36 Framework Manager: Testing a relationship

Figure 9-37 shows the result of the test.



CINUM	DESCRIPTION	LANGCODE	CLASSSTRUCTUREID	CILOCATION	STATUS	CHANGEDATE	CHANGE
TI0003-SYS1	TI0003-sys1	EN	2389	OFFICEHQ	NOT READY	Jun 14, 2010 7:02:56.765 AM	MAXADM
DENEME	TI0003-sys9	EN	2384	OFFICEHQ	NOT READY	Jun 15, 2010 3:11:36.125 AM	MAXADM
DENEME	TI0003-sys9	EN	2384	OFFICEHQ	NOT READY	Jun 15, 2010 3:11:36.125 AM	MAXADM
TI0003-SYS1	TI0003-sys1	EN	2389	OFFICEHQ	NOT READY	Jun 14, 2010 7:02:56.765 AM	MAXADM
DENEME	TI0003-sys9	EN	2384	OFFICEHQ	NOT READY	Jun 15, 2010 3:11:36.125 AM	MAXADM
DENEME	TI0003-sys9	EN	2384	OFFICEHQ	NOT READY	Jun 15, 2010 3:11:36.125 AM	MAXADM
DENEME	TI0003-sys9	EN	2384	OFFICEHQ	NOT READY	Jun 15, 2010 3:11:36.125 AM	MAXADM
DENEME	TI0003-sys9	EN	2384	OFFICEHQ	NOT READY	Jun 15, 2010 3:11:36.125 AM	MAXADM
TI0003-SYS1	TI0003-sys1	EN	2389	OFFICEHQ	NOT READY	Jun 14, 2010 7:02:56.765 AM	MAXADM
TI0003-SYS3	TI0003-SYS3	EN	2389	OFFICEHQ	NOT READY	Jun 16, 2010 8:26:54.750 AM	MAXADM
TI0003-SYS14	TI0003-SYS14	EN	2388	OFFICEHQ	NOT READY	Jun 14, 2010 9:47:25.015 AM	MAXADM
TI0003-SYS3	TI0003-SYS3	EN	2389	OFFICEHQ	NOT READY	Jun 16, 2010 8:26:54.750 AM	MAXADM
TI0003-SYS14	TI0003-SYS14	EN	2388	OFFICEHQ	NOT READY	Jun 14, 2010 9:47:25.015 AM	MAXADM

Figure 9-37 Framework Manager: Test Result CI and INCIDENT

15. Create a relationship between the Tivoli Service Request Manager CI and the managed system in IBM Tivoli Monitoring. See Figure 9-38 on page 274 and Figure 9-39 on page 274.

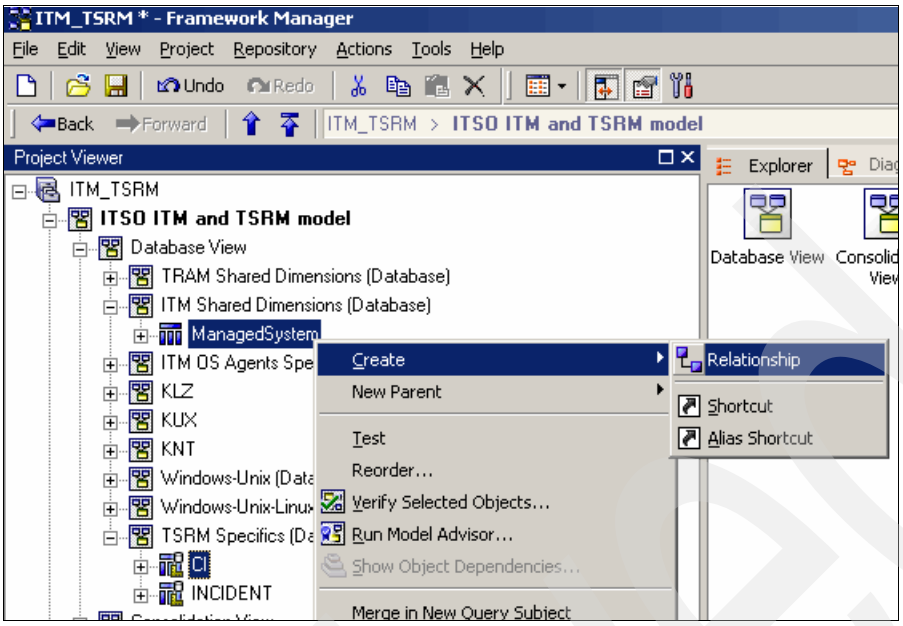


Figure 9-38 Framework Manager: Create relationship between ManagedSystem and CI

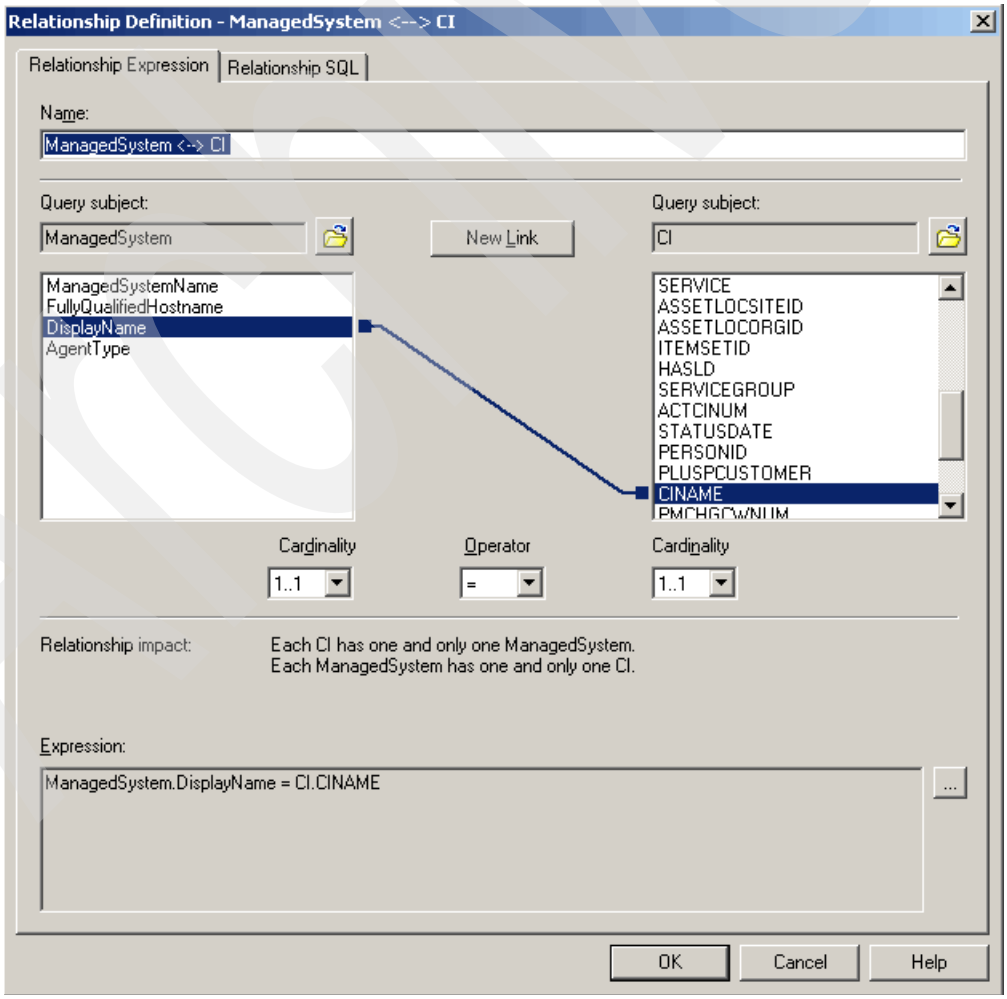


Figure 9-39 Framework Manager: Relationship Definition Managed System and CI

16. In our case, the join is between `ManagedSystem.DisplayName` and `CI.CINAME` because this is the standard we decided on and the person promoting the CIs in Tivoli Service Request Manager has followed this standard. Although your setup might differ, you must find some information in the CI that uniquely identifies the CI to an IBM Tivoli Monitoring managed system. See Figure 9-40.

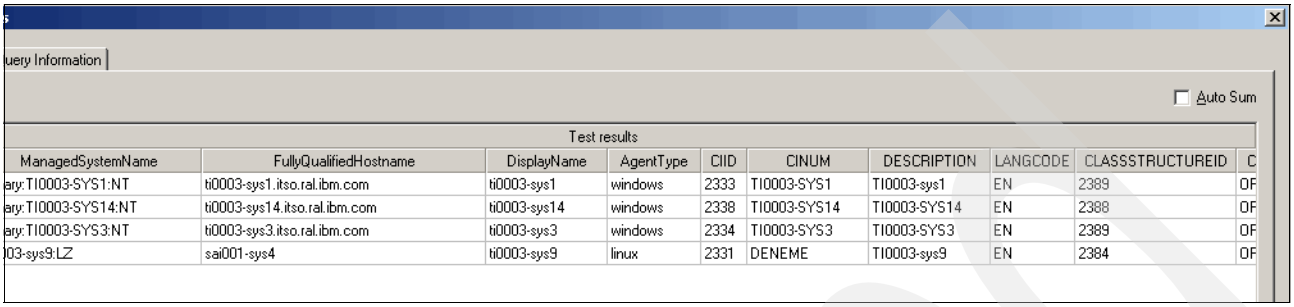


Figure 9-40 Framework Manager: Test results CI and ManagedSystem

We have now created a relationship between INCIDENT and ManagedSystem through CI. See Figure 9-41.

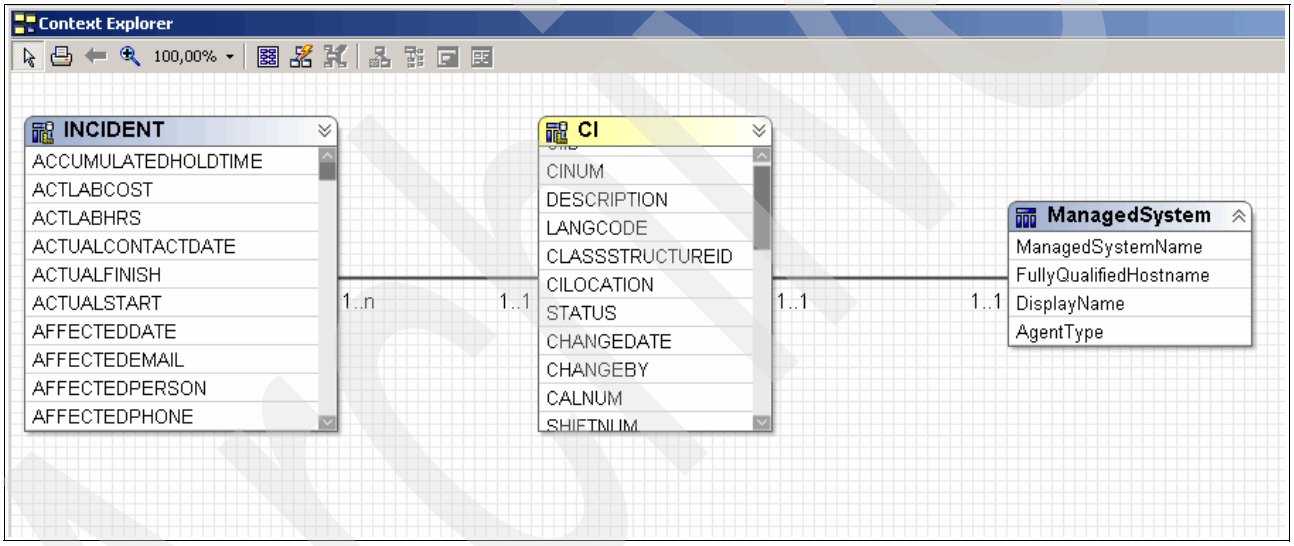


Figure 9-41 Framework Manager: Context Explorer CI

17. To test that the end-to-end relationship works, select **INCIDENT** and **ManagedSystem** and run a test. Framework Manager can join these query subjects by using CI in the background. See Figure 9-42 on page 276.

Test results

CREATIONDATE	DESCRIPTION
Jun 7, 2010 6:23:14.500 AM	NT_Process_CPU_Warning:Primary:TI0003-SYS1:NT:vmware-vmx:ITM_NT_Process
Jun 7, 2010 9:07:21.250 AM	NT_Process_CPU_Warning:Primary:TI0003-SYS1:NT:vmware-vmx:ITM_NT_Process
Jun 8, 2010 10:26:49.516 AM	NT_Process_CPU_Critical:Primary:TI0003-SYS1:NT:vmware-vmx:ITM_NT_Process
Jun 14, 2010 1:08:28.530 PM	NT_Process_CPU_Critical:Primary:TI0003-SYS14:NT:java:ITM_NT_Process
Jun 14, 2010 1:46:36.702 PM	NT_Physical_Disk_Busy_Warning:Primary:TI0003-SYS14:NT:0 C::ITM_NT_Physical_Disk
Jun 16, 2010 5:41:13.437 AM	NT_Physical_Disk_Busy_Critical:Primary:TI0003-SYS14:NT:0 C::ITM_NT_Physical_Disk
Jun 16, 2010 8:23:42.062 AM	NT_Process_CPU_Critical:Primary:TI0003-SYS14:NT:java:ITM_NT_Process
Jun 16, 2010 8:32:42.937 AM	NT_Process_CPU_Critical:Primary:TI0003-SYS14:NT:java:ITM_NT_Process

Figure 9-42 Framework Manager: Test result CI and ManagedSystem

9.4.4 Publishing the framework model package

We have created the relationships necessary to create our report and we are ready to publish the package to TCR V1.3:

1. Create a shortcut to Tivoli Service Request Manager. See Figure 9-43.

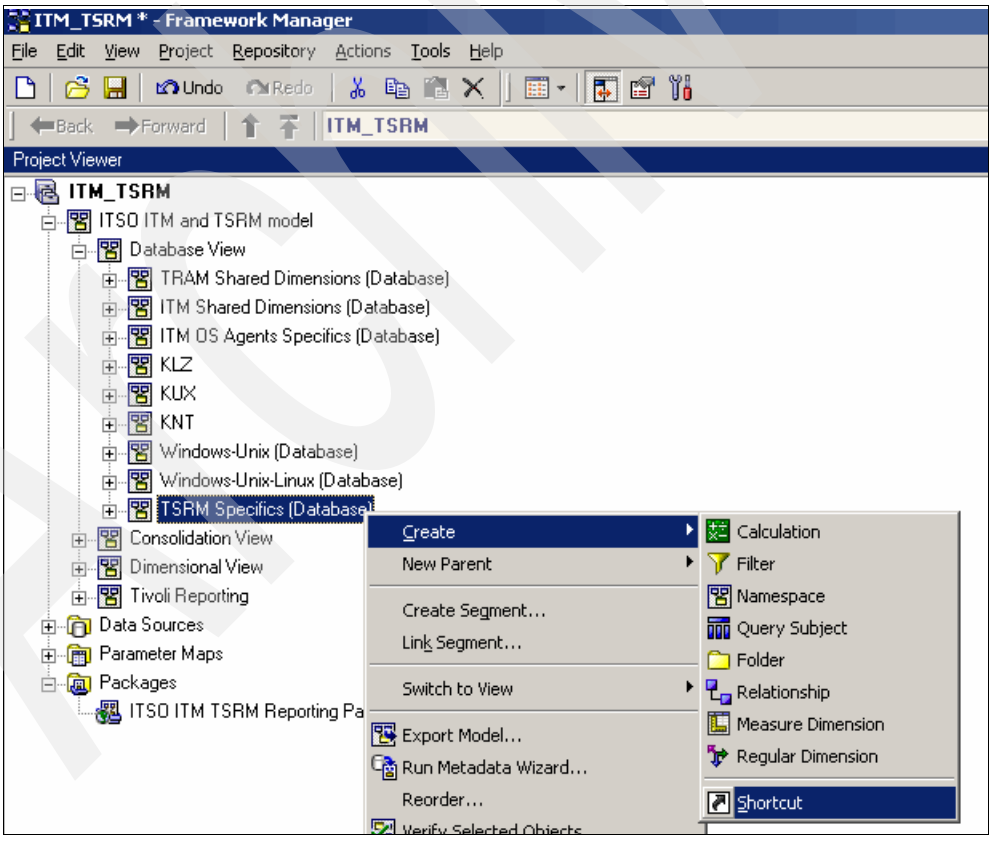


Figure 9-43 Framework Manager: Tivoli Service Request Manager Create Shortcut

2. Move the shortcut to under Tivoli Reporting and rename it to `TSRM Incident Model`. See Figure 9-44 on page 277.

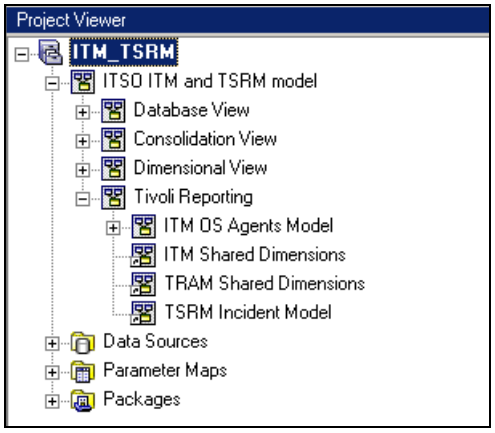


Figure 9-44 Framework Manager: Tivoli Service Request Manager Incident Model shortcut

3. Run the publishing wizard, as shown in Figure 9-45.

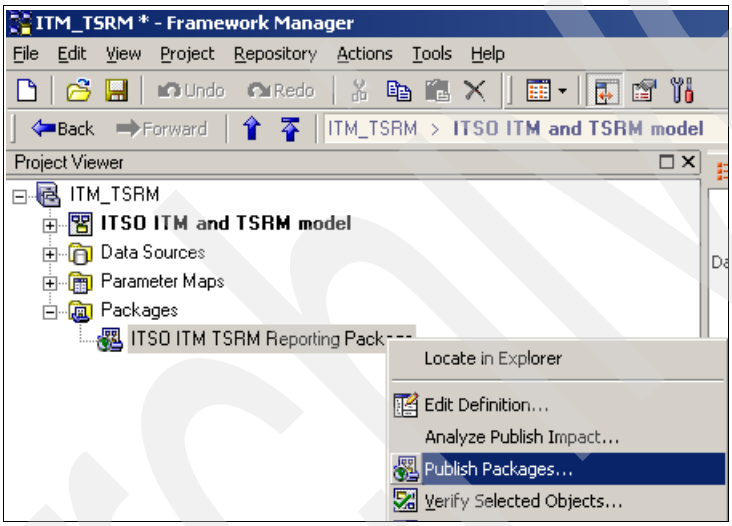


Figure 9-45 Framework Manager: Publish Package wizard

4. Step through the “Publish Package” windows by selecting default values. The package is verified and you might receive verification messages. If these messages are not more serious than warnings, you may ignore them. See Figure 9-46.

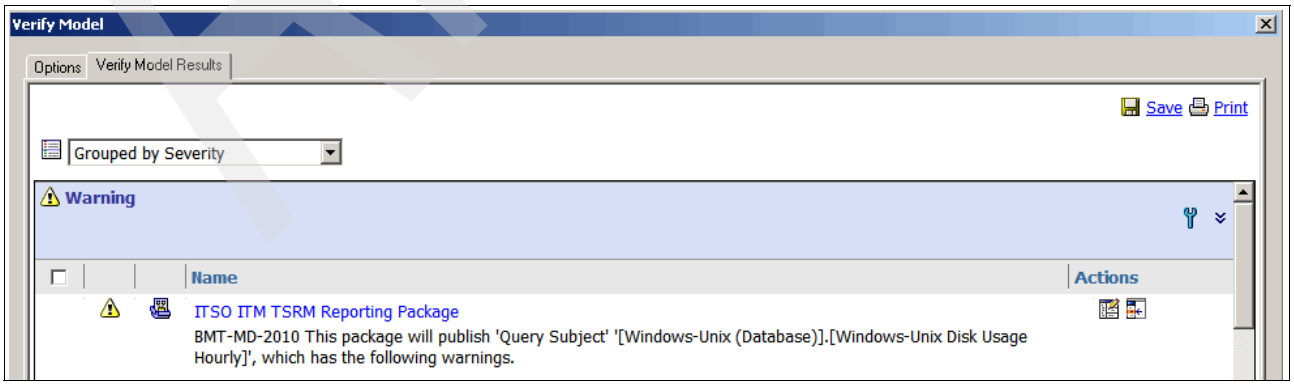


Figure 9-46 Framework Manager: Verify Model warning messages

9.5 Creating reports

Now we are ready to switch to Tivoli Common Reporting and the Report Studio:

1. Log in to Tivoli Integrated Portal and select **Common Reporting**. See Figure 9-47.

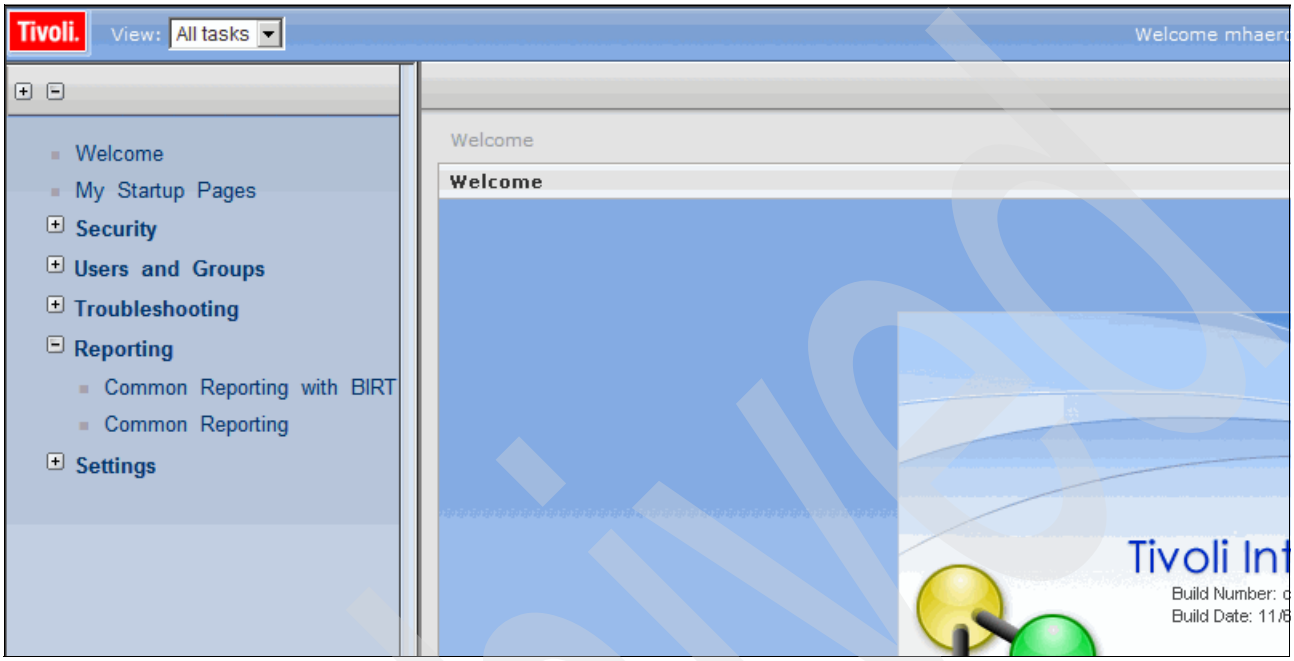


Figure 9-47 Tivoli Integrated Portal: Select Common Reporting

2. On the right side of the window, select **Launch and Report Studio**. See Figure 9-48.

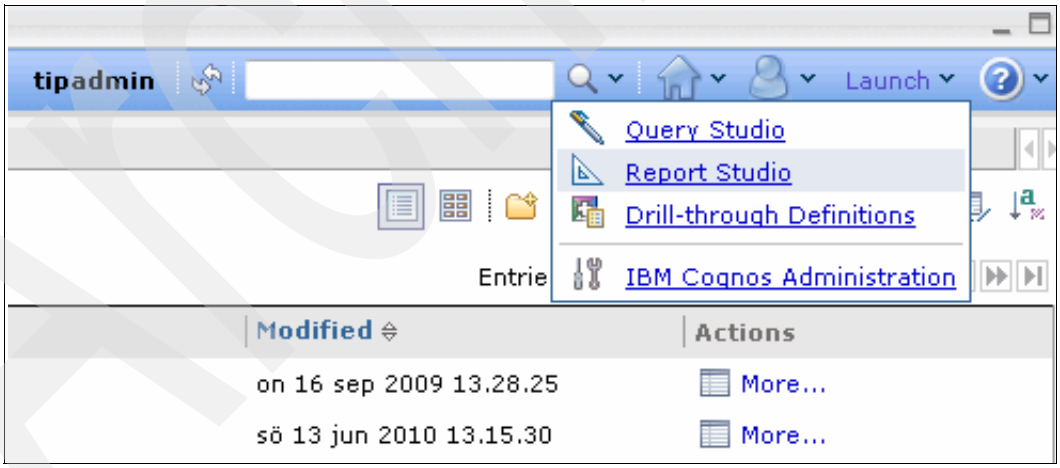


Figure 9-48 TCR: Launching Report Studio

3. Select the package you just published. See Figure 9-49 on page 279.

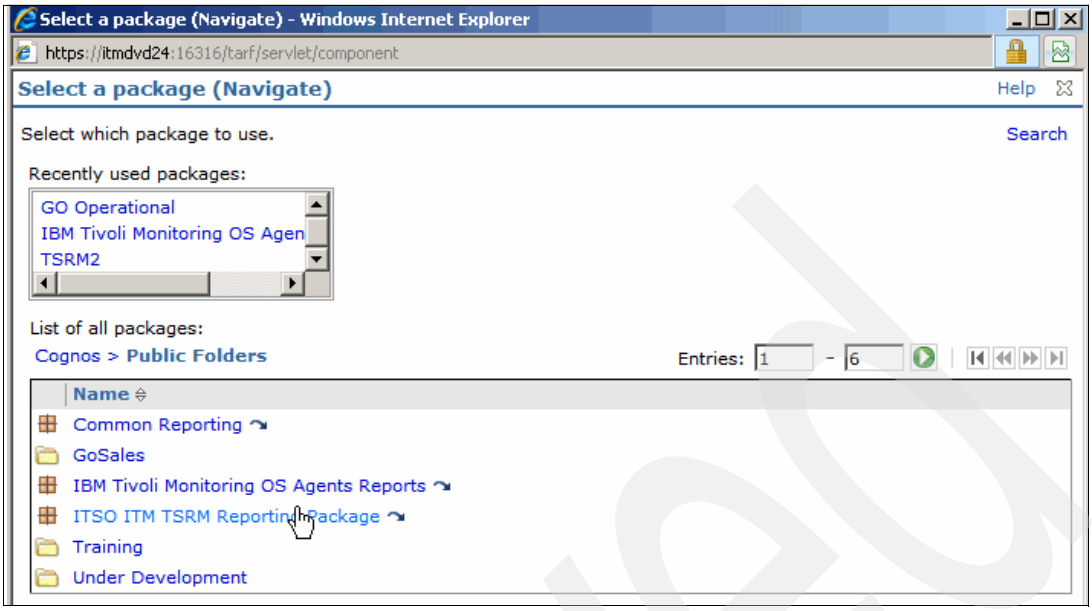


Figure 9-49 TCR Report Studio: Select Reporting Package

4. Select **Create a new report or template**. See Figure 9-50.

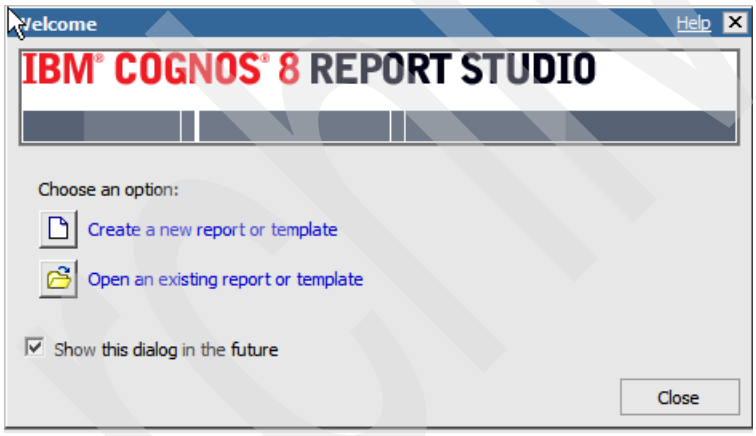


Figure 9-50 TCR Report Studio: Create a new report

Tivoli Common Reporting V2.1: In Tivoli Common Reporting V2.1, which was not available at the time of writing this book, Cognos references will be removed in the Tivoli Common Reporting panels. For example the title of this window will be changed to REPORT STUDIO.

9.5.1 Server Statistics Chart report

Our report actually consists of three reports, as outlined earlier. The entry report is a list of servers and the total number of incidents logged to them. Clicking one of them, or drilling down, shows all the incidents logged to the server. Drilling down on one incident shows graphs of important performance metrics for the server for the day the incident was reported.

Perform the following steps:

1. We start with creating the details report so select **Chart** as the type for the report, and then select **Column** and **Column with 3-D Visual Effect**. See Figure 9-51.

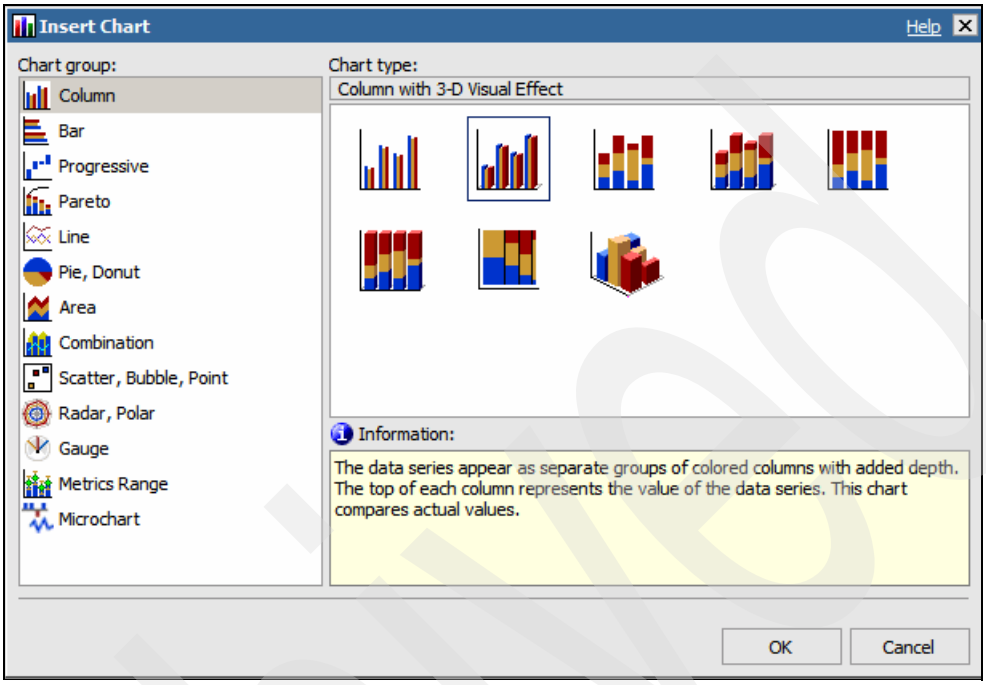


Figure 9-51 TCR Report Studio: Insert Chart

2. Expand **ITSO ITM TSRM Reporting Package** → **Tivoli Reporting** → **ITM OS Agents Model** → **Windows-Linux** → **CPU Hourly in the Insertable Objects** and drag all CPU metrics to the Series place holder in the chart. See Figure 9-52.

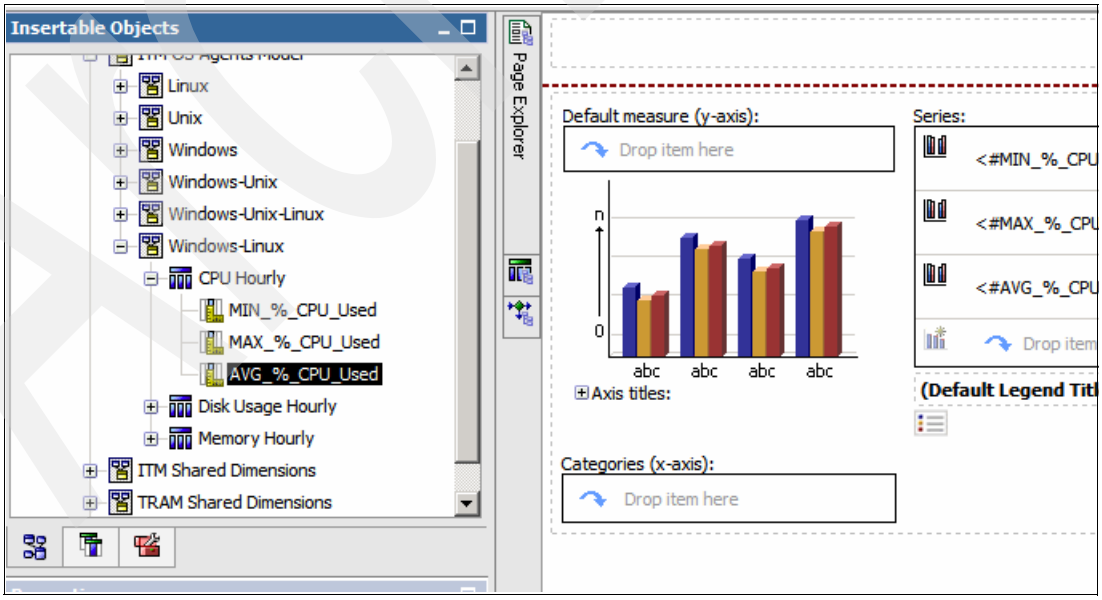


Figure 9-52 TCR Report Studio: Insert CPU metrics

3. Expand **ITSO ITM TSRM Reporting Package** → **Tivoli Reporting** → **ITM Shared Dimensions** → **Managed Systems** and drag Managed System Name to Default measure (y-axis). See Figure 9-53.
4. Expand **ITSO ITM TSRM Reporting Package** → **Tivoli Reporting** → **TRAM Shared Dimensions** → **Time** and drag Hour to Default measure (x-axis).

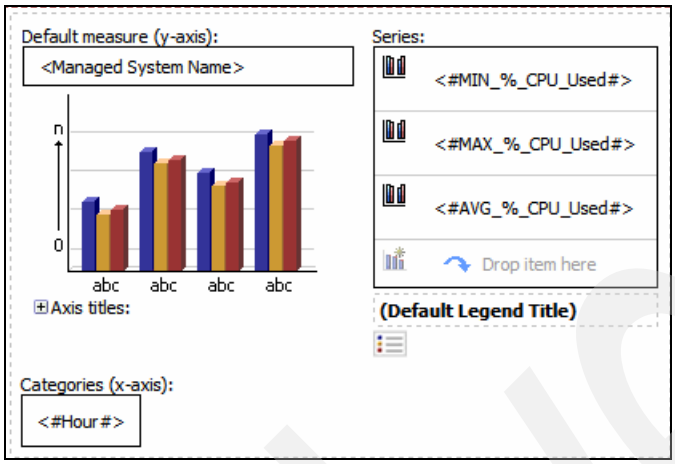


Figure 9-53 TCR Report Studio: CPU chart

5. Select the properties for the y-axis and change Minimum Value to 0 and Maximum Value to 100. Click where it says "**0** → **n**" on the left side. The values are changed in the Properties windows at the bottom left. See Figure 9-54 on page 282.

Archived

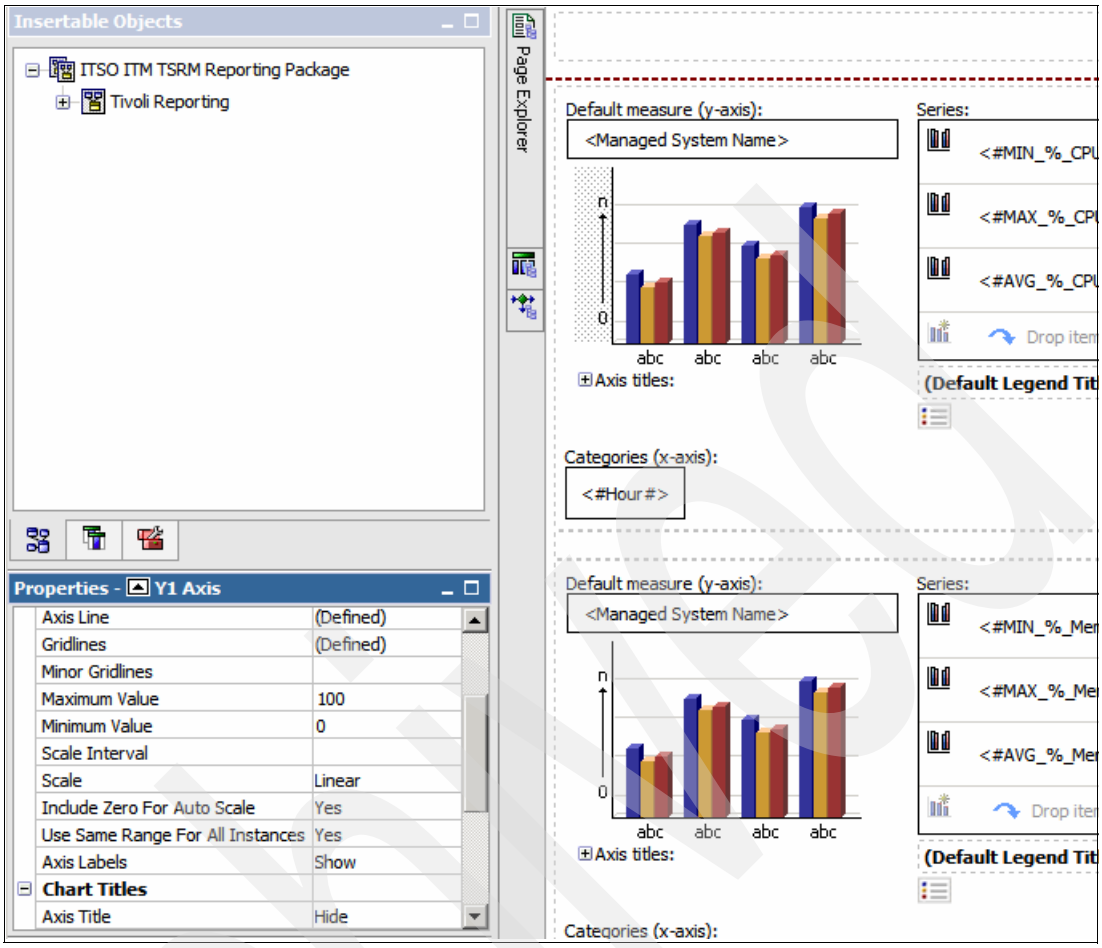


Figure 9-54 TCR Report Studio: Y1 Axis properties

6. Create two filters to use as drill-through parameters. A *drill-through parameter* begins and ends with a question mark (?) symbol. Click **Detail Filters** and then click **Add**. See Figure 9-55.

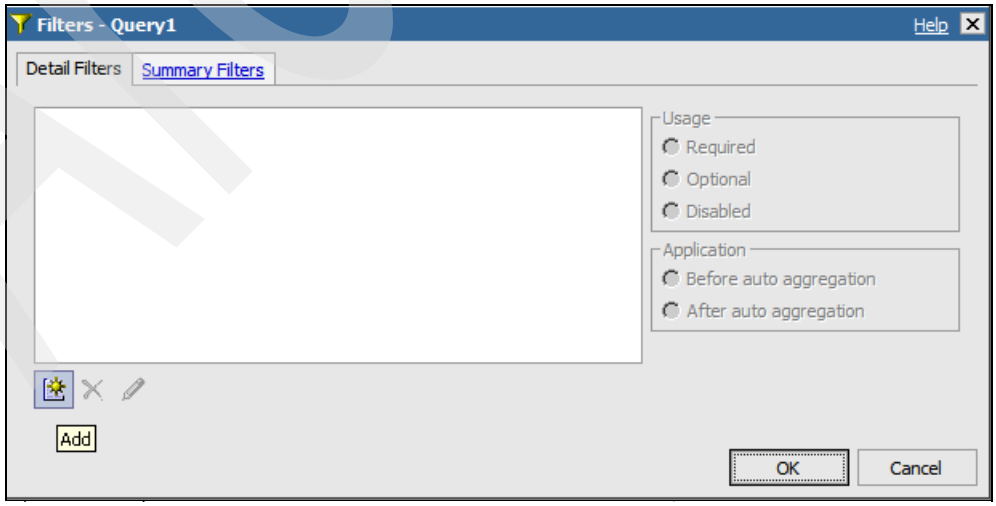


Figure 9-55 TCR Report Studio: Add Filter

7. Type the following text in the Expression Definition window (Figure 9-56):

   ```
   [ITM Shared Dimensions].[Managed Systems].[System Name]=?ms_SN?
   ```

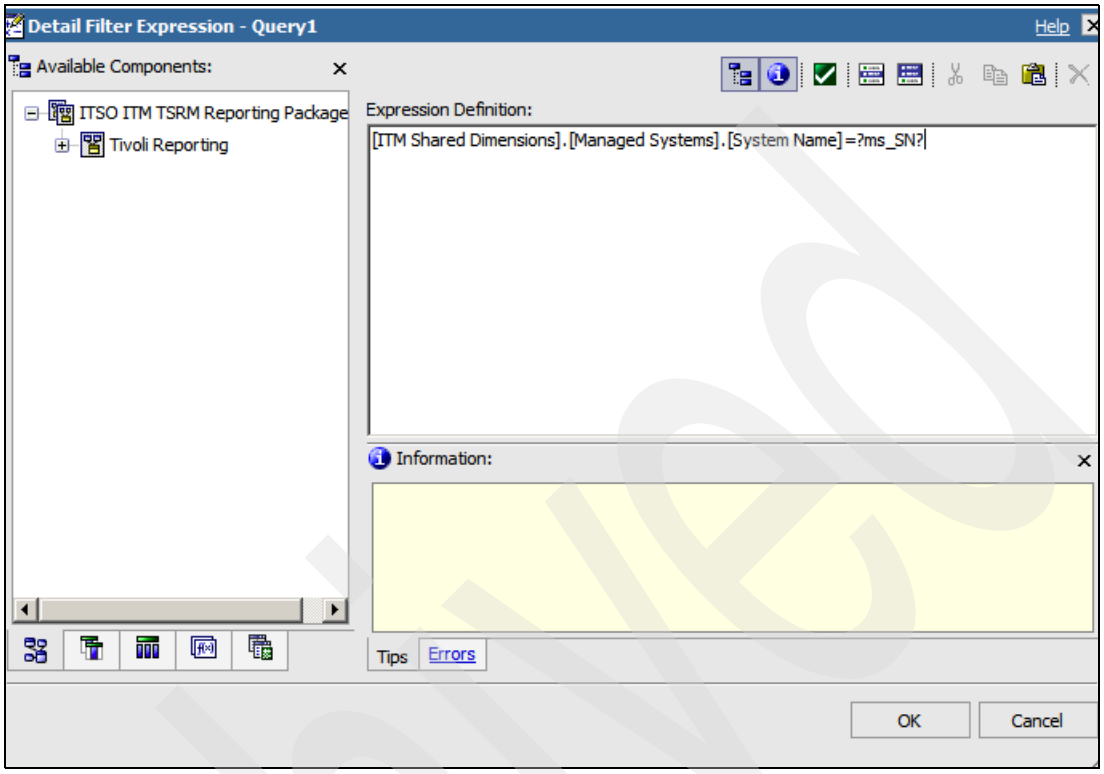


Figure 9-56 TCR Report Studio: System Name filter

8. Create another filter for the date with the following definition:

   ```
   [TRAM Shared Dimensions].[Time].[Date]=?t_D?]
   ```

9. Save the report as **Server Statistics Chart**. Run the report to test it. You are prompted for a system name and a date (Figure 9-57 on page 284).

Archived

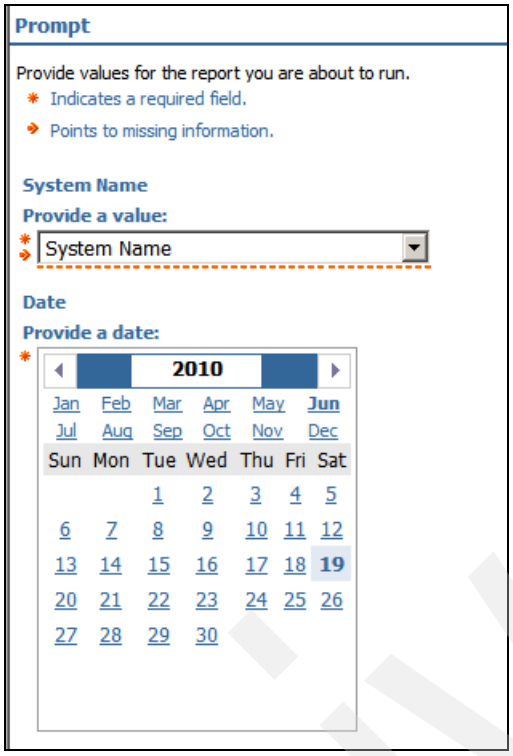


Figure 9-57 TCR Report Studio: Server Statistics Chart prompts

10. Create the next graph under the first by dragging a box object to the right of the graph:
    a. Select the Toolbox tab in Insertable objects pane.
    b. Select the **Block object type** and drag it to the right of the graph. See Figure 9-58.

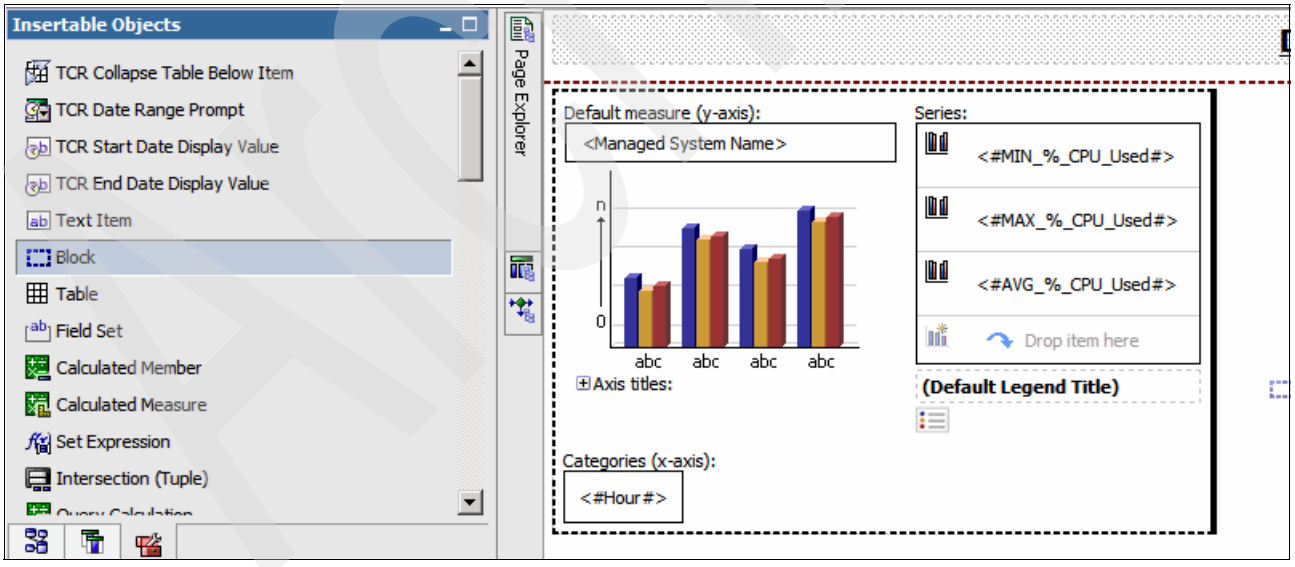


Figure 9-58 TCR Report Studio: Block object

11. Select the graph, and then copy and paste to create the next graph. After pasting it below the block object, you can remove the CPU metrics from the Series and add three memory metrics instead. For the disk usage metrics, repeat by adding a block, copy and paste the graph and replace the metrics. Make sure that you use some of the percentage metrics so all graphs have the same type of y value. Test and save the report.

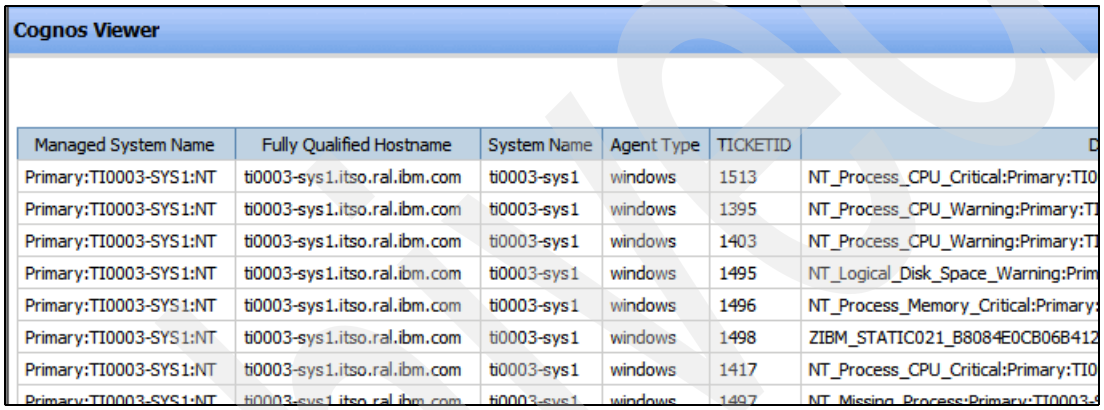
9.5.2 Server Incident List report

The next report we create is the list of incidents for one system. Start by creating a new report and select a **List** report. Insert all data from **Tivoli Reporting** → **ITM Shared Dimensions** → **Managed Systems** by dragging.

Add data from Tivoli Service Request Manager by double-clicking the items in the following order:

1. Tivoli Reporting → TSRM Incident Model → INCIDENT → TICKETID
2. Tivoli Reporting → TSRM Incident Model → DESCRIPTION
3. Tivoli Reporting → TSRM Incident Model → REPORTDATE

Test the report and save as Incident List. See Figure 9-59.



Managed System Name	Fully Qualified Hostname	System Name	Agent Type	TICKETID	D
Primary:TI0003-SYS1:NT	ti0003-sys1.itso.ral.ibm.com	ti0003-sys1	windows	1513	NT_Process_CPU_Critical:Primary:TI0
Primary:TI0003-SYS1:NT	ti0003-sys1.itso.ral.ibm.com	ti0003-sys1	windows	1395	NT_Process_CPU_Warning:Primary:TI
Primary:TI0003-SYS1:NT	ti0003-sys1.itso.ral.ibm.com	ti0003-sys1	windows	1403	NT_Process_CPU_Warning:Primary:TI
Primary:TI0003-SYS1:NT	ti0003-sys1.itso.ral.ibm.com	ti0003-sys1	windows	1495	NT_Logical_Disk_Space_Warning:Prim
Primary:TI0003-SYS1:NT	ti0003-sys1.itso.ral.ibm.com	ti0003-sys1	windows	1496	NT_Process_Memory_Critical:Primary:
Primary:TI0003-SYS1:NT	ti0003-sys1.itso.ral.ibm.com	ti0003-sys1	windows	1498	ZIBM_STATIC021_B8084E0CB06B412
Primary:TI0003-SYS1:NT	ti0003-sys1.itso.ral.ibm.com	ti0003-sys1	windows	1417	NT_Process_CPU_Critical:Primary:TI0
Primary:TI0003-SYS1:NT	ti0003-sys1.itso.ral.ibm.com	ti0003-sys1	windows	1497	NT_Missing_Process:Primary:TI0003-S

Figure 9-59 TCR Report Studio: Server Incident List unsorted

Select the **REPORTDATE** column and click the sort icon on the toolbar to sort ascending or descending order.

The next step is to define the drill-trough to the chart report:

1. Click **Data** → **Drill Behavior**.
2. On the basic tab, under Report output drill capabilities, select the **Allow this report to be a package-based drill-through source** check box and click **OK**.
3. Click one of the cells in the column Managed System Name and click **Drill-Through Definitions**. (Do not click the header; clicking must be in one of the cells.)
4. Click **New Drill Through**.
5. Under Report, click the **ellipse (...)** button.
6. Select the **Server Statistics Chart report** you created and click **Open**.
7. Under Action, click **Run the Report.**
8. Under Format, click **HTML.**
9. Select the **Open check box** in the new window
10. Click **Edit** (a pen icon below the Parameters area).
11. For item `ms_SN`, under Method, click **Pass data item value**, and under Value, click **System Name**.

Archived

12. For item `t_D`, under Method, click **Pass data item value**, and under Value, click **REPORTDATE**. See Figure 9-60

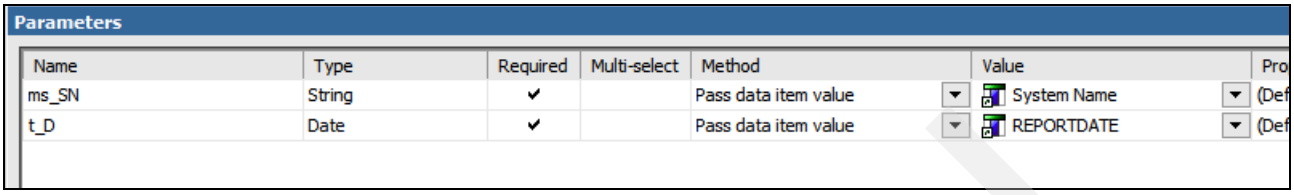


Figure 9-60 TCR Report Studio: Drill through parameters Incident List

13. Create a a drill-through parameter for this report to use from the entry report. A drill-through parameter begins and ends with a question mark (?) symbol.
14. Click **Filters**.
15. Click **Add** and type the following text in the Expression Definition window:

    ```
    [ITM Shared Dimensions].[Managed Systems].[System Name]=?ms_SN?
    ```

16. Run the report and check that you can drill through to the chart report. Save as Incident List.

9.5.3 Top Servers with Incident List report

The third and final report we create is the summary list report of number of incidents per server:

1. Create a new report.
2. In the New dialog box, click **List** and click **OK**.
3. In the Insertable objects pane, on the Source tab, expand **Tivoli Reporting** → **ITM Shared Dimensions** → **Managed Systems**.
4. Drag System Name to the report.
5. In the Insertable objects pane, on the Toolbox tab, double-click **Query Calculation**.
6. Name the calculation `Number of Incidents`.
7. In the Expression Definition, enter the following information:

   ```
   count([TSRM Incident Model].[INCIDENT].[TICKETID])
   ```

8. Click **OK**.
9. Select to sort in Descending order on the Number of Incidents column.
10. Click **Data** → **Drill Behavior**.
11. On the Basic tab, under Report output drill capabilities, select the **Allow this report to be a package-based drill-through source** check box and click **OK**.
12. Right-click a cell in the System Name column (not the header) and click **Drill-Through Definitions**.
13. Click **New Drill Through**.
14. Under Report, click the **ellipse (...)** button.
15. Select the Incident List report you created, and click **Open**.
16. Under Action, click **Run the Report**.
17. Under Format, click **HTML**.

Archived

18. Click **Edit**. Any existing drill-through parameters are listed. You see the parameter you created for Incident List.
19. For item `ms_SN`, under Method, click the **Pass data item value**, and under Value, click the **System Name**. See Figure 9-61.

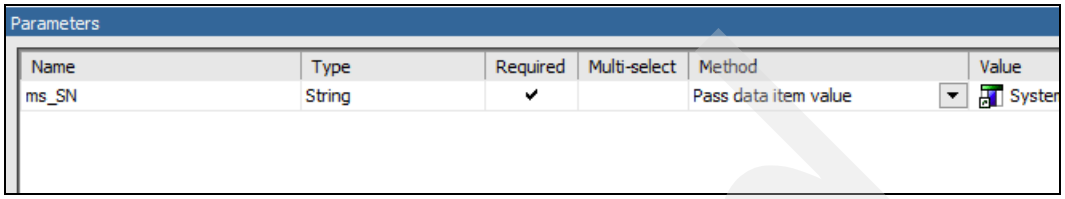


Figure 9-61 TCR Report Studio: Parameters for Incident List report

20. Click **OK**.
21. Save the report as `Top Servers with Incident List`.
22. Run the report.

    You can now run the Top Servers with Incident List report. This report shows us servers that are monitored with IBM Tivoli Monitoring and that have incidents logged to them in Tivoli Service Request Manager. The list shows the servers with the highest number of incidents listed first.
23. Click a system name and run the Incident List report and from there select a managed system name and run the Server statistics report. Figure 9-62 shows the Top Server Incidents List.

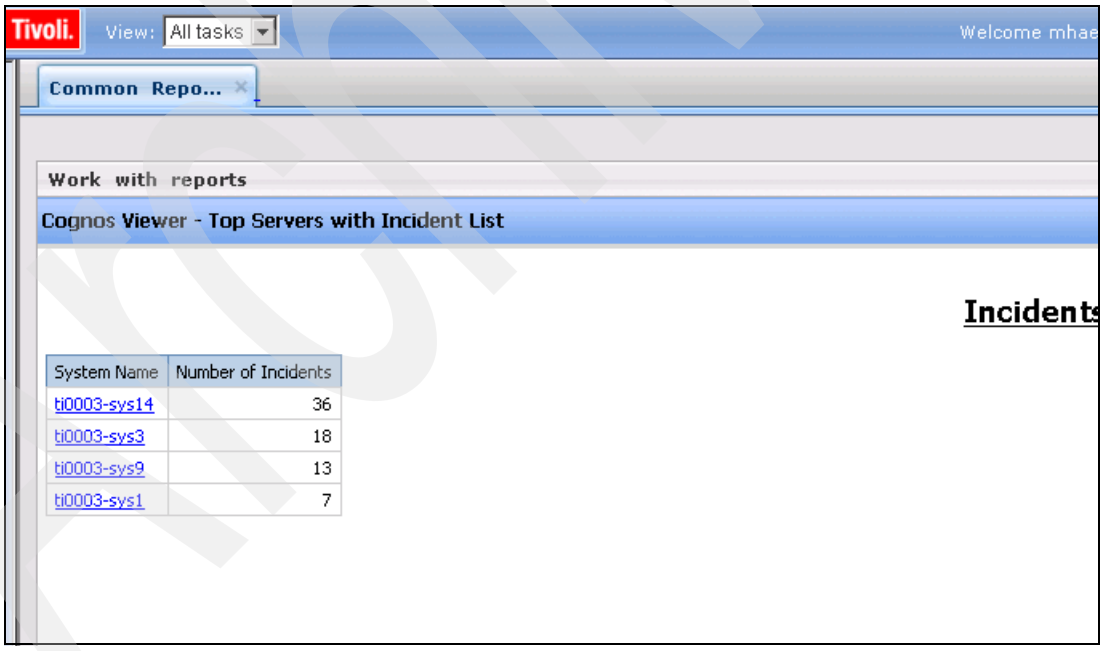
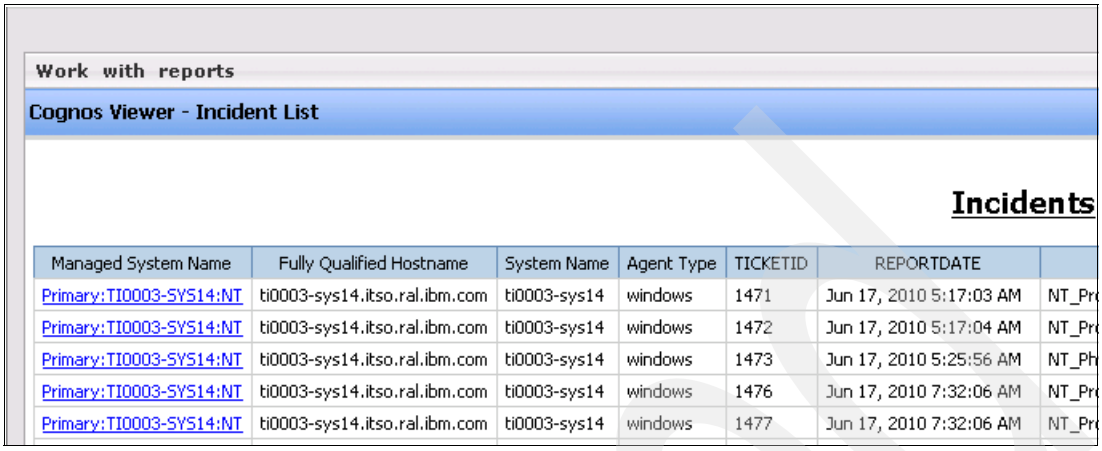


Figure 9-62 Top Server Incidents List

After you select a server by clicking the system name, a list of all incidents that are logged to this server are listed in chronological order. See Figure 9-63.



Managed System Name	Fully Qualified Hostname	System Name	Agent Type	TICKETID	REPORTDATE	
Primary:TI0003-SYS14:NT	ti0003-sys14.itso.ral.ibm.com	ti0003-sys14	windows	1471	Jun 17, 2010 5:17:03 AM	NT_Pr
Primary:TI0003-SYS14:NT	ti0003-sys14.itso.ral.ibm.com	ti0003-sys14	windows	1472	Jun 17, 2010 5:17:04 AM	NT_Pr
Primary:TI0003-SYS14:NT	ti0003-sys14.itso.ral.ibm.com	ti0003-sys14	windows	1473	Jun 17, 2010 5:25:56 AM	NT_Ph
Primary:TI0003-SYS14:NT	ti0003-sys14.itso.ral.ibm.com	ti0003-sys14	windows	1476	Jun 17, 2010 7:32:06 AM	NT_Pr
Primary:TI0003-SYS14:NT	ti0003-sys14.itso.ral.ibm.com	ti0003-sys14	windows	1477	Jun 17, 2010 7:32:06 AM	NT_Pr

Figure 9-63 Server Incident List

24. Click one of the links on the page, to open a page with three graphs. The graphs show hourly CPU, memory, and disk performance metrics for the same day that the incident was reported. The figures show the following information:
   - Figure 9-64 shows the hourly CPU metrics.
   - Figure 9-65 on page 289 shows the hourly memory metrics.
   - Figure 9-66 on page 289 shows the hourly disk metrics.

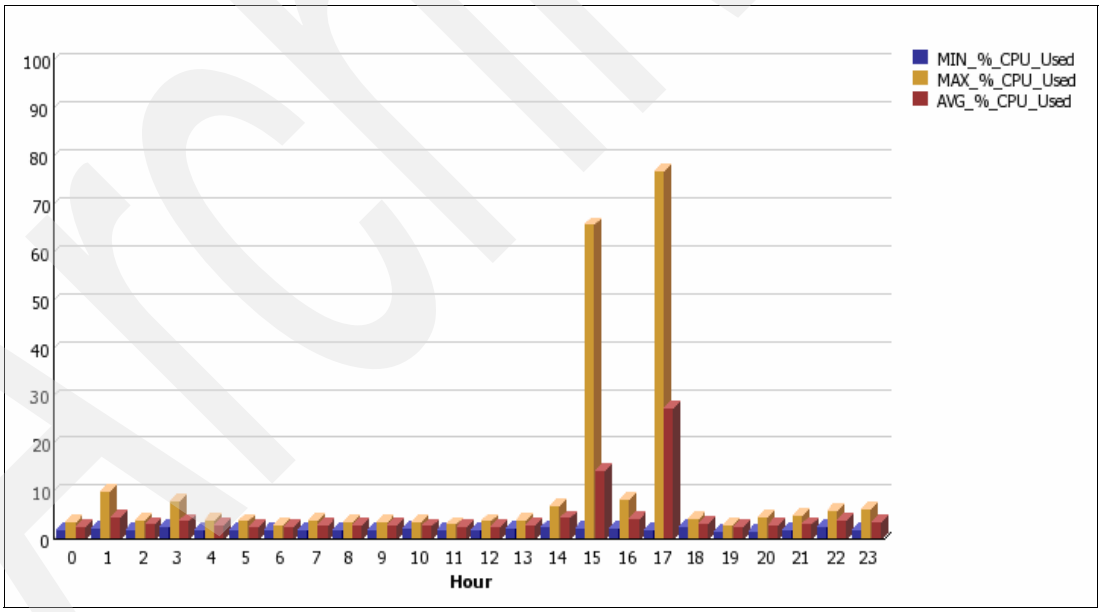


Figure 9-64 Hourly CPU metrics

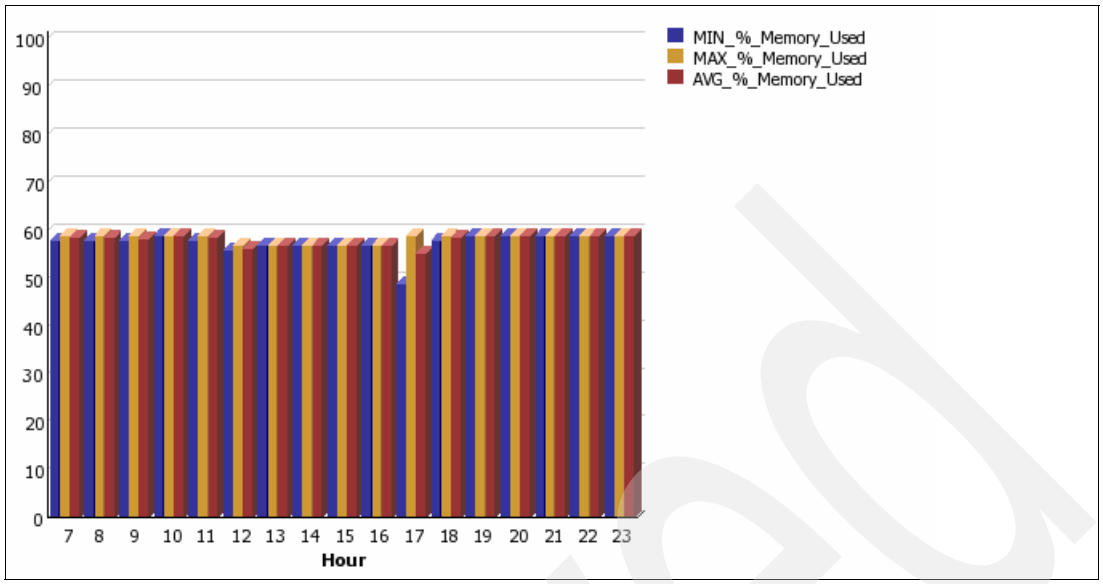


Figure 9-65 Hourly memory metrics

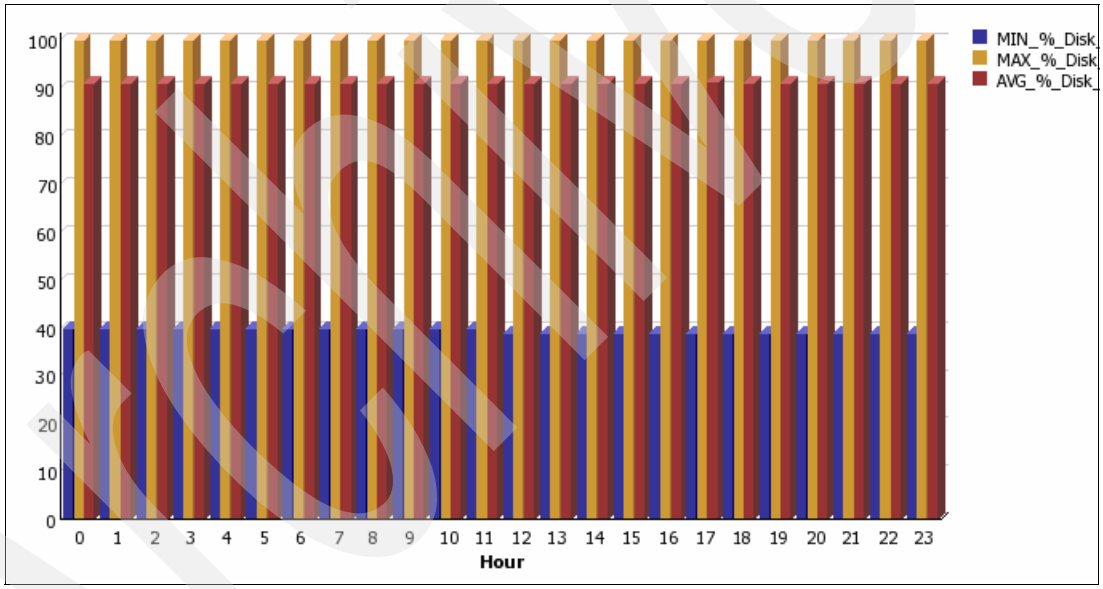


Figure 9-66 Hourly disk metrics

Archived

9.6 Summary

Figure 9-67 shows the summary of this integration.

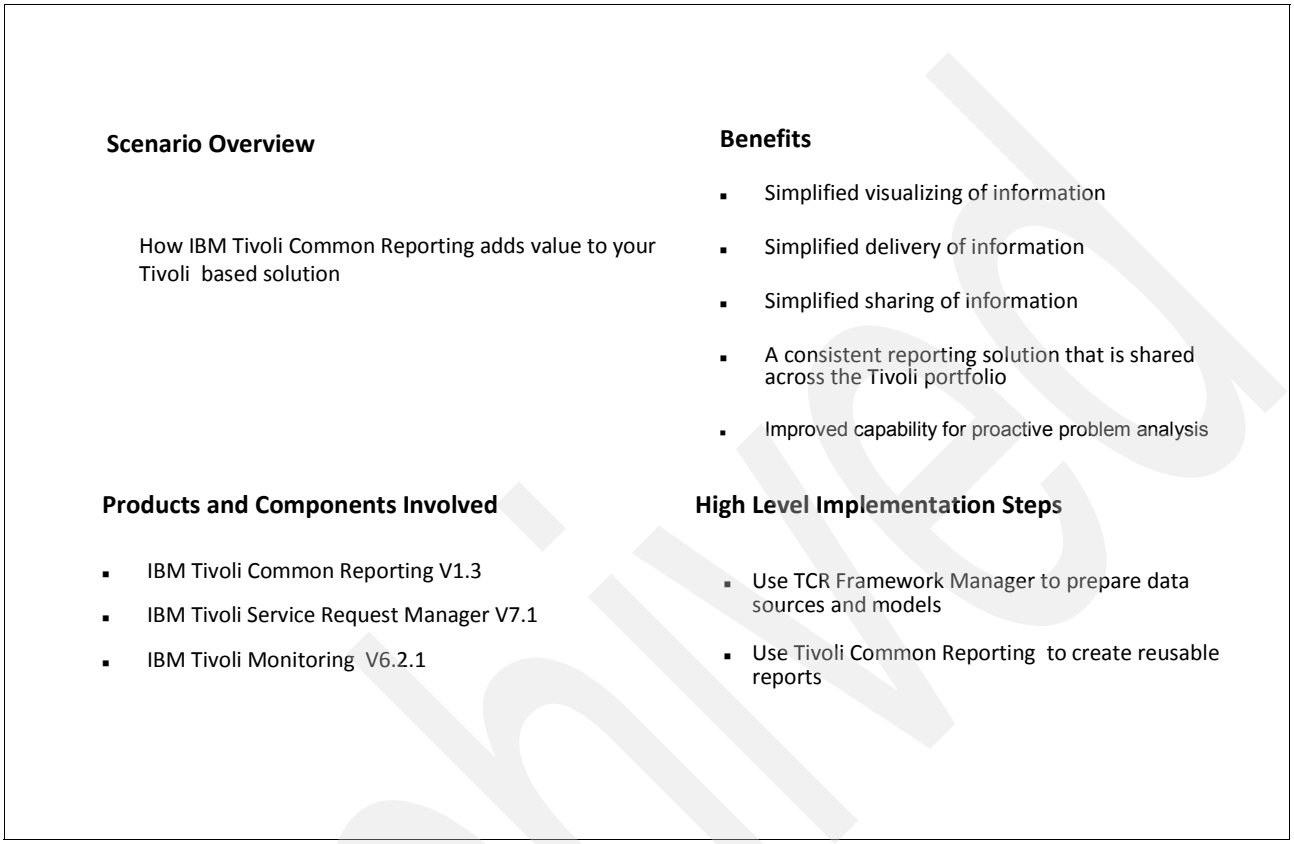


Figure 9-67 Quick summary

Archived

10

Protecting Tivoli Integrated Portal with Tivoli Access Manager for e-business

This chapter describes the value of using a high-performance, multi-threaded reverse proxy (Tivoli Access Manager WebSEAL component) to “front-end” the Tivoli Integrated Portal interface. This integrated solution provides an authentication and authorization framework to allow controlled access to the Tivoli Integrated Portal. This solution is extremely valuable in cases where access to the Tivoli Integrated Portal is required outside of an organization over the Internet.

This chapter contains the following topics:

- 10.1, “Scenario overview and products involved” on page 292
- 10.2, “Benefits” on page 293
- 10.3, “Integrating a Tivoli Integrated Portal environment with IBM Tivoli Access Manager WebSEAL” on page 295
- 10.4, “Summary” on page 301

© Copyright IBM Corp. 2011. All rights reserved.

10.1 Scenario overview and products involved

The Tivoli Access Manager *extended Trust Association Interceptor* (referred to as eTAI in this chapter) establishes “trust” in WebSphere Application Server, performs user authentication, and inserts credentials information into the HTTP request.

Trust associations: WebSphere Application Server supports single sign-on (SSO) with perimeter authentication services through *trust associations*. When trust associations are enabled, WebSphere Application Server does not need to authenticate a user if a request arrives through a trusted source that has already performed the authentication.

The Tivoli Integrated Portal with eTAI plug-in provides validation of trust with the perimeter authentication service and extraction of credential information from the request. A user that is accessing the Tivoli Integrated Portal leveraging the extended Trust Association Interceptor with WebSEAL goes through the flow shown in Figure 10-1.

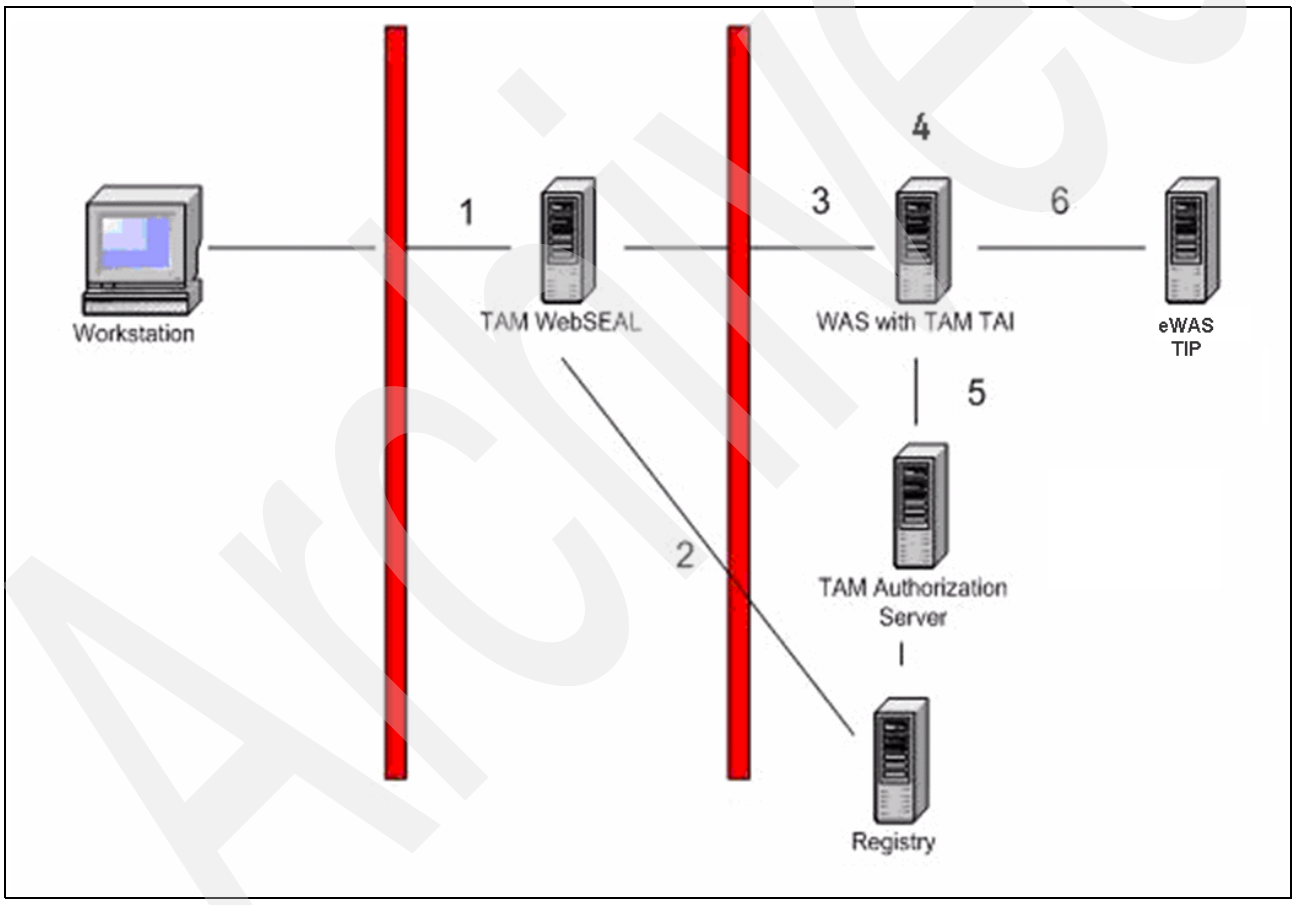


Figure 10-1 Tivoli Integrated Portal and WebSEAL integration flow

The figure shows the following steps:

1. The user login request is intercepted by WebSEAL (possibly through other proxies) and is prompted to authenticate.
2. WebSEAL authenticates the user and acquires credentials for the user from the user registry.

3. WebSEAL augments the request with an additional HTTP header (iv-creds) that contains the user's credentials. The password contained in the header is changed so it matches a configured SSO user. This request is sent to WebSphere Application Server.
4. Tivoli Integrated Portal receives the request and calls a TAI method (isTargetInterceptor) to determine whether the request is from a perimeter authentication service that has already authenticated the user.
5. Tivoli Integrated Portal calls a TAI method (negotiateValidateandEstablishTrust) to establish trust with the perimeter authentication server. This method establishes trust with WebSEAL by checking that the BA header contains the correct password for the configured SSO user. Trust between WebSEAL and Tivoli Integrated Portal cannot be established using mutually authenticated SSL sessions and can only be established by verifying the SSO password. No checking of certificates is performed by the TAI.
6. Tivoli Integrated Portal then retrieves the credentials. The iv-creds header is extracted from the request and used to retrieve the short name of the WebSEAL authenticated user, and credential object containing user and group information. Tivoli Integrated Portal then returns the authenticated user information and WebSphere Application Server has valid credentials that it can use for making authorization decisions in the usual J2EE manner.

The important points to note here are as follows:

- WebSEAL must insert the iv-creds header into the request.
- The new TAI can be configured to authenticate trust using either the Tivoli Access Manager Authorization Server or the WebSphere Application Server user registry directly.
- The credential object is inserted into the Subject by the TAI, which means that WebSphere Application Server does not have to perform any additional user registry searches as part of the authentication process.
- The user information that is extracted from the iv-creds header can have the dn format mapped from the initial format into the required format of the WebSphere Application Server user registry.

10.2 Benefits

In this section, we discuss the key benefits of protecting the Tivoli Integrated Portal with the IBM Tivoli Access Manager WebSEAL solution.

- Value of IBM Tivoli Access Manager for WebSEAL
- Value of Tivoli Integrated Portal
- Value to Tivoli Integrated Portal and WebSEAL Integrated Solution

10.2.1 Value of Tivoli Access Manager WebSEAL

IBM Tivoli Access Manager WebSEAL is the resource manager responsible for managing and protecting web-based information and resources. WebSEAL is a high performance, multi-threaded web server that applies fine-grained security policy to the Tivoli Access Manager protected web object space. WebSEAL can provide single sign-on solutions and incorporate back-end web application server resources into its security policy. WebSEAL normally acts as a reverse web proxy by receiving HTTP/HTTPS requests from a browser and delivering content from its own web server or from junctioned back-end web application servers.

Archive

Requests passing through WebSEAL are evaluated by the Tivoli Access Manager authorization service to determine whether the user is authorized to access the requested resource. WebSEAL provides the following features:

- Supports multiple authentication methods
- Accepts HTTP and HTTPS requests
- Integrates and protects back-end server resources through WebSEAL junction technology
- Manages fine-grained access control for the local and back-end server web space
- Supports resources including URLs, URL-based regular expressions, CGI programs, HTML files, Java servlets, and Java class files.
- Performs as a reverse web proxy
- Centrally manages user access and enforces Web SSO to the Tivoli Integrated Portal, and also to the Tivoli Common Reporting and other web applications across the enterprise.

WebSEAL appears as a web server to clients and appears as a web browser to the junctioned back-end servers it is protecting.

10.2.2 Value of Tivoli Integrated Portal

Web-based products built on the Tivoli Integrated Portal Infrastructure share a common user interface where you can launch applications and share information. Tivoli Integrated Portal helps the interaction and secure passing of data between Tivoli products through a common portal. You can launch from one application to another and within the same dashboard view to research different aspects of your managed enterprise. Tivoli Integrated Portal provides the following features:

- Services to support a console for individual products and for integrating multiple products.
- Views that span server instances, such as the Tivoli Netcool/OMNIbus ObjectServer and Tivoli Enterprise Portal Server
- A web-based user interface with inter-view messaging between products such that the data in, for example, a single tabulated report, can be gathered from multiple products.
- A common task navigator for multiple products that makes for convenient selection by the task and not necessarily by the product you need to use to perform that task.
- An integration point for Netcool products. Products that in previous versions were running on the Netcool GUI Foundation (NGF) now use Tivoli Integrated Portal, enabling greater interoperability among products that are built on this converged platform or are compatible with it.

10.2.3 Value to Tivoli Integrated Portal and WebSEAL Integrated Solution

Starting with Tivoli Integrated Portal 2.1, Tivoli Integrated Portal is a J2EE application running in embedded WebSphere Application Server so it can take advantage of and benefit from many of the functions that WebSEAL, provides other web applications. In this case, Tivoli Access Manager WebSEAL is used as a reverse proxy intercepting any incoming HTTP or HTTPS requests and ensuring that users are authenticated and authorized for the request to continue onto the necessary Tivoli Integrated Portal server. In addition, WebSEAL allows for single sign-on of the authenticated user where the credential is passed from WebSEAL to the downstream Tivoli Integrated Portal Server which ensures that the user does not need to reauthenticate at any time.

10.3 Integrating a Tivoli Integrated Portal environment with IBM Tivoli Access Manager WebSEAL

In this section, we discuss the steps required to integrate Tivoli Integrated Portal and IBM Tivoli Access Manager WebSEAL:

- Configuring for integration: Tivoli Integrated Portal
- Configuring for integration: Tivoli Access Manager WebSEAL

10.3.1 Configuring for integration: Tivoli Integrated Portal

The following key configuration steps are performed in the Tivoli Integrated Portal for this integration:

1. Enable Trust Association in the Tivoli Integrated Portal Console; click **Security** → **Secure administration, applications, and infrastructure**. See Figure 10-2.
2. Click **Web Security** → **Trust Association**.

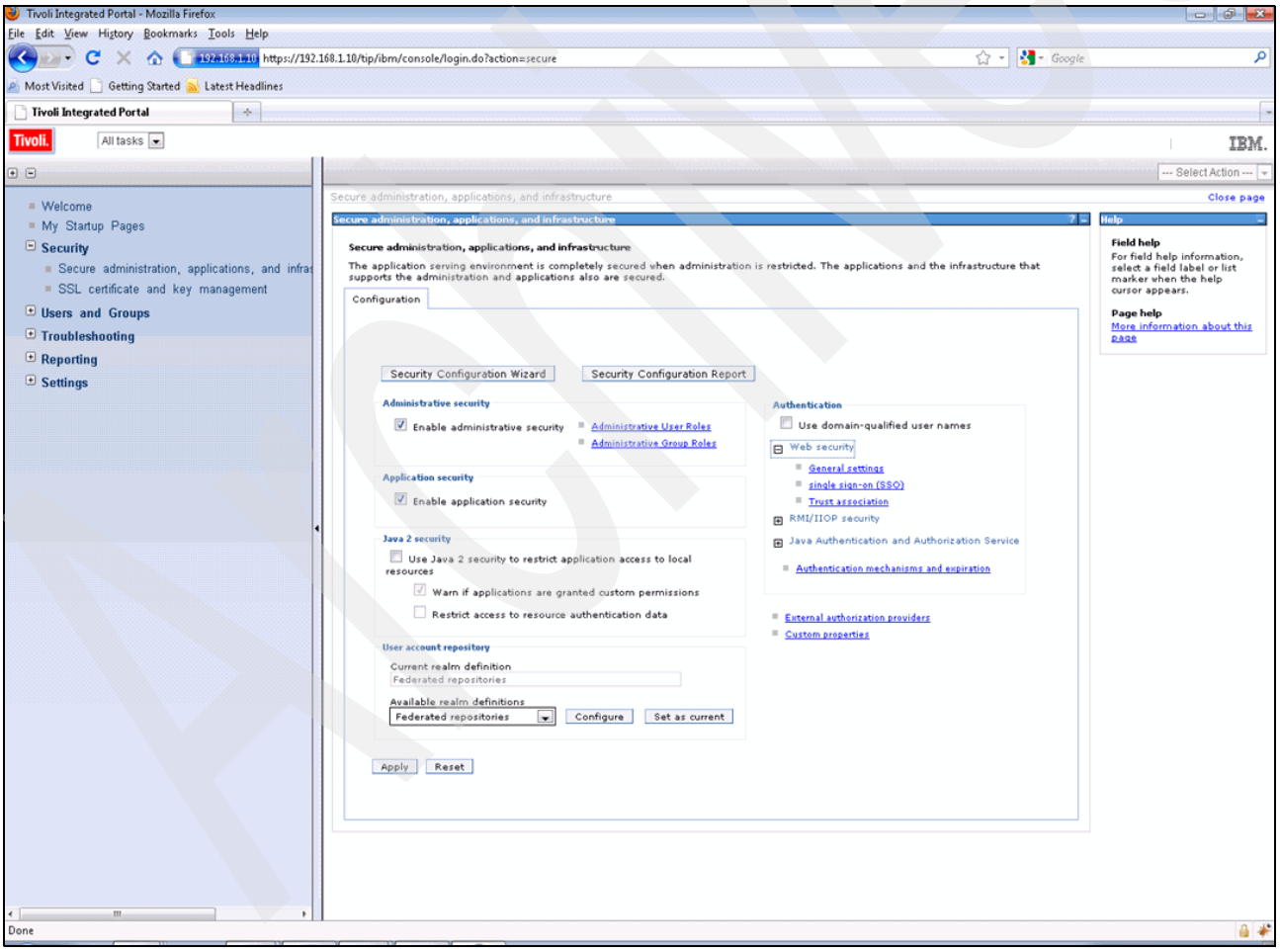


Figure 10-2 Enabling Trust Association

3. Enable trust association must be selected. If it is not already selected, select it and click **Apply**.
4. Save the configuration changes.

5. Add the extended trust association interceptor by starting at the Trust association page and click **Interceptors**. See Figure 10-3.

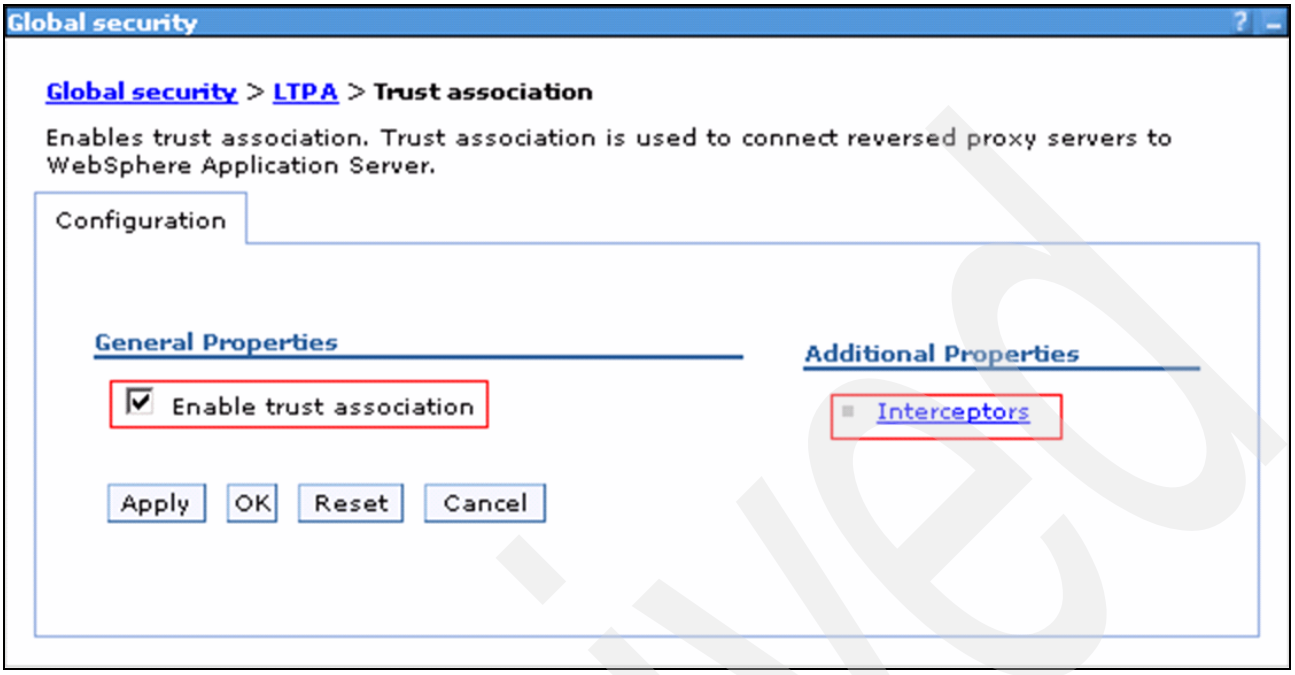


Figure 10-3 Interceptors

6. If `com.ibm.sec.authn.tai.TAMETai` class is not defined, click **New**. On the following window enter `com.ibm.sec.authn.tai.TAMETai` into the Interceptor class name field, and click A**pply**.
7. Save the configuration changes.
8. Next add custom properties to TAMETai by starting at the Interceptors page.
9. Select the `com.ibm.sec.authn.tai.TAMETai` Interceptor class.
10. Go to the Custom properties page.
11. Click **Custom Properties**. See Figure 10-4 on page 297.

Archived

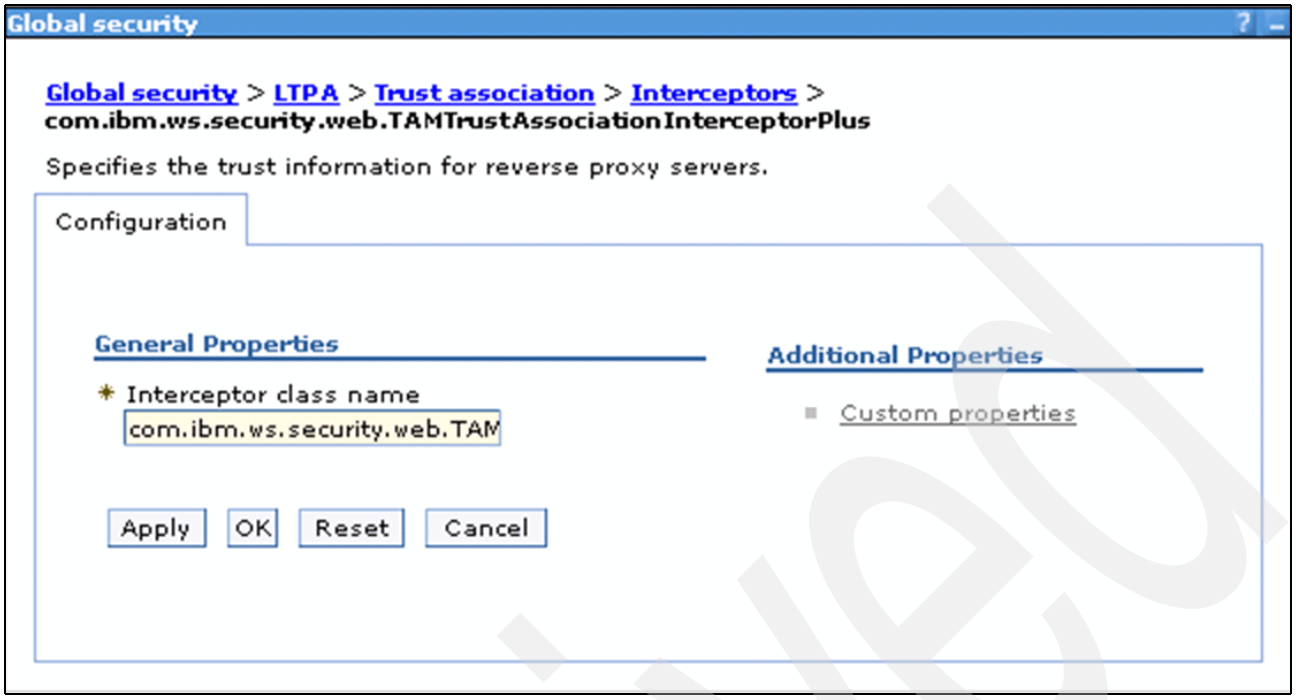


Figure 10-4 Custom Properties

12. For each required property that is not defined, click **New and** then enter information in the required Name and Value fields. Click **Apply**. See Figure 10-5.

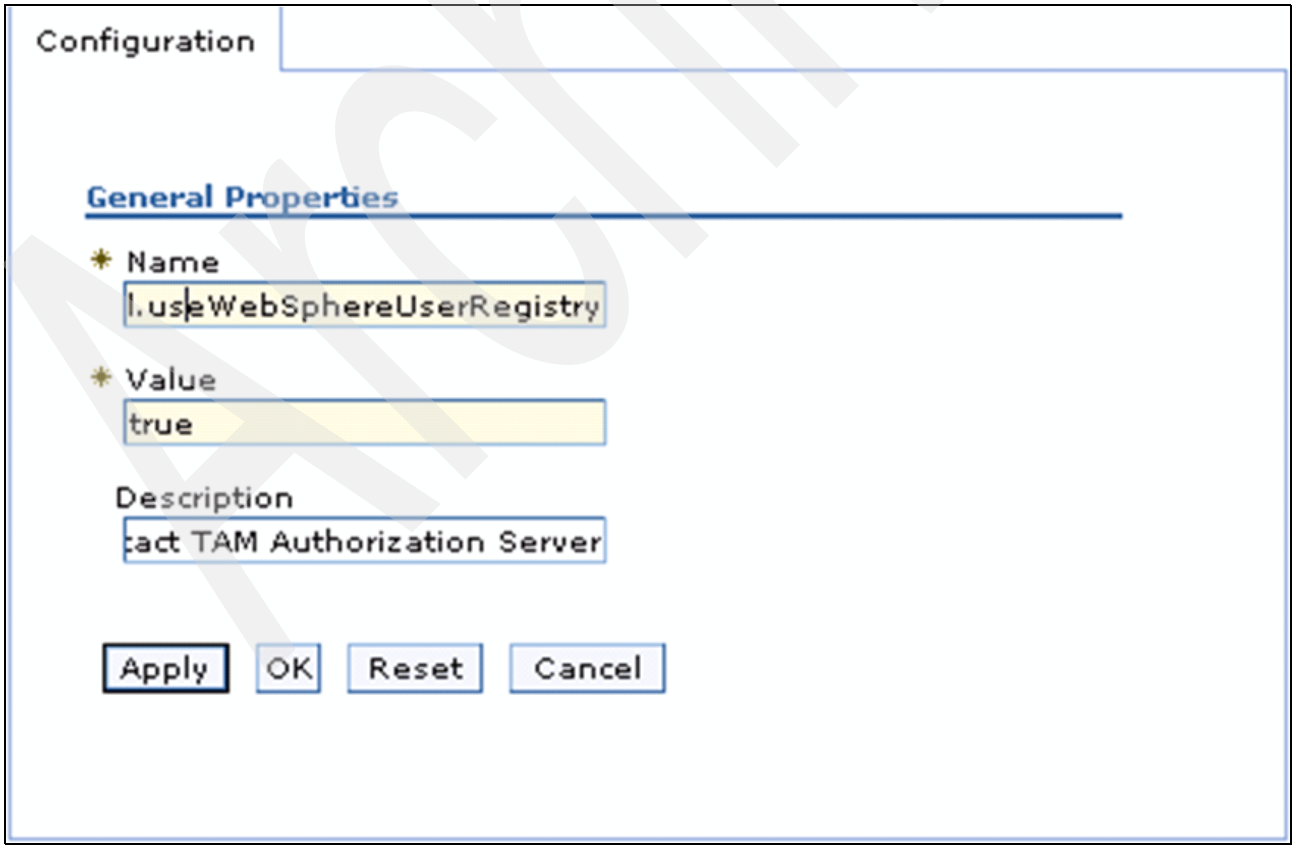


Figure 10-5 Define the properties

Archived

The results are in the Custom property definition as shown in Figure 10-6.



Figure 10-6 Custom property definition

13. If the custom property already exists but does not contain the correct Name, Value, or Description, select that property, change what is necessary, and click **Apply**.
14. Repeat for all required properties that are listed in the table in Figure 10-7 on page 299.

Archive

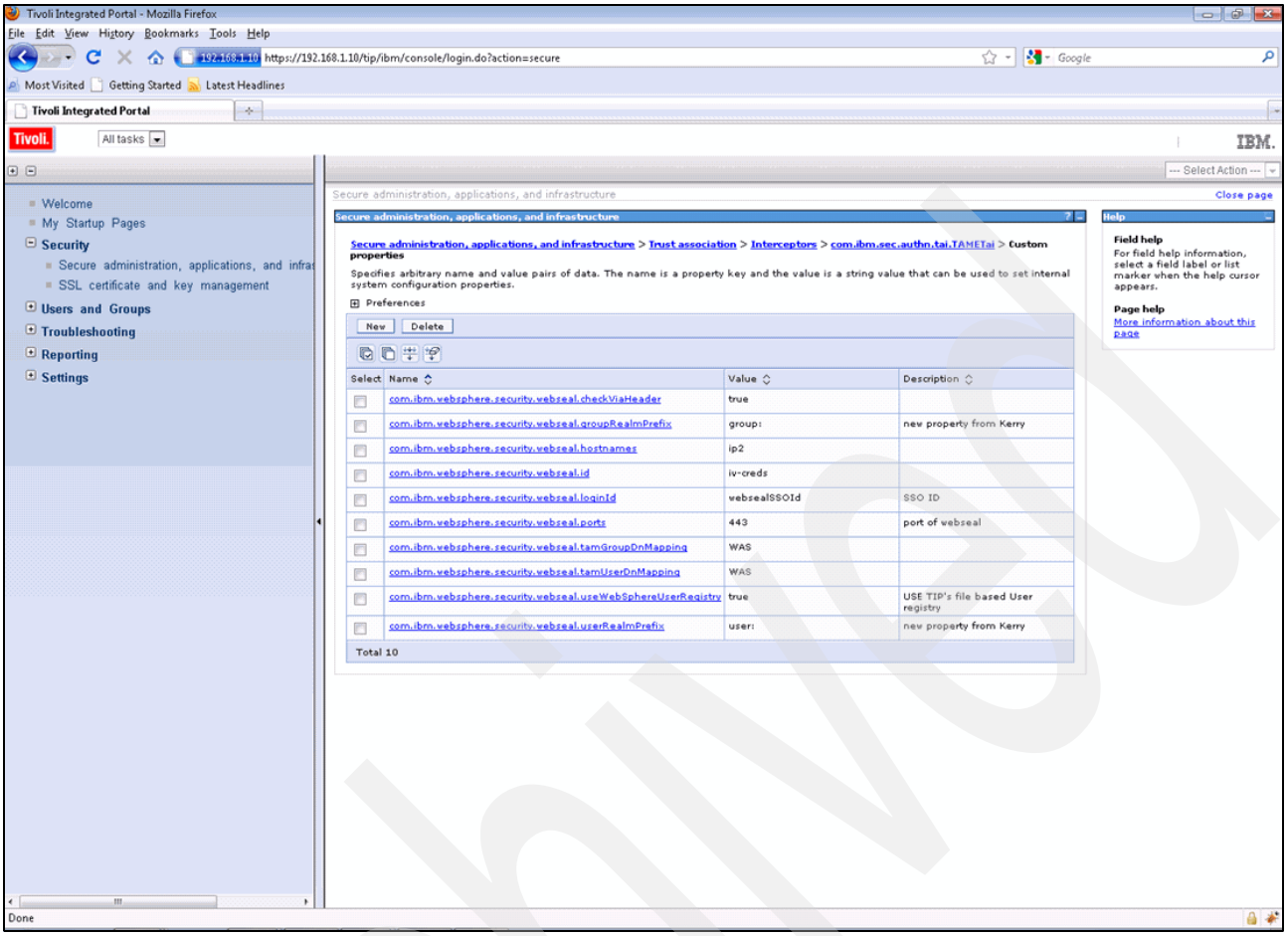


Figure 10-7 WebSEAL

15. Save the configuration changes.
16. Restart the Tivoli Integrated Portal Server.

10.3.2 Configuring for integration: Tivoli Access Manager WebSEAL

The key configuration steps in this section are performed in the Tivoli Access Manager WebSEAL for this integration.

For the eTAI to be able to accept requests from WebSEAL, several tasks must be performed on the WebSEAL server to ensure that an eTAI-targeted HTTP request is sent:

- Creating the junction with the required parameters
- Creating a trusted single sign on user
- Setting the dummy password in the configuration file

Archived

Perform the following steps:

1. Create the junction with the required parameters. The following two parameters are required by the eTAI:

   -b supply Ensures that WebSEAL inserts a basic authentication header in the HTTP request. The eTAI requires the dummy password in the BA header; the user name is not used. The header is used as part of the trust establishment, which is separate from the credentials originating from the client.

   -c iv-creds Ensures that WebSEAL passes the logged-in users credential in an iv-creds header in the HTTP request. The eTAI requires that this header exists or it will not handle the request. Other headers can also be passed such as iv-user but the iv-creds header must also be passed.

   The following command creates the junction:

   ```
   server task "webseal_instance_name" create -b supply -c iv-creds -t tcp -h
   "websphere_hostname" -p "websphere_app_port_number" "junction_name"
   ```

2. To create a trusted single sign on user the eTAI requires a user to exist in the user registry that will be used to authenticate trust. This user and its password become the central part of establishing trust between WebSEAL and Tivoli Integrated Portal.

   The value of the following custom property will be set to this user and the dummy password that is sent by WebSEAL in the inserted BA header will be set to this user's password.

   ```
   com.ibm.websphere.security.webseal.loginId
   ```

   If the following custom property is not set to true, this user must be created in the Tivoli Access Manager user registry.

   ```
   com.ibm.websphere.security.webseal.useWebSphereUserRegistry
   ```

   Creating a user can be done by using the pdadmin utility, as shown the following example:

   ```
   user create sso cn=sso,o=ibm,c=au sso sso ssopwd
   user modify ssouser account-valid yes
   ```

3. WebSEAL provides a mechanism for specifying the password that is passed in the basic authentication header of the HTTP request. This is done by setting the dummy password in the WebSEAL instance configuration file and using the **–b supply** junction parameter. The configuration file to update is located in your `<webseal_home>/etc` directory and has the following name:

   ```
   webseald-<instancename.conf>
   ```

4. Open this file, search for `basicauth-dummy-passwd`, and set the value of this property to be the password of the trusted single sign on user.

The scenario is now customized.

10.4 Summary

Figure 10-8 shows the quick summary of this integration.

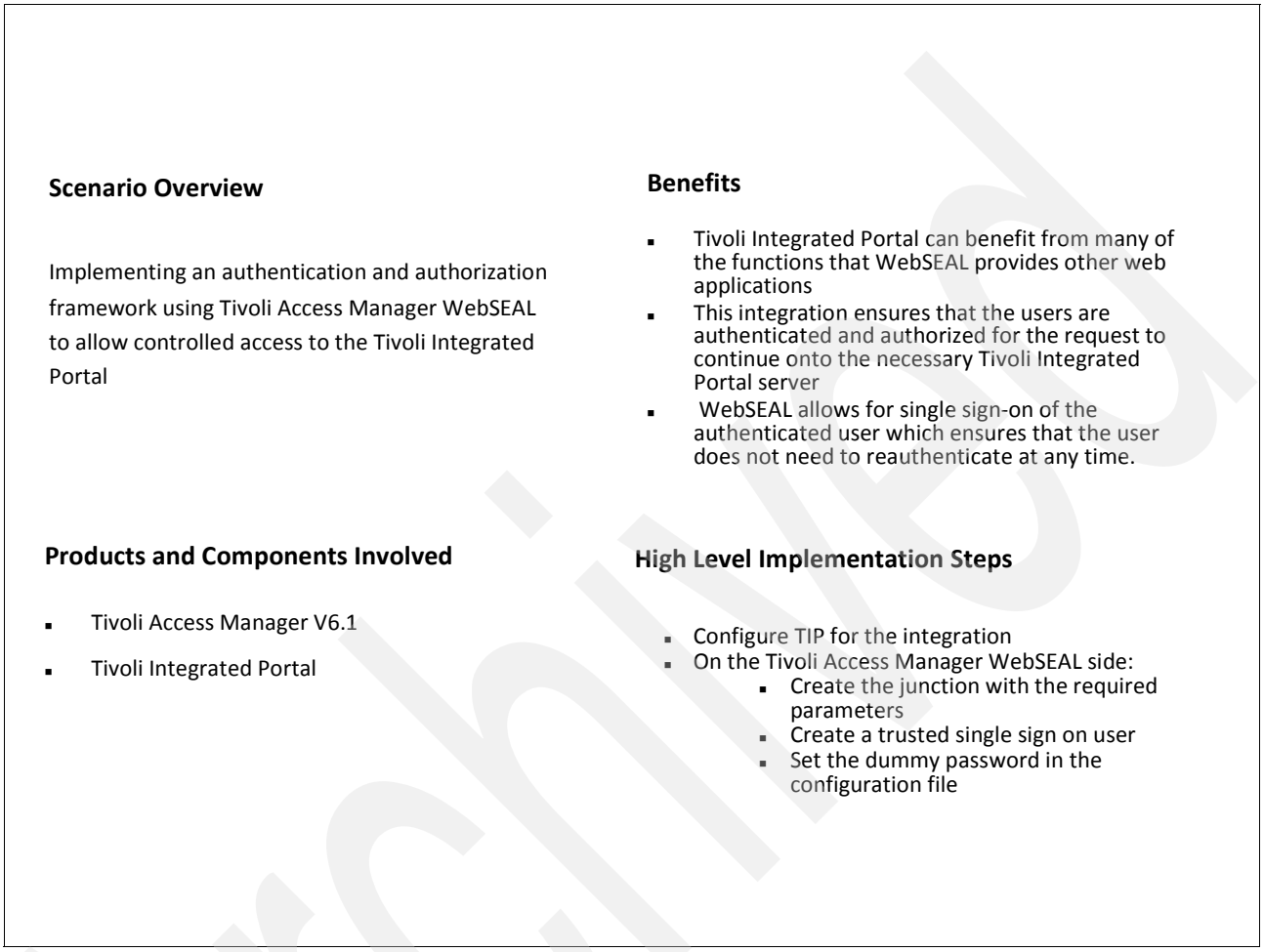


Figure 10-8 Quick summary

Archived

Archived

11

WebGUI launch to IBM Tivoli Monitoring

This chapter describes an integration that can be provided by using the WebGUI component of IBM Tivoli Netcool/OMNIbus with other Tivoli products in your environment. After adding this customization, WebGUI users can drill down for additional details to uncover the root cause of the events shown in the WebGUI. WebGUI users will be able to more quickly determine and resolve issues in their IT environments. This integration requires the development of several custom scripts and additional setup, as outlined in other sections of this book.

This chapter contains the following topics:

- 11.1, “Scenario description” on page 304
- 11.2, “Products involved” on page 304
- 11.3, “Benefits” on page 304
- 11.4, “Architectural diagram of the integration” on page 305
- 11.5, “Implementation steps” on page 305
- 11.6, “Scenario walk-through” on page 319
- 11.7, “Summary” on page 322

© Copyright IBM Corp. 2011. All rights reserved.

11.1 Scenario description

This scenario demonstrates the launch-in-context integration between the IBM Tivoli Monitoring 6.2.2 and IBM Tivoli Netcool/OMNIbus Web GUI 7.3 products.

This chapter accomplishes the following tasks for this scenario:

- Install the products used in Integration.
- Outline the steps involved in the setup of Event Forwarding.
- Confirm through the IBM Tivoli Netcool/OMNIbus Web GUI 7.3's Active Event List (AEL) portlet that our new event fields are being populated and accessible through AEL scripts.

  IBM Tivoli Netcool/OMNIbus alerts.status fields and EIF Probe rules are updated to allow the communication between IBM Tivoli Monitoring 6.2.2 and IBM Tivoli Netcool/OMNIbus Web GUI 7.3.
- Define a new AEL tool to use to launch into the Tivoli Enterprise Portal server UI, which is associated with the IBM Tivoli Monitoring 6.2.2 that sends the event, passing to the script the parameters that open the workspace that is associated with the resource in the event.

11.2 Products involved

The products and product components involved in this scenario are as follows:

- IBM Tivoli Monitoring 6.2.2
- IBM Tivoli Netcool/OMNIbus Web GUI 7.3
- IBM Tivoli Netcool/OMNIbus 7.3
- IBM Tivoli Netcool EIF Probe 9.0
- IBM Tivoli Monitoring Situation Update Forwarder (SUF) Component
- LDAP server on Microsoft Active Directory

11.3 Benefits

Benefits of this integration are as follows:

- Ability to drill down for additional details to uncover the root cause of the events.
- More quickly determine and resolve issues in the IT environment.
- Visualize event data efficiently with the capabilities of consoles.

Archived

11.4 Architectural diagram of the integration

Figure 11-1 shows the architectural diagram of the integration.

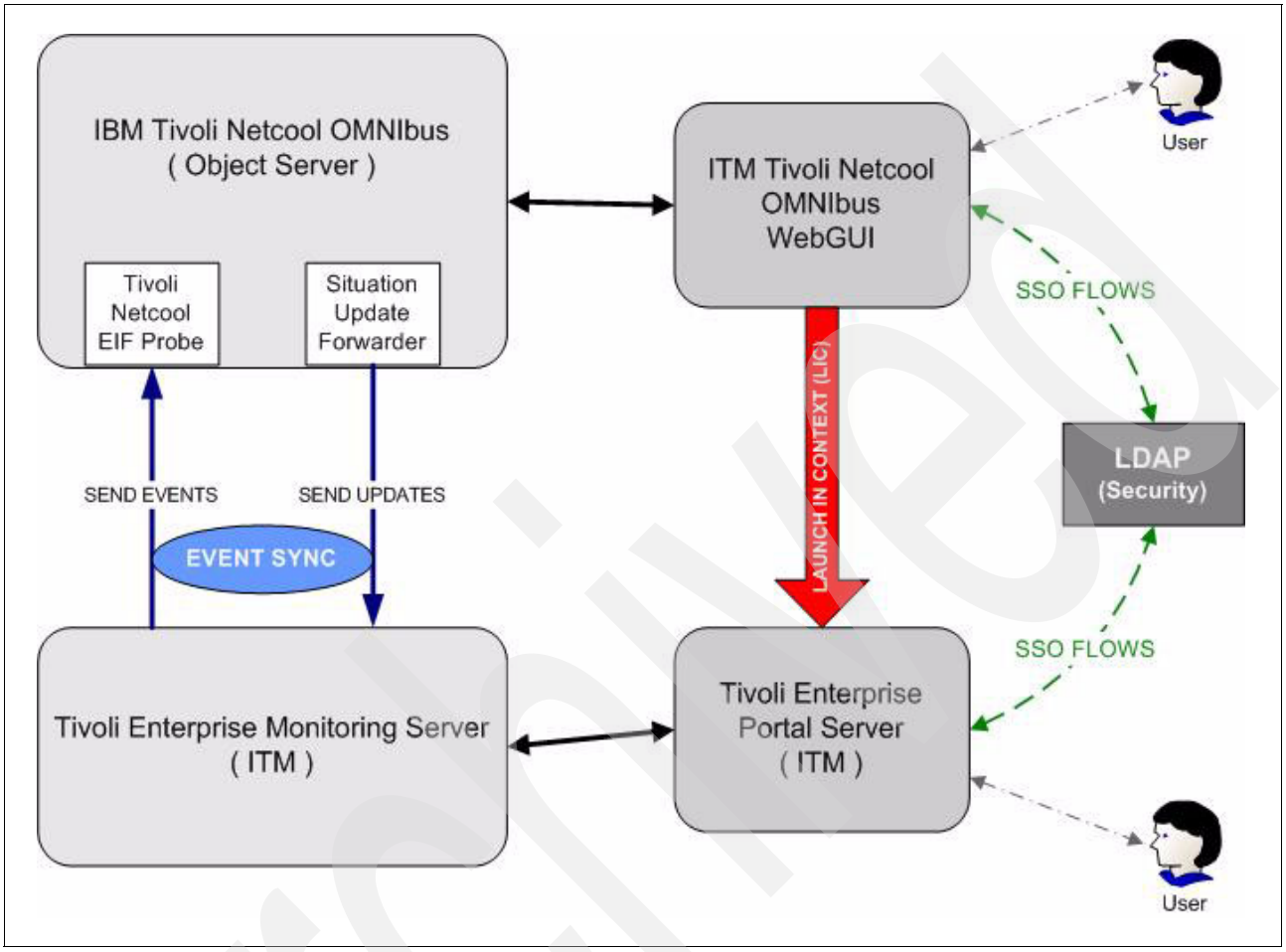


Figure 11-1 Overview of IBM Tivoli Netcool/OMNIbus Web GUI integration with Tivoli Enterprise Portal

11.5 Implementation steps

This section describes the implementation steps of this scenario.

11.5.1 Product installations

A description of the products that are involved in this scenario are provided in this section. Web addresses for information about installing the products are also provided.

IBM Tivoli Monitoring 6.2.2

IBM Tivoli Monitoring monitors and manages system and network applications for a variety of operating systems, tracks the availability and performance of your enterprise system, and provides reports to track trends and troubleshoot problems. IBM Tivoli Monitoring is the launch-into product from the IBM Tivoli Netcool/OMNIbus Web GUI 7.3 used to provide additional details.

Instructions for installing and configuring the IBM Tivoli Monitoring environment are in the *IBM Tivoli Monitoring Installation and Setup Guide*:

http://publib.boulder.ibm.com/infocenter/tivihelp/v15r1/topic/com.ibm.itm.doc_6.2.2/itm_install.htm

IBM Tivoli Netcool/OMNIbus 7.3

Tivoli Netcool/OMNIbus is a service level management (SLM) system that delivers real-time, centralized monitoring of complex networks and IT domains. Tivoli Netcool/OMNIbus can consolidate information from different domain-limited network management platforms in remote locations. Tivoli Netcool/OMNIbus tracks alert information in a high-performance, in-memory database, and presents information of interest to specific users through filters and views that can be configured individually. Tivoli Netcool/OMNIbus has automation functions that can perform intelligent processing on managed alerts.

IBM Tivoli Netcool/OMNIbus integrates with IBM Tivoli Monitoring solution by using its probe component. Tivoli Enterprise Monitoring Server component of IBM Tivoli Monitoring sends the alarms that are created by situations and Tivoli Netcool/OMNIbus EIF Probe receives them as the integration and creates events in Object Server.

The scenario, described in this chapter, can help improve the IBM Tivoli Netcool/OMNIbus, IBM Tivoli Monitoring integration by enabling communication between both web consoles. It is possible to open Tivoli Enterprise Console from IBM Tivoli Netcool/OMNIbus Web GUI by using a customized script within the context of the event.

Installation and configuration details are at the following web address:

http://publib.boulder.ibm.com/infocenter/tivihelp/v8r1/topic/com.ibm.netcool_OMNIbus.doc_7.3.0/omnibus/wip/install/concept/omn_ins_installingupgradinguninstalling.html

IBM Tivoli Netcool/OMNIbus Web GUI 7.3

IBM Tivoli Netcool/OMNIbus Web GUI is the user console of Tivoli Netcool/OMNIbus product. It is the web-based user interface that lists and operates on the events that are stored in Object Server.

An important entity of the Web GUI product is the Active Event List (AEL), which is a list of events that the user can configure to monitor the environment. With the help of tools, the user can work on the stored events and take actions.

This scenario adds a new action option to the alerts menu for Active Event Lists. The user can then open the Tivoli Enterprise Portal console that is associated with the event from the IBM Tivoli Netcool/OMNIbus Web GUI.

Installation and configuration details are at the following web address:

http://publib.boulder.ibm.com/infocenter/tivihelp/v8r1/topic/com.ibm.netcool_OMNIbus.doc_7.3.0/webtop/wip/task/web_ins_installing.html

IBM Tivoli Netcool EIF Probe 9.0

To integrate IBM Tivoli Monitoring and IBM Tivoli Netcool/OMNIbus, Tivoli Netcool/OMNIbus EIF Probe is needed. EIF Probe is a component of Tivoli Netcool/OMNIbus.

Archive

Note: Starting with Tivoli Netcool/OMNIbus V7.3, Tivoli Netcool EIF Probe is installed with the same installer as the core Tivoli Netcool/OMNIbus product. In previous versions of Tivoli Netcool/OMNIbus, the EIF Probe had a different installation package and was installed on Tivoli Netcool/OMNIbus as a patch.

You configure IBM Tivoli Monitoring Tivoli Enterprise Monitoring Server to send situation alarms to the EIF Probe, and the EIF Probe receives and processes those alarms. Tivoli EIF Probe can do the event field matching with the help of rule files, and create an event for each of them in Object Server.

You can install Tivoli Netcool/OMNIbus EIF Probe on the same server with Tivoli Netcool/OMNIbus, and on a separate server.

Installation and configuration details are at the following web address:

http://publib.boulder.ibm.com/infocenter/tivihelp/v8r1/topic/com.ibm.netcool_OMNIbus.doc/probes/tivoli_eif/common/topicref/pro_install_post73.html

IBM Tivoli Monitoring: Situation Update Forwarder

Situation Update Forwarder, known as SUF, is a tool of IBM Tivoli Monitoring that is used to send the updates back to IBM Tivoli Monitoring from Tivoli Netcool/OMNIbus. Acknowledgement and status change operations are the possible operations performed on the stored events at WebGUI.

SUF is installed between IBM Tivoli Monitoring and Tivoli Netcool/OMNIbus, waits for the updates coming from Object Server, and sends them back to IBM Tivoli Monitoring. With the help of Tivoli Netcool/OMNIbus EIF Probe, Situation Update Forwarder supports the bidirectional event and data flow between IBM Tivoli Monitoring and IBM Tivoli Netcool/OMNIbus.

SUF installation package is included with the IBM Tivoli Monitoring v6.2.2 Tools package. SUF must be installed on the same server as the IBM Tivoli Netcool/OMNIbus ObjectServer. To perform the installation and configuration, follow the instructions at the following address:

http://publib.boulder.ibm.com/infocenter/tivihelp/v15r1/topic/com.ibm.itm.doc_6.2.2fp2/omni_esync_install.htm#omni_esync_install

11.5.2 Event integration

To integrate IBM Tivoli Netcool/OMNIbus with IBM Tivoli Monitoring bidirectionally, implement the following flows:

- Events from IBM Tivoli Monitoring to IBM Tivoli Netcool/OMNIbus (with the help of the Tivoli EIF Probe)
- Event updates from IBM Tivoli Netcool/OMNIbus to IBM Tivoli Monitoring (with the help of Situation Update Forwarder)

Event flow from IBM Tivoli Monitoring to IBM Tivoli Netcool/OMNIbus

Tivoli Netcool/OMNIbus is the main event destination for various event sources. Sources such as IBM Tivoli Monitoring can send alarms they produce to OMNIbus Object Server. This event integration enhances the event management capability of the users.

For the integration of IBM Tivoli Monitoring and OMNIbus, the events that are created in IBM Tivoli Monitoring should be forwarded to the Tivoli Netcool/OMNIbus.

On the Tivoli Netcool/OMNIbus side, the forwarded events are received and processed by Tivoli Netcool/OMNIbus EIF Probe. The probe performs field matching to create new events in the Object Server for the coming IBM Tivoli Monitoring alarms.

To implement the event flow from IBM Tivoli Monitoring to IBM Tivoli Netcool/OMNIbus, use the following steps:

- Configure the IBM Tivoli Monitoring to forward events.
- Install and configure the Tivoli Netcool/OMNIbus EIF Probe to receive and process the events.

Configuring Tivoli Enterprise Monitoring Server is located at the following address:

http://publib.boulder.ibm.com/infocenter/tivihelp/v15r1/topic/com.ibm.itm.doc_6.2.2/itm_install457.htm#omnibus_install

Configuring the Tivoli Netcool/OMNIbus EIF Probe is located at the following address:

http://publib.boulder.ibm.com/infocenter/tivihelp/v8r1/topic/com.ibm.netcool_OMNIbus.doc/probes/tivoli_eif/tivoli_eif/wip/concept/tveif_intro.html

Event updates flow from IBM Tivoli Netcool/OMNIbus to IBM Tivoli Monitoring

When the alarms in Tivoli Enterprise Monitoring Server are sent to IBM Tivoli Netcool/OMNIbus, the operators can check and work on them by using Tivoli Netcool/OMNIbus Web GUI product. They can use the tools in the Alerts menu on the console to operate on those alarms. To send updates on events back to the IBM Tivoli Monitoring, you need the Situation Update Forwarder (SUF) tool.

SUF is included with the IBM Tivoli Monitoring 6.2.2 Tools package and must be installed on the server where Tivoli Netcool/OMNIbus resides.

During implementation of the SUF, Tivoli Netcool/OMNIbus EIF Probe and Object Server components are updated with the SQL and rules files from the installation package of SUF.

To update Object Server to create the required fields, triggers, and procedures, run the SQL files that are included with the SUF package.

To configure the Tivoli Netcool/OMNIbus EIF Probe to collect and match correct information, update the rules file with the one that is included with the SUF installation package.

Installation and configuration details are at the following address:

http://publib.boulder.ibm.com/infocenter/tivihelp/v15r1/index.jsp?topic=/com.ibm.itm.doc_6.2.2fp2/omnibus_install.htm

11.5.3 Event flow check

After implementing the bidirectional integration of IBM Tivoli Monitoring and Tivoli Netcool/OMNIbus, a good practice is to check the event and data flow between them before implementing the launch-in-context integration of Tivoli Netcool/OMNIbus Web GUI and Tivoli Enterprise Portal console of IBM Tivoli Monitoring.

Archived

Perform the following steps:

1. Open the Tivoli Enterprise Portal console of IBM Tivoli Monitoring to see whether agents are collecting alarms or not, as shown in Figure 11-2.

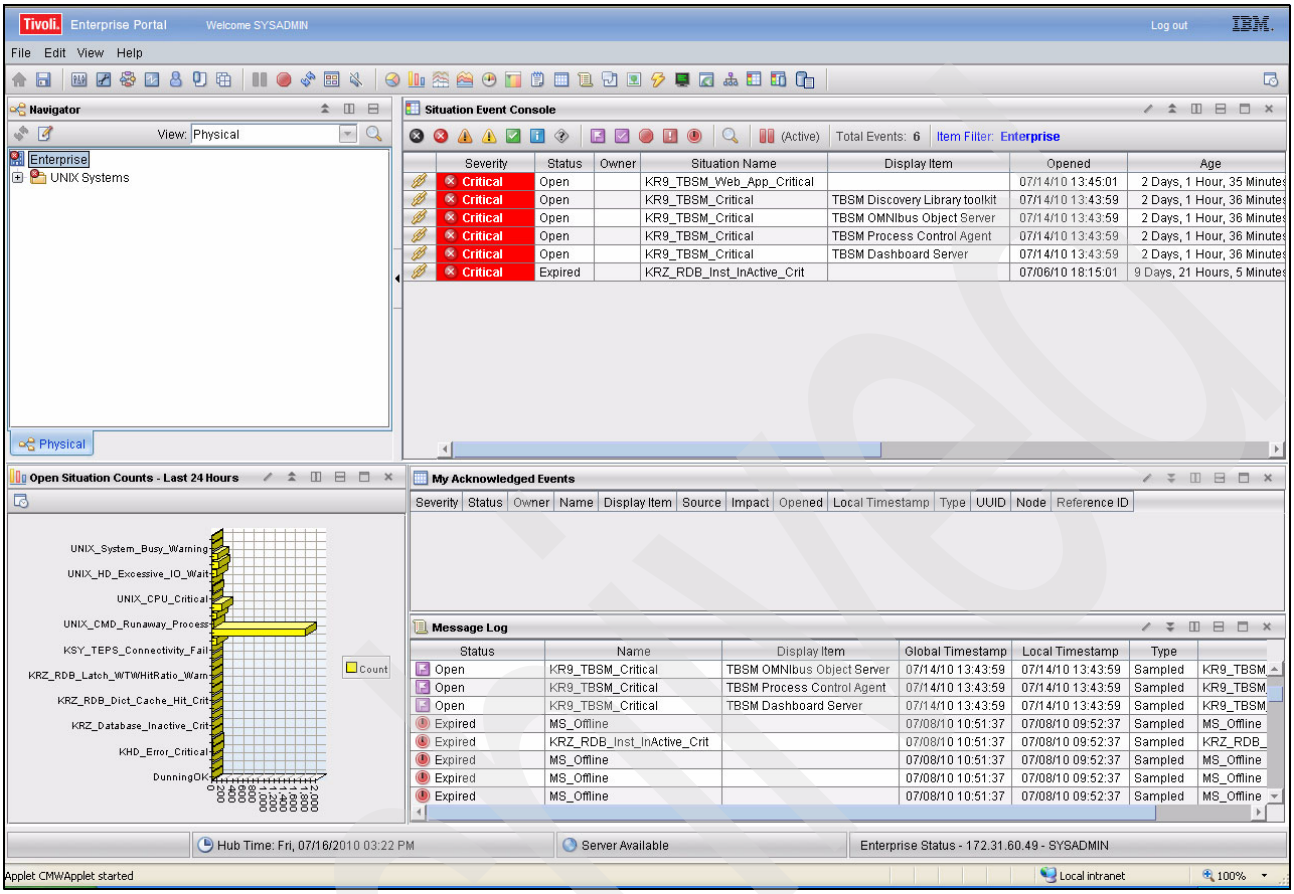


Figure 11-2 Events listed in Tivoli Enterprise Portal console of IBM Tivoli Monitoring

2. If you have correctly configured the IBM Tivoli Monitoring to forward alarms to Tivoli Netcool/OMNIbus EIF Probe, and Tivoli Netcool/OMNIbus EIF Probe to process the incoming alarms, you see the events in Tivoli Netcool/OMNIbus Web GUI also.

See Figure 11-3 on page 310.

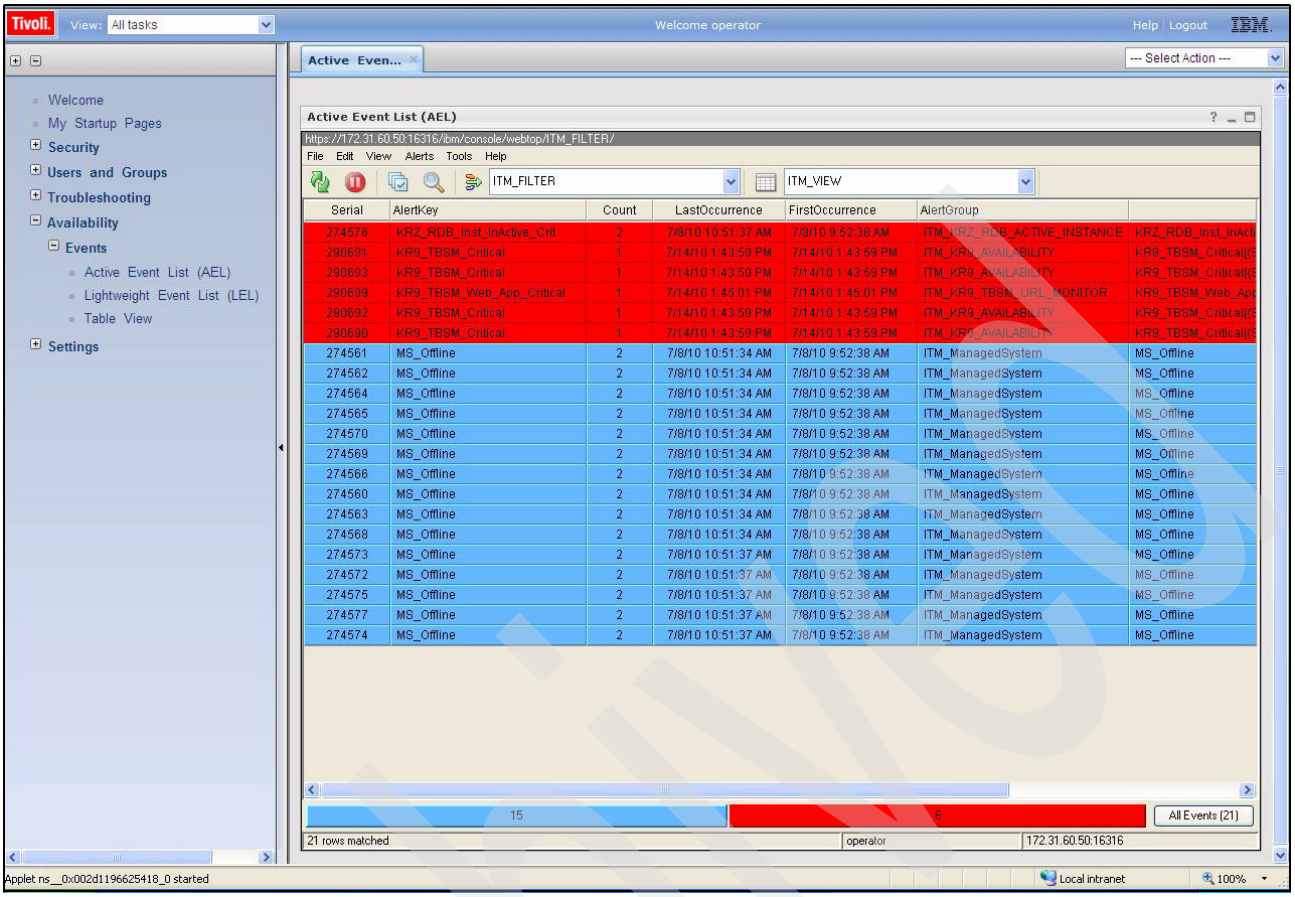


Figure 11-3 Events from IBM Tivoli Monitoring are listed in Tivoli Netcool/OMNIbus Web GUI

To apply the script that is described in this scenario, the Node and ITMHostname fields must be populated with the correct information. To do make sure the fields are correct, apply the Situation Update Forwarder configuration steps as described in 11.5.2, “Event integration” on page 307. The important step is to run the SQL files of the SUF package correctly and replace the `tivoli_eif.rules` file of the probe with the one that is included with the SUF package, and run the probe with the new rules file. The EIF Probe working with the updated rule file is able to populate the Node and ITMHostname fields correctly.

You can check the information details of an event by double-clicking it. Check the information in the Node and ITMHostname fields. Node is the name of the managed system affected by the event. ITMHostname is the host name of the Hub TEMS (Figure 11-4 on page 311).

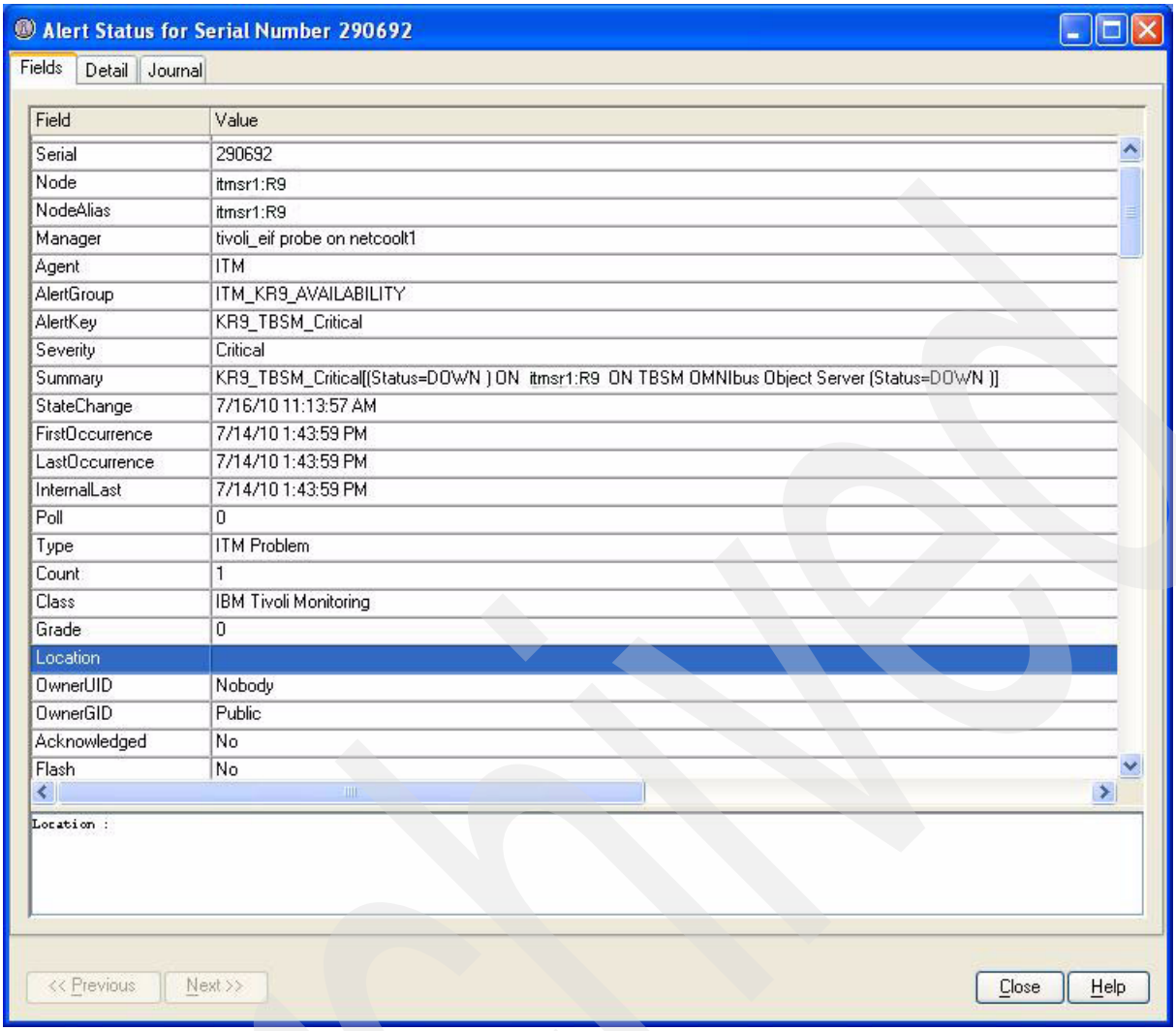


Figure 11-4 Information Details of an event in Web GUI

11.5.4 Implementing launch-in-context integration between user consoles

In this scenario, we implement launch-in-context integration of Tivoli Netcool/OMNIbus Web GUI and Tivoli Enterprise Portal console of IBM Tivoli Monitoring. In 11.5.1, “Product installations” on page 305 and 11.5.2, “Event integration” on page 307 we have described the basic steps including installation, configuration and event synchronization of the products. In this topic, the main purpose of the scenario will be described.

In Tivoli Enterprise Portal console, all monitored systems have workspaces. Operators can see detailed information for the alarms that occurred with the help of available charts and data configurations. However, Tivoli Netcool/OMNIbus stores only the event data sent by the source; sometimes, to diagnose a problem, operators must launch the Tivoli Enterprise Portal console to see more details related to the event.

By using launch-in-context integration between IBM Tivoli Netcool/OMNIbus Web GUI and IBM Tivoli Monitoring Tivoli Enterprise Portal console, you can launch into the Tivoli Enterprise Portal console in context of the managed system that is affected by the event. This new option is added to the Alerts menu that the operator can use from Tivoli Netcool/OMNIbus Web GUI console to open the workspace that is associated with the

managed system name of the event in IBM Tivoli Monitoring Tivoli Enterprise Portal console. The integration increases the ease of use of the monitoring products for the operators.

To implement the launch-in-context integration of consoles, perform the following steps:

1. Log in to the Tivoli Netcool/OMNIbus WebGUI console as a user that has ncw_admin role.

   You may check the user and role management of the Web GUI at the following address:

   http://publib.boulder.ibm.com/infocenter/tivihelp/v8r1/topic/com.ibm.netcool_OMNIbus.doc_7.3.0/webtop/wip/concept/web_adm_webtopusersrolesgroups.html

2. In the left navigation menu, click **Administration** → **Event Management Tools** → **Tool Creation** (Figure 11-5).

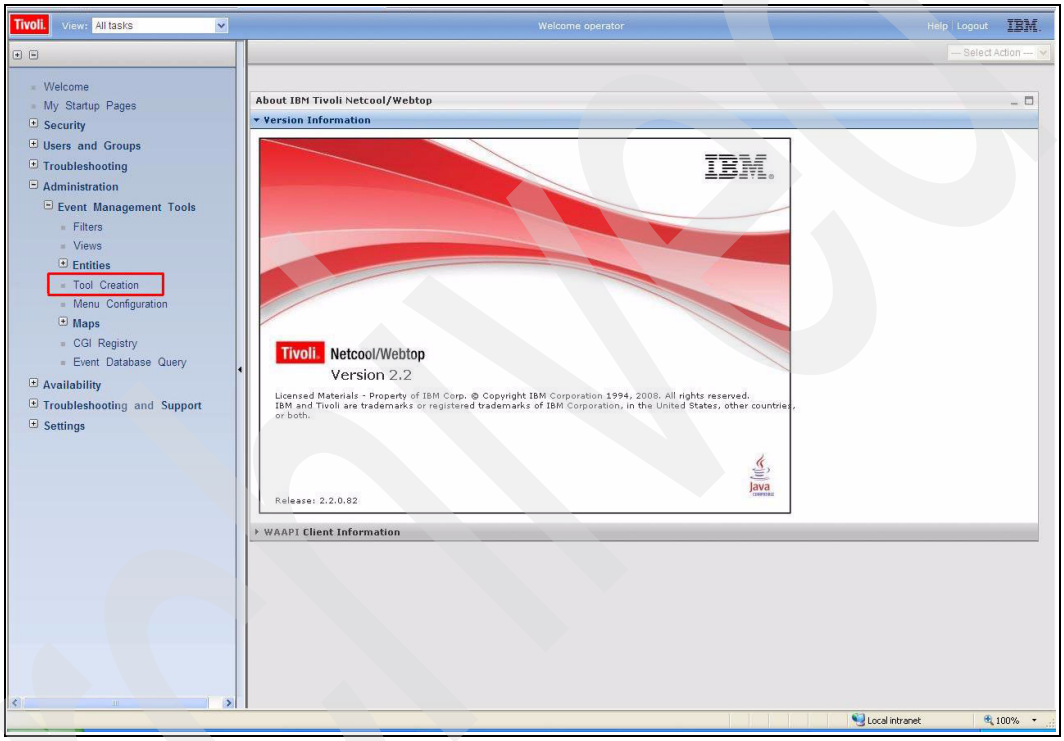


Figure 11-5 Tool Creation page at WebGUI console

> **Note:** If you do not see this page at the left navigation menu, check the roles of the user you have logged in with.

3. Click **New Tool** to create a new tool for the events. (Figure 11-6 on page 313).

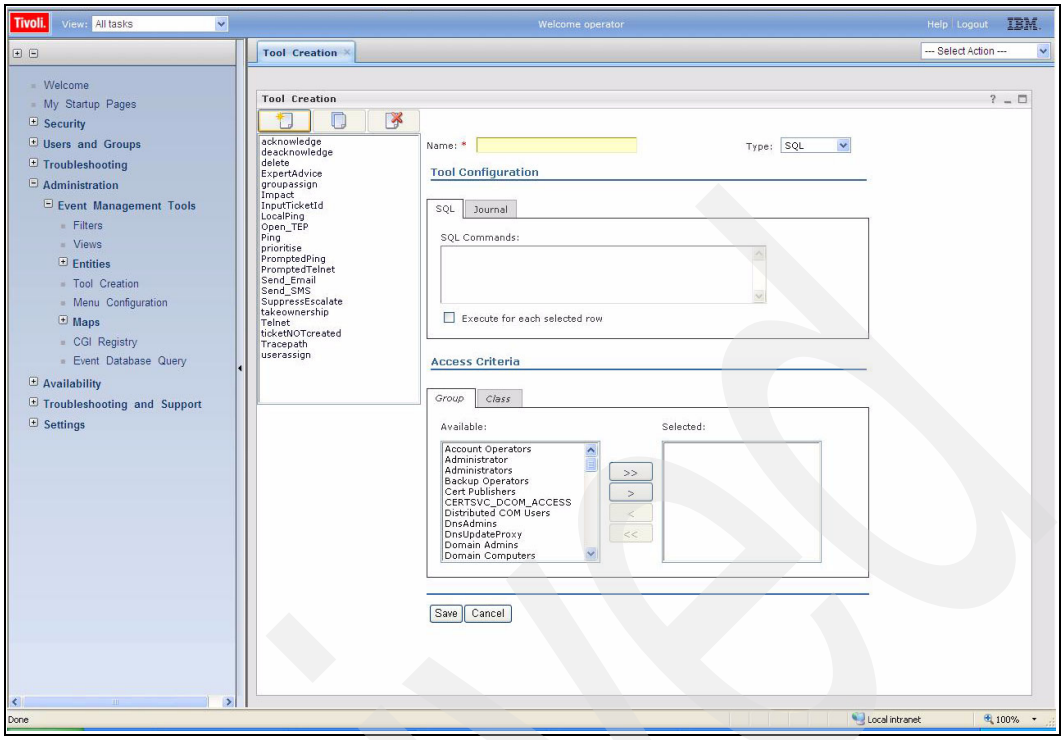


Figure 11-6 Adding new Script Tool for launch-in-context integration of the console

4. Select type **Script** and enter a meaningful name such as Open_TEP.
5. For the Script Commands area, use the script that is suitable for your environment, single or multiple Tivoli Enterprise Portal Server:
   - Single Tivoli Enterprise Portal Server

     If you have a single Tivoli Enterprise Portal Server in the environment and Tivoli Enterprise Portal Server and Tivoli Enterprise Monitoring Server components of IBM Tivoli Monitoring are running on the *same* server, add the script command that is shown in Example 11-1.

     Important: In the examples, we use HTTP and port number 1920. If, in your environment, you use HTTPS and a different port number, you must customize the scripts to reflect that environment.

     Example 11-1 Single Tivoli Enterprise Portal Server: TEPS and TEMS on the same server

```
var strsystem={@Node};
var ustrsystem = strsystem.replace(/'/g, "");
var strtems={@ITMHostname};
var ustrtems = strtems.replace(/'/g, "");
window.open("http://"+
ustrtems+":1920///cnp/kdh/lib/cnp.html?managed_system_name=" +ustrsystem);
```

     If you have single Tivoli Enterprise Portal Server in the environment, but Tivoli Enterprise Portal Server and Tivoli Enterprise Monitoring Server components of IBM

Tivoli Monitoring are running on *separate* servers, add the script command, shown in Example 11-2.

Example 11-2 Single Tivoli Enterprise Portal Server: TEPS and TEMS on separate servers

```
var strsystem={@Node};
var ustrsystem = strsystem.replace(/'/g, "");
window.open("http://ithilien.tivlab.austin.ibm.com:1920///cnp/kdh/lib/cnp.ht
ml?managed_system_name=" +ustrsystem);
```

In the script, *<tep_host >* is the IP address or the host name of the system where IBM Tivoli Monitoring Tivoli Enterprise Portal Server is running.

Important: In IBM Tivoli Monitoring 6.2.2 FP1, IBM Tivoli Monitoring added the capability to send private situation events directly from an agent to IBM Tivoli Netcool/OMNIbus. For these types of events, you cannot launch back to the Tivoli Enterprise Portal because the Tivoli Enterprise Portal might have no knowledge of the agent (for example, the agent might be autonomous and not connected to a Tivoli Enterprise Monitoring Server and Tivoli Enterprise Portal Server). To ignore events from an agent for launch-in-context, the script command would need to check the `@ITMApplLabel` OMNIbus variable (which only exists if you are using Tivoli Monitoring 6.2.2 FP1 or later) and skip the launch-in-context if this label is set to `"A:P:S"` or `"A:P:L"`.

– Multiple Tivoli Enterprise Portal Servers

If you have multiple Tivoli Enterprise Portal Servers in the environment that is connected to separate Tivoli Enterprise Monitoring Server, add the script commands, shown in Example 11-3.

Important: In the example, we use HTTP and port number 1920. If, in your environment, you use HTTPS and a different port number, you must customize the scripts to reflect that environment.

Also, make sure that all monitoring servers are sending the alarms to the same Object Server.

Example 11-3 Multiple Tivoli Enterprise Portal Server servers

```
var strsystem={@Node};
var ustrsystem = strsystem.replace(/'/g, "");
var strtems={@ITMHostname};
if( strtems == "reg-10.tivlab.austin.ibm.com" )
{ window.open("http://ithilien.tivlab.austin.ibm.com:1920///cnp/kdh/lib/cnp.html?man
aged_system_name=" +ustrsystem);}
else if( strtems == "nc185071.tivlab.austin.ibm.com")
{ window.open("http://nc185071.tivlab.austin.ibm.com:1920///cnp/kdh/lib/cnp.html?man
aged_system_name=" +ustrsystem);}
```

This script uses the following definitions:

<tems_host1> This is the IP address or the host name of the server where first Tivoli Enterprise Monitoring Server exists. Make sure to use the host name or IP address in the @ITMHostname field of the event.

<tep_host1> This is the IP address or the host name of the server where first Tivoli Enterprise Portal Server exists.

<tems_host2> This is the IP address or the host name of the server where second Tivoli Enterprise Monitoring Server exists. Be sure whether host name or IP address is used in @ITMHostname field of the event and use the same.

<tep_host2> This is the IP address or the host name of the server where the second Tivoli Enterprise Portal Server exists.

6. After writing the script command that matches your environment, click **Save** to save the new tool (Figure 11-7).

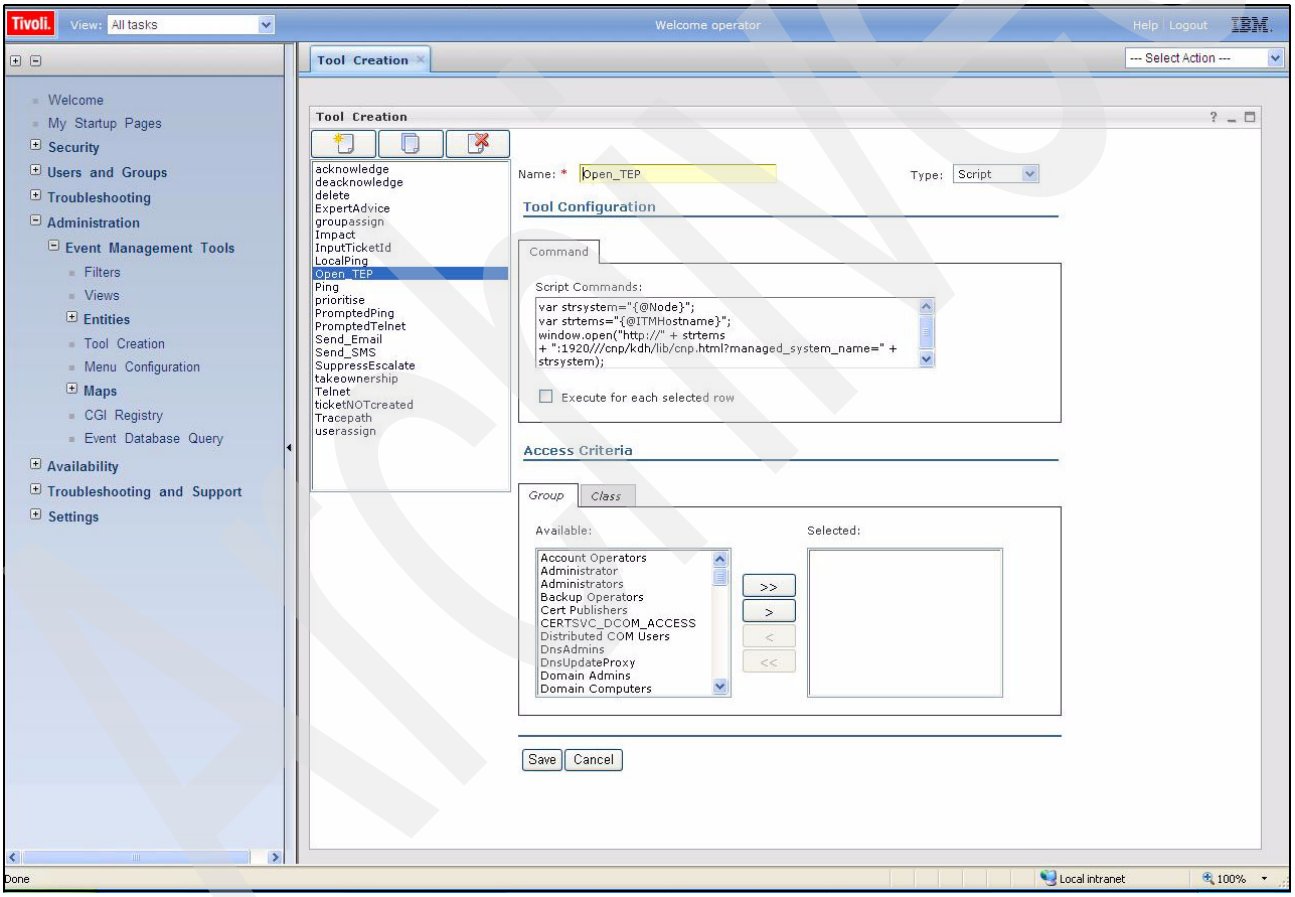


Figure 11-7 Script to open Tivoli Enterprise Portal console from WebGUI of Tivoli Netcool/OMNIbus

7. In the left navigation menu, click **Administration** → **Event Management Tools** → **Menu Configuration**. See Figure 11-8.

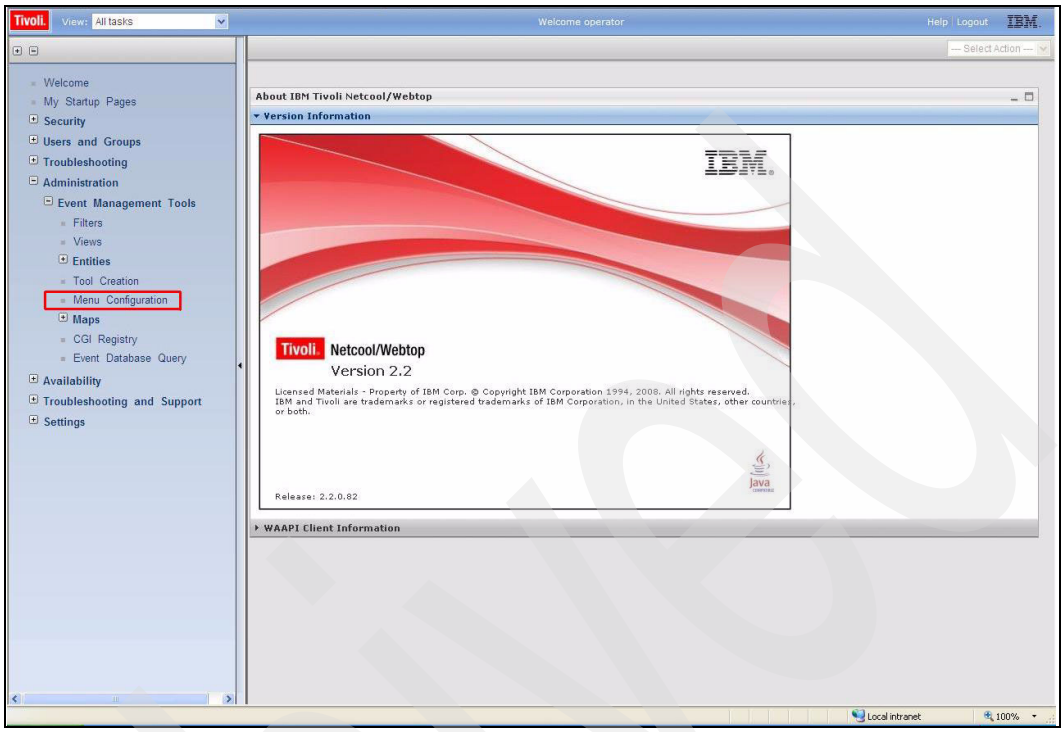


Figure 11-8 Menu Configuration page of Web GUI

8. Add the newly created Open_TEP tool in the Alerts menu as a new option. At the Available menus table, you see that the tool lists that were created before. Select **Alerts** and click **Modify** to update it. See Figure 11-9 on page 317.

Archive

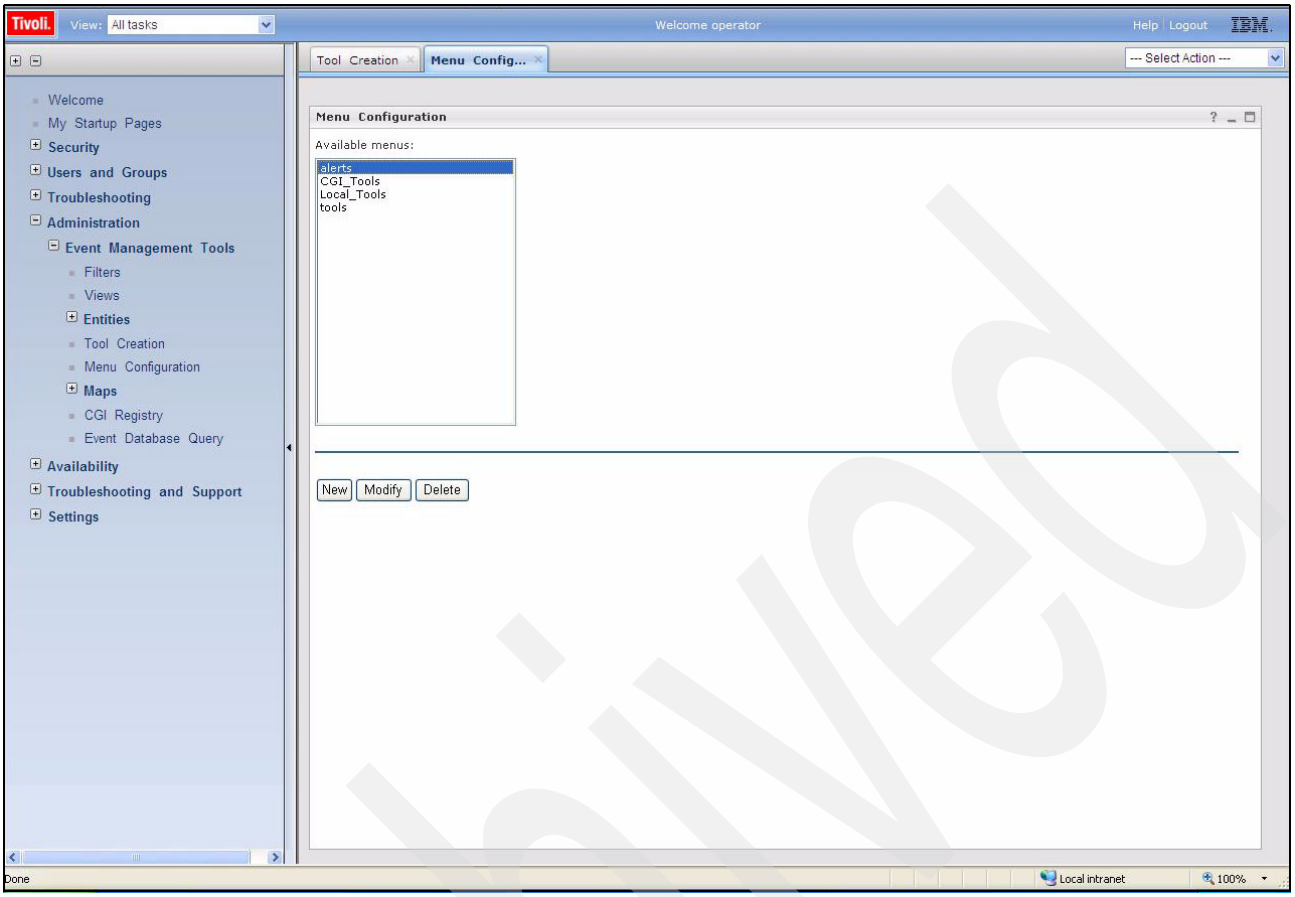


Figure 11-9 Update alerts menu to add new tool as new option

9. The available tools are listed in the table on the left. Select the **Open_TEP** tool and click the right arrow button to add it to the list at the right.
10. The tools that are listed in the table on the right are options that will be available when a user right-clicks an event in Active Event Lists. After adding the new tool to the Alerts menu, click **Save** to save it. See Figure 11-10 on page 318.

Archived

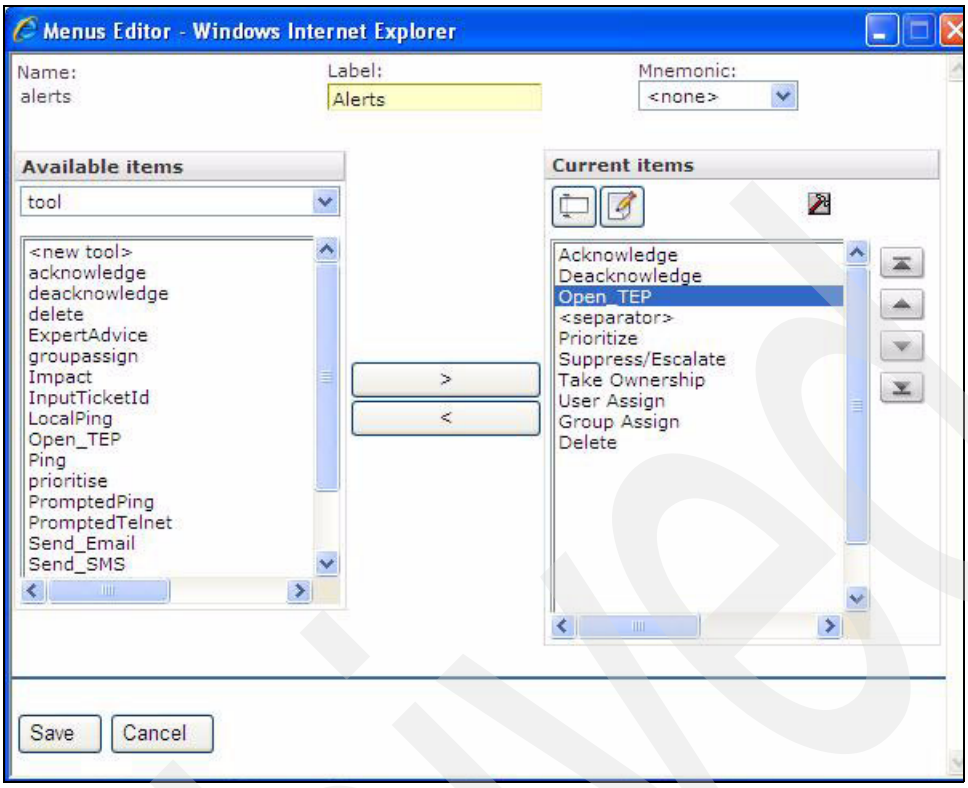


Figure 11-10 Adding new tool Open_TEP to Alerts menu

By following these steps, a new tool with the name Open_TEP is created and added in the Alerts menu as an option. By using this new option, users can use the launch-in-context integration easily by just selecting this new tool from the options list.

The launch-in-context integration of the consoles will be shown in detail in the following sections.

11.5.5 Single sign-on option for the products

Both Tivoli Netcool/OMNIbus WebGUI and IBM Tivoli Monitoring Tivoli Enterprise Portal console run on Tivoli Integrated Portal. You can configure both consoles to authenticate users from an external LDAP repository.

If you do this configuration and add the single sign-on (SSO) capability between both servers (WebGUI of IBM Tivoli Netcool/OMNIbus and Tivoli Enterprise Portal console of IBM Tivoli Monitoring), you can open and work on both consoles with same user credentials (authenticated by LDAP server). You do not need to log in separately to each console.

If you do not enable SSO option for the consoles, you must log in to each server with authorized local users.

To configure Tivoli Enterprise Portal console for the SSO option, go to the following address:

`http://publib.boulder.ibm.com/infocenter/tivihelp/v15r1/index.jsp?topic=/com.ibm.itm.doc_6.2.2/userauthenticate_tepssso_info.htm`

The LDAP configuration of Tivoli Enterprise Portal console is also defined in section 4.7 of *Integrating Tivoli Products*, SG24-7757.

To configure the WebGUI console for single sign-on option, go to the following address:

http://publib.boulder.ibm.com/infocenter/tivihelp/v8r1/topic/com.ibm.netcool_OMNIbus.doc_7.3.0/tip/1.1.1/ttip_install_sso.html

The LDAP configuration of IBM Tivoli Netcool/OMNIbus console is also defined in section 4.6 of *Integrating Tivoli Products*, SG24-7757.

The Single Sign-On implementation of the user console is also defined in section 4.3.2 of *Integrating Tivoli Products*, SG24-7757.

11.6 Scenario walk-through

In this section, we demonstrate how this integration works in a typical real-life scenario.

To use the launch-in-context solution for both consoles, perform the following steps:

1. Log in to WebGUI console of Tivoli Netcool/OMNIbus, open an Active Event List, select an event and right-click it to see the options in Alerts menu. (Figure 11-11).

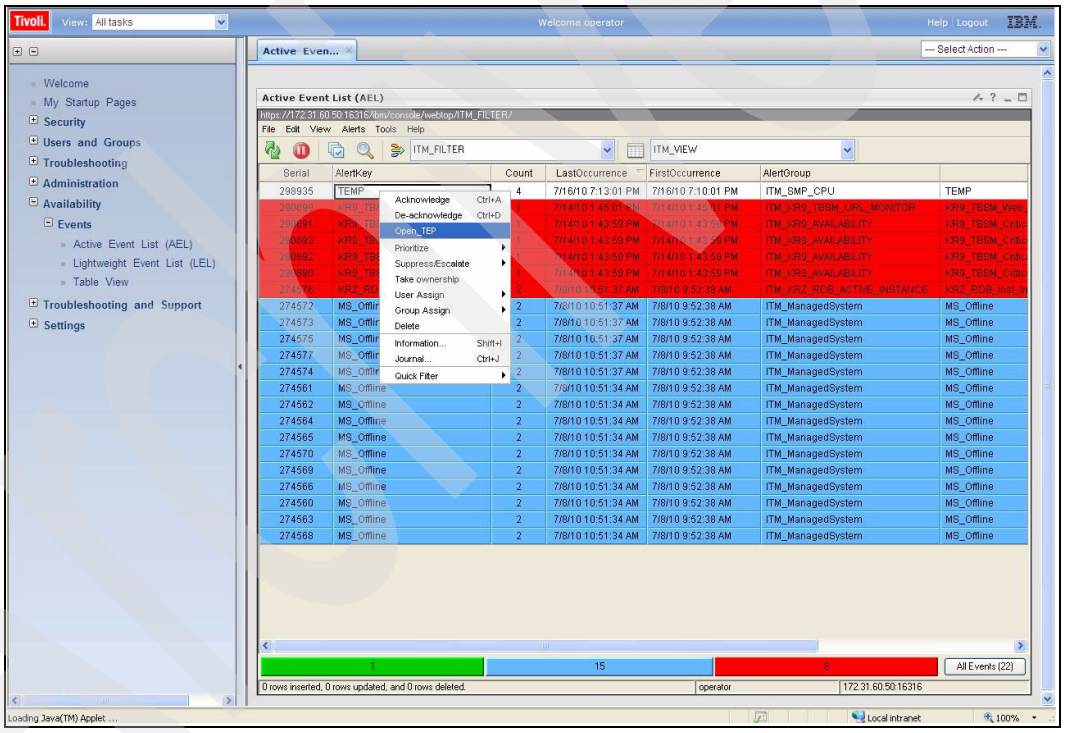


Figure 11-11 Using created tool to open Tivoli Enterprise Portal console for selected event

You see the newly created Open_TEP script tool in the list.

2. Select the **Open_TEP** from the list. A window opens showing the Tivoli Enterprise Portal address of the managed system name (Figure 11-12).

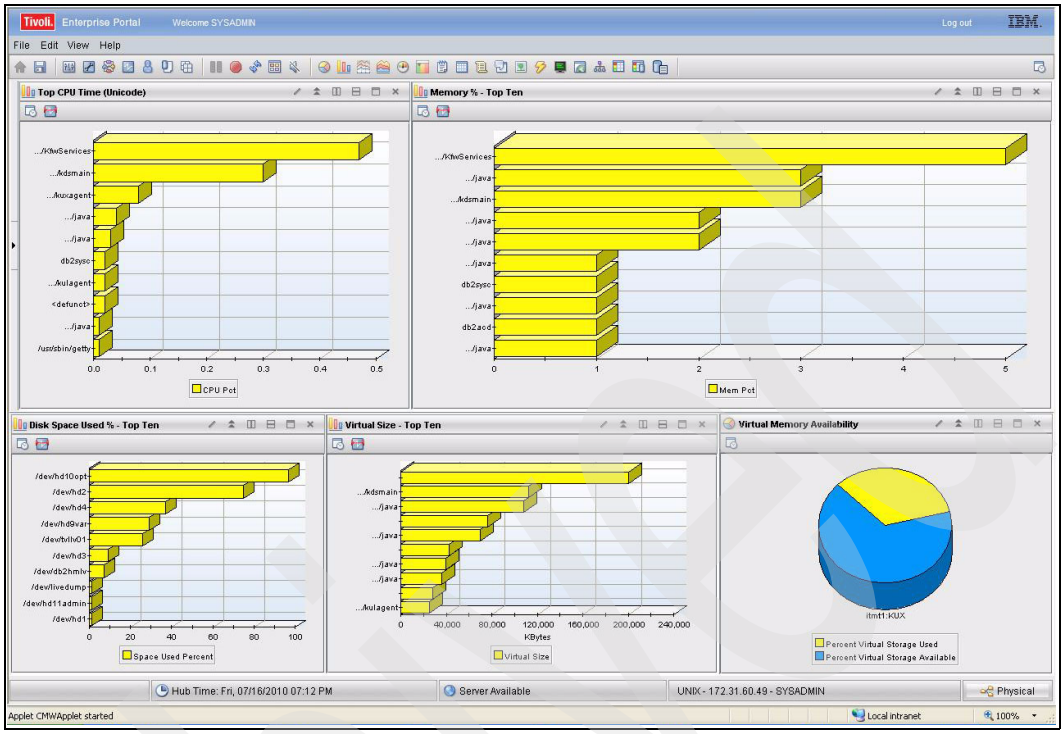


Figure 11-12 Workspace in Tivoli Enterprise Portal for system related to event in IBM Tivoli Netcool/OMNIbus

Archive

3. If you have not configured the single sign-on capability between both consoles, log in to the opened Tivoli Enterprise Portal console as an authorized user. Before the Tivoli Enterprise Portal console opens, a window opens, as shown in Figure 11-13.

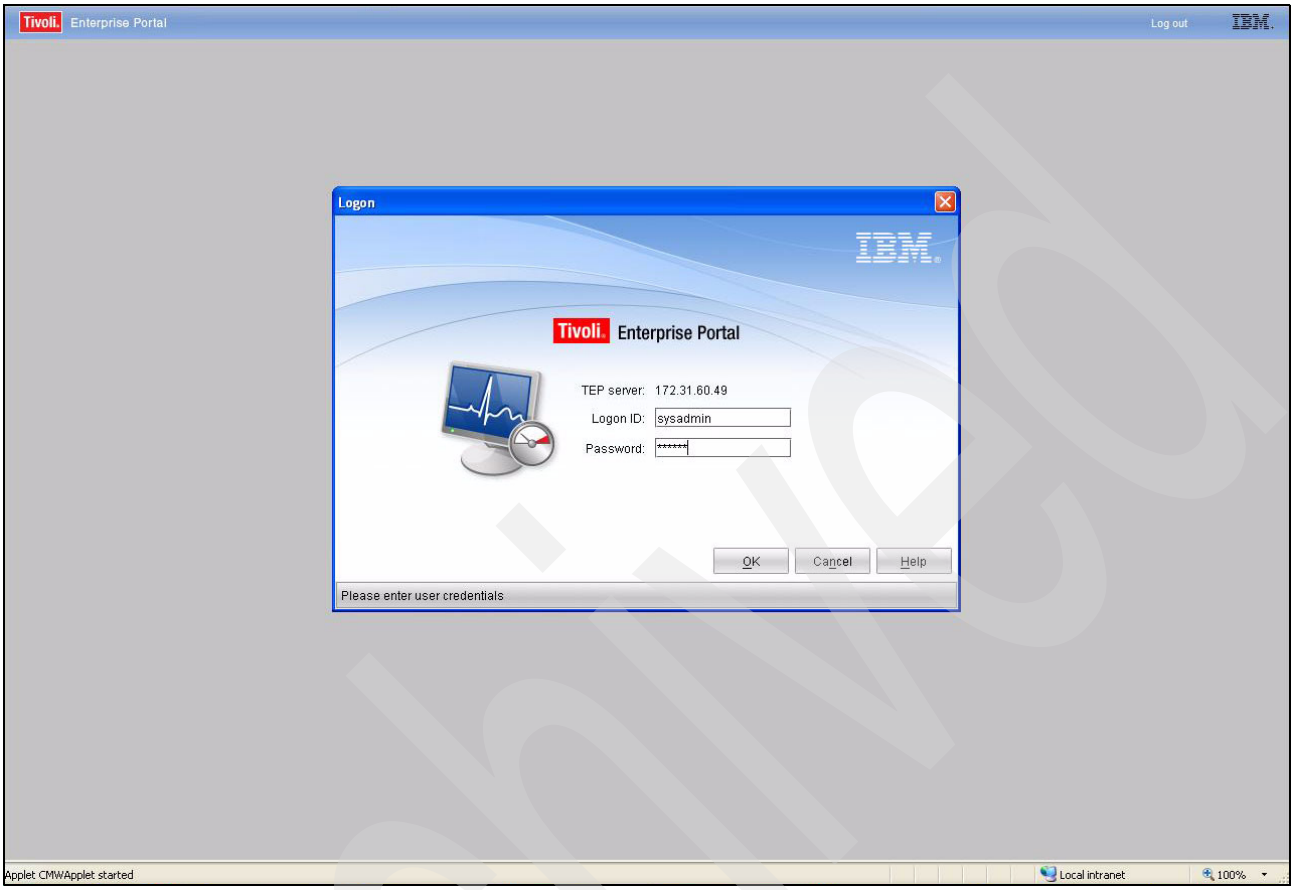


Figure 11-13 Log in to Tivoli Enterprise Portal console if SSO is not enabled

Archive

11.7 Summary

Figure 11-14 shows the quick summary of this integration.

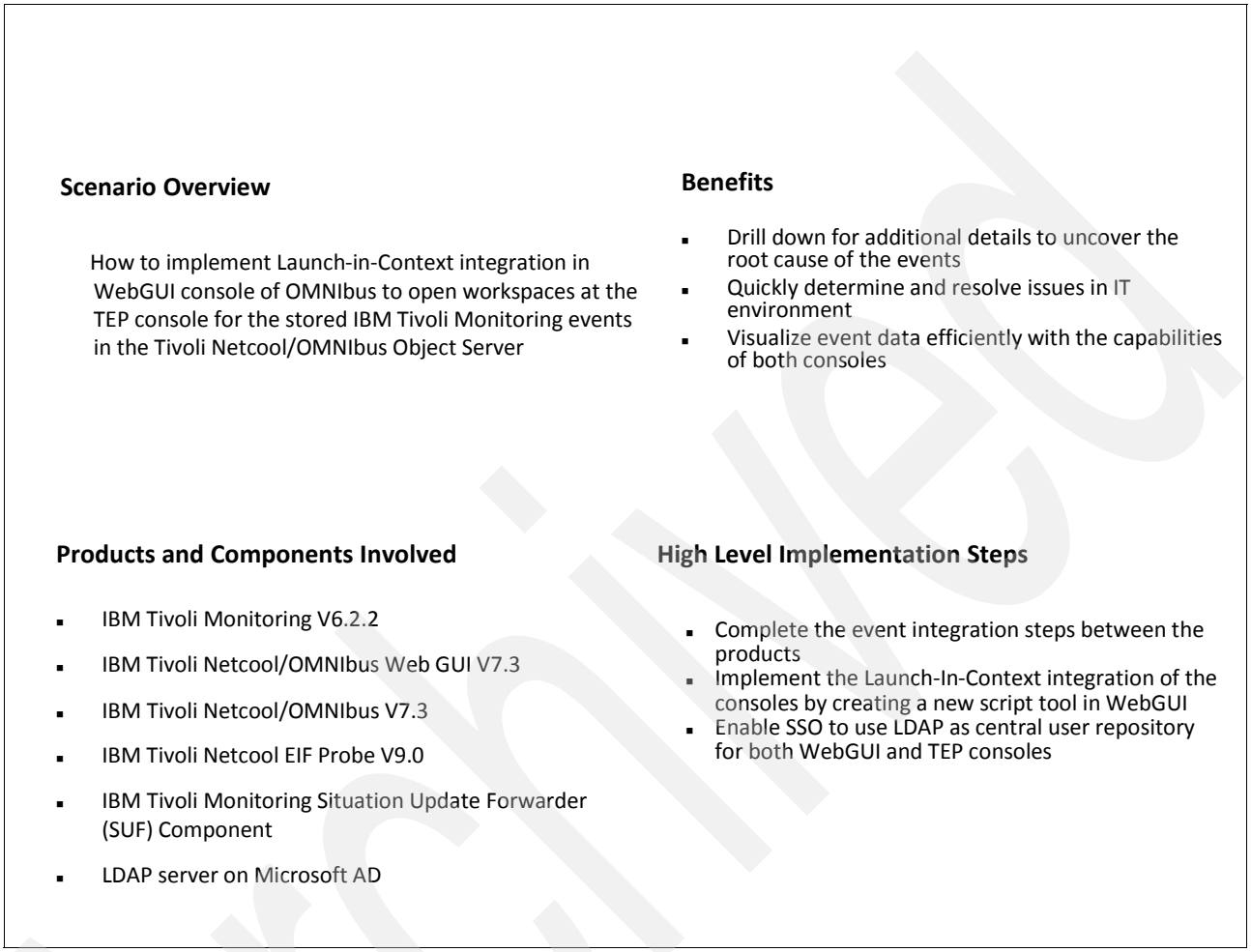


Figure 11-14 Quick summary

Archived

Part 4

Cloud computing and Provisioning integration scenarios

In this part, we cover the Cloud computing and Provisioning integration scenarios.

This part contains the following chapters:

Chapter 12, “Tivoli Service Automation Manager and Cloud Computing” on page 325

Chapter 13, “Automated Storage Provisioning” on page 379

Chapter 14, “Tivoli Provisioning Manager discovery of TADDM discovered CIs” on page 399

Chapter 15, “Tivoli Provisioning Manager and Tivoli Identity Manager integration” on page 405

© Copyright IBM Corp. 2011. All rights reserved.

Archived

12

Tivoli Service Automation Manager and Cloud Computing

This chapter describes how various Tivoli products can be integrated with Tivoli Service Automation Manager to support the delivery of Cloud Computing solutions.

This chapter contains the following topics:

- 12.1, “Cloud Computing overview” on page 326
- 12.2, “Products involved” on page 326
- 12.3, “Benefits” on page 327
- 12.4, “Scenario overview” on page 327
- 12.5, “Creating the service definition” on page 328
- 12.6, “Monitoring the business service” on page 351
- 12.7, “Triggering automated service delivery” on page 369
- 12.8, “Summary” on page 378

© Copyright IBM Corp. 2011. All rights reserved.

12.1 Cloud Computing overview

The automation of service delivery in Cloud Computing requires that several capabilities be integrated. In a Cloud environment, each layer of the infrastructure may be virtualized (for example the core infrastructure, the runtime platform, or the software used to run any actual business services). As a result, Cloud service providers must be able to trace observed *symptoms* as they affect hosted applications to their actual *causes* and trigger the appropriate remediation *based on contracted levels of service*.

Automated remediation, based not only on the triggering event but also on contracted service levels, is a core differentiator between Cloud Computing and traditional models for Event Management. When the performance of some service component is degraded, the effect on the hosted application must be evaluated against the customer's entitlements, which might vary depending on the level of service for which the customer has paid. A decline in performance of the overall application might appear severe in absolute terms, but might not be sufficiently severe that it exceeds the contracted service-level agreement (SLA) between the provider and the customer. An integrated solution that combines service performance metrics, tracing of events to their effects on hosted applications, and automated remediation based on defined service levels is a key component of Cloud Computing services delivery.

We divide the discussion into the following sections:

- Setup and prerequisites for the scenario, which lists and describes the products used for the implementation and gives pointers to installation instructions for each. The base installation of the products that are used is beyond the scope of this publication. See 12.4, "Scenario overview" on page 327.
- Defining a service using Tivoli Service Automation Manager, with which service administrators can create service definitions and the execution flows that are required for automated delivery of each service and service level. See 12.5, "Creating the service definition" on page 328.
- Measuring the health of the hosted application or business service using IBM event management and monitoring tools, which includes determining the service components to monitor, configuring the monitors, and ensuring that detected events are correctly correlated with affected services. See 12.6, "Monitoring the business service" on page 351.
- Creation and activation of automated remediation actions to ensure that contracted service levels are maintained. See 12.7, "Triggering automated service delivery" on page 369.

12.2 Products involved

The products involved in this scenario are as follows:

- IBM Tivoli Composite Application Manager for Transactions 7.2.0.1
- IBM Tivoli Monitoring V6.2.2 FP2
- IBM Tivoli Netcool/OMNIbus V7.2.1
- IBM Tivoli Provisioning Manager V7.1.1 (bundled with Tivoli Service Automation Manager)
- IBM Tivoli Service Automation Manager V7.2 FP3

The middleware products used in this scenario, which all implement Application Response Measurement (ARM), are as follows:

- WebSphere Application Server
- IBM HTTP Server with the WebSphere plug-in (ARM is implemented by the plug-in)
- DB2 Enterprise Server Edition

We describe how each of these tools can be integrated within the context of Cloud Computing. We also discuss the tool-specific customizations to enable the automation detection, tracing, and remediation of Service-Affecting Events (SAE).

Additional integration with IBM Tivoli Usage and Accounting Manager: Another product that you can integrate in this scenario is IBM Tivoli Usage and Accounting Manager. This integration can provide service usage reporting and enable accurate billing. An example of this for Energy Management is discussed in Chapter 4, “Managing business service with energy and environment” on page 63.

12.3 Benefits

The scenario has the following benefits:

- Better response time to resolve a performance issue
- Provisioning new virtual servers without any human interaction based on defined policies

12.4 Scenario overview

This scenario requires that certain products already be installed and configured. The prerequisites are as follows:

- An instance of Tivoli Service Automation Manager
- Integrated monitoring and event management using IBM Tivoli Monitoring and Tivoli Netcool/OMNIbus
- A web-based business service to monitor. Although this scenario could apply to non-web-based applications by changing the selection of monitoring agents used and their configurations, the use case we demonstrate is that of a web-based business service hosted in a cloud. We use the WebSphere Application Server benchmarking application Trade, which is a full J2EE application encompassing multiple application tiers (web/UI, business logic, persistence) and components, and provides a fuller demonstration of the scenario’s applicability to real-world usage.

Because of the selection of IBM Tivoli Composite Application Manager for Transactions, an important aspect is that the application (or the middleware components used to host it) implement ARM, which is an open standard for the generation and collection of transaction processing times. The standard includes APIs for both C and Java and is published by the Open Group. It is located at the following address:

http://www.opengroup.org/management/arm/

Important: The REST API that is used to integrate Tivoli Netcool/OMNIbus and Tivoli Service Automation Manager is not supported in Tivoli Service Automation Manager. A REST API is supported in Tivoli Service Automation Manager 7.2.1, but because at the time of writing this book Tivoli Service Automation Manager 7.2.1 was not available, we used Tivoli Service Automation Manager 7.2 in this integration scenario.

To implement this scenario in an officially supported way, use Tivoli Service Automation Manager 7.2.1 and the REST API as documented for Tivoli Service Automation Manager 7.2.1. You can use the instructions in this chapter as a template for implementing your solution in Tivoli Service Automation Manager 7.2.1, but you must adapt the scripts and the overall solution for your specific environment and for Tivoli Service Automation Manager 7.2.1. Implementing your own solution exactly as described here means you cannot migrate to Tivoli Service Automation Manager 7.2.1.

12.5 Creating the service definition

We start the scenario by describing how to create the service definition on Tivoli Service Automation Manager.

12.5.1 Using the Tivoli Service Automation Manager REST interface

Tivoli Service Automation Manager has several mechanisms to enable integration with external systems. One mechanism is the Web Service facility. This mechanism allows external systems to access the offering catalog by using a Representational State Transfer (REST) interface.

By using the REST interface, for example, you can access the offering catalog and submit a new catalog request, which can start a new provisioning or even take an action in a server.

To use the REST interface, you must define the object structures in the Integration Framework. However, Tivoli Service Automation Manager already provides a set of object structures that are used by the user interface. Be sure that you do not change these object structures, otherwise, the Tivoli Service Automation Manager user interface might stop work.

More details about the object structure are in *Deployment Guide Series: Maximo Asset Management V7.1*, SG24-7640.

Figure 12-1 on page 329 shows the Tivoli Service Automation Manager objects that are ready to use. To see the list of Tivoli Service Automation Manager objects, open the Object Structures application (click **Go to** → **Integration** → **Object Structures**) and filter for the following object structure:

```
PMZHBR1_*
```

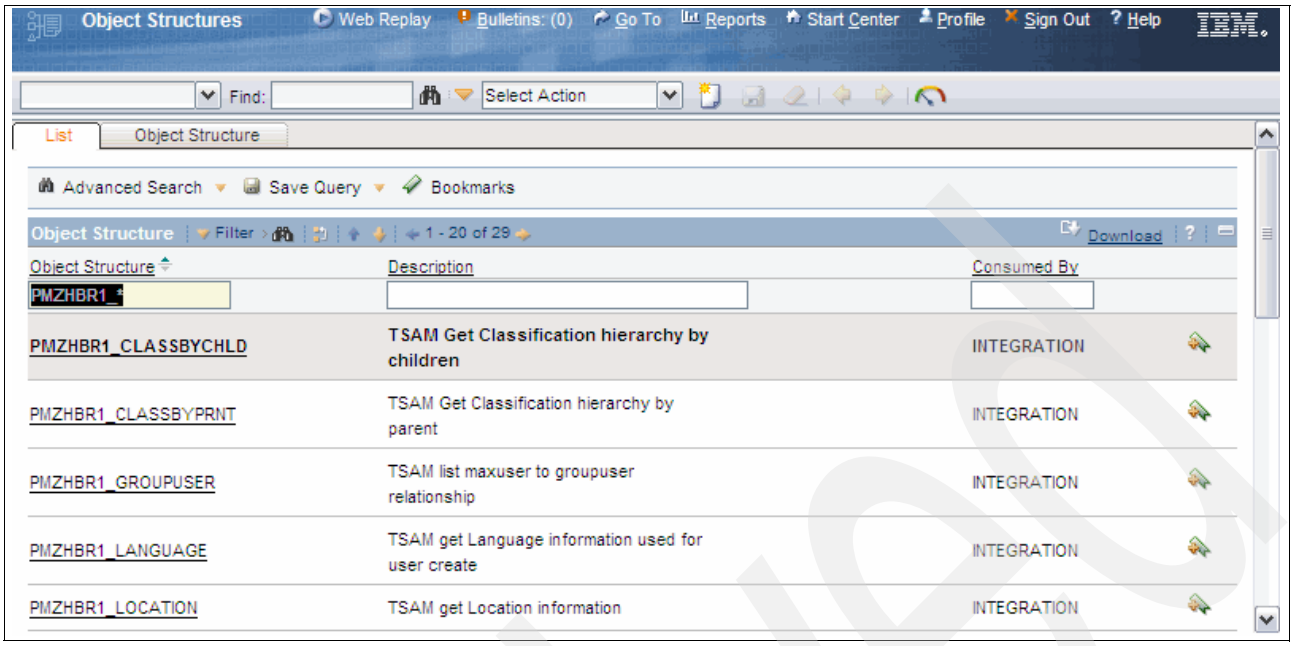


Figure 12-1 Tivoli Service Automation Manager object structures

These Tivoli Service Automation Manager object structures are the interface that is used to perform the following tasks:

- Create and update requests.
- Query for data structures.

12.5.2 Understanding the request creation object

The object structure used to create or update a request is PMZHBR1_PMSCMRCREATE. The source objects for this object structure are shown in Figure 12-2.

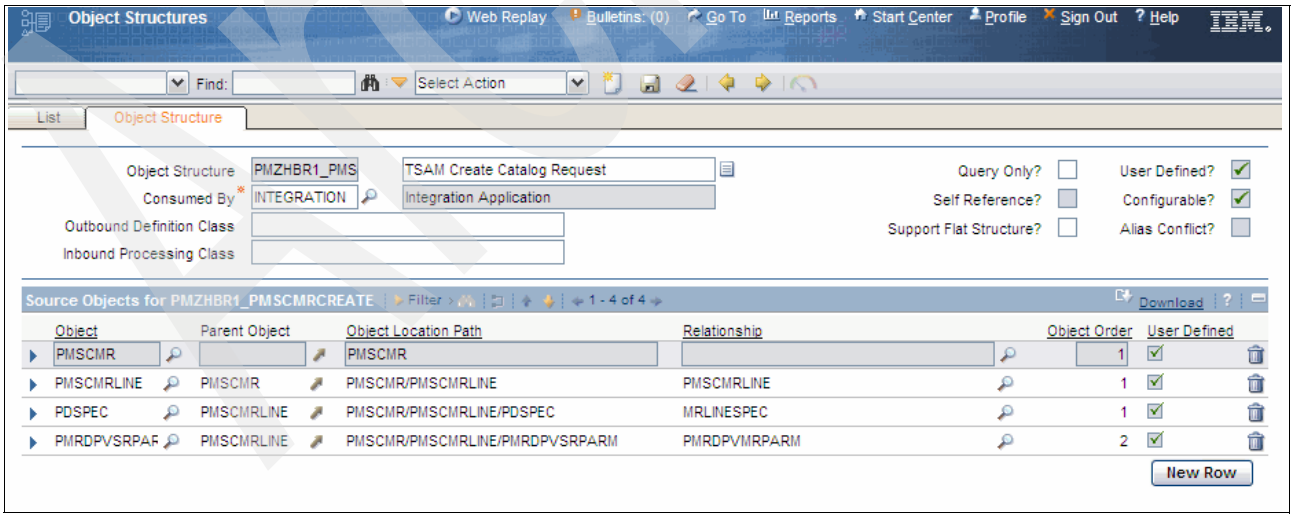


Figure 12-2 Source Objects definition

Each of these objects has a relationship that is represented by the Parent Object column. The attributes for each of these objects can be located using the Database configuration

Application (click **Go To** → **System Configuration** → **Platform Configuration** → **Database Configuration**). Figure 12-3 shows an example of attributes of the PMSCMR object.

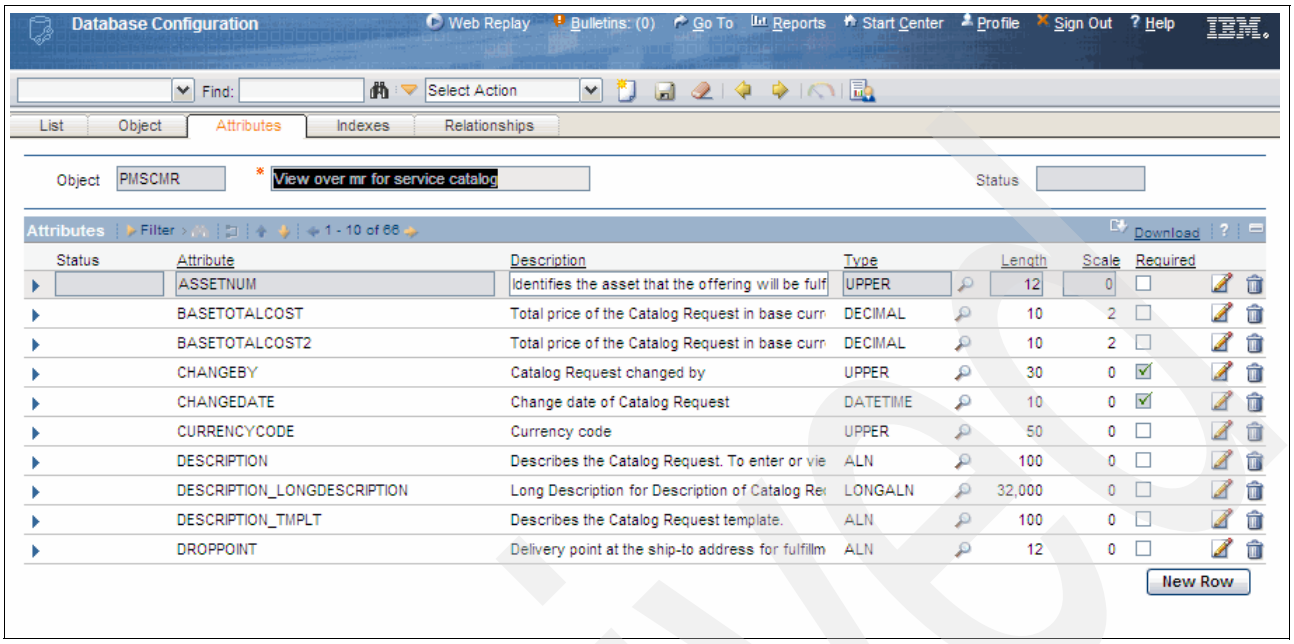


Figure 12-3 List of attributes of the PMSCMR object

12.5.3 Obtaining the resource pool name

Before submitting a request for new provisioning, you need the available resource pool name. To obtain this information, use a web service. to see the list of available web services, open the Web Services Library application (click **Go to** → **Integration** → **Web Services Library**). Press Enter to show all available Web Services. See Figure 12-4.

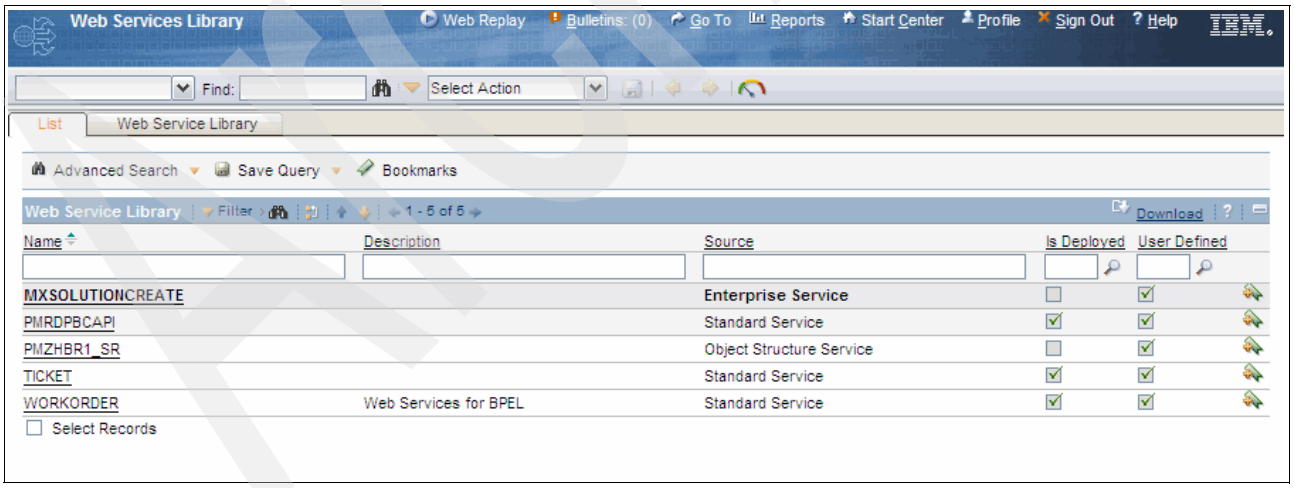


Figure 12-4 List of available Web Services

The web service PMRDPBCAPI can be used to query the available resource pool. A method named `pmrdpbcapigetAvailablePoolList` returns information about each defined resource pool. See Figure 12-5 on page 331.

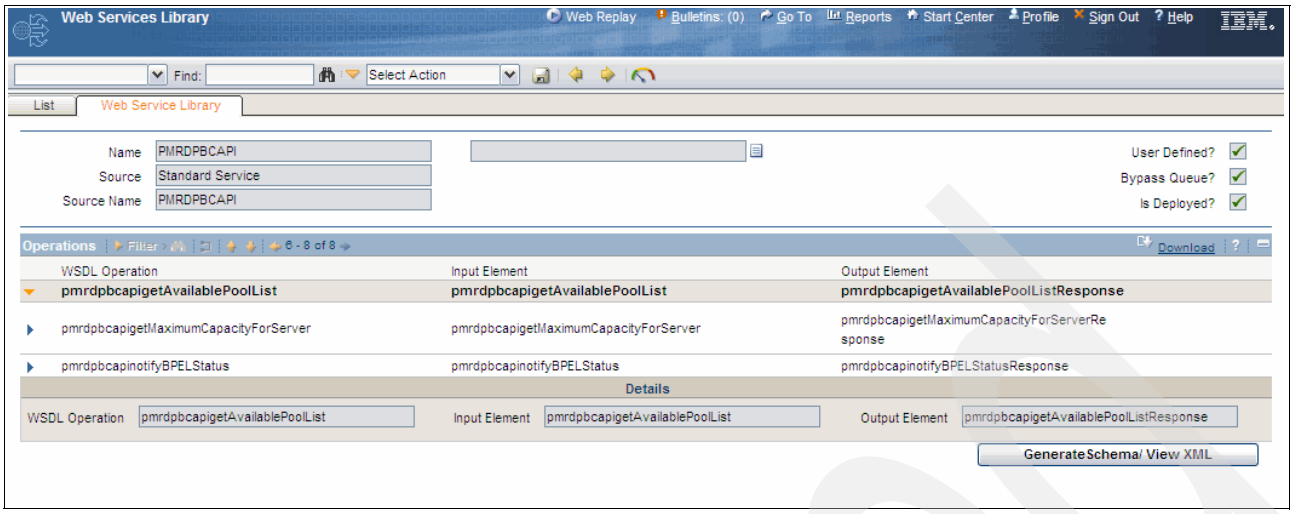


Figure 12-5 Method used to query the available resource pools

To access the information in the web service, a simple request XML statement is required (Example 12-1), and the response is in XML also (Example 12-2).

Example 12-1 Request XML

```
<?xml version="1.0" encoding="utf-8"?>
<soapenv:Envelope
   xmlns:q0="http://www.ibm.com/maximo"
   xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
   <soapenv:Body>
      <q0:pmrdpbcapigetAvailablePoolList
         creationDateTime="2010-06-10T14:56:42Z"
         maximoVersion="7115"
         messageID="193" />
   </soapenv:Body>
</soapenv:Envelope>
```

Example 12-2 Response XML

```
<?xml version="1.0" encoding="utf-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
<soapenv:Body>
   <pmrdpbcapigetAvailablePoolListResponse
      creationDateTime="2010-06-10T10:57:11-05:00"
      transLanguage="EN" baseLanguage="EN"
      messageID="1276185431062621164"
      xmlns="http://www.ibm.com/maximo"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
   <PMRDPBCPOOLSMboSet>
      <PMRDPBCPOOLS>
         <NAME>Xen Local Disk</NAME>
         <ID>/cloud/rest/pools/0/</ID>
         <PLATFORMTYPE>Xen</PLATFORMTYPE>
      </PMRDPBCPOOLS>
      <PMRDPBCPOOLS>
```

Archive

```
         <NAME>System p LPAR</NAME>
         <ID>/cloud/rest/pools/1/</ID>
         <PLATFORMTYPE>LPAR</PLATFORMTYPE>
      </PMRDPBCPOOLS>
      <PMRDPBCPOOLS>
         <NAME>VMware System x</NAME>
         <ID>/cloud/rest/pools/3/</ID>
         <PLATFORMTYPE>VMware</PLATFORMTYPE>
      </PMRDPBCPOOLS>
   </PMRDPBCPOOLSMboSet>
   </pmrdpbcapigetAvailablePoolListResponse>
</soapenv:Body>
</soapenv:Envelope>
```

The resource pools available in the environment is shown between the following tags:

`<PMRDPBCPOOLS> </PMRDPBCPOOLS>`

The information required for the next step is the string in the `<ID>` tag.

12.5.4 Obtaining the images list

A REST-based method is used to obtain the list of images in a resource pool. To access the information, an HTTP GET method must be issued in the REST interface. Example 12-3 shown a URL of a REST interface.

Note: The URL is an example only and might change based on the specifics of your Tivoli Service Automation Manager environment.

Example 12-3 URL used to obtain the images list

```
https://itsam72.ral.ibm.com:9443/ilrest/rest/os/TPV01IMGLIBENTRYMSTR?_format=json&
_compact=1
```

To access the REST interface, a user ID and password must be provided. The default user ID is `maxadmin`. The response for the URL is in *json* format. See Example 12-4.

Example 12-4 Images list

```
{
   "QueryTPV01IMGLIBENTRYMSTRResponse": {
      "rsStart": 0,
      "rsCount": 1,
      "rsTotal": 1,
      "TPV01IMGLIBENTRYMSTRSet": {
         "ILIMAGE": [
            {
               "COLLECTION": false,
               "CONFIGSWAPSPACE": false,
               "CREATED": "2010-06-10T05:30:01-04:00",
               "CREATOR": "MAXADMIN",
               "GUESTOSTYPE": "Unknown",
               "HYPERVISOR": "VMware",
               "IMAGEFORMAT": "None",
               "IMAGEID": 1,
               "IMAGELIBRARYID": 1,
               "NAME": "LinuxTemplate",
```

```
"OWNER": "MAXADMIN",
"POOL_ID": "\/cloud\/rest\/pools\/3\/",
"REPOSITORYID": 1,
"REPOSITORYNAME": "NFS Repository",
"REPOSITORYTYPE": "NFS",
"SOFTWARESTACKID": 9487,
"STATE": "Created",
"TPIMAGEID": 9489,
"TYPE": "Master",
"UPDATED": "2010-06-10T05:30:01-04:00",
"UPDATER": "MAXADMIN",
"VERSION": "1",
"VIRTUALSERVERTEMPLATEID": 9414,
"VIRTUALSERVERTEMPLATEMINID": 9410,
"TPIMAGE": [
   {
      "ID": 9489,
      "IMAGE_TYPE_ID": 1,
      "NAME": "LinuxTemplate"
   }
],
"TPSOFTWARESTACK": [
   {
      "ID": 9487,
      "NAME": "LinuxTemplate"
   }
],
"TPVIRTUALSERVERTEMPLATE": [
   {
      "ID": 9410,
      "TPRESOURCEREQUIREMENT": [
         {
            "HOW_MANY": 9,
            "NAME": "Minimum Disk Space",
            "RESOURCE_SIZE": 9.0,
            "RESOURCE_TYPE_ID": 2
         },
         {
            "HOW_MANY": 102,
            "NAME": "Minimum Memory",
            "RESOURCE_SIZE": 102.0,
            "RESOURCE_TYPE_ID": 6
         },
         {
            "HOW_MANY": 1,
            "NAME": "Minimum CPU",
            "RESOURCE_SIZE": 1.0,
            "RESOURCE_TYPE_ID": 3
         }
      ]
   },
   {
      "ID": 9414,
      "TPRESOURCEREQUIREMENT": [
         {
            "HOW_MANY": 10,
            "NAME": "Recommended Disk Space",
            "RESOURCE_SIZE": 10.0,
            "RESOURCE_TYPE_ID": 2
         },
```


Archived

```
                        {
                           "HOW_MANY": 102,
                           "NAME": "Recommended Memory",
                           "RESOURCE_SIZE": 102.0,
                           "RESOURCE_TYPE_ID": 6
                        },
                        {
                           "HOW_MANY": 1,
                           "NAME": "Recommended CPU",
                           "RESOURCE_SIZE": 10.0,
                           "RESOURCE_TYPE_ID": 3
                        }
                     ]
                  }
               ],
               "ILIMAGE": [
               ]
            }
         ]
      }
   }
}
```

12.5.5 Creating a new catalog request

A catalog request is the beginning of the process to provision a new server. The process consists of defining the Catalog Request with basic information and the item that represents the action of this request. The first step of the process creates a request in draft status and as a result, a list of unique identifiers that will be used in the second part of the process to update the request with all the necessary information to the action.

The process consists of three steps:

1. Create a catalog request (PMSCMR Object Instance).
2. Obtain the unique identifiers.
3. Update the request with required information.

The REST interface is used to create and update the request. The `PMZHBR1_PMSCMRCREATE` object structure is used to create the request. As mentioned before, the object instance PMSCMR is a source of this object. The method used to input the data into the REST interface must be HTTP POST.

Example 12-5 is a URL for the REST interface.

Note: The URL can change, based on the specifics of your Tivoli Service Automation Manager environment.

Example 12-5 URL to create the Catalog Request

```
https://itsam72.ral.ibm.com:9443/maxrest/rest/os/PMZHBR1_PMSCMRCREATE?_format=json&_compact
=1&SITEID=PMSCRTP&ORGID=PMSCIBM&REQUESTEDBY=CLOUDADMIN&REQUESTEDFOR=&DESCRIPTION=Create%20P
roject%20with%20VMware%20Servers&ENDDATE=2010-06-22&PMSCMRLINE.1.CLASSSTRUCTUREID=8100210&P
MSCMRLINE.1.ITEMNUM=PMRDP_0201A&PMSCMRLINE.1.ITEMSETID=PMSCS1&PMSCMRLINE.1.ORDERUNIT=EACH&P
MSCMRLINE.1.LINETYPE=SRMOFF&PMSCMRLINE.1.QTY=1
```

Archived

The Table 12-1 shows the URL attributes that are used in this environment.

Table 12-1 Object Parameters

Attribute	Value	Comment
https://itsam72.ral.ibm.com:9443/maxrest/rest/os/PMZHBR1_PMSCMRCREATE	<hostname> <IP>	The host name and IP of the Tivoli Service Automation Manager server
_format	json	This is the only acceptable value
_compact	1	This is the only acceptable value
SITEID	PMSCRTP	The site defined in the Tivoli Service Automation Manager. To see the site name, open the Organizations application (**Go To** → **Administration** → **Organizations)**
ORGID	PMSCIBM	The organization defined in the Tivoli Service Automation Manager. To see the organization name, open the Organizations application (**Go To** → **Administration** → **Organizations)**
REQUESTEDBY	CLOUDADMIN	The requester ID for this service. The requester must be defined in the environment.
REQUESTEDFOR	<offering description>	The offering description. Do not use spaces here.
ENDDATE	date	The end date for this project. (yyyy-mm-aa)
PMSCMRLINE.1.CLASSSTRUCTUREID	8100210	This value is optional. If you do not know the CLASSSTRUCTUREID, remove this attribute.
PMSCMRLINE.1.ITEMNUM	PMRDP_0201A	Represents the offering name in the catalog. To see all offerings, open the offerings application (**Go To** → **Service Request Manager catalog** → **Service Inventory** → **Offerings**
PMSCMRLINE.1.ITEMSETID	PMSCS1	Item Set defined for the offering.
MSCMRLINE.1.ORDERUNIT	EACH	Order unit by which the offering is requested.
PMSCMRLINE.1.LINETYPE	SRMOFF	Line type for the offering. The only acceptable value is SRMOFF.
PMSCMRLINE.1.QTY	1	Number of requests

Archive

After posting the HTTP, the return is similar to Example 12-6.

Example 12-6 Catalog request response, in json format


```
{
   "CreatePMZHBR1_PMSCMRCREATEResponse": {
      "rsStart": 0,
      "rsCount": 1,
      "rsTotal": 1,
      "PMZHBR1_PMSCMRCREATESet": {
         "PMSCMR": {
            "CHANGEBY": "MAXADMIN",
            "CHANGEDATE": "2010-06-24T00:04:27-04:00",
            "DESCRIPTION": "Start Server vm009042171039",
            "ENTERBY": "MAXADMIN",
            "ENTERDATE": "2010-06-24T00:04:28-04:00",
            "HISTORYFLAG": false,
            "MRDATE": "2010-06-24T00:04:28-04:00",
            "MRID": 69,
            "MRNUM": "1093",
            "MRSTATUSSEQ": 240,
            "ORGID": "PMSCIBM",
            "PMSCSDSCMAPID": 69,
            "PRIORITY": 1,
            "REQUESTEDBY": "PMRDPCAUSR",
            "SITEID": "PMSCRTP",
            "STATUS": "DRAFT",
            "STATUSDATE": "2010-06-24T00:04:28-04:00",
            "TOTALCOST": 0.0,
            "TYPE": "STANDARD",
            "PMSCMRLINE": [
               {
                  "CHARGESTORE": false,
                  "CLASSIFICATIONID": "PMRDP_SR_VS_MTG \\
PMRDP_SR_CREATE_PROJECT_VMWARE_SERVER",
                  "CLASSSTRUCTUREID": "8100210",
                  "COMMODITY": "VSERVER",
                  "COMMODITYGROUP": "SRVAUTOM",
                  "COMPLETE": false,
                  "CURRENCYCODE": "USD",
                  "DESCRIPTION": "Create Project with VMware Servers",
                  "DIRECTREQ": true,
                  "ENTEREDASTASK": false,
                  "EXCHANGERATE": 1.0,
                  "INSPECTIONREQUIRED": true,
                  "ISDISTRIBUTED": false,
                  "ITEMNUM": "PMRDP_0201A",
                  "ITEMSETID": "PMSCS1",
                  "LINECOST": 0.0,
                  "LINECOST1": 0.0,
                  "LINETYPE": "SRMOFF",
                  "MKTPLCITEM": false,
                  "MRLINEID": 68,
                  "MRLINENUM": 1,
                  "ORDERUNIT": "EACH",
                  "ORGID": "PMSCIBM",
                  "PARTIALISSUE": true,
                  "QTY": 1.0,
                  "UNITCOST": 0.0,
                  "PDSPEC": [
                     {
```


Archived

```
    "ASSETATTRID": "PMRDPCLCPR_DESCRIPTION",
    "CHANGEBY": "MAXADMIN",
    "CHANGEDATE": "2010-06-24T00:04:28-04:00",
    "CLASSSPECID": 925,
    "CLASSSTRUCTUREID": "8100210",
    "DISPLAYSEQUENCE": 7,
    "ITEMNUM": "PMRDP_0201A",
    "MANDATORY": false,
    "ORGID": "PMSCIBM",
    "PDOWNERID": 69,
    "PDSPECID": 376,
    "REFOBJECTID": 68,
    "REFOBJECTLINENUM": 1,
    "REFOBJECTNAME": "PMSCMRLINE"
  },
  {
    "ASSETATTRID": "PMRDPCLCPR_PERSONGROUP",
    "CHANGEBY": "MAXADMIN",
    "CHANGEDATE": "2010-06-24T00:04:29-04:00",
    "CLASSSPECID": 926,
    "CLASSSTRUCTUREID": "8100210",
    "DISPLAYSEQUENCE": 18,
    "ITEMNUM": "PMRDP_0201A",
    "MANDATORY": false,
    "ORGID": "PMSCIBM",
    "PDOWNERID": 69,
    "PDSPECID": 388,
    "REFOBJECTID": 68,
    "REFOBJECTLINENUM": 1,
    "REFOBJECTNAME": "PMSCMRLINE"
  },
  {
    "ASSETATTRID": "PMRDPCLCPR_PROJECTACCOUNT",
    "CHANGEBY": "MAXADMIN",
    "CHANGEDATE": "2010-06-24T00:04:28-04:00",
    "CLASSSPECID": 927,
    "CLASSSTRUCTUREID": "8100210",
    "DISPLAYSEQUENCE": 1,
    "ITEMNUM": "PMRDP_0201A",
    "MANDATORY": false,
    "ORGID": "PMSCIBM",
    "PDOWNERID": 69,
    "PDSPECID": 369,
    "REFOBJECTID": 68,
    "REFOBJECTLINENUM": 1,
    "REFOBJECTNAME": "PMSCMRLINE"
  },
  {
    "ASSETATTRID": "PMRDPCLCPR_PROJECTID",
    "CHANGEBY": "MAXADMIN",
    "CHANGEDATE": "2010-06-24T00:04:28-04:00",
    "CLASSSPECID": 928,
    "CLASSSTRUCTUREID": "8100210",
    "DISPLAYSEQUENCE": 6,
    "ITEMNUM": "PMRDP_0201A",
    "MANDATORY": false,
    "ORGID": "PMSCIBM",
    "PDOWNERID": 69,
    "PDSPECID": 375,
    "REFOBJECTID": 68,
```


Archived

```
      "REFOBJECTLINENUM": 1,
      "REFOBJECTNAME": "PMSCMRLINE"
   },
   {
      "ASSETATTRID": "PMRDPCLCPR_PROJECTNAME",
      "CHANGEBY": "MAXADMIN",
      "CHANGEDATE": "2010-06-24T00:04:28-04:00",
      "CLASSSPECID": 929,
      "CLASSSTRUCTUREID": "8100210",
      "DISPLAYSEQUENCE": 5,
      "ITEMNUM": "PMRDP_0201A",
      "MANDATORY": false,
      "ORGID": "PMSCIBM",
      "PDOWNERID": 69,
      "PDSPECID": 374,
      "REFOBJECTID": 68,
      "REFOBJECTLINENUM": 1,
      "REFOBJECTNAME": "PMSCMRLINE"
   },
   {
      "ASSETATTRID": "PMRDPCLCPR_SERVERQTY",
      "CHANGEBY": "MAXADMIN",
      "CHANGEDATE": "2010-06-24T00:04:28-04:00",
      "CLASSSPECID": 930,
      "CLASSSTRUCTUREID": "8100210",
      "DISPLAYSEQUENCE": 8,
      "ITEMNUM": "PMRDP_0201A",
      "MANDATORY": false,
      "ORGID": "PMSCIBM",
      "PDOWNERID": 69,
      "PDSPECID": 377,
      "REFOBJECTID": 68,
      "REFOBJECTLINENUM": 1,
      "REFOBJECTNAME": "PMSCMRLINE"
   },
   {
      "ALNVALUE": "RDPVS",
      "ASSETATTRID": "PMRDPCLCPR_SERVICEDEFINITIONNUM",
      "CHANGEBY": "MAXADMIN",
      "CHANGEDATE": "2010-06-24T00:04:28-04:00",
      "CLASSSPECID": 931,
      "CLASSSTRUCTUREID": "8100210",
      "DISPLAYSEQUENCE": 3,
      "ITEMNUM": "PMRDP_0201A",
      "MANDATORY": false,
      "ORGID": "PMSCIBM",
      "PDOWNERID": 69,
      "PDSPECID": 372,
      "REFOBJECTID": 68,
      "REFOBJECTLINENUM": 1,
      "REFOBJECTNAME": "PMSCMRLINE"
   },
   {
      "ASSETATTRID": "PMRDPCLCPR_SERVICEDEFINITIONREVISION",
      "CHANGEBY": "MAXADMIN",
      "CHANGEDATE": "2010-06-24T00:04:28-04:00",
      "CLASSSPECID": 932,
      "CLASSSTRUCTUREID": "8100210",
      "DISPLAYSEQUENCE": 2,
      "ITEMNUM": "PMRDP_0201A",
```


Archived

```
      "MANDATORY": false,
      "NUMVALUE": 2.0,
      "ORGID": "PMSCIBM",
      "PDOWNERID": 69,
      "PDSPECID": 371,
      "REFOBJECTID": 68,
      "REFOBJECTLINENUM": 1,
      "REFOBJECTNAME": "PMSCMRLINE"
   },
   {
      "ASSETATTRID": "PMRDPCLCPR_SERVICEINSTANCEID",
      "CHANGEBY": "MAXADMIN",
      "CHANGEDATE": "2010-06-24T00:04:28-04:00",
      "CLASSSPECID": 933,
      "CLASSSTRUCTUREID": "8100210",
      "DISPLAYSEQUENCE": 1,
      "ITEMNUM": "PMRDP_0201A",
      "MANDATORY": false,
      "ORGID": "PMSCIBM",
      "PDOWNERID": 69,
      "PDSPECID": 370,
      "REFOBJECTID": 68,
      "REFOBJECTLINENUM": 1,
      "REFOBJECTNAME": "PMSCMRLINE"
   },
   {
      "ASSETATTRID": "PMRDPCLCVS_MEMORY",
      "CHANGEBY": "MAXADMIN",
      "CHANGEDATE": "2010-06-24T00:04:29-04:00",
      "CLASSSPECID": 934,
      "CLASSSTRUCTUREID": "8100210",
      "DISPLAYSEQUENCE": 9,
      "ITEMNUM": "PMRDP_0201A",
      "MANDATORY": false,
      "MEASUREUNITID": "MBYTE",
      "ORGID": "PMSCIBM",
      "PDOWNERID": 69,
      "PDSPECID": 378,
      "REFOBJECTID": 68,
      "REFOBJECTLINENUM": 1,
      "REFOBJECTNAME": "PMSCMRLINE"
   },
   {
      "ASSETATTRID": "PMRDPCLCVS_NUMBER_CPUS",
      "CHANGEBY": "MAXADMIN",
      "CHANGEDATE": "2010-06-24T00:04:29-04:00",
      "CLASSSPECID": 935,
      "CLASSSTRUCTUREID": "8100210",
      "DISPLAYSEQUENCE": 10,
      "ITEMNUM": "PMRDP_0201A",
      "MANDATORY": false,
      "ORGID": "PMSCIBM",
      "PDOWNERID": 69,
      "PDSPECID": 379,
      "REFOBJECTID": 68,
      "REFOBJECTLINENUM": 1,
      "REFOBJECTNAME": "PMSCMRLINE"
   },
   {
      "ASSETATTRID": "PMRDPCLCVS_NUMBER_PCPUS",
```


Archived

```
      "CHANGEBY": "MAXADMIN",
      "CHANGEDATE": "2010-06-24T00:04:29-04:00",
      "CLASSSPECID": 936,
      "CLASSSTRUCTUREID": "8100210",
      "DISPLAYSEQUENCE": 11,
      "ITEMNUM": "PMRDP_0201A",
      "MANDATORY": false,
      "ORGID": "PMSCIBM",
      "PDOWNERID": 69,
      "PDSPECID": 380,
      "REFOBJECTID": 68,
      "REFOBJECTLINENUM": 1,
      "REFOBJECTNAME": "PMSCMRLINE"
   },
   {
      "ASSETATTRID": "PMRDPCLCVS_RESGRPNAME",
      "CHANGEBY": "MAXADMIN",
      "CHANGEDATE": "2010-06-24T00:04:29-04:00",
      "CLASSSPECID": 937,
      "CLASSSTRUCTUREID": "8100210",
      "DISPLAYSEQUENCE": 17,
      "ITEMNUM": "PMRDP_0201A",
      "MANDATORY": false,
      "ORGID": "PMSCIBM",
      "PDOWNERID": 69,
      "PDSPECID": 386,
      "REFOBJECTID": 68,
      "REFOBJECTLINENUM": 1,
      "REFOBJECTNAME": "PMSCMRLINE"
   },
   {
      "ASSETATTRID": "PMRDPCLCVS_RESGRPNUM",
      "CHANGEBY": "MAXADMIN",
      "CHANGEDATE": "2010-06-24T00:04:29-04:00",
      "CLASSSPECID": 938,
      "CLASSSTRUCTUREID": "8100210",
      "DISPLAYSEQUENCE": 16,
      "ITEMNUM": "PMRDP_0201A",
      "MANDATORY": false,
      "ORGID": "PMSCIBM",
      "PDOWNERID": 69,
      "PDSPECID": 385,
      "REFOBJECTID": 68,
      "REFOBJECTLINENUM": 1,
      "REFOBJECTNAME": "PMSCMRLINE"
   },
   {
      "ASSETATTRID": "PMRDPCLCVS_STORAGE_SIZE",
      "CHANGEBY": "MAXADMIN",
      "CHANGEDATE": "2010-06-24T00:04:29-04:00",
      "CLASSSPECID": 939,
      "CLASSSTRUCTUREID": "8100210",
      "DISPLAYSEQUENCE": 12,
      "ITEMNUM": "PMRDP_0201A",
      "MANDATORY": false,
      "ORGID": "PMSCIBM",
      "PDOWNERID": 69,
      "PDSPECID": 381,
      "REFOBJECTID": 68,
      "REFOBJECTLINENUM": 1,
```


Archived

```
      "REFOBJECTNAME": "PMSCMRLINE"
   },
   {
      "ASSETATTRID": "PMRDPCLCVS_SWAP_SIZE",
      "CHANGEBY": "MAXADMIN",
      "CHANGEDATE": "2010-06-24T00:04:29-04:00",
      "CLASSSPECID": 940,
      "CLASSSTRUCTUREID": "8100210",
      "DISPLAYSEQUENCE": 14,
      "ITEMNUM": "PMRDP_0201A",
      "MANDATORY": false,
      "ORGID": "PMSCIBM",
      "PDOWNERID": 69,
      "PDSPECID": 383,
      "REFOBJECTID": 68,
      "REFOBJECTLINENUM": 1,
      "REFOBJECTNAME": "PMSCMRLINE"
   },
   {
      "ALNVALUE": "VMware",
      "ASSETATTRID": "PMRDPCLCVS_TYPE",
      "CHANGEBY": "MAXADMIN",
      "CHANGEDATE": "2010-06-24T00:04:29-04:00",
      "CLASSSPECID": 941,
      "CLASSSTRUCTUREID": "8100210",
      "DISPLAYSEQUENCE": 19,
      "ITEMNUM": "PMRDP_0201A",
      "MANDATORY": false,
      "ORGID": "PMSCIBM",
      "PDOWNERID": 69,
      "PDSPECID": 389,
      "REFOBJECTID": 68,
      "REFOBJECTLINENUM": 1,
      "REFOBJECTNAME": "PMSCMRLINE"
   },
   {
      "ASSETATTRID": "PMRDPCLSWS_IMAGEID",
      "CHANGEBY": "MAXADMIN",
      "CHANGEDATE": "2010-06-24T00:04:29-04:00",
      "CLASSSPECID": 942,
      "CLASSSTRUCTUREID": "8100210",
      "DISPLAYSEQUENCE": 13,
      "ITEMNUM": "PMRDP_0201A",
      "MANDATORY": false,
      "ORGID": "PMSCIBM",
      "PDOWNERID": 69,
      "PDSPECID": 382,
      "REFOBJECTID": 68,
      "REFOBJECTLINENUM": 1,
      "REFOBJECTNAME": "PMSCMRLINE"
   },
   {
      "ASSETATTRID": "PMRDPCLSWS_MONITORING",
      "CHANGEBY": "MAXADMIN",
      "CHANGEDATE": "2010-06-24T00:04:29-04:00",
      "CLASSSPECID": 943,
      "CLASSSTRUCTUREID": "8100210",
      "DISPLAYSEQUENCE": 15,
      "ITEMNUM": "PMRDP_0201A",
      "MANDATORY": false,
```


Archived

```
      "ORGID": "PMSCIBM",
      "PDOWNERID": 69,
      "PDSPECID": 384,
      "REFOBJECTID": 68,
      "REFOBJECTLINENUM": 1,
      "REFOBJECTNAME": "PMSCMRLINE"
   },
   {
      "ASSETATTRID": "PMRDPCLSWS_SWIDS",
      "CHANGEBY": "MAXADMIN",
      "CHANGEDATE": "2010-06-24T00:04:29-04:00",
      "CLASSSPECID": 944,
      "CLASSSTRUCTUREID": "8100210",
      "DISPLAYSEQUENCE": 17,
      "ITEMNUM": "PMRDP_0201A",
      "MANDATORY": false,
      "ORGID": "PMSCIBM",
      "PDOWNERID": 69,
      "PDSPECID": 387,
      "REFOBJECTID": 68,
      "REFOBJECTLINENUM": 1,
      "REFOBJECTNAME": "PMSCMRLINE"
   },
   {
      "ALNVALUE": "CANPROJECT",
      "ASSETATTRID": "PMRDPCLVSRV_DELETE_MPNUM",
      "CHANGEBY": "MAXADMIN",
      "CHANGEDATE": "2010-06-24T00:04:29-04:00",
      "CLASSSPECID": 945,
      "CLASSSTRUCTUREID": "8100210",
      "DISPLAYSEQUENCE": 21,
      "ITEMNUM": "PMRDP_0201A",
      "MANDATORY": false,
      "ORGID": "PMSCIBM",
      "PDOWNERID": 69,
      "PDSPECID": 391,
      "REFOBJECTID": 68,
      "REFOBJECTLINENUM": 1,
      "REFOBJECTNAME": "PMSCMRLINE"
   },
   {
      "ALNVALUE": "NEWPROJECT",
      "ASSETATTRID": "PMRDPCLVSRV_MPNUM",
      "CHANGEBY": "MAXADMIN",
      "CHANGEDATE": "2010-06-24T00:04:28-04:00",
      "CLASSSPECID": 946,
      "CLASSSTRUCTUREID": "8100210",
      "DISPLAYSEQUENCE": 4,
      "ITEMNUM": "PMRDP_0201A",
      "MANDATORY": false,
      "ORGID": "PMSCIBM",
      "PDOWNERID": 69,
      "PDSPECID": 373,
      "REFOBJECTID": 68,
      "REFOBJECTLINENUM": 1,
      "REFOBJECTNAME": "PMSCMRLINE"
   },
   {
      "ALNVALUE": "NOTIFYPENDINGDELETEPR",
      "ASSETATTRID": "PMRDPCLVSRV_NOTIFY_MPNUM",
```


Archived

```
                    "CHANGEBY": "MAXADMIN",
                    "CHANGEDATE": "2010-06-24T00:04:29-04:00",
                    "CLASSSPECID": 947,
                    "CLASSSTRUCTUREID": "8100210",
                    "DISPLAYSEQUENCE": 20,
                    "ITEMNUM": "PMRDP_0201A",
                    "MANDATORY": false,
                    "ORGID": "PMSCIBM",
                    "PDOWNERID": 69,
                    "PDSPECID": 390,
                    "REFOBJECTID": 68,
                    "REFOBJECTLINENUM": 1,
                    "REFOBJECTNAME": "PMSCMRLINE"
                }
            ]
        }
    ]
  }
 }
}
```


All these parameters represent the definitions of the request. You can see the request in the Maximo console as draft status by opening the View Catalog Requests application (**Go To** → **Service Request Manager Catalog** → **View Catalog Requests**). Press Enter to list all the requests. See Figure 12-6.

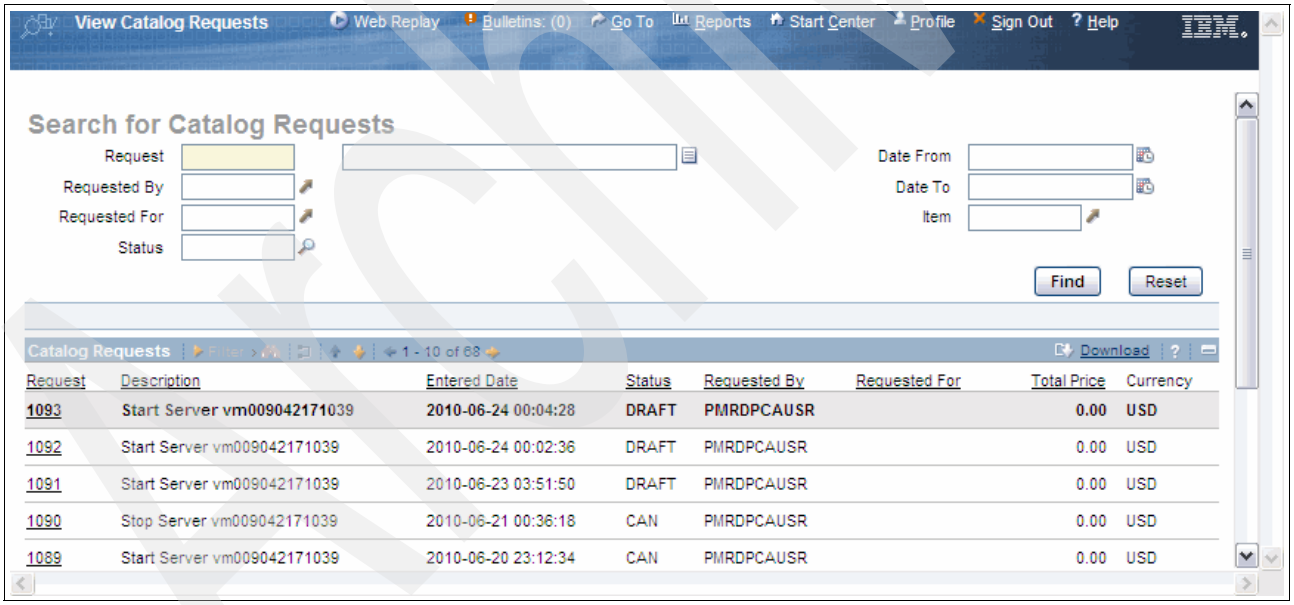


Figure 12-6 List of Catalog requests

By opening the request, you see the details of your request (Figure 12-7) and the items added (Figure 12-8).

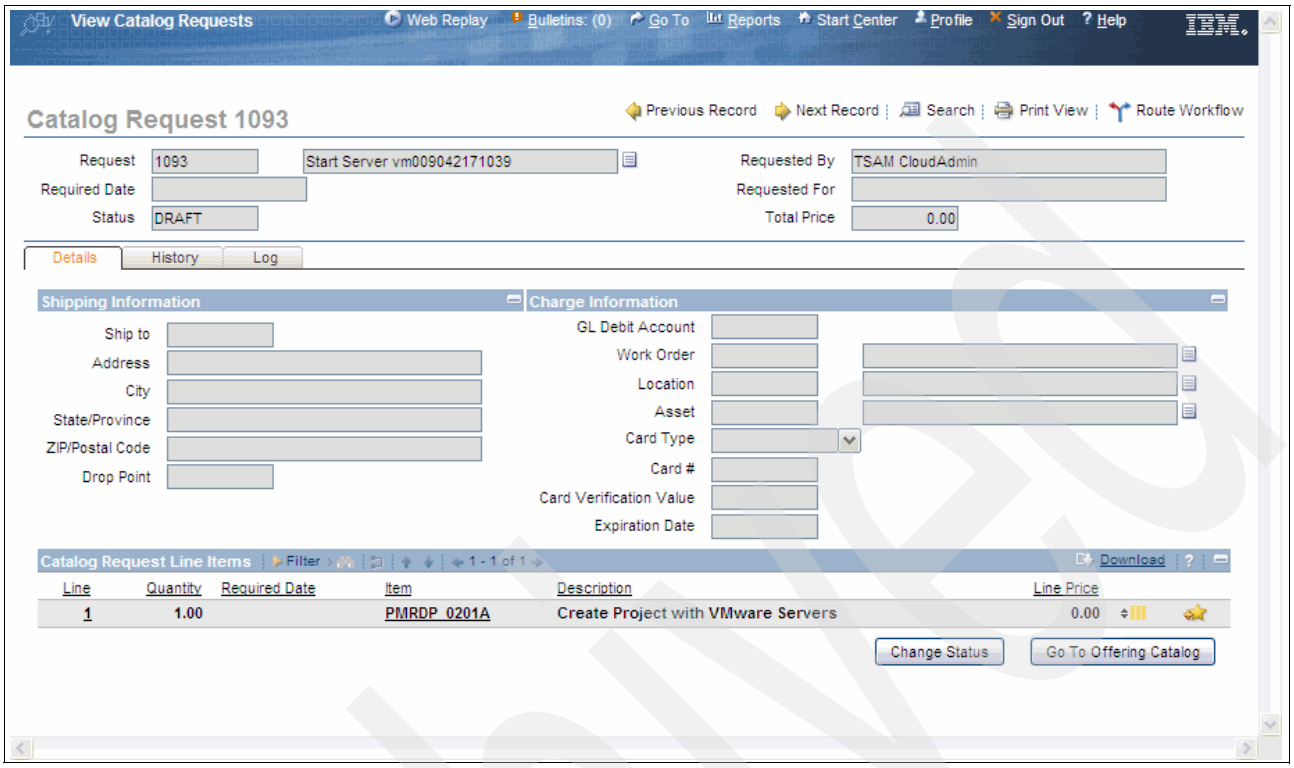


Figure 12-7 Details of the request

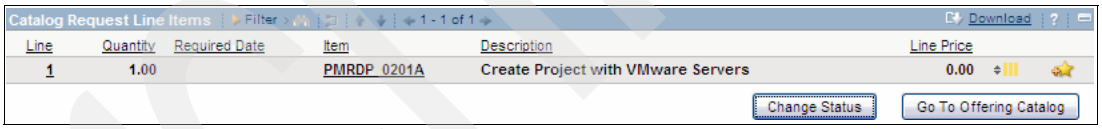


Figure 12-8 Items added in the request

The items represent the offering of the request. Each offering has the required parameters that must be provided before changing the status of the request.

Archive

Click the item to see the details, as shown in Figure 12-9.

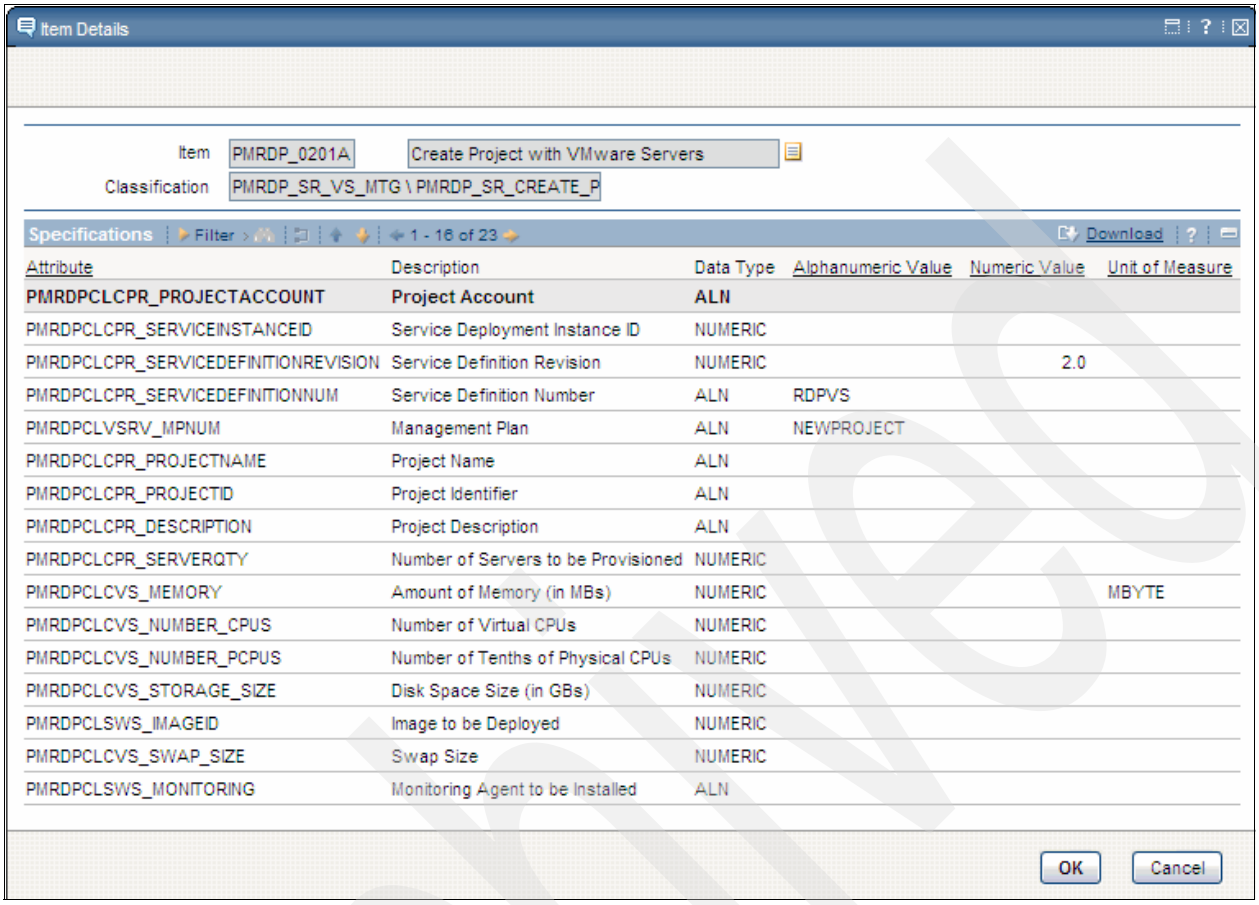


Figure 12-9 Required parameters for the offering

12.5.6 Updating the request with the required information

The request now must receive the attributes to provision a new virtual machine. The process to update the request is the same as to create it. However, for now, you must input all the attributes in the REST interface.

The URL in Example 12-7 on page 346 can be used to update the request with all the attributes. All attributes between the opening and closing angle brackets (< >) in the example must be changed for the correspondent value. The information about each attribute is in Table 12-2 on page 347.

Note: The URL is an example only and can change based on the specifics of your Tivoli Service Automation Manager environment.

Example 12-7 Updating the request with the required information

```
https://itsam72.ral.ibm.com:9443/maxrest/rest/os/PMZHBR1_PMSCMRCREATE/69?_format=json&MRID=
69&STATUS=WAPPR&PMSCMRLINE.1.MRLINEID=68&PMSCMRLINE.1.COMMODITY=VSERVER&PMSCMRLINE.1.COMMOD
ITYGROUP=SRVAUTOM&PMSCMRLINE.1.DESCRIPTION=Create%20Project%20with%20VMware%20Servers&PMSCM
RLINE.1.PDSPEC.1.ALNVALUE=Request%20using%20REST%20API&PMSCMRLINE.1.PDSPEC.1.REFOBJECTID=26
0&PMSCMRLINE.1.PDSPEC.1.REFOBJECTNAME=PMSCMRLINE&PMSCMRLINE.1.PDSPEC.1.ASSETATTRID=PMRDPCLC
PR_DESCRIPTION&PMSCMRLINE.1.PDSPEC.1.SECTION=&PMSCMRLINE.1.PDSPEC.1.PDSPECID=4375&PMSCMRLIN
E.1.PDSPEC.2.ALNVALUE=ITSO&PMSCMRLINE.1.PDSPEC.2.REFOBJECTID=260&PMSCMRLINE.1.PDSPEC.2.REFO
BJECTNAME=PMSCMRLINE&PMSCMRLINE.1.PDSPEC.2.ASSETATTRID=PMRDPCLCPR_PERSONGROUP&PMSCMRLINE.1.
PDSPEC.2.SECTION=&PMSCMRLINE.1.PDSPEC.2.PDSPECID=4387&PMSCMRLINE.1.PDSPEC.3.ALNVALUE=ITSOPr
oject&PMSCMRLINE.1.PDSPEC.3.REFOBJECTID=260&PMSCMRLINE.1.PDSPEC.3.REFOBJECTNAME=PMSCMRLINE&
PMSCMRLINE.1.PDSPEC.3.ASSETATTRID=PMRDPCLCPR_PROJECTNAME&PMSCMRLINE.1.PDSPEC.3.SECTION=&PMS
CMRLINE.1.PDSPEC.3.PDSPECID=4373&PMSCMRLINE.1.PDSPEC.4.NUMVALUE=1&PMSCMRLINE.1.PDSPEC.4.REF
OBJECTID=260&PMSCMRLINE.1.PDSPEC.4.REFOBJECTNAME=PMSCMRLINE&PMSCMRLINE.1.PDSPEC.4.ASSETATTR
ID=PMRDPCLCPR_SERVERQTY&PMSCMRLINE.1.PDSPEC.4.SECTION=&PMSCMRLINE.1.PDSPEC.4.PDSPECID=4376&
PMSCMRLINE.1.PDSPEC.5.NUMVALUE=1024&PMSCMRLINE.1.PDSPEC.5.REFOBJECTID=260&PMSCMRLINE.1.PDSP
EC.5.REFOBJECTNAME=PMSCMRLINE&PMSCMRLINE.1.PDSPEC.5.ASSETATTRID=PMRDPCLCVS_MEMORY&PMSCMRLIN
E.1.PDSPEC.5.SECTION=&PMSCMRLINE.1.PDSPEC.5.PDSPECID=4377&PMSCMRLINE.1.PDSPEC.6.NUMVALUE=1&
PMSCMRLINE.1.PDSPEC.6.REFOBJECTID=260&PMSCMRLINE.1.PDSPEC.6.REFOBJECTNAME=PMSCMRLINE&PMSCMR
LINE.1.PDSPEC.6.ASSETATTRID=PMRDPCLCVS_NUMBER_CPUS&PMSCMRLINE.1.PDSPEC.6.SECTION=&PMSCMRLIN
E.1.PDSPEC.6.PDSPECID=4378&PMSCMRLINE.1.PDSPEC.7.NUMVALUE=5&PMSCMRLINE.1.PDSPEC.7.REFOBJECT
ID=260&PMSCMRLINE.1.PDSPEC.7.REFOBJECTNAME=PMSCMRLINE&PMSCMRLINE.1.PDSPEC.7.ASSETATTRID=PMR
DPCLCVS_NUMBER_PCPUS&PMSCMRLINE.1.PDSPEC.7.SECTION=&PMSCMRLINE.1.PDSPEC.7.PDSPECID=4379&PMS
CMRLINE.1.PDSPEC.8.ALNVALUE=%2Fcloud%2Frest%2Fpools%2F3%2F&PMSCMRLINE.1.PDSPEC.8.REFOBJECTI
D=260&PMSCMRLINE.1.PDSPEC.8.REFOBJECTNAME=PMSCMRLINE&PMSCMRLINE.1.PDSPEC.8.ASSETATTRID=PMRD
PCLCVS_RESGRPNUM&PMSCMRLINE.1.PDSPEC.8.SECTION=&PMSCMRLINE.1.PDSPEC.8.PDSPECID=4384&PMSCMRL
INE.1.PDSPEC.9.NUMVALUE=5&PMSCMRLINE.1.PDSPEC.9.REFOBJECTID=260&PMSCMRLINE.1.PDSPEC.9.REFOB
JECTNAME=PMSCMRLINE&PMSCMRLINE.1.PDSPEC.9.ASSETATTRID=PMRDPCLCVS_STORAGE_SIZE&PMSCMRLINE.1.
PDSPEC.9.SECTION=&PMSCMRLINE.1.PDSPEC.9.PDSPECID=4380&PMSCMRLINE.1.PDSPEC.10.NUMVALUE=1.000
&PMSCMRLINE.1.PDSPEC.10.REFOBJECTID=260&PMSCMRLINE.1.PDSPEC.10.REFOBJECTNAME=PMSCMRLINE&PMS
CMRLINE.1.PDSPEC.10.ASSETATTRID=PMRDPCLCVS_SWAP_SIZE&PMSCMRLINE.1.PDSPEC.10.SECTION=&PMSCMR
LINE.1.PDSPEC.10.PDSPECID=4382&PMSCMRLINE.1.PDSPEC.11.NUMVALUE=1&PMSCMRLINE.1.PDSPEC.11.REF
OBJECTID=260&PMSCMRLINE.1.PDSPEC.11.REFOBJECTNAME=PMSCMRLINE&PMSCMRLINE.1.PDSPEC.11.ASSETAT
TRID=PMRDPCLSWS_IMAGEID&PMSCMRLINE.1.PDSPEC.11.SECTION=&PMSCMRLINE.1.PDSPEC.11.PDSPECID=438
1
```

Table 12-2 information about the attributes

Attribute	Value	Comment
https://itsam72.ral.ibm.com:9443/maxrest/rest/os/PMZHBR1_PMSCMRCREATE/69?	Host name, IP address, and MRID	MRID is the PMSCMR unique identifier. In the Example 12-6 on page 336, the MRID is 69.
_json	json	-
MRID	MRID	The PMSCMR unique identifier
STATUS	WAPPR	This is the status Waiting on Approval. This status will start the process workflow.
PMSCMRLINE.1.MRLINEID	PMSCMRLINE.1.MRLINEID	The PMSCMRLINE unique identifier
PMSCMRLINE.1.COMMODITY	VSERVER	The value defined in the Tivoli Service Automation Manager
PMSCMRLINE.1.COMMODITYGROUP	SRVAUTOM	The value defined in the Tivoli Service Automation Manager
PMSCMRLINE.1.DESCRIPTION	description	The description of the project
Note: The following attributes represent each object inside the array PDSPEC in the Example 12-6 on page 336. Objects inside array PDSPEC is represented by brackets. To update the objects, change the signal % in the following attributes to the sequential number. The quantity of objects can vary for each offering.		
PMSCMRLINE.1.PDSPEC.%.ALNVALUE PMSCMRLINE.1.PDSPEC.%.NUMVALUE	ALNVALUE - String value NUMVALUE - Numeric value	The value to be set for the attribute
PMSCMRLINE.1.PDSPEC.%.REFOBJECTID	The REFORBJECTID of the object	-
PMSCMRLINE.1.PDSPEC.%.REFOBJECTNAME	PMSCMRLINE	The object in reference. This is the only acceptable value
PMSCMRLINE.1.PDSPEC.%.ASSETATTRID	The name of the attribute	The list of the attributes can be found in the HTTP response.
PMSCMRLINE.1.PDSPEC.%.SECTION	Usually null	The section for a group of attributes. This is usually null.
PMSCMRLINE.1.PDSPEC.$.PDSPECID	unique value	-

Archive

Using the HTTP response from Example 12-6 on page 336, the parameters are listed in Example 12-8.

Example 12-8 HTTP response

```
MRID=69
STATUS=WAPPR
PMSCMRLINE.1.MRLINEID=68
PMSCMRLINE.1.COMMODITY=VSERVER
PMSCMRLINE.1.COMMODITYGROUP=SRVAUTOM
PMSCMRLINE.1.DESCRIPTION=Create%20Project%20with%20VMware%20Servers
PMSCMRLINE.1.PDSPEC.1.ALNVALUE=Request%20using%20REST%20API
PMSCMRLINE.1.PDSPEC.1.REFOBJECTID=68
PMSCMRLINE.1.PDSPEC.1.REFOBJECTNAME=PMSCMRLINE
PMSCMRLINE.1.PDSPEC.1.ASSETATTRID=PMRDPCLCPR_DESCRIPTION
PMSCMRLINE.1.PDSPEC.1.SECTION=
PMSCMRLINE.1.PDSPEC.1.PDSPECID=376
PMSCMRLINE.1.PDSPEC.2.ALNVALUE=ITSO
PMSCMRLINE.1.PDSPEC.2.REFOBJECTID=68
PMSCMRLINE.1.PDSPEC.2.REFOBJECTNAME=PMSCMRLINE
PMSCMRLINE.1.PDSPEC.2.ASSETATTRID=PMRDPCLCPR_PERSONGROUP
PMSCMRLINE.1.PDSPEC.2.SECTION=
PMSCMRLINE.1.PDSPEC.2.PDSPECID=388
PMSCMRLINE.1.PDSPEC.3.ALNVALUE=ITSOProject
PMSCMRLINE.1.PDSPEC.3.REFOBJECTID=68
PMSCMRLINE.1.PDSPEC.3.REFOBJECTNAME=PMSCMRLINE
PMSCMRLINE.1.PDSPEC.3.ASSETATTRID=PMRDPCLCPR_PROJECTNAME
PMSCMRLINE.1.PDSPEC.3.SECTION=
PMSCMRLINE.1.PDSPEC.3.PDSPECID=374
PMSCMRLINE.1.PDSPEC.4.NUMVALUE=1
PMSCMRLINE.1.PDSPEC.4.REFOBJECTID=68
PMSCMRLINE.1.PDSPEC.4.REFOBJECTNAME=PMSCMRLINE
PMSCMRLINE.1.PDSPEC.4.ASSETATTRID=PMRDPCLCPR_SERVERQTY
PMSCMRLINE.1.PDSPEC.4.SECTION=
PMSCMRLINE.1.PDSPEC.4.PDSPECID=377
PMSCMRLINE.1.PDSPEC.5.NUMVALUE=1024
PMSCMRLINE.1.PDSPEC.5.REFOBJECTID=68
PMSCMRLINE.1.PDSPEC.5.REFOBJECTNAME=PMSCMRLINE
PMSCMRLINE.1.PDSPEC.5.ASSETATTRID=PMRDPCLCVS_MEMORY
PMSCMRLINE.1.PDSPEC.5.SECTION=
PMSCMRLINE.1.PDSPEC.5.PDSPECID=378
PMSCMRLINE.1.PDSPEC.6.NUMVALUE=1
PMSCMRLINE.1.PDSPEC.6.REFOBJECTID=68
PMSCMRLINE.1.PDSPEC.6.REFOBJECTNAME=PMSCMRLINE
PMSCMRLINE.1.PDSPEC.6.ASSETATTRID=PMRDPCLCVS_NUMBER_CPUS
PMSCMRLINE.1.PDSPEC.6.SECTION=
PMSCMRLINE.1.PDSPEC.6.PDSPECID=379
PMSCMRLINE.1.PDSPEC.7.NUMVALUE=5
PMSCMRLINE.1.PDSPEC.7.REFOBJECTID=68
PMSCMRLINE.1.PDSPEC.7.REFOBJECTNAME=PMSCMRLINE
PMSCMRLINE.1.PDSPEC.7.ASSETATTRID=PMRDPCLCVS_NUMBER_PCPUS
PMSCMRLINE.1.PDSPEC.7.SECTION=
PMSCMRLINE.1.PDSPEC.7.PDSPECID=380
PMSCMRLINE.1.PDSPEC.8.ALNVALUE=%2Fcloud%2Frest%2Fpools%2F3%2F
PMSCMRLINE.1.PDSPEC.8.REFOBJECTID=68
PMSCMRLINE.1.PDSPEC.8.REFOBJECTNAME=PMSCMRLINE
PMSCMRLINE.1.PDSPEC.8.ASSETATTRID=PMRDPCLCVS_RESGRPNUM
PMSCMRLINE.1.PDSPEC.8.SECTION=
PMSCMRLINE.1.PDSPEC.8.PDSPECID=385
PMSCMRLINE.1.PDSPEC.9.NUMVALUE=5
PMSCMRLINE.1.PDSPEC.9.REFOBJECTID=68
```

```
PMSCMRLINE.1.PDSPEC.9.REFOBJECTNAME=PMSCMRLINE
PMSCMRLINE.1.PDSPEC.9.ASSETATTRID=PMRDPCLCVS_STORAGE_SIZE
PMSCMRLINE.1.PDSPEC.9.SECTION=
PMSCMRLINE.1.PDSPEC.9.PDSPECID=381
PMSCMRLINE.1.PDSPEC.10.NUMVALUE=1.000
PMSCMRLINE.1.PDSPEC.10.REFOBJECTID=68
PMSCMRLINE.1.PDSPEC.10.REFOBJECTNAME=PMSCMRLINE
PMSCMRLINE.1.PDSPEC.10.ASSETATTRID=PMRDPCLCVS_SWAP_SIZE
PMSCMRLINE.1.PDSPEC.10.SECTION=
PMSCMRLINE.1.PDSPEC.10.PDSPECID=383
PMSCMRLINE.1.PDSPEC.11.NUMVALUE=1
PMSCMRLINE.1.PDSPEC.11.REFOBJECTID=68
PMSCMRLINE.1.PDSPEC.11.REFOBJECTNAME=PMSCMRLINE
PMSCMRLINE.1.PDSPEC.11.ASSETATTRID=PMRDPCLSWS_IMAGEID
PMSCMRLINE.1.PDSPEC.11.SECTION=
PMSCMRLINE.1.PDSPEC.11.PDSPECID=382
```

Note: You do not need to use all the ASSETATTRID from the offerings.

After the HTTP POST is performed, the return is a JSON response that can be used as a confirmation that the REQUEST has been updated.

To check the status of the request, open the View Catalog Requests application (**Go To** → **Service Request Manager Catalog** → **View Catalog Requests**), as shown in Figure 12-10.

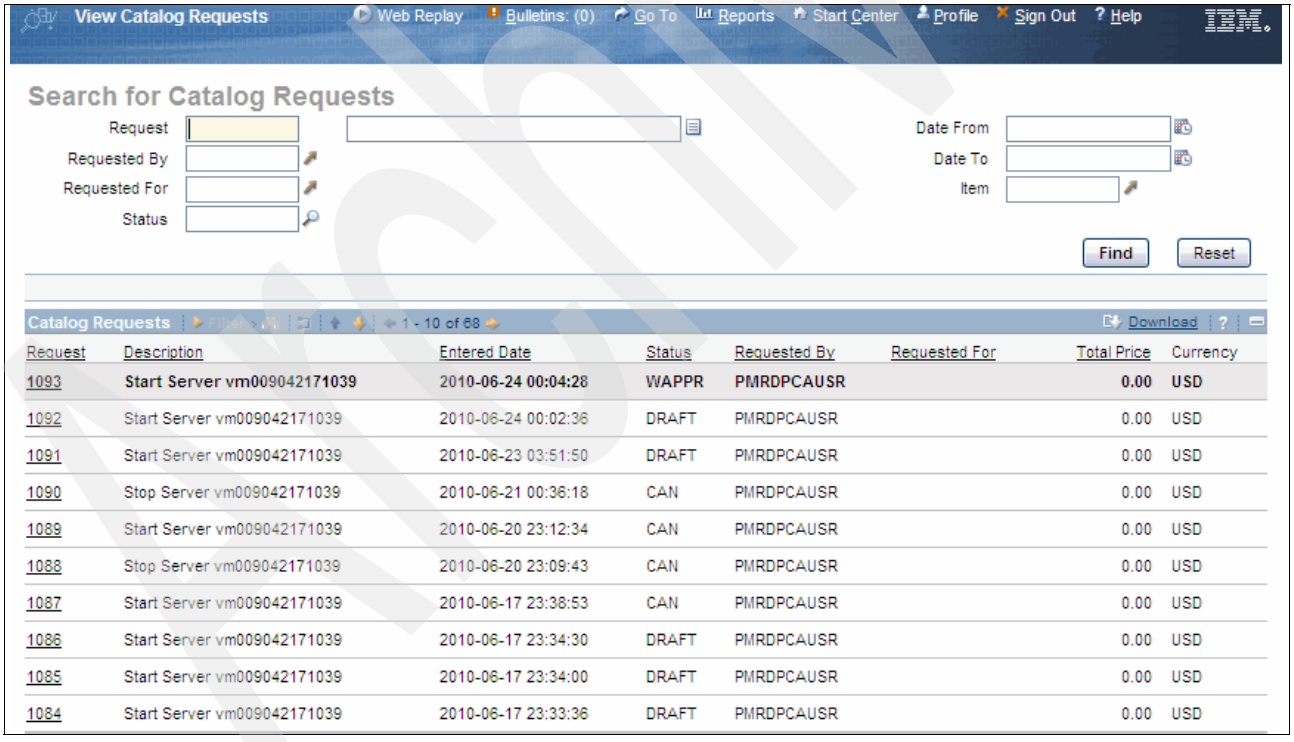


Figure 12-10 List of Catalog Requests

Open the Catalog Request and verify whether the parameters were updated correctly. See Figure 12-11 on page 350.

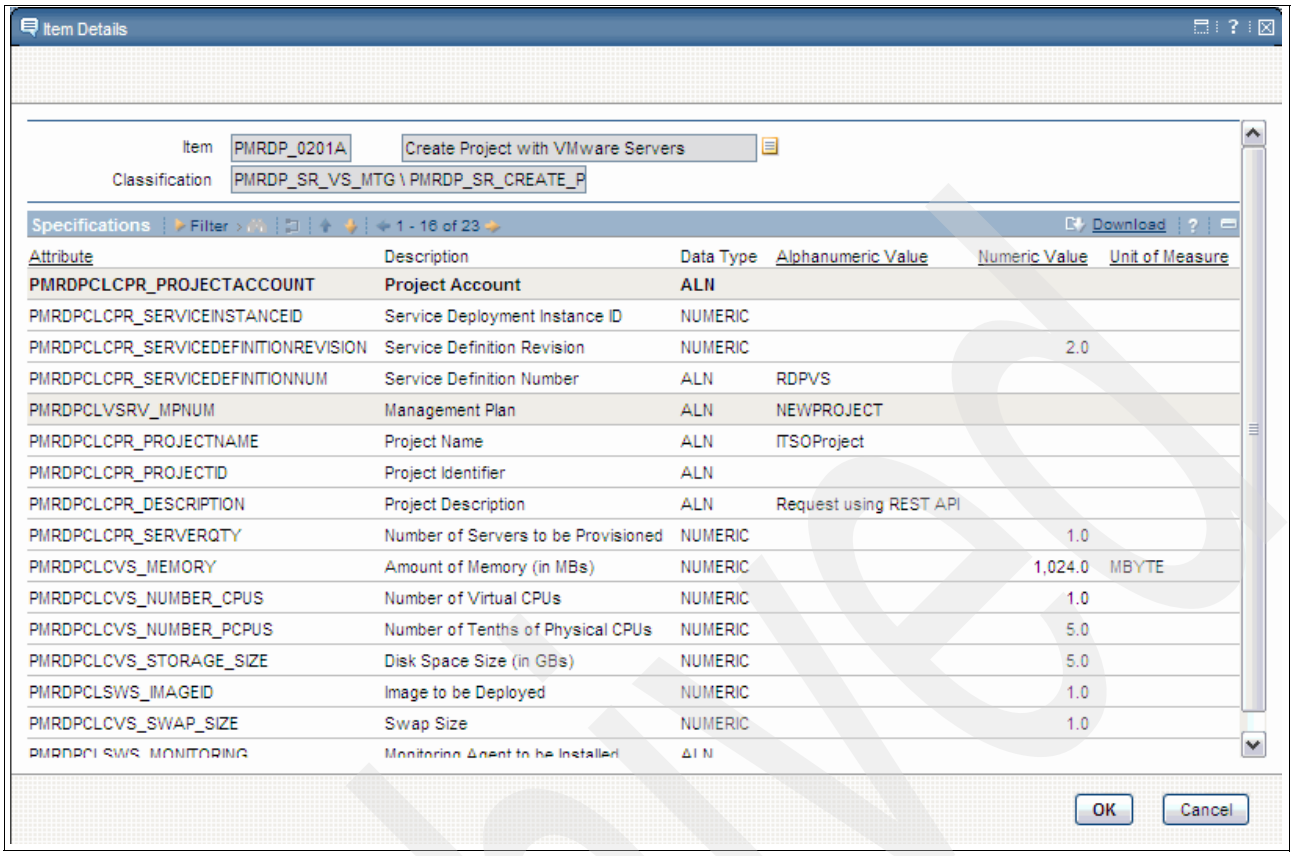


Figure 12-11 The list of parameters updated

After approval of this request, a service request is created to perform the transaction. To see the request, open the Service Requests application (**Go To** → **Service Desk** → **Service Requests**), as shown in Figure 12-12:

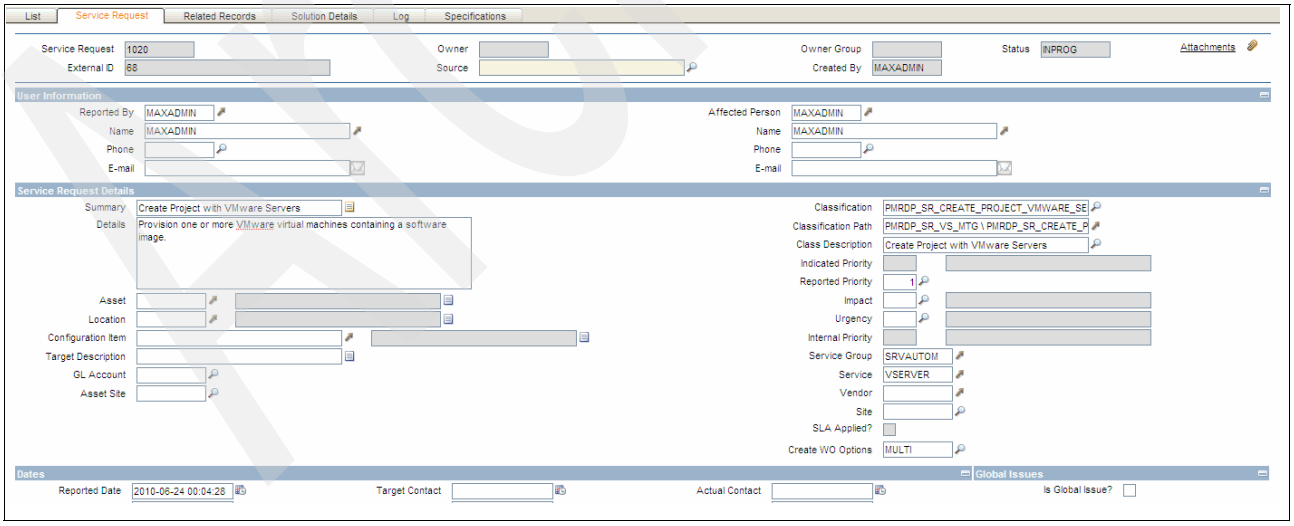


Figure 12-12 Example of Service Request in progress

When provisioning is finished, the service request status is changed to Resolved.

12.6 Monitoring the business service

Event Management and Monitoring (EMM) includes many types of monitoring such as measuring of both IT and non-IT infrastructure health, detecting component outages or degradations, and monitoring IT resource utilization. In a Cloud Computing environment, however, you must be capable of monitoring both the underlying IT infrastructure and the business services and applications that run on top of them, particularly with regards to the quality of service delivery.

12.6.1 Selecting monitoring agents

Because we are concerned primarily with service delivery and specifically with ensuring that contracted service levels are achieved, this scenario uses IBM Tivoli Composite Application Manager for Transactions as the primary monitoring component, in conjunction with IBM Tivoli Monitoring. IBM Tivoli Composite Application Manager for Transactions includes three types of monitoring capabilities:

- Internet Service Monitoring, which measures the performance and availability of various network services against defined service level parameters
- Response Time, which provides automated measurements of HTTP and HTTPS service response time
- Transaction Tracking, which tracks user transactions through a composite application and measures the time each application component spends handling the transaction

For this scenario, we deployed the Internet Service Monitoring and Transaction Tracking components of IBM Tivoli Composite Application Manager for Transactions. Internet Service Monitoring executes HTTP and other service requests, and measures performance and availability against user-defined service levels; Transaction Tracking traces each transaction through the business service and identifies any bottlenecks which may be impacting service levels.

12.6.2 Configuring Internet Service Monitoring

To configure Internet Service Monitoring to probe and measure service availability and performance, you must use the Tivoli Enterprise Portal component of IBM Tivoli Monitoring:

1. In the Tivoli Enterprise Portal Client, click **ISM Configuration** icon. See Figure 12-13.

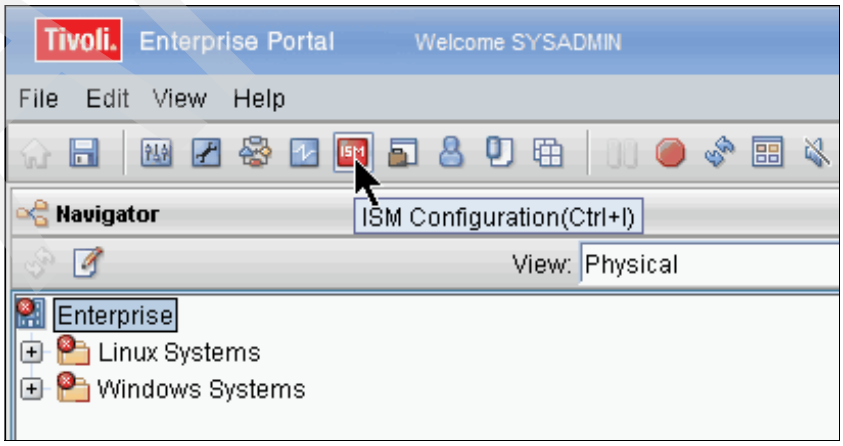


Figure 12-13 Accessing the ISM Configuration in the Tivoli Enterprise Portal

2. In the dialog box, click the **Create Profile** icon at the top left to create a new profile. See Figure 12-14.

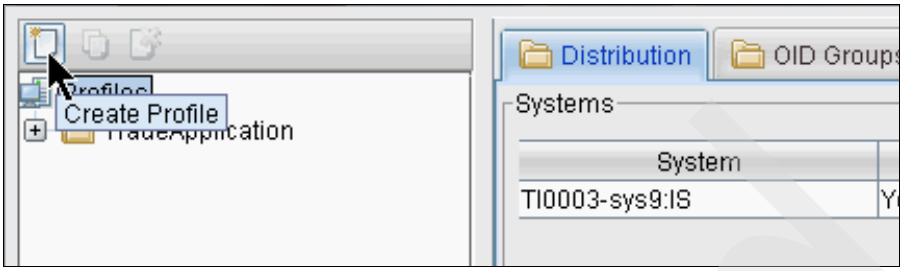


Figure 12-14 Creating a new Internet Service Monitoring profile

3. In the dialog window that opens, select a name for your application. This application name refers to the name of the monitored business service; you enter it at several points during component configuration.

> **Important:** Be sure to use the same name for the application throughout the scenario.

For our scenario, we enter `TradeApplication` as our application name.

After creating the profile, create monitoring elements. Each element represents an instance of a service that must be monitored and includes details about the application protocol to use, timeout, poll interval, and service level thresholds, among other parameters. The configurable parameters vary according to the type of monitoring element being created.

To create an element, perform the following steps:

1. Select your profile by clicking it.
2. Select the type of monitor from the menu in the “Add monitor type to profile” pane.
3. Click **Add**, as shown in Figure 12-15.

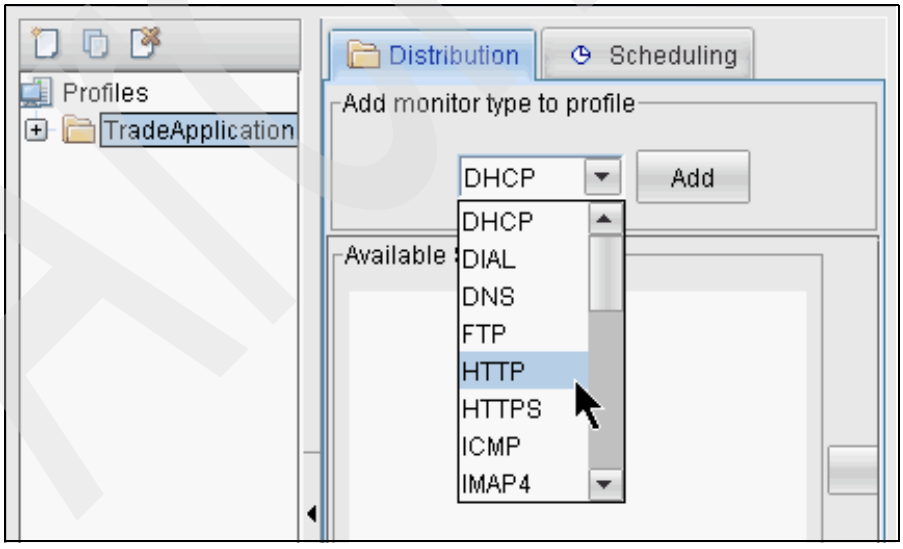


Figure 12-15 Adding a new HTTP monitoring element to the ISM profile

Configuring an HTTP monitoring element

The most basic way of monitoring a web application is to use an HTTP (or HTTPS, if supported by the application) monitoring element. Figure 12-16 shows the selection of HTTP as the monitoring element type.

This section details how to configure the element in more detail.

After adding a new HTTP element, a panel shows the list of all HTTP elements within the same profile, including any that already exist, at the top. If you click an element to select it, the element details are at the bottom. See Figure 12-16.

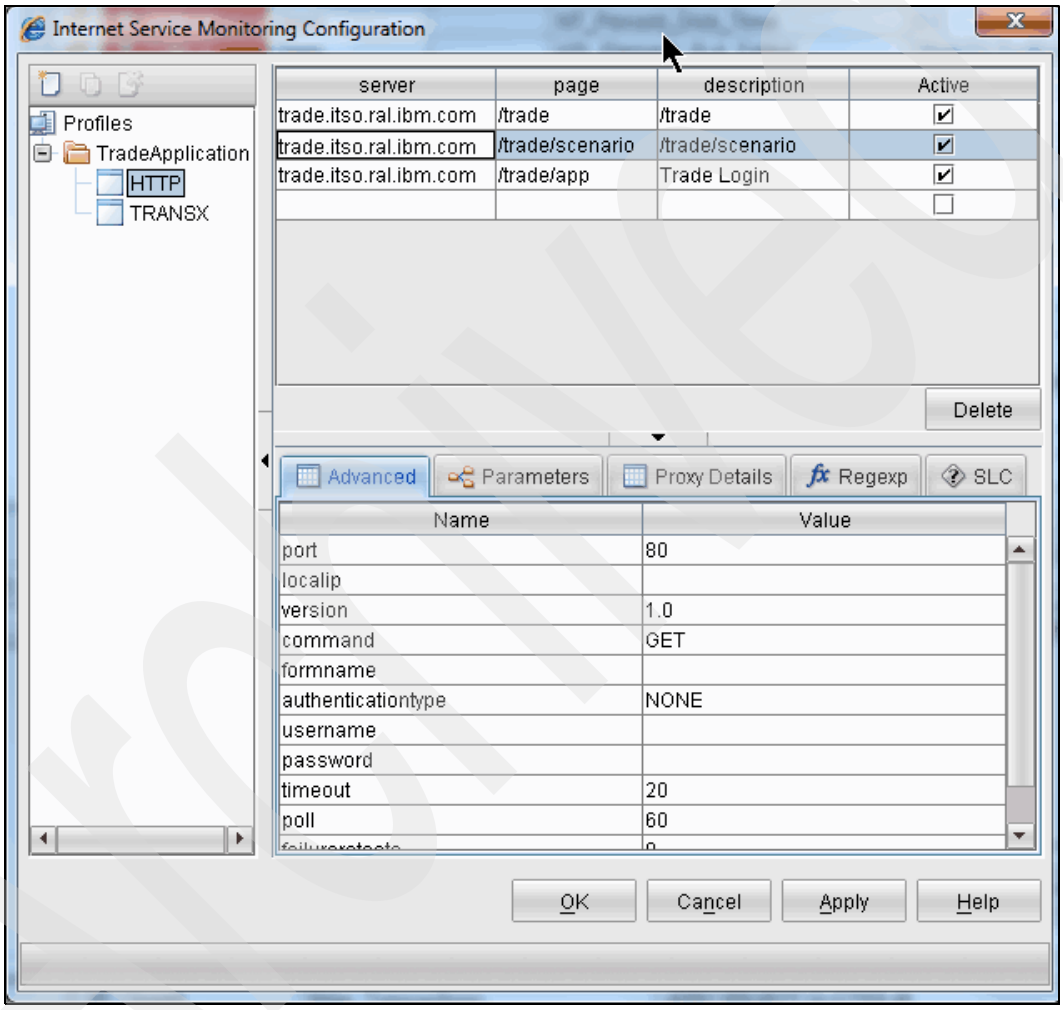


Figure 12-16 HTTP Element Configuration for Internet Service Monitoring

Although we describe the most commonly used parameters for an HTTP monitor element, other less commonly used parameters and parameters for other types of Internet Service Monitoring elements are not covered. For those, see the *IBM Tivoli Composite Application Manager for Transactions User's Guide*, which provides a full reference for all monitor attributes:

http://publib.boulder.ibm.com/infocenter/tivihelp/v24r1/index.jsp?topic=/com.ibm.itcamt.doc_7.2.0.1/tt/dita/topic/ug.html

To create an HTTP element, enter information (from left to right) in the columns at the top of the configuration window: the host name or IP address of the HTTP server (which can be a

cluster address, such as a Virtual IP, also known as a VIP), the URI path of the page to be requested, and a short description. For the purposes of this scenario, we have made the description the same as the page URI path, which facilitates correlating events later. See Figure 12-17.

server	page	description	Active
trade.itso.ral.ibm.com	/trade	/trade	☑
trade.itso.ral.ibm.com	/trade/scenario	/trade/scenario	☑

Figure 12-17 Basic configuration details for HTTP element

In the lower part of the pane are several tabs:

- Advanced, which has more detailed configuration parameters
- Parameters, which are the parameters to be passed in as part of the request, such as with a form
- Proxy Details, in case the monitored service must be accessed through a proxy server
- Regexp, which you use to check returned content for the presence (or absence) of one or more regular expressions
- SLC, in which you configure the service level classification parameters. These are used to assign a service level to the monitored service based on aspects such as response time, protocol-specific return codes, and the presence or absence of particular regular expressions

We describe the configuration of the Advanced and SLC tabs only at this point. For many basic HTTP probe configurations, you may not need to change any values in the Advanced tab at all from the defaults. Figure 12-18 shows that we have modified only the *timeout* and *poll* values.

Advanced | Parameters | Proxy Details | Regexp | SLC

Name	Value
port	80
localip	
version	1.0
command	GET
formname	
authenticationtype	NONE
username	
password	
timeout	20
poll	60
failureretests	0
retestinterval	0

Figure 12-18 Advanced configuration details for HTTP element

The timeout parameter controls how long the probe will wait for a response from the server before giving up; the poll parameter controls the interval between requests. Both values are in seconds. The value for poll obviously is greater than that for timeout.

Configuring service level classification

In many ways, the SLC tab is the most important one for any Internet Service Monitoring element, because it is in this tab that you configure the thresholds that are used to determine the service level classification assigned to each request cycle. The available classifications are GOOD, MARGINAL, and FAILED in decreasing order of service quality. Therefore, Internet Service Monitoring users must be careful about what values they select and should revisit monitor configurations periodically to be sure they reflect actual target service levels.

The Internet Service Monitoring configuration divides service level classification into condition groups, which are themselves divided into individual conditions. Each condition group is associated with a service level classification; the conditions within the group are combined together (using AND) for purposes of evaluation, and if all evaluate to true, the associated classification is assigned to the monitored service.

For an HTTP monitoring element, three condition groups are configured by default, although you can add or delete groups as needed. The first group checks HTTP response codes; the second and third groups both check the total time to complete the request, including the time to receive the full response from the server. Most users need a minimum to modify the time thresholds for the second and third groups. The second group, if true, results in a service level classification of FAILED; the third results in a classification of MARGINAL. Figure 12-19 shows the values we used for our element.

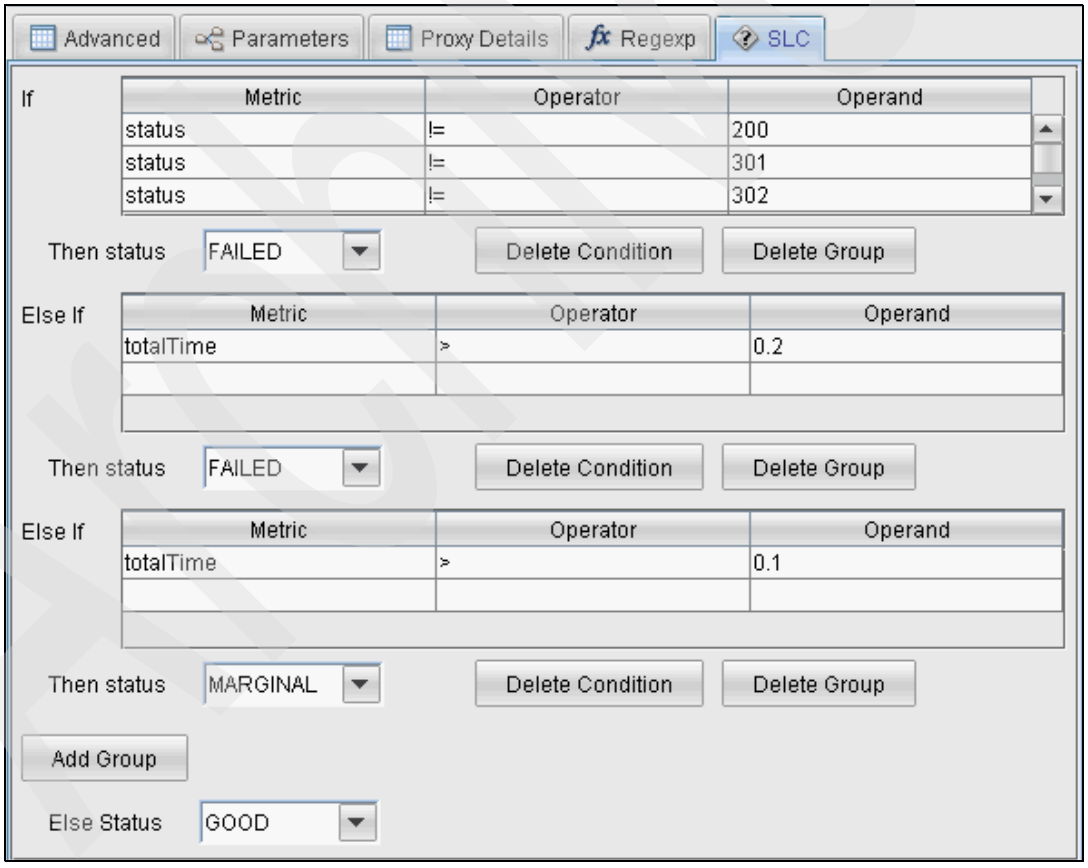


Figure 12-19 Service level classification configuration details for HTTP element

The configuration shown in Figure 12-19 on page 355 results in the following evaluation flow:

1. If the HTTP status code is not one of 200, 301, or 302, then classify the service level as FAILED and stop the request.
2. If the total time to send the request and receive the full response (including time for nameservice lookups and handshaking, if applicable) is greater than 0.2 seconds, classify the service level as FAILED and stop.
3. If the total time is greater than 0.1 seconds, classify the service level as MARGINAL and stop.
4. If none of the preceding three conditions (technically condition groups in Internet Service Monitoring) applies, then classify the service level as GOOD.

Distributing the profile to monitoring agents

With most Tivoli Enterprise Monitoring Agents, you only need to distribute situations to them, because situations determine the monitoring thresholds, which trigger the raising of events. But when deploying BM Tivoli Composite Application Manager for Transactions, you must also distribute monitoring profiles which define the additional configuration details that are required to monitor your application correctly. For Internet Service Monitoring, you do this within the same configuration window that you used to create the profile and monitoring element.

You may distribute profiles in two ways using the ISM configuration window:

- Distribute a single profile to one or more deployed agents.
- Distribute one or more profiles to a single deployed agent.

The exact mechanics for the two ways are similar but not exactly identical. We demonstrate both methods for profile distribution.

To distribute a single profile to one or more agents, select the profile by clicking it in the profile list pane at the left. See Figure 12-20, which shows the distributing of a single profile to one or more deployed Internet Service Monitoring agents. To the right is the list from which you can select monitor element types to add; below that are two lists showing the agents that are available for profile distribution and those to which this particular profile has already been distributed.

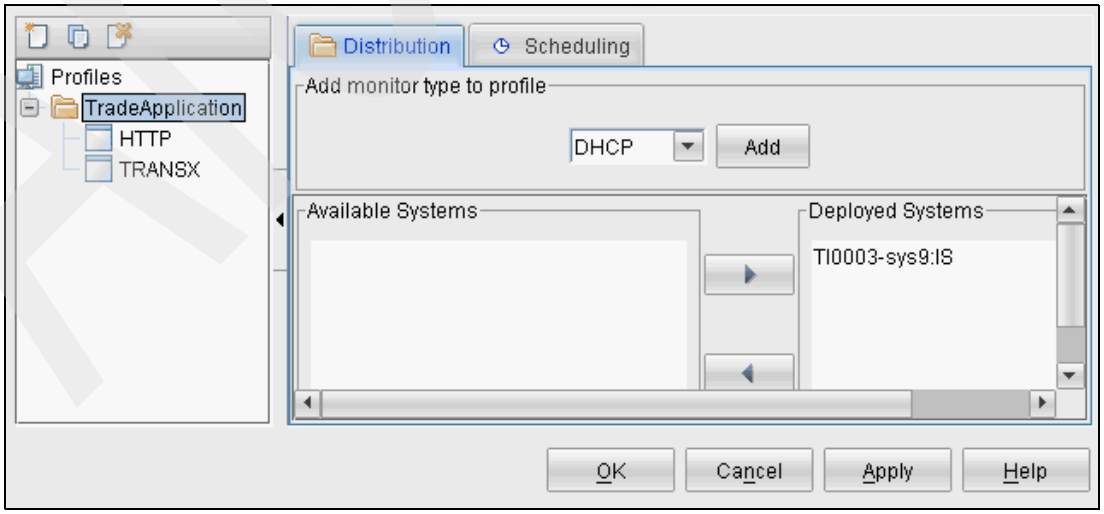


Figure 12-20 Distributing single profile

In Figure 12-20 on page 356, we have already distributed our profile to a single agent, `TI0003-sys9:IS`. If we had additional Internet Service Monitoring agents deployed in our infrastructure to which we had not yet distributed this profile, they would appear in the list on the left under Available Systems. Simply select any agents to be targeted for distribution from the Available Systems list, then click the button containing the arrow pointing to the right to move those agents into the Deployed Systems list. After you click either **OK** or **Apply**, the profile is distributed to the agents selected.

To distribute one or more profiles to a single selected deployed agent, click the top-level Profiles node in the profile list on the left. At the right, agents that are deployed in your infrastructure are listed. Select the agent to which you want to distribute profiles. Figure 12-21 shows a list of those profiles (on the left) that are available for distribution; on the right is a list of those profiles that are already distributed to this particular agent.

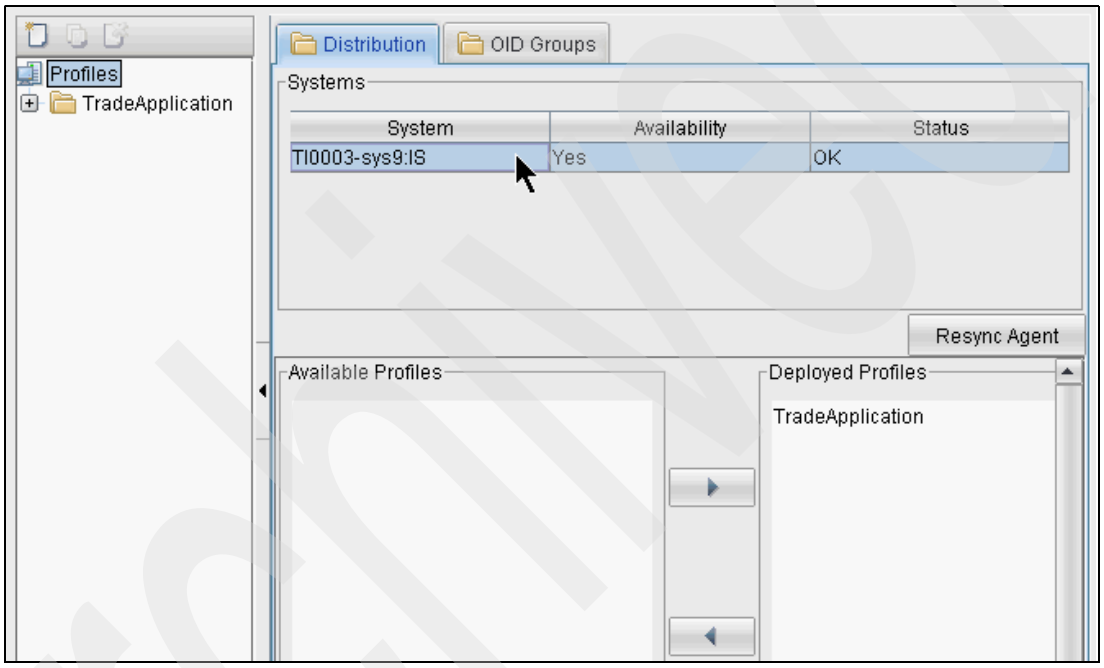


Figure 12-21 Distribution of profiles to a selected Internet Service Monitoring agent

To deploy a profile to the selected agent, simply move it from the Available Profiles list to the Deployed Profiles list and click **Apply**.

Because distributed profiles are stored as XML files at the agents to which they have been deployed, this method is also useful if those files should become corrupted or deleted. If you select an agent from the list at the top of the window and then click **Resync Agent**, IBM Tivoli Monitoring will redeploy all profiles previously distributed to that agent, overwriting any locally stored profiles and restoring the agent to the specified configuration.

Archive

12.6.3 Configuring Transaction Tracking

The Transaction Tracking component of IBM Tivoli Composite Application Manager for Transactions enables the tracing of a service transaction through each tier of the composite application. Although Internet Service Monitoring can emulate the user experience and measure adherence to contracted service levels, Transaction Tracking enables you to identify how much time a transaction is being handled at each tier and thereby identify which component or inter-component communication may be responsible for any service degradation. Combined with Internet Service Monitoring, Transaction Tracking provides a powerful platform for the management of business services.

The steps are as follows:

1. To configure Transaction Tracking, you must create an application configuration and then specify the types of transactions that belong to the application. You then associate this application's transactions with a distribution profile and distribute it to Transaction Collector agents. All of this configuration is done in the Application Management Configuration Editor, which you access by clicking **Application Management Configuration Editor** in the Tivoli Enterprise Portal toolbar, as shown in Figure 12-22.

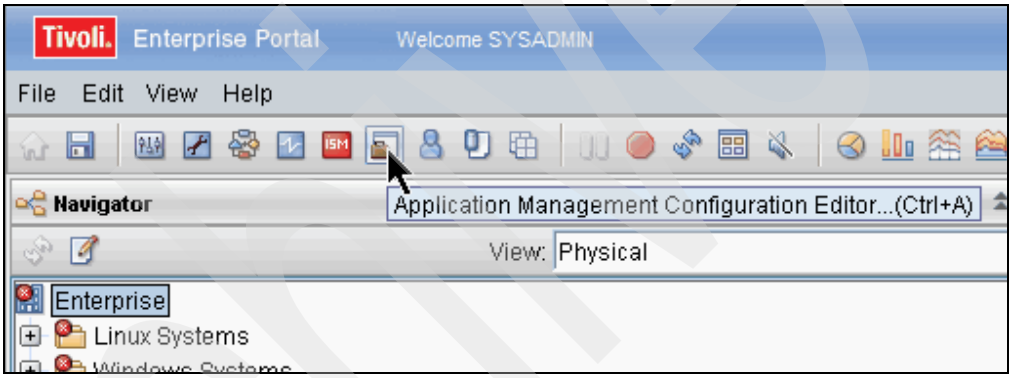


Figure 12-22 Accessing the Application Management Configuration Editor

The Application Management Configuration Editor is installed as part of the Application Management Console, which is a subcomponent of both Transaction Tracking and Response Time. The editor, shown in Figure 12-23 on page 359, is similar to the one that is used to configure Internet Service Monitoring.

Archived

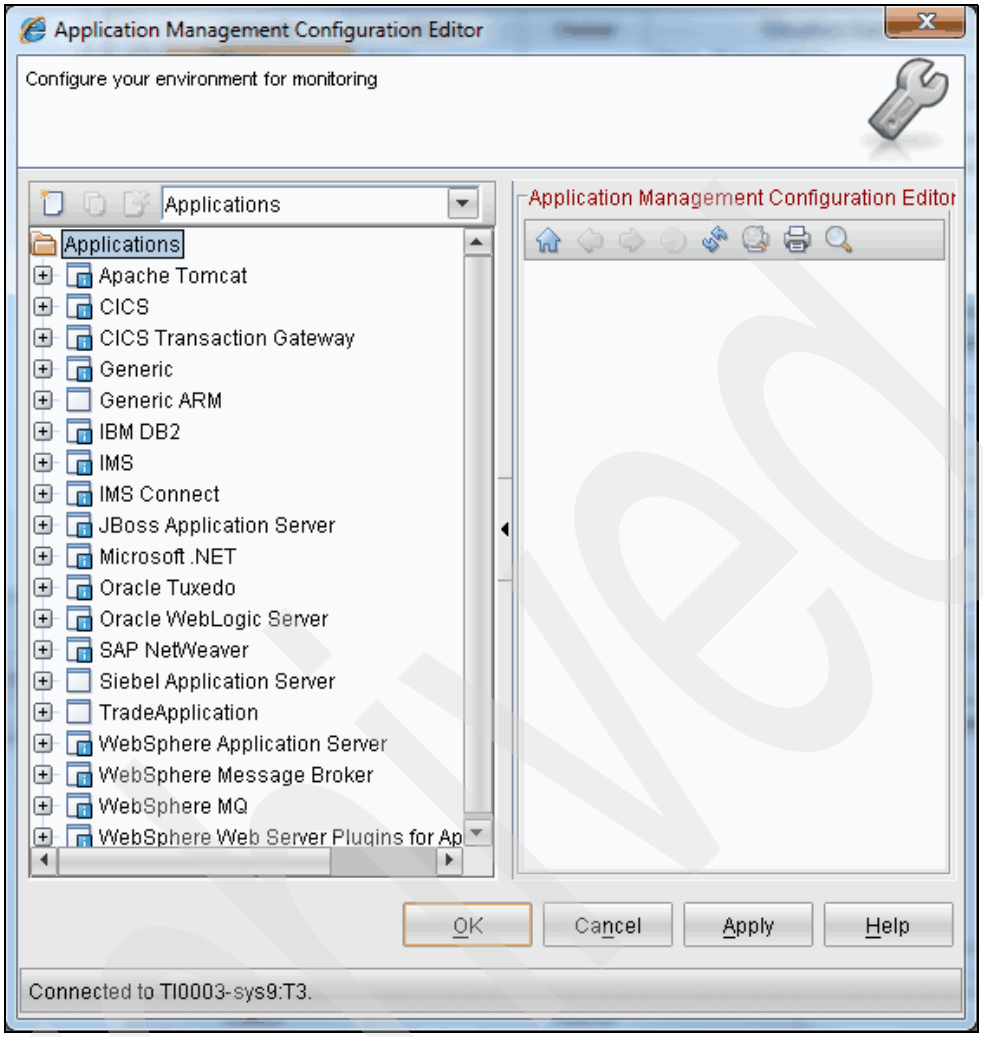


Figure 12-23 Application Management Configuration Editor window

On the left is a navigator tree, containing all configured applications. By default, applications are supplied, corresponding to each middleware application that Transaction Tracking is capable of monitoring. All transactions belonging to these included applications are associated with a distribution profile named Default.

Archived

Figure 12-24 shows the editor window with the Default profile selected on the left under the application IBM DB2 and All DB2 Subtransactions through ARM.

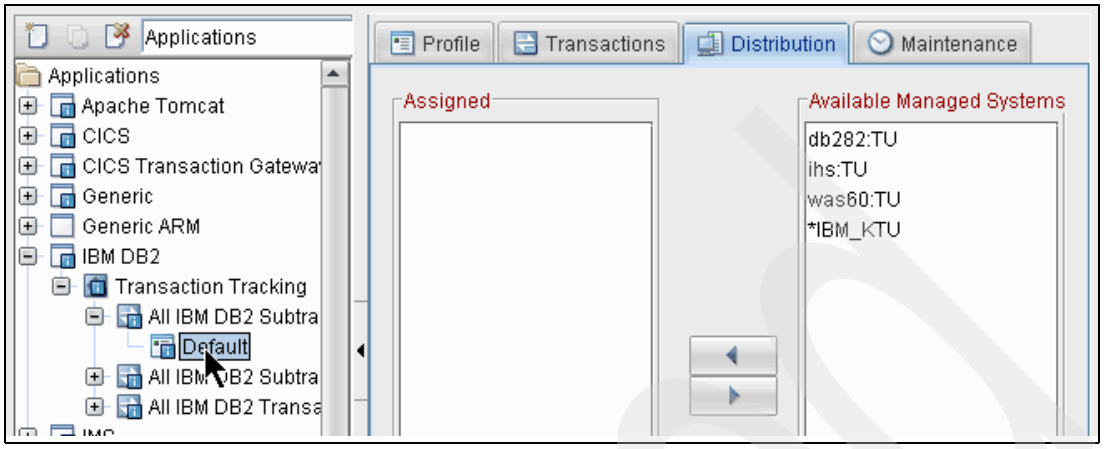


Figure 12-24 Editor window with the Default profile selected

2. On the right, you see that this particular profile is not distributed to any managed systems (agents). Click the **Transactions** tab for this profile to see the transactions that are associated with the profile. See Figure 12-25.

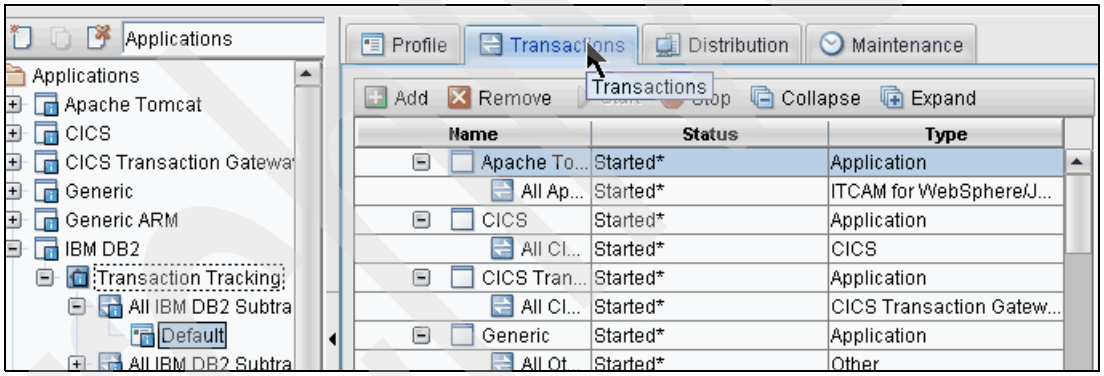


Figure 12-25 Transactions associated with Default profile

3. You can also see the profile configuration by selecting **Profiles** from the drop-down menu at the top of the navigator pane, and then selecting a profile. Note in Figure 12-25 that, although we have navigated to the profile Default under the application IBM DB2, this profile includes transactions from many other applications also. Any profile can contain transactions that are part of multiple applications. The distribution of a profile to a particular Transaction Collector means that the agent will collect those transactions matching those configured for the profile, and in certain cases, multiple applications may submit their transaction data through ARM to the same Transaction Collector agent.
4. To configure Transaction Tracking for our application, use the following steps:
   a. Be sure that **Applications** is selected in the drop-down menu above the navigator.
   b. Click the top-level Applications node in the navigator.
   c. Click **Create New Application**. See Figure 12-26 on page 361.

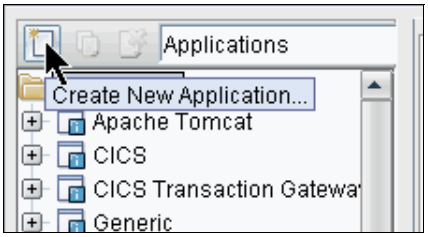


Figure 12-26 Create New Application

In 12.6.2, “Configuring Internet Service Monitoring” on page 351, we first created a profile that we called `TradeApplication`. Be sure to use the same name for the *application name* in Transaction Tracking.

5. After creating your application, create the transactions belonging to that application. In the navigator, select your application, and then click **Create**, which is now the **Create New Transaction** button (Figure 12-27).

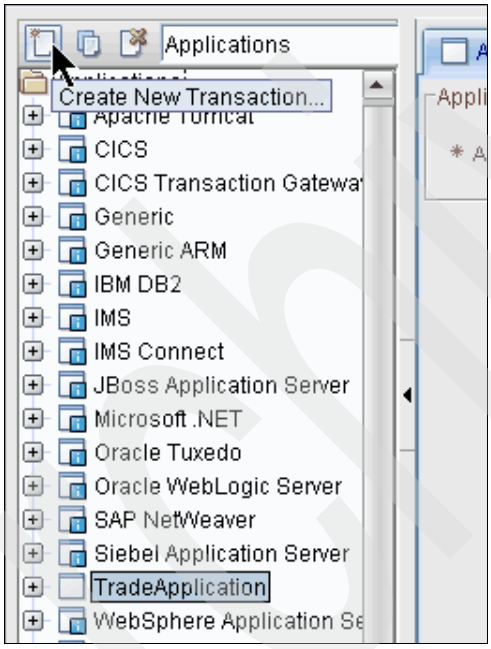


Figure 12-27 Create New Transaction

6. In the Create Transaction window that opens, enter the details for your new transaction, which include the application name (this is already be selected for you, but you may select a different application from the drop-down menu), the transaction name, a description (this is not required), the agent type (because we are not using Response Time, the only option is Transaction Tracking), and the transaction type.

Archived

Figure 12-28 shows the values we use for capturing transaction metrics from the WebSphere plug-in for IBM HTTP Server.

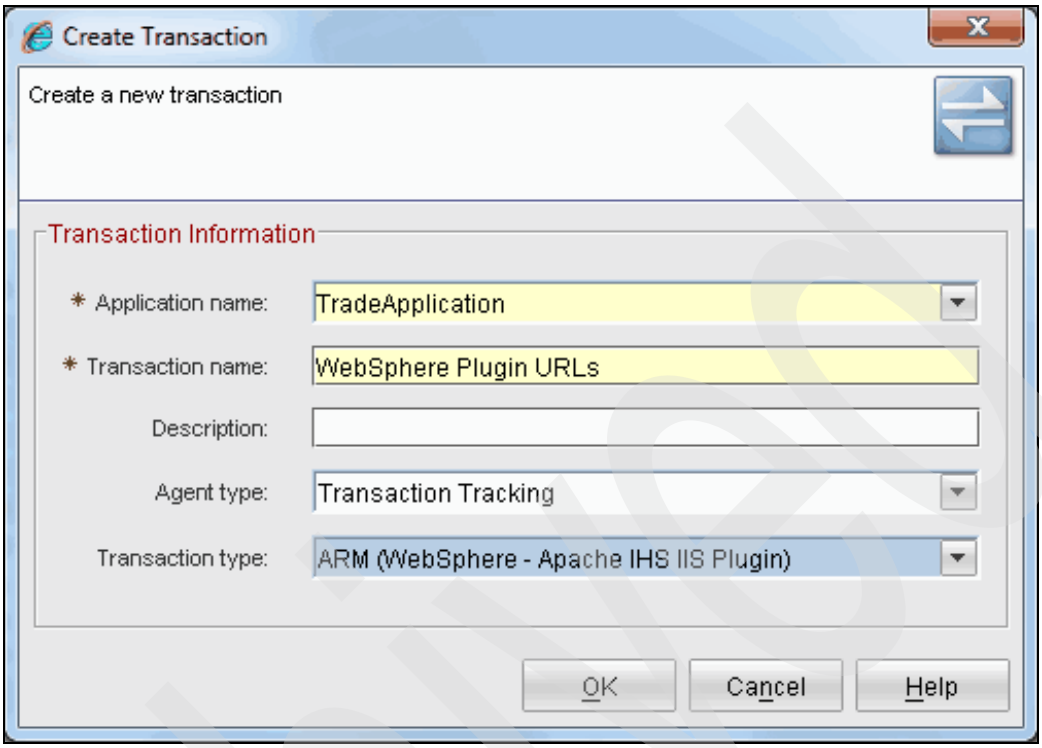


Figure 12-28 New transaction details for IBM HTTP Server transactions

7. After clicking **OK** to create the transaction, make sure your new transaction is selected in the navigator and then click the **Filter** tab in the pane on the right. In this pane, you configure the filter that is used to determine which transactions to collect and which not to collect. Keep in mind that most ARM-enabled middleware components send all transaction metrics to the Transaction Collector, even those that you do not need. Therefore, use your filter to capture those transactions that you really need. Capturing transactions that have unimportant measurements wastes resources in terms of both load on the monitoring infrastructure and space used to store historical transaction data in the Tivoli Data Warehouse.

   The filters you apply for a given transaction can vary among monitored components and among applications. In our scenario, we are filtering based on the URI path of requests sent to IBM HTTP Server and handled by the WebSphere plug-in. As the Transaction Collector receives metrics, it must determine whether the transaction is being reported by the correct component and whether the transaction has the correct path. The next window shows how we configure this information for the trade application and IBM HTTP Server using the WebSphere plug-in. See Figure 12-29 on page 363.

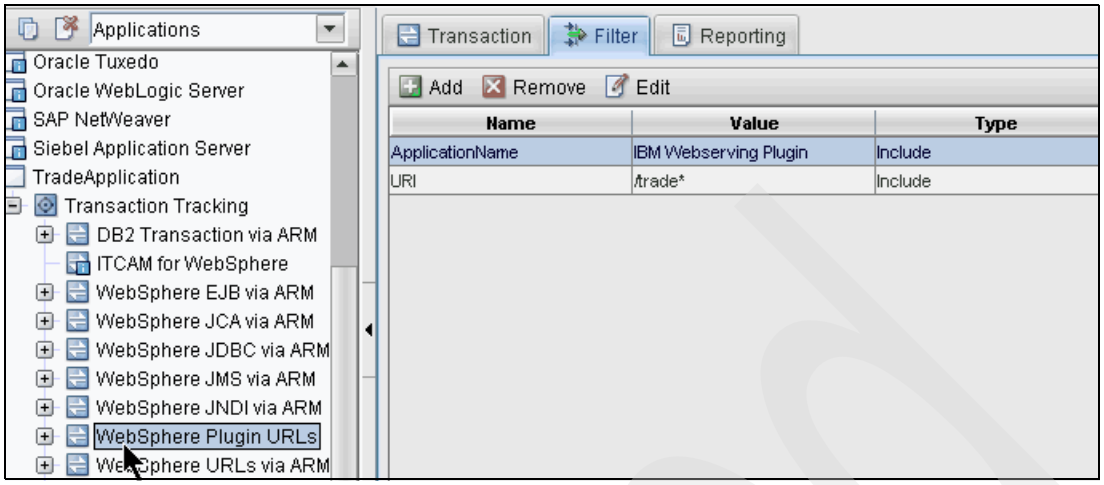


Figure 12-29 Filter configuration for WebSphere plug-in transactions

8. The ApplicationName parameter is set to `IBM Webserving Plugin`, which is the value automatically inserted into the request metric information by the plug-in. Each ARM-enabled application inserts its own value, which can be used to identify transaction data as coming from that component. Because IBM Tivoli Composite Application Manager for Transactions includes a robust set of default transactions, you can browse these to get examples of how to capture transactions from a variety of middleware and application products.

   Because we are gathering data from the WebSphere plugin, any requests to any URIs the plugin is configured to handle will be tracked and reported, but we only want data about those requests for the Trade application. We use the URI parameter with a value of `/trade*` to capture only transaction data pertaining to requests that have a path beginning with that string. On the right side, note that the Type is set to **Include**. It is also possible to use Exclude to filter *out* certain requests, but that is not useful in this particular case.

   Although we do not describe details of the process here, you may also use the same URI filter when capturing request metrics from the application server itself, which in this scenario is WebSphere Application Server. In addition to the WebSphere Plugin URLs transaction, we configured several other transactions for TradeApplication, and Figure 12-30 shows the full list used in this scenario.

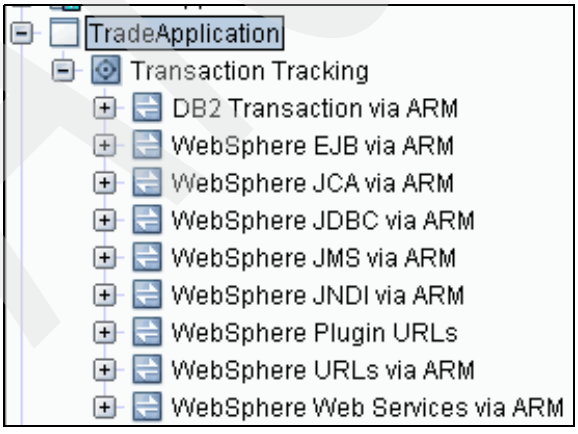


Figure 12-30 Transactions configured for TradeApplication

9. After creating the transaction and defining the filters, you must configure how the transaction data is reported by the Collector. Be sure that you can correctly identify transactions as belonging to the same composite application or business service and use the data to generate events that can be correlated. You configure the reporting individually for each transaction as you create it. After you have created your filters, click the **Reporting** tab and enter values for the following items:
   - Application name: Enter *exactly the same name* as you used when creating the application and the Internet Service Monitoring profile.
   - Transaction name: How you configure this parameter can vary greatly depending how you identify service requests for the given application. Be sure to examine the set of transactions that are included with the product for examples. For web services, however, a good default is to use the value **URI** for both the web server and application server layer, because this value can be used to track the request most readily as it is handed off. We use this value for both **WebSphere Plugin URLs** and **WebSphere URLs via ARM**.
   - Server name: For this parameter, we use **$Hostname$** for all of our transactions, which will be dynamically evaluated to the host name of the server that is reporting the transaction. This is probably a good default to use unless you have specific requirements otherwise. If you have a complex WebSphere topology, for example, with multiple application servers, you may need to include
   - Component name: This scenario uses the value **$ApplicationName$** for all transactions other than **WebSphere Plugin URLs**, for which we use **$ApplicationGroup$**. Note that this value does *not* necessarily evaluate to the name of your application for most middleware components, it evaluates to the name of the middleware product, for example IBM DB2 Universal Database™. Again, in more complex environments, you may choose to use other properties to populate this or other fields to assist in transaction identification and tracking. For WebSphere Application Server, version 6.1, the list of available properties and their meanings are at the following address:

     `http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.base.doc/info/aes/ae/rprf_armproperties.html`

   See the appropriate product documentation for your middleware components to see what other properties might be available for populating any of the other fields. Dynamic properties must always be enclosed in single dollar signs, for example:

   `$ApplicationName$`

   Figure 12-31 shows the settings configured for the WebSphere Plugin URLs transaction within TradeApplication.

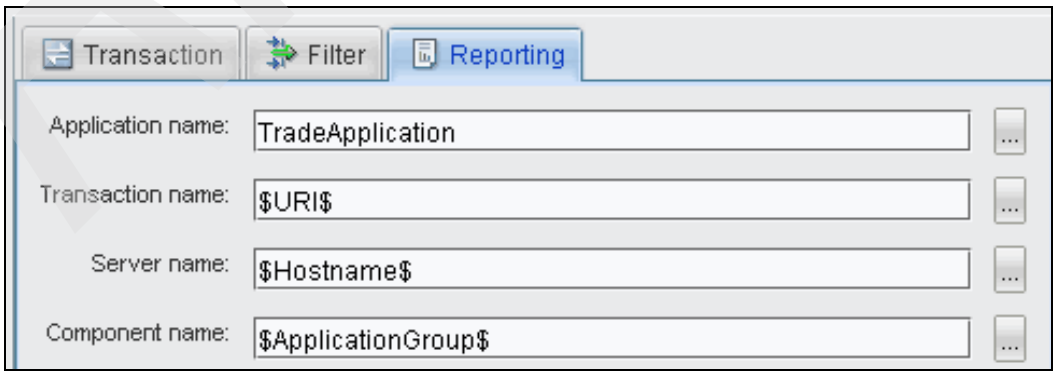


Figure 12-31 Transaction reporting configuration for WebSphere Plugin URLs

Note: We have used ARM instrumentation for DB2, and WebSphere Application Server and IBM Tivoli Composite Application Manager for Transactions as the transaction monitoring solution in this scenario. Alternatively, you may use the IBM Tivoli Composite Application Manager for Application Diagnostics product as the monitoring solution. This product can be used for WebSphere monitoring and transaction tracking (including JDBC tracking which will cover DB2). See the IBM Tivoli Composite Application Manager for Application Diagnostics library for more information about this product:

http://publib.boulder.ibm.com/infocenter/tivihelp/v24r1/index.jsp?topic=/com.ibm.itcamfad.doc_7.1/ABD001/about%20this%20book/kyj62ig06.html

Distributing the Transaction Tracking configuration

After creating your application and set of transactions, you must create a profile for distribution, associate the transactions with it, and then distribute the profile to deployed Transaction Collector agents:

1. Click **Apply** to save any changes you have made to the configuration, and then select **Profiles** from the drop-down menu above the navigator pane in the Application Management Configuration Editor. Click **Create New Profile** to create the new distribution profile, as shown in Figure 12-32.



Figure 12-32 Create new profile

2. Enter the profile details in Figure 12-33.

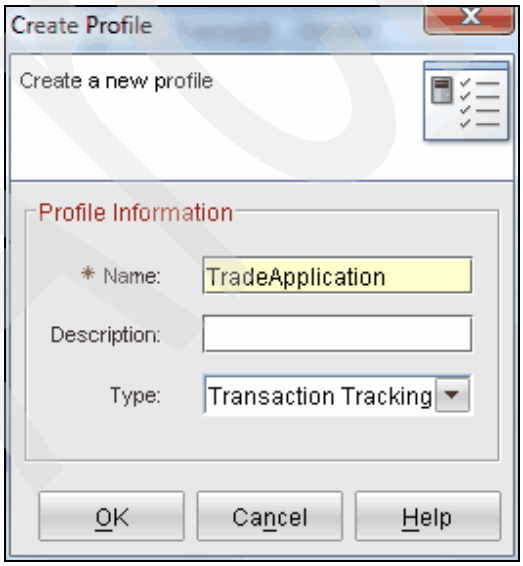


Figure 12-33 New profile details

Note: We use the same name for our profile as we used for the application. Doing so is not necessary, and in many production environments, it might not even be advisable. Although we do this in our scenario, in production, particularly where you might have multiple Transaction Collectors deployed and firewalls separating security zones, you might want several transactions distributed through profiles to some collectors and not to others. Keep in mind the requirements of your particular application deployment topology when designing your Transaction Tracking profiles.

3. After creating the profile, add transactions to it and then distribute the profile. To add transactions, select the profile in the navigator (this is done by default immediately after creating the profile) and then, in the pane to the right, click the **Transactions** tab. Click **Add** to open the Transaction Selection window, shown in Figure 12-34.

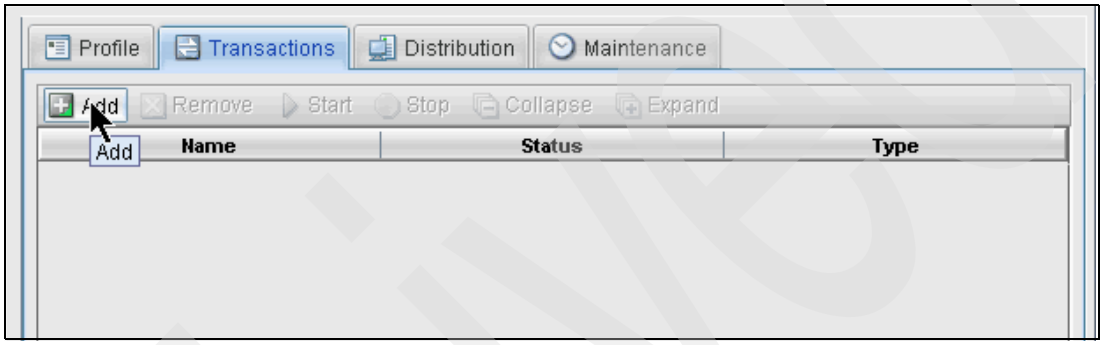


Figure 12-34 Add transaction to Transaction Tracking profile

4. As you can see in Figure 12-34, newly created profiles do not include any transactions. After clicking **Add**, the Transaction Selection window opens. Only those applications containing transactions not already added to the profile are displayed (if all the transactions in an application are already part of the profile, the application is not shown at all). Click the twisty next to the application containing the transactions to be added and then click one or more transactions. You can press Ctrl+Select and Shift+Select to select multiple transactions as in most other list selection applications.

   Figure 12-35 on page 367 shows that we have selected all the transactions within TradeApplication to be added to our profile.

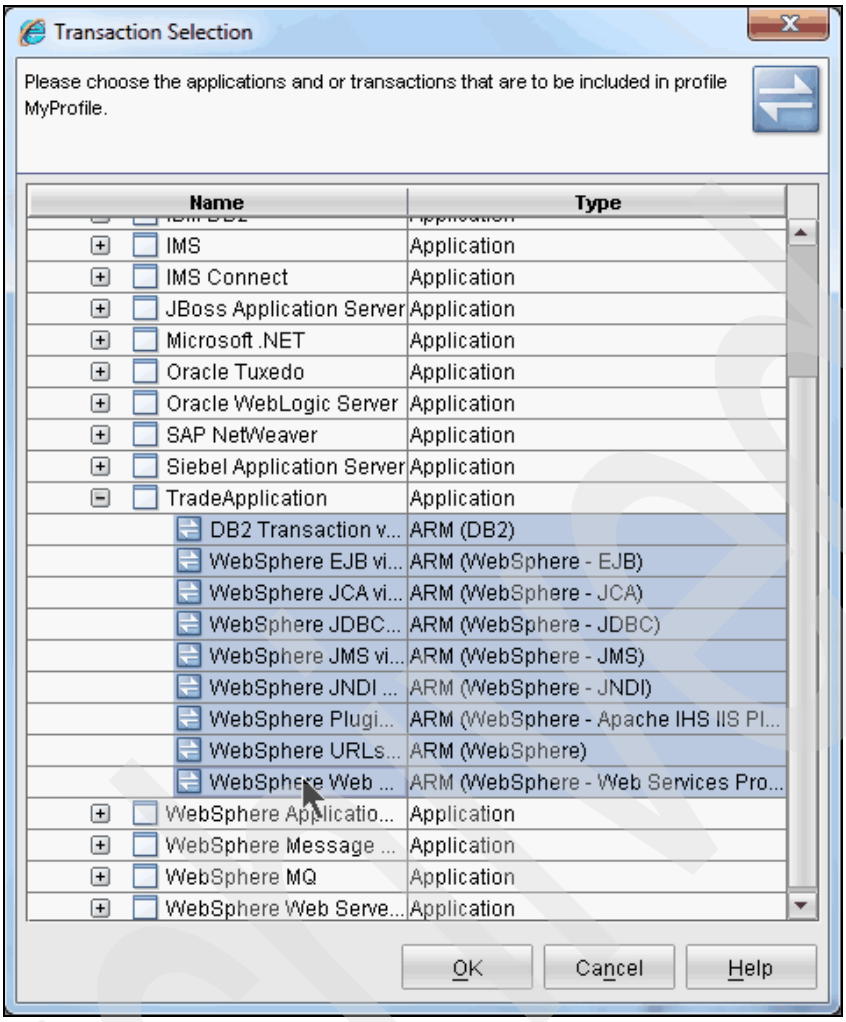


Figure 12-35 Transaction selection for addition to a profile

5. Click **OK** to add the selected transactions to your profile, and then distribute the profile. Click the **Distribution** tab to see a familiar pair of lists. See Figure 12-36.

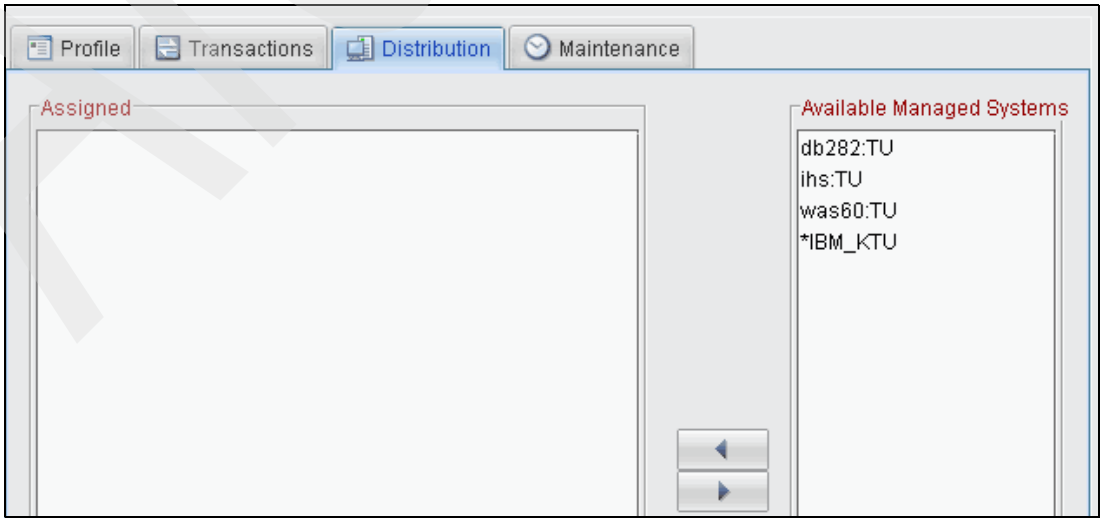


Figure 12-36 Application Management Configuration Editor profile distribution

6. Select the targeted agents for distribution from the **Available Managed Systems** list and move them to the **Assigned** list by clicking the left arrow button. Click **Apply** or **OK** to complete your Transaction Tracking configuration (if you click **Apply** you still need to click **OK** to exit the Application Management Configuration Editor). If you click **Apply** and then return to the Transactions tab, you see that all your transactions are labelled as Started. You may stop or start collection of metrics for any or all associated transactions from this window.

Note: Be aware that the Started label can be deceptive; by default, a new profile and any associated transactions are both in a Started state (just as with newly created IBM Tivoli Monitoring situations), but until the profile has been distributed to agents through the window shown in Figure 12-36, the status for the transactions will show as `Started*` (the asterisk is a visual clue that the profile has not been distributed to any agents yet, and you still need to do that task).

Configuring monitored applications for ARM

The method to enable the collection of ARM metrics varies widely between applications and is outside the scope of this book. Configuration details for many of the more common applications and platforms are in the IBM Tivoli Composite Application Manager for Transactions documentation at:

`http://publib.boulder.ibm.com/infocenter/tivihelp/v24r1/index.jsp?topic=/com.ibm.itcamt.doc_7.2.0.1/tt/dita/concept/kto_tools_arm_oview.html`

You may have to perform one important step, however, particularly if you are using a distributed Transaction Collector deployment in which multiple applications must report their metrics to a single Transaction Collector. Each monitored application, whether or not it reports to a local or remote Transaction Collector, must have an instance of the Transaction Collector installed locally. This Transaction Collector may not be used to collect the metrics from reporting applications, but it is still required to serve as a profile distribution endpoint as shown in Figure 12-36 on page 367. The installation of the Transaction Collector also provides the ARM client libraries used by the monitored application and includes a configuration file, which can be used to specify a remote Transaction Collector agent for metrics reporting. This file is in the following location:

`$CANDLE_HOME/tmaitm6/arm/armconfig.xml`

In the path, `$CANDLE_HOME` is the root installation directory for IBM Tivoli Monitoring on the system. Update the file by adding the XML element (or modifying the existing element):

```
<ttconnectionstring>tcp:collector_hostname_or_address:collector_port</ttconnectionstring>
```

Note that *`collector_hostname_or_address`* is the host name or IP address of the remote Transaction Collector host and *`collector_port`* is the port on which the Transaction Collector is listening (5455 by default). The monitored application must be restarted after this file is updated. Full documentation about updating this file is at the following address:

`http://publib.boulder.ibm.com/infocenter/tivihelp/v24r1/index.jsp?topic=/com.ibm.itcamt.doc_7.2.0.1/tt/dita/concept/kto_tools_arm_remote_coll.html`

12.7 Triggering automated service delivery

This section describes how to use the monitoring components to detect an impact to service delivery and trigger a remediation action by requesting a service instance from Tivoli Service Automation Manager.

12.7.1 Detecting a service event

Detecting events affecting service levels requires two steps, both pertaining to event-slot mappings to enable proper correlation in Tivoli Netcool/OMNIbus:

- Updating the Event Integration Facility configuration for existing situations in IBM Tivoli Monitoring
- Updating the Event Integration Facility probe rules files to incorporate mappings for IBM Tivoli Composite Application Manager for Transactions

Situation configuration

For this scenario, we use two situations, both of which are included with IBM Tivoli Composite Application Manager for Transactions. One situation checks for a service-level failure based on Internet Service Monitoring; the other checks for a transaction slowdown at a particular application tier. In real-world production usage, expect to use more complex configurations, perhaps requiring more situations, but those uses are still derived from the same basic model, which is detect a service-level failure and identify the component causing the failure.

In our scenario, which is meant to represent a software as a service (SaaS) cloud, we detect a service level failure for a web-based business service. then correlate that with a component-level failure with regards to configured response time thresholds; the correlation of the two events then triggers the service request to Tivoli Service Automation Manager. Although this scenario is relatively simple, it demonstrates a flow that can be used to maintain service levels through near-real-time automated remediation. In real-world usage, additional monitoring agents, such as those for operating systems or IBM Tivoli Composite Application Manager for Application Diagnostics, might also be used to detect issues with the underlying middleware or to detect events which could be used to identify and remediate more fine-grained capacity issues:

The steps are as follows:

1. Modify the event slot mappings for two situations, both of which are created when you install the application support for IBM Tivoli Composite Application Manager for Transactions:
   a. Open the Situation Editor in the Tivoli Enterprise Portal,
   b. Navigate to the situation **Internet Service Monitoring** → **KIS_Element_SLA_Failed**.
   c. Click the **EIF** tab and be sure that the **Forward Events to an EIF Receiver** check box is selected.
   d. Select the appropriate severity (we have used **`Critical`**) and EIF Receiver, as shown in Figure 12-37 on page 370.

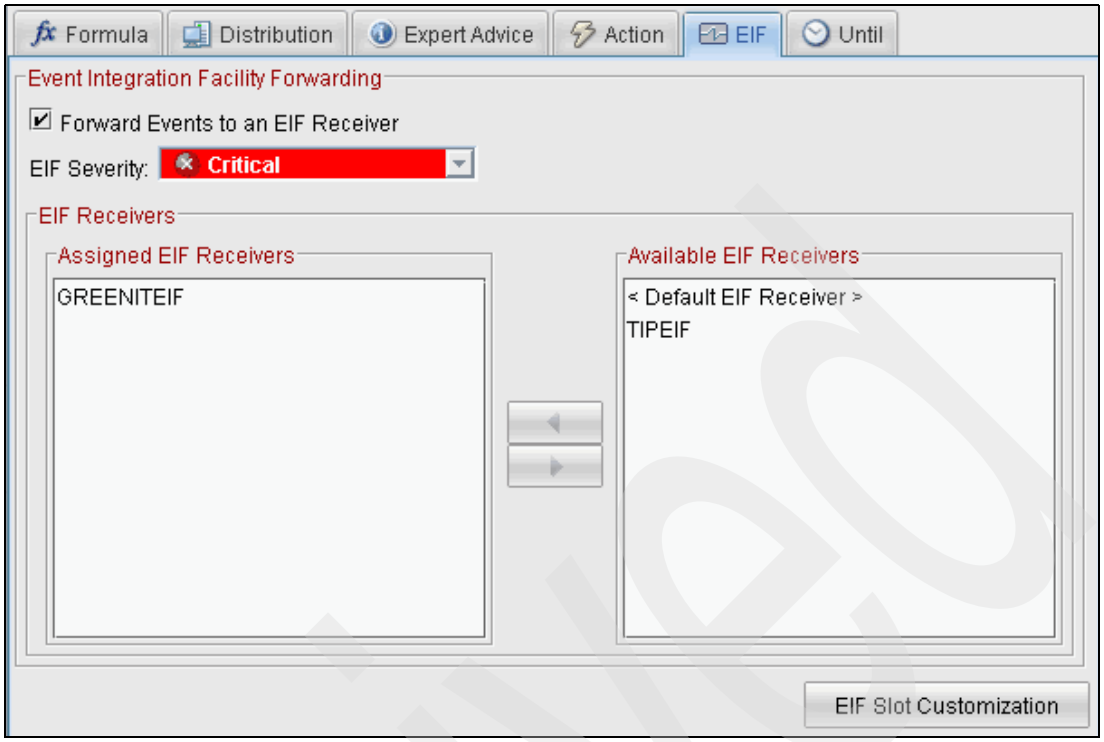


Figure 12-37 Basic EIF configuration for situation

2. Click **EIF Slot Customization**. This next section assumes that you have used the **tedgen** utility to update your slot mapping XML file and configured your Tivoli Enterprise Portal Server to use the updated file. If you have not, the instructions for doing so are in the IBM Tivoli Monitoring documentation:

   http://publib.boulder.ibm.com/infocenter/tivihelp/v15r1/index.jsp?topic=/com.ibm.itm.doc_6.2.2fp2/tec_eifslot_tedgen.htm

   The instructions are identical for IBM Tivoli Monitoring 6.2.2 fixpack 1 and fixpack 2, so the instructions for the link are valid for either version. Previous versions of **tedgen** do require a Tivoli Enterprise Console. As of IBM Tivoli Monitoring 6.2.2 fixpack 1, the **tedgen** utility is part of the tools distribution and no longer requires that you have a Tivoli Enterprise Console installation.

   After you start the EIF Slot Customization window, you see a box at the top showing the currently selected event class for mapping the situation event (Figure 12-38), and below that you see currently mapped slots. Beside the box showing the class is a button you can use to select another event class; click that button to change the class.

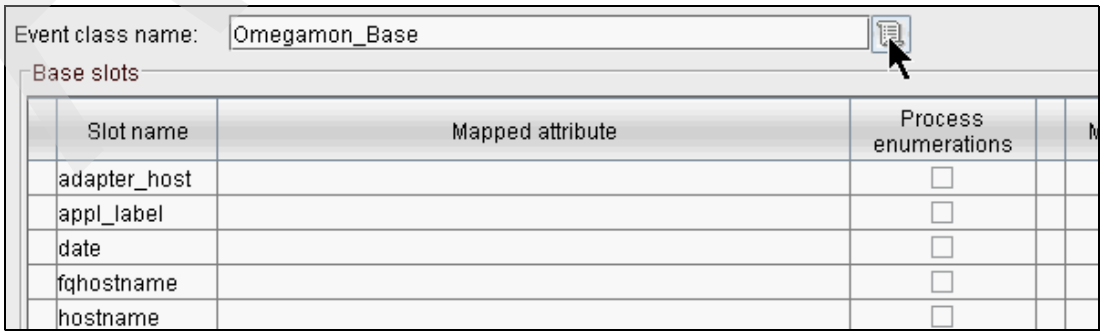


Figure 12-38 EIF Slot Customization

3. Click the button to the right of the Event class name (Figure 12-38 on page 370 shows the cursor over this button) to open the window (shown in Figure 12-39) that lists available event classes.

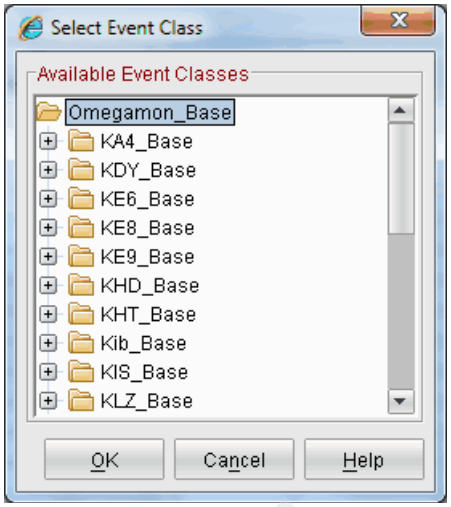


Figure 12-39 Select Event Class dialog for Situation EIF Slot Customization

4. Every situation starts with the assigned default **Omegamon_Base** class. The classes are organized hierarchically in the tree with the various agent-specific base classes listed alphabetically.

   In the class navigator, select **KIS_Base** → **ITM_KIS_SERVICE_INSTANCE_STATISTICS** and click **OK**. Initially you may not see much difference in the list of mapped slots, with the Extended slots pane empty. In the EIF Slot Customization window, be sure the **Map all attributes** check box is selected. See Figure 12-40.

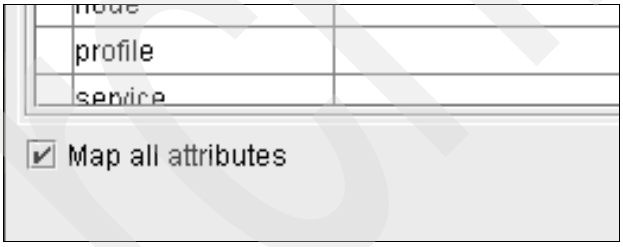


Figure 12-40 Map all attributes

   The Extended slots pane is populated with the class-specific slots, which you can use to map specific attributes onto slots. In this scenario, we do not customize the mappings any further; the purpose is to enable the mapping so that all slots are populated when the event is submitted through the Event Integration Facility.

5. Save the Event Integration Facility configuration by clicking **OK**. Return to the main situation definition tab and click **Advanced**, then set the situation display item to **Description** from the selection menu. Click **OK**, then click **Apply** to save the situation configuration.

6. Perform the same steps for the Transaction Tracking situation except for setting the display item. Although you may set this item (and setting it is good practice generally when creating or configuring situations), it is not required for the integration to work. You are still in the Situation Editor at this point, if not, open it again. Navigate to the situation **Transaction Reporter** → **Slow_Transactions** and click the EIF tab. Follow the same

steps as for the previous situation, ensuring that the **Forward Event to an EIF Receiver** check box is selected and that the appropriate EIF Receiver is listed under **Assigned EIF Receivers**. Next, open the EIF Slot Customization window and click the button to select the event class. For the Transaction Reporter situation, select the **KTO_Base** → **ITM_Aggregates** class. See Figure 12-41.

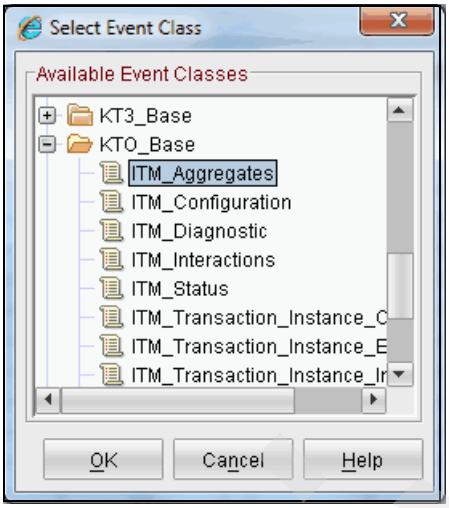


Figure 12-41 Event class selection for situation Slow_Transactions

7. Make sure the **Map all attributes** check box is selected and save your configuration. Be aware that, when configuring the severity used when forwarding events through the Event Integration Facility, we used a lower severity for Slow_Transactions than for KIS_Element_SLA_Failure. The situation by itself might not indicate a service failure, whereas the second situation does in our scenario, and so we assigned the severities to indicate the relative impact.

   By default, both situations are distributed to all installed agents of the appropriate type; although you may change the distribution, you must have only a single Transaction Reporter (KTO) agent deployed in your infrastructure.

Updating the Tivoli Netcool/OMNIbus configuration

We next discuss modifying the probe rules files used by the Event Integration Facility probe to process incoming events before passing them on to the Tivoli Netcool/OMNIbus ObjectServer:

1. Before performing the update, perform the integration steps to create the rules files and modify the ObjectServer database appropriately for IBM Tivoli Composite Application Manager for Transactions. To do this, extract the contents of the integrations package to a directory on your ObjectServer system. This directory tree should be navigable and writeable by the user under whom the ObjectServer runs. After extracting the contents, you see four subdirectories:

   ```
   tivoli@tbsm421:/root/itcamftx> ls
   dla  omnibus  tbsm  tcr
   ```

2. Update the permissions on the directory tree to ensure that your Tivoli Netcool/OMNIbus user can write to the tree:

   ```
   tbsm421:~/itcamftx # chown tivoli:tivoli *
   tbsm421:~/itcamftx # chmod -R g+rw
   ```

   In our example, the ObjectServer is installed and runs as the user `tivoli`; this user belongs to the group that is also named `tivoli`.

3. Navigate to the `omnibus` subdirectory. The integration package includes the `.baroc` files, defining the classes for most of the IBM Tivoli Composite Application Manager for Transactions agents. However, two are absent:

   – Those for the Internet Service Monitoring agent and the Transaction Reporter agent.
   – Those are the two which are most important for our purposes.

   They are both installed in the `$CANDLE_HOME/tables/`*`HTEMS_NAME`*`/TECLIB` directory on your Hub Tivoli Enterprise Monitoring Server (where *`HTEMS_NAME`* is the name of your Monitoring Server) and are called `kis.baroc` and `kto.baroc`, respectively. Copy them both to the same `omnibus` subdirectory where you extracted the integration package on your ObjectServer system. Then, as your ObjectServer user (`tivoli` in our scenario), execute the following commands in Example 12-9.

   Example 12-9 Execute the commands

```
tivoli@tbsm421:~> cd /itcamftx_integration_dir/omnibus/
tivoli@tbsm421:/root/itcamftx/omnibus> ./omnibusUpdater.sh -b kis.baroc
kt3.baroc kt4.baroc kt5.baroc kt6.baroc kto.baroc -u objserv_admin_user -p
password
```

   In the example, *`itcamftx_integration_dir`* is the directory into which you extracted the contents of the integration package, *`objserv_admin_user`* is the name of an ObjectServer superuser or user with the appropriate permissions to modify the ObjectServer database schema, and *`password`* is the password for the admin user.

   This command updates the `alerts.status` table in the ObjectServer database to include specific columns for IBM Tivoli Composite Application Manager for Transactions, and create rules files specific to each event class, which are then included in the main `tivoli_eif.rules` file for the Event Integration Facility Probe.

We next describe further manual modifications we make to those rules files to populate the new `alerts.status` columns appropriately for our scenario.

Archived

Modifying the Event Integration Facility rules files

If you examine your `tivoli_eif.rules` file, you see that an additional set of lines has been inserted to include the newly created rules files. See Example 12-10.

Example 12-10 Additional set of lines

```
# including the rules file generated by the omnibusUpdater script.
        include
"/opt/IBM/tivoli/netcool/omnibus/probes/linux2x86/custom_mapping.rules"

        # including the rules file generated by the omnibusUpdater script.
        include "/opt/IBM/tivoli/netcool/omnibus/probes/linux2x86/kis.rules"

        # including the rules file generated by the omnibusUpdater script.
        include "/opt/IBM/tivoli/netcool/omnibus/probes/linux2x86/kt3.rules"

        # including the rules file generated by the omnibusUpdater script.
        include "/opt/IBM/tivoli/netcool/omnibus/probes/linux2x86/kt4.rules"

        # including the rules file generated by the omnibusUpdater script.
        include "/opt/IBM/tivoli/netcool/omnibus/probes/linux2x86/kt5.rules"

        # including the rules file generated by the omnibusUpdater script.
        include "/opt/IBM/tivoli/netcool/omnibus/probes/linux2x86/kt6.rules"

        # including the rules file generated by the omnibusUpdater script.
        include "/opt/IBM/tivoli/netcool/omnibus/probes/linux2x86/kto.rules"
```

We modify the `kis.rules` and `kto.rules` files:

1. Open the `kis.rules` file with an editor. The file contains a series of conditionals, which check the class of the received event and process the slot values appropriately. Find the stanza that begins with the following lines:

   ```
   if( match($ClassName, "ITM_KIS_SERVICE_INSTANCE_STATISTICS"))
   {
   ```

2. At the end of this stanza before the closing brace, add the following lines:

   ```
   @CAM_Application_Name = $profile
   @CAM_Profile_Name = $profile
   @CAM_Response_Time = $totaltime
   @CAM_Transaction_Name = $situation_displayitem
   @Node = $host
   ```

   Remember that the `$situation_displayitem` for situation events of this class contain the probe element `Description` attribute, which we set previously to be the same as the URI path that is used in the probe request.

Archived

3. Save the `kis.rules` file and open the `kto.rules` file for editing. At the bottom of this file, add the following stanza *in its entirity:*

```
if (match($ClassName, "ITM_Aggregates"))
{
        if(regmatch($enclosing_server, "'(.*)'"))
        {
                $enclosing_server = extract($enclosing_server, "'(.*)'")
        }

        if(regmatch($enclosing_component, "'(.*)'"))
        {
                $enclosing_component = extract($enclosing_component, "'(.*)'")
        }

        if(regmatch($total_time, "'(.*)'"))
        {
                $total_time = extract($total_time, "'(.*)'")
        }

        @CAM_Application_Name = $enclosing_application
        @CAM_Details = $enclosing_component
        @CAM_Profile_Name = $enclosing_application
        @CAM_Response_Time = $total_time
        @CAM_Transaction_Name = $aggregate
        @Node = $enclosing_server
}
```

4. Save this file and either send a **`SIGHUP`** signal to the Event Integration Facility probe or restart it to force a reload of the rules files.

12.7.2 Triggering the service instance request

Now we describe configuring the Tivoli Netcool/OMNIbus ObjectServer to automate generating the request to Tivoli Service Automation Manager. We address two primary tasks:

- Creating a simple procedure to call a script which communicates with Tivoli Service Automation Manager,
- Creating a simple trigger which will run the procedure based on the correlation of the two events received.

Creating the procedure

Execution of external tasks is done in Tivoli Netcool/OMNIbus by means of procedures, which can either execute SQL statements or external commands. In our scenario, we use the latter type of procedure. We use a simple procedure such as shown in Figure 12-42 on page 376.

Archived

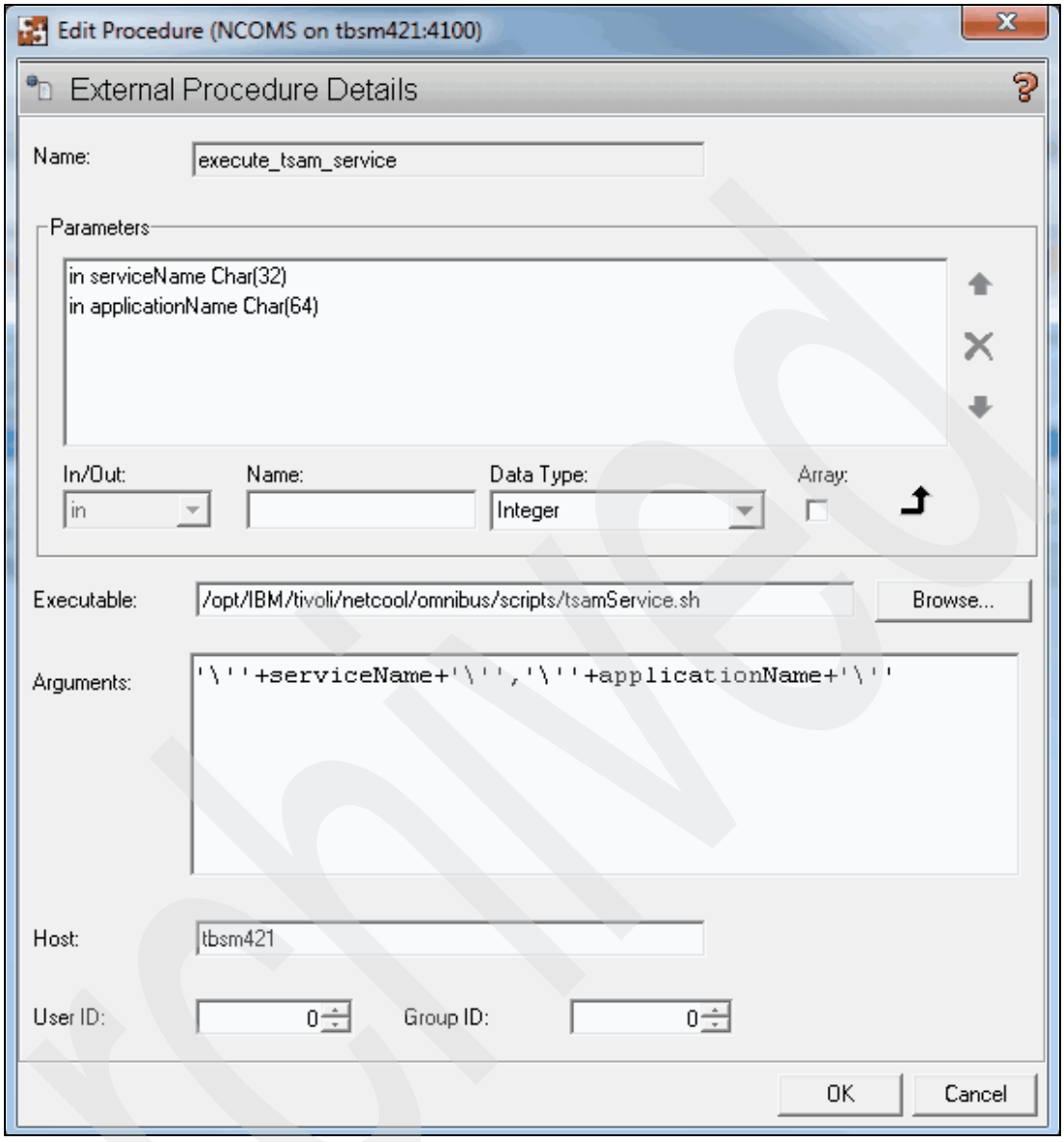


Figure 12-42 Edit procedure

This procedure simply calls the `tsamService.sh` script, passing in as arguments the name of the Tivoli Service Automation Manager service to execute and the name of the hosted application for which the service is to be executed.

Creating the trigger

For this scenario, we create the trigger manually in Tivoli Netcool/OMNIbus, but that is not the only approach. Starting with the release of version 7.3 of Tivoli Netcool/OMNIbus, IBM Tivoli Netcool/OMNIbus users receive IBM Tivoli Netcool/Impact with a limited use license (such as data access to only Tivoli products), and we believe that many users are more likely to use Tivoli Netcool/Impact to create their automations. Another approach is to create yet a third situation in IBM Tivoli Monitoring. You can use a correlated situation to evaluate both of the other situations we described, then send only the single event to Tivoli Netcool/OMNIbus. We however, found this approach less robust and less applicable to production uses, in which users are more likely to want to compare the specific event attributes for correlation purposes.

Our trigger, which we created as a timer trigger set to fire once every minute, has the basic logic shown in Example 12-11.

Example 12-11 Basic logic

```
begin
    -- Find service element failures matched by tx slowdown traceable to a service
component
    for each row txslowdown in alerts.status where
        txslowdown.AlertGroup = 'ITM_Aggregates'
        and txslowdown.Type = 20 and txslowdown.Acknowledged = 0
        and (txslowdown.CAM_Application_Name + txslowdown.CAM_Transaction_Name) in
           (select CAM_Application_Name + CAM_Transaction_Name from alerts.status
where AlertGroup = 'ITM_KIS_SERVICE_ELEMENT_FAILURE' and Type = 20 )
    begin
        --You could use SupressEscl just as well here
      set txslowdown.Acknowledged = 1;
        --Rememeber that CAM_Details contains the name of the failing component,
such as IBM_HTTP_Server
        execute execute_tsam_service( txslowdown.CAM_Details,
txslowdown.CAM_Application_Name );
     end
end;
```

The trigger checks for unacknowledged transaction slowdowns corresponding to a known service failure, acknowledges the slowdown event, and then requests a new service component instance from Tivoli Service Automation Manager. If the transaction slowdown event cannot be matched with a known service failure, no action is taken.

Archived

12.8 Summary

Figure 12-43 shows the quick summary for this integration.

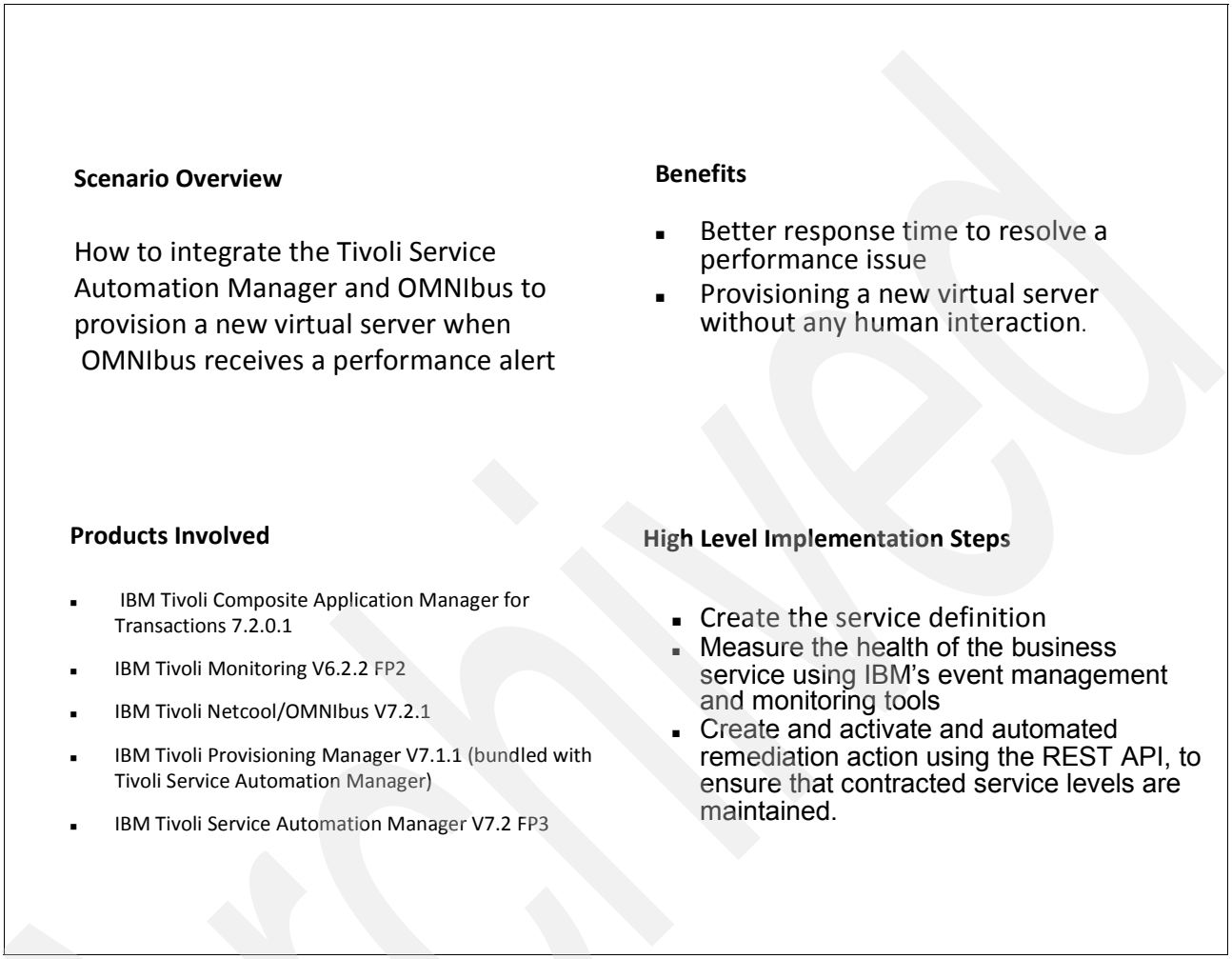


Figure 12-43 Quick summary

Archived

13

Automated Storage Provisioning

This chapter contains the following topics:

- 13.1, “Scenario overview” on page 380
- 13.2, “Products involved” on page 380
- 13.3, “Benefits” on page 380
- 13.4, “Integration configuration steps” on page 381
- 13.5, “Scenario walk-through” on page 390
- 13.6, “Summary” on page 398

Archived

© Copyright IBM Corp. 2011. All rights reserved.

13.1 Scenario overview

In this chapter, we discuss the value of integrating an automated storage provisioning solution as part of an end-to-end service management solution. We achieve automated storage provisioning as part of an end-to-end application provisioning solution by using Tivoli Provisioning Manager to deploy the various components of an application and Tivoli Storage Productivity Center to deploy and manage the storage needs for a given business service as part of an integrated workflow. Although we do not show the installation steps for the individual products, we do describe the components and map them to the machines in our environment. In addition, we explain the default parameters and what changes are needed to perform the product integrations.

13.2 Products involved

The products involved in this scenario are as follows:

- Tivoli Storage Productivity Center Server V4.1.1
- Tivoli Provisioning Manager V7.1.1 or higher

13.3 Benefits

In this section, we discuss the key benefits of provisioning and storage management solutions and the value an integrated end-to-end business service provisioning solution provides:

- Benefits of Automated Provisioning
- Benefits of Storage Management
- Benefits of an integrated end-to-end strategy

13.3.1 Benefits of Automated Provisioning

End-to-end application provisioning helps to automate IT processes and optimizes resource utilization. This approach can address changing business demands and help meet service-level agreements for key business services.

Many IT organizations today manage their silos of expertise (network, storage, operating system, and application teams) for the key business services of an organization, which can cause many operational challenges. First, deployment errors can result in roll-backs and unplanned down-time of business-critical applications, which can cause business customer dissatisfaction. In addition, siloed operations can cause the inability to ensure that the test environment matches production, which can result in quality problems and slow problem resolution, causing a loss in revenue and productivity. Moreover, this type of operation tends to lead to a lack of an auditable deployment process that hinders the ability to comply with external regulations, which can cause financial penalties or loss of shareholder value. Finally, this type of operation tends to lead to poor asset management which tends to result in “just-in-case” capacity provisioning and minimal amounts of virtualization because of the demand by finance teams for usage reporting and chargeback, which causes low-resource sharing and low-utilization rates and a high cost per asset.

Archived

13.3.2 Benefits of Storage Management

In addition to the challenge of end-to-end application provisioning, storage management has its own set of challenges that can affect the business. Out-of-control storage growth causes the demand for better utilization of existing resources. In addition, as storage grows, the importance of several aspects becomes important: minimizing performance bottlenecks and planning for capacity upgrades, which causes the need for performance analytics with root cause identification and capacity management. Finally, minimizing outages and keeping up with changes made to the storage area network and multivendor storage devices forces organizations to look at tools that can be used for many systems and allow for change and configuration management. Figure 13-1 shows the cost of disks has consistently decreased, however, the quantity of disks needed as exponential grown.

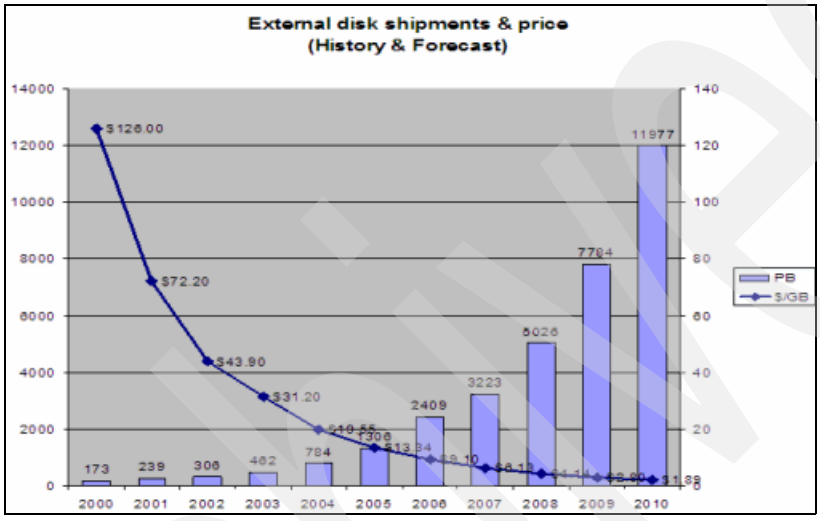


Figure 13-1 Cost of disks has decrease; quantity of disks needed has increased

13.3.3 Benefits of an integrated end-to-end strategy

Integrating storage and application provisioning in an end-to-end process eliminates much of the error-prone manual work that is traditionally required to build an application environment. This type of workflow can also reduce the amount of manual communication between IT silos that, in many cases, adds several hours or not weeks to the application building process. Because application, server, and storage teams in many organizations are part of separate team with various expertise, an automated single tool that all teams can use to provision a solution is extremely beneficial.

13.4 Integration configuration steps

This section explains the key configuration steps of integrating Tivoli Provisioning Manager and Tivoli Storage Productivity Center. It describes the following information:

- Useful resources
- Integration prerequisites and roles
- Configuring Tivoli Storage Productivity Center
- Configuring Tivoli Provisioning Manager security
- Defining the server in Tivoli Provisioning Manager

13.4.1 Useful resources

Many useful resources can help you implement and deploy the integration between Tivoli Provisioning Manager and Tivoli Storage Productivity Center.

- Tivoli Provisioning Manager Information Center, which reviews the steps necessary to integrate Tivoli Provisioning Manager and Tivoli Storage Productivity Center:

  http://publib.boulder.ibm.com/infocenter/tivihelp/v28r1/index.jsp?topic=/com.ibm.tivoli.tpm.scenario.doc/tips/ttip_tpc_launch.html

- *Automated Storage Provisioning for Windows using TotalStorage Productivity Center and Tivoli Provisioning Manager*, REDP-4242
- *Exploring Storage Management Efficiencies and Provisioning - Understanding IBM TotalStorage Productivity Center and IBM TotalStorage Productivity Center with Advanced Provisioning*, SG24-6373
- *An Introduction to Storage Provisioning with Tivoli Provisioning Manager and TotalStorage Productivity Center*, REDP-3900
- *IBM Tivoli Provisioning Manager V7.1.1: Deployment and IBM Service Management Integration Guide*, SG24-7773

13.4.2 Integration prerequisites and roles

For Tivoli Provisioning Manager to integrate with Tivoli Storage Productivity Center, certain prerequisites and roles are required:

- Software prerequisites:
  - Tivoli Storage Productivity Center Version 4.1.1
  - Tivoli Provisioning Manager Version 7.1.1
  - Tivoli Storage Productivity Center automation package (`IBMTPC.tcdriver`), which must be installed in Tivoli Provisioning Manager.
- Security roles
  - Tivoli Provisioning Manager

    Any role
  - Tivoli Storage Productivity Center

    Any of the following roles:
    - Superuser
    - Productivity Center Administrator
    - Data Operator
    - Data Administrator

Archived

13.4.3 Configuring Tivoli Storage Productivity Center

Perform the following steps to configure Tivoli Storage Productivity Center for this scenario:

1. Open the Tivoli Integrated Portal in a web browser.

   In our case, the login is `http://tpcprod:16310`, where `tpcprod` is the host name of the computer where Tivoli Storage Productivity Center is installed.

2. Log in as a user with administrative privileges. See Figure 13-2.

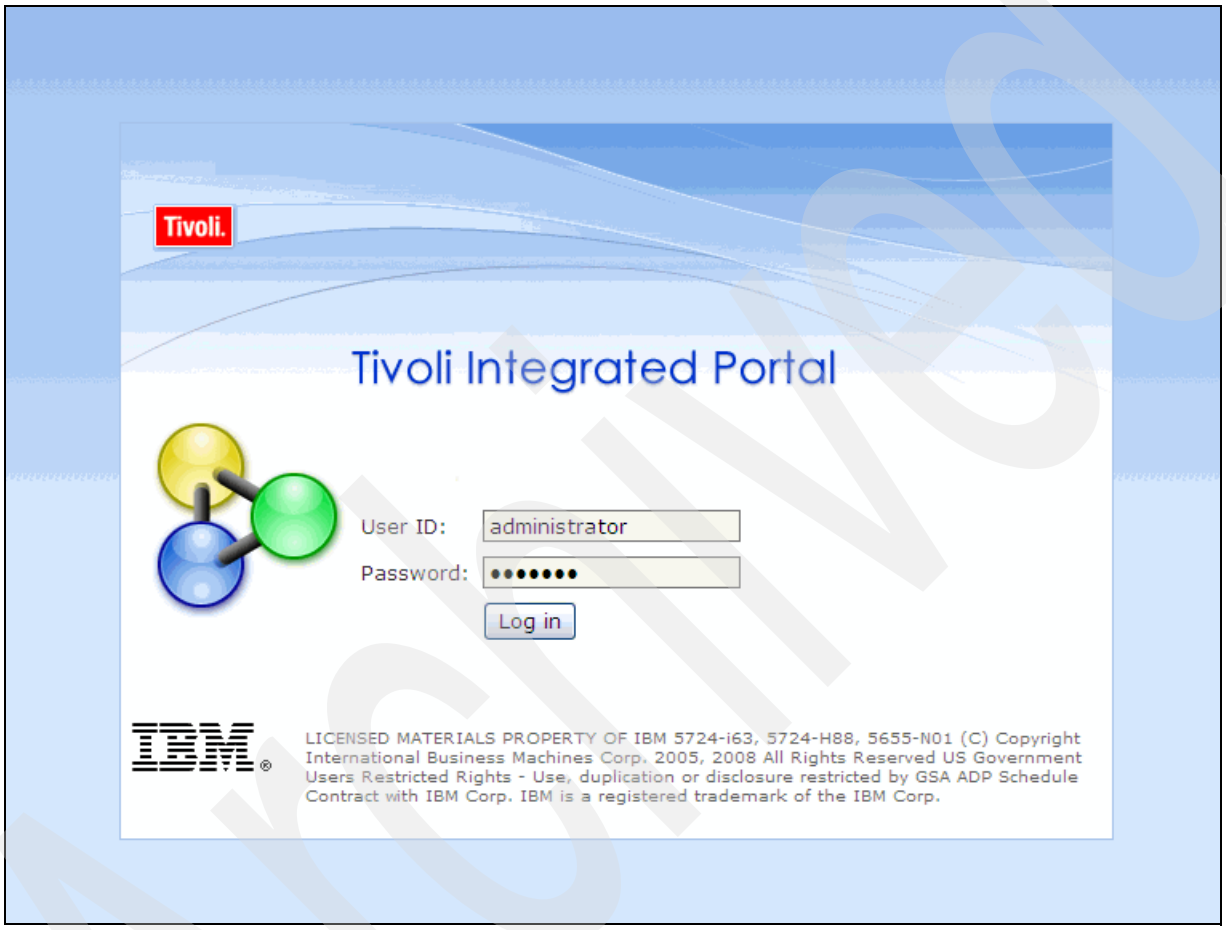


Figure 13-2 Log in to Tivoli Integrated Portal

3. Click **Security** → **Secure administration, applications, and infrastructure** → **Web Security** → **Single Sign-On**.

Archive

4. Verify that single sign-on is enabled and a valid domain name is configured. See Figure 13-3.



Figure 13-3 Ensure single sign-on is enabled

5. Click **Security** → **Secure administration, applications, and infrastructure** → **Authentication mechanisms and expiration**.
6. In the Cross-cell single sign-on section, enter the following information:
   a. Enter and confirm a Tivoli Integrated Portal key file password of your choice.
   b. Enter a Tivoli Integrated Portal name of your choice.

Archive

7. Click **Export Keys**. See Figure 13-4.

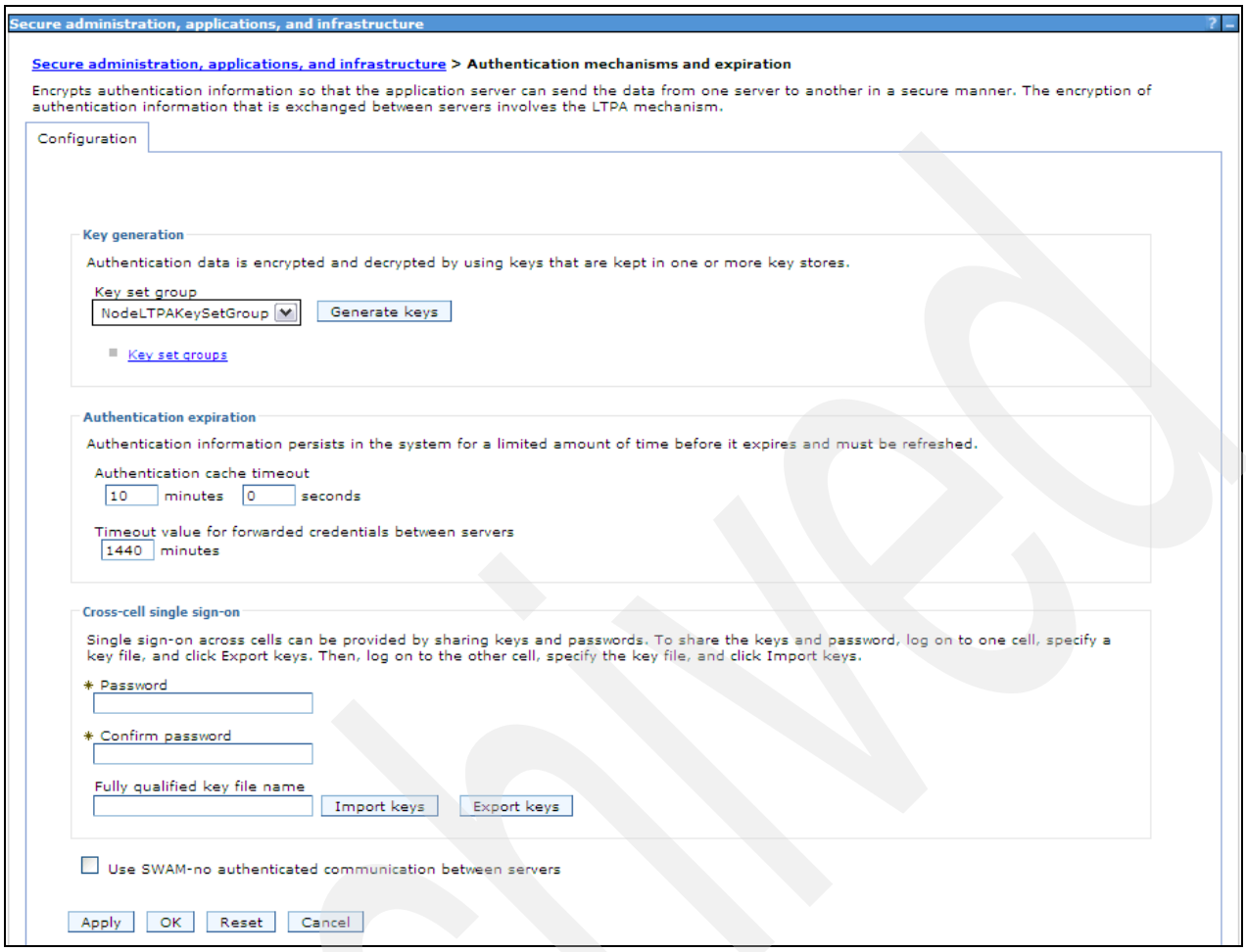


Figure 13-4 Export Keys

8. Log out of the Tivoli Integrated Portal.

13.4.4 Configuring Tivoli Provisioning Manager security

Perform the following steps to configure Tivoli Provisioning Manager security:

1. On the provisioning server, log in to the WebSphere Application Server administrative console.

   In our environment, the WebSphere Application Server administrative console was located at `http://tpmprod:9060/admin`. In our case, `tpmprod` was the host name of the provisioning server and we used the default port number of 9060.

2. Log in as a user with administrative privileges, which in our case was the user wsadmin.
3. Click **Security** → **Secure administration, applications, and infrastructure** → **Web Security** → **Single Sign-On**. Verify that single sign-on is enabled and that the domain name matches the one you used to configure single sign-on for Tivoli Storage Productivity Center.

Archived

4. Verify that the same LDAP realm name is configured for both Tivoli Provisioning Manager and Tivoli Storage Productivity Center by clicking **Security** → **Secure administration, applications, and infrastructure**.
5. Under Account Repository, click **Configure**.
6. Under General Properties verify that the same LDAP Realm name is configured for both Tivoli Provisioning Manager and Tivoli Storage Productivity Center. See Figure 13-5.

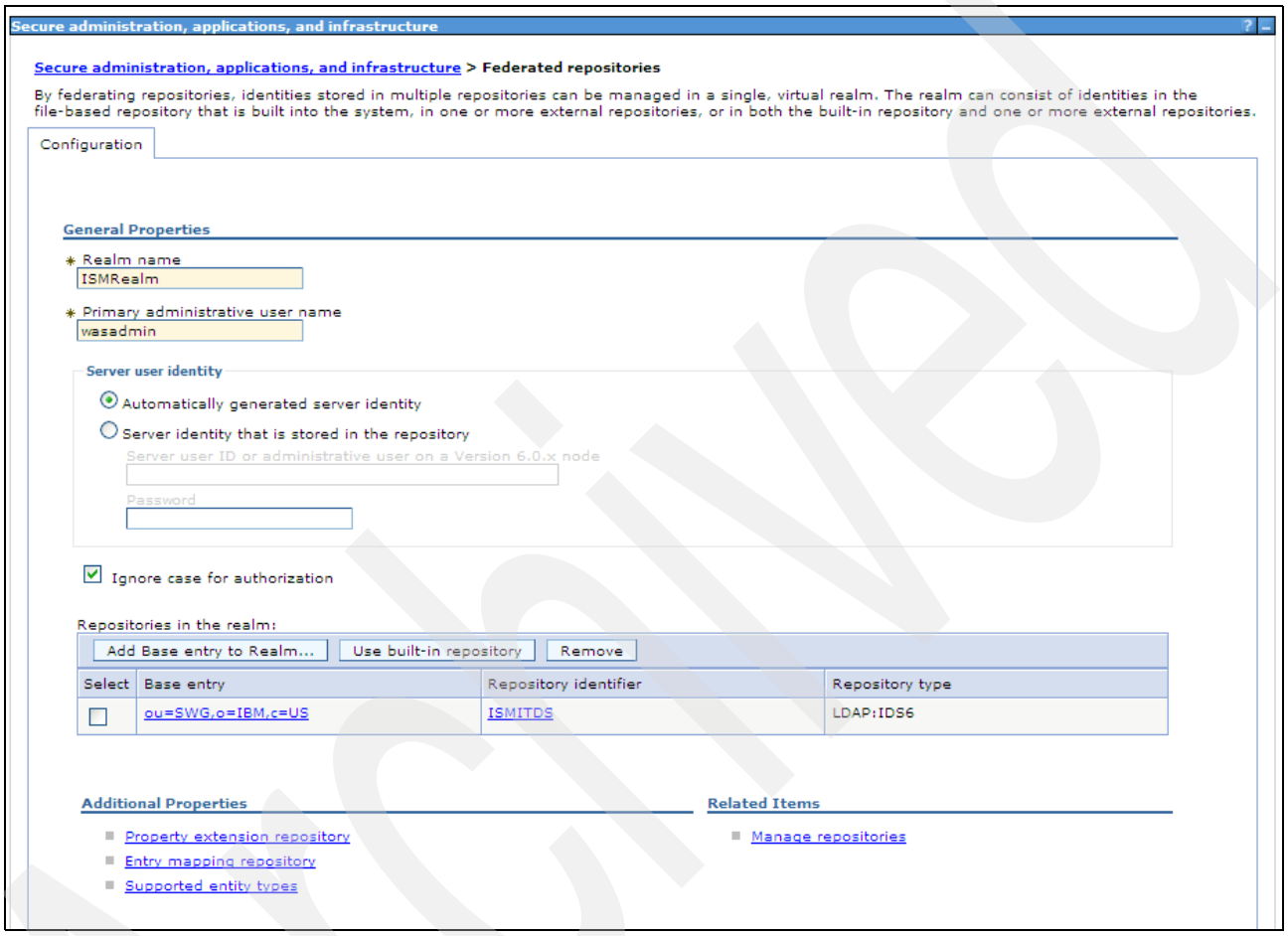


Figure 13-5 Verifying the Realm name

7. Click **Security** → **Secure administration, applications, and infrastructure** → **Authentication mechanisms and expiration**.
8. In the Cross-cell single sign-on section, enter the following information:
   a. Enter and confirm the Tivoli Integrated Portal key file password.
   b. Enter the Tivoli Integrated Portal key file name.

Archived

9. Click **Import Keys**. See Figure 13-6.

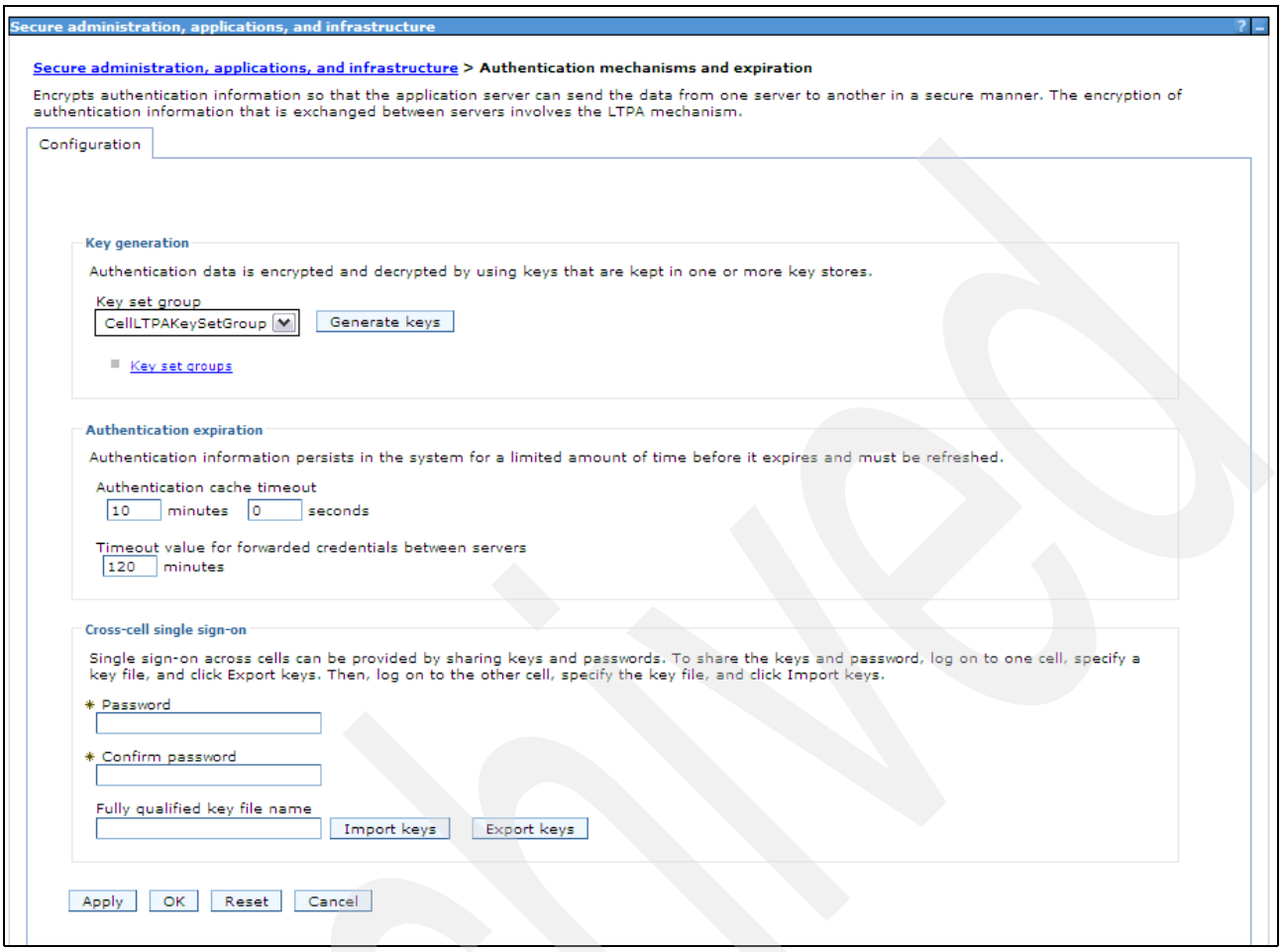


Figure 13-6 Importing the Tivoli Storage Productivity Center Security Keys

10. Log out of the WebSphere Application Server administrative console.
11. Restart Tivoli Provisioning Manager.

13.4.5 Defining the server in Tivoli Provisioning Manager

Define the Tivoli Storage Productivity Center server in Tivoli Provisioning Manager by running `TPC_DefineTPCServer.wkf`, the provisioning workflow. You must perform these steps for each Tivoli Storage Productivity Center that you want to add to Tivoli Provisioning Manager.

1. Click **Go To** → **Administration** → **Provisioning** → **Provisioning Workflows**.
2. Search for TPC_DefineTPCServer in the list.
3. From the **Select Action** menu, click **Run**.
4. Select the provisioning workflow.

Archived

5. Provide the parameters listed in Table 13-1, and then click **Run**.

Table 13-1 Workflow parameters

Parameter name	Description	Lab value
hostname	The host name of Tivoli Storage Productivity Center Server.	tpcprod
ipaddress	The IP address to use for communication with the Tivoli Storage Productivity Center Server.	10.66.17.6
portnum	The port number to connect to the Tivoli Storage Productivity Center Device server. The default is 9550.	9550
username	The user name that has Superuser privileges on the Tivoli Storage Productivity Center server.	administrator
password	The password to match the provided user name.	******

Figure 13-7 shows defining of the Tivoli Storage Productivity Center server in Tivoli Provisioning Manager by running `TPC_DefineTPCServer.wkf`, the provisioning workflow.

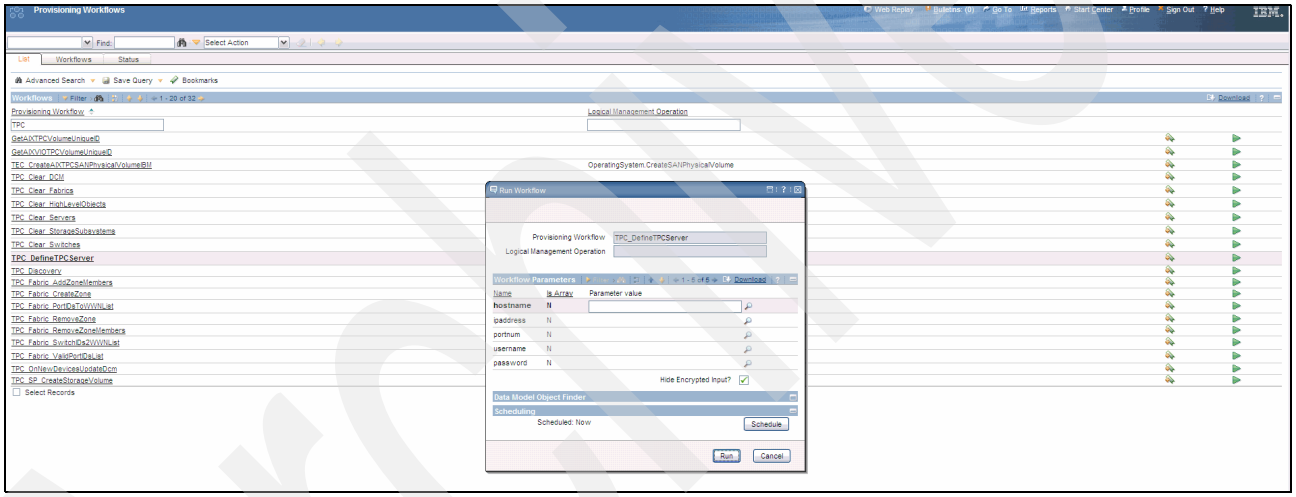


Figure 13-7 Running the TPC_DefineTPCServer.wkf provisioning workflow

6. Run TPC Discovery to populate the TotalStorageProductivity Center data, such as storage subsystems, storage pools, fabrics, and managed computers, in Tivoli Provisioning Manager:
   a. Click **Go To** → **Discovery** → **Provisioning Discovery** → **Discovery Configurations**.
   b. In the list of discovery configurations, click **TPC Discovery**.
   c. Click **Run Discovery**.
   d. Select the Tivoli Provisioning Manager as the target server of the discovery.
   e. Click **Submit**. See Figure 13-8.

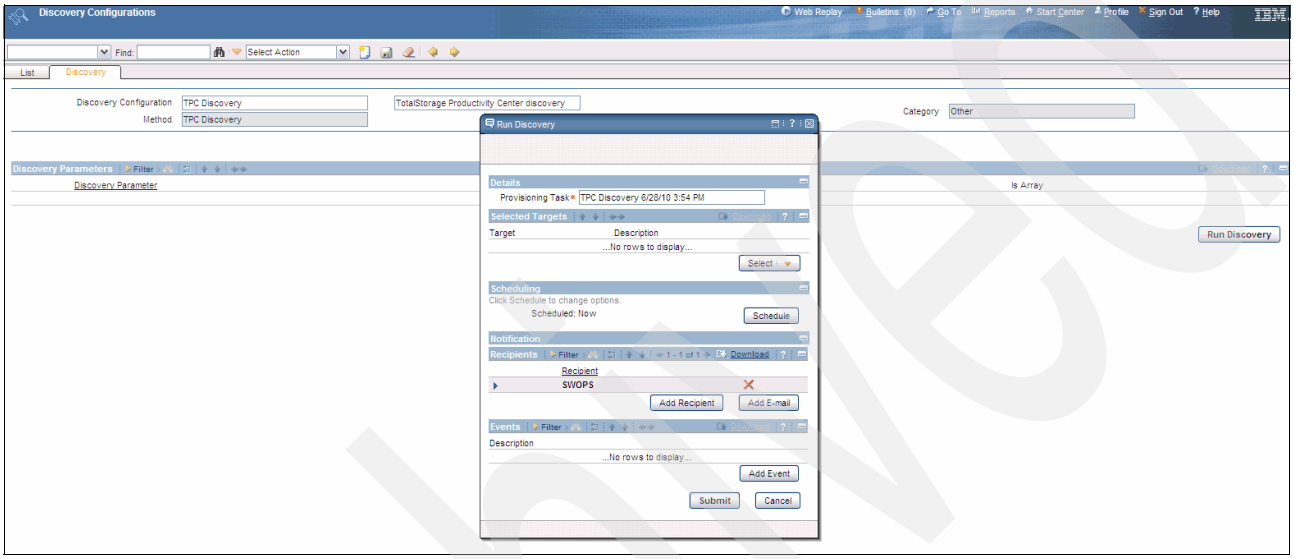


Figure 13-8 Running Discovery of Tivoli Storage Productivity Center

Archived

13.5 Scenario walk-through

In this section, we follow the overall end-to-end application provisioning solution and show how storage provisioning fits into an end-to-end solution:

- Solution architecture
- Environment prior to end-to-end provisioning workflow
- Triggering the end-to-end provisioning workflow
- Viewing the environment after the workflow has completed

13.5.1 Solution architecture

In this scenario, we use Tivoli Provisioning Manager to automate the provisioning of the Trade application. This application is hosted on three machines: AIX, Linux, and Windows. The Windows and Linux machines are built as VMWare ESX guests; the AIX system is built as an LPAR on the IBM Power 570. Storage is provisioned to the AIX LPAR through Tivoli Storage Productivity Center through a San Volume Controller and provisioned on DS4800. Finally, the Trade application is deployed and started. See Figure 13-9.

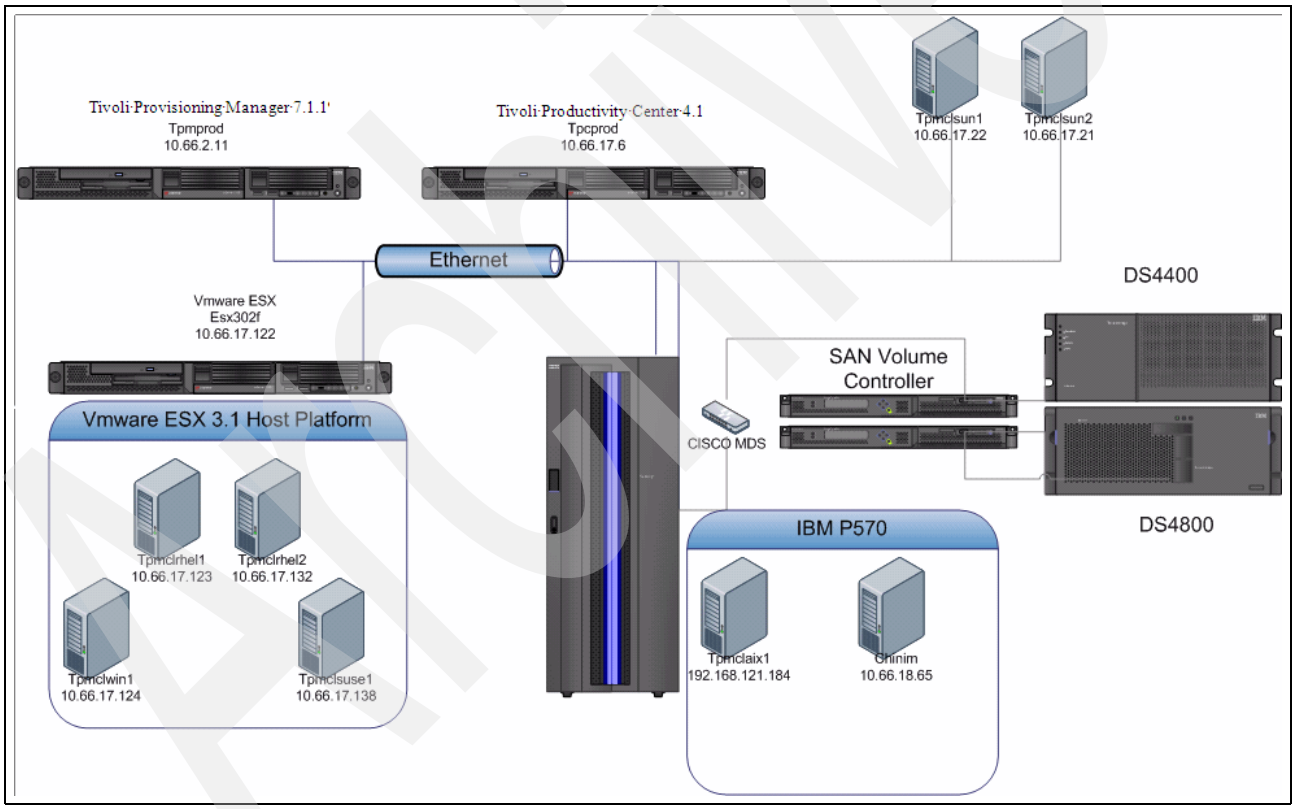


Figure 13-9 Physical solution architecture

Proof

Based on the physical architecture described here, the complete solution provisions in 12 - 17 minutes. See Figure 13-10.

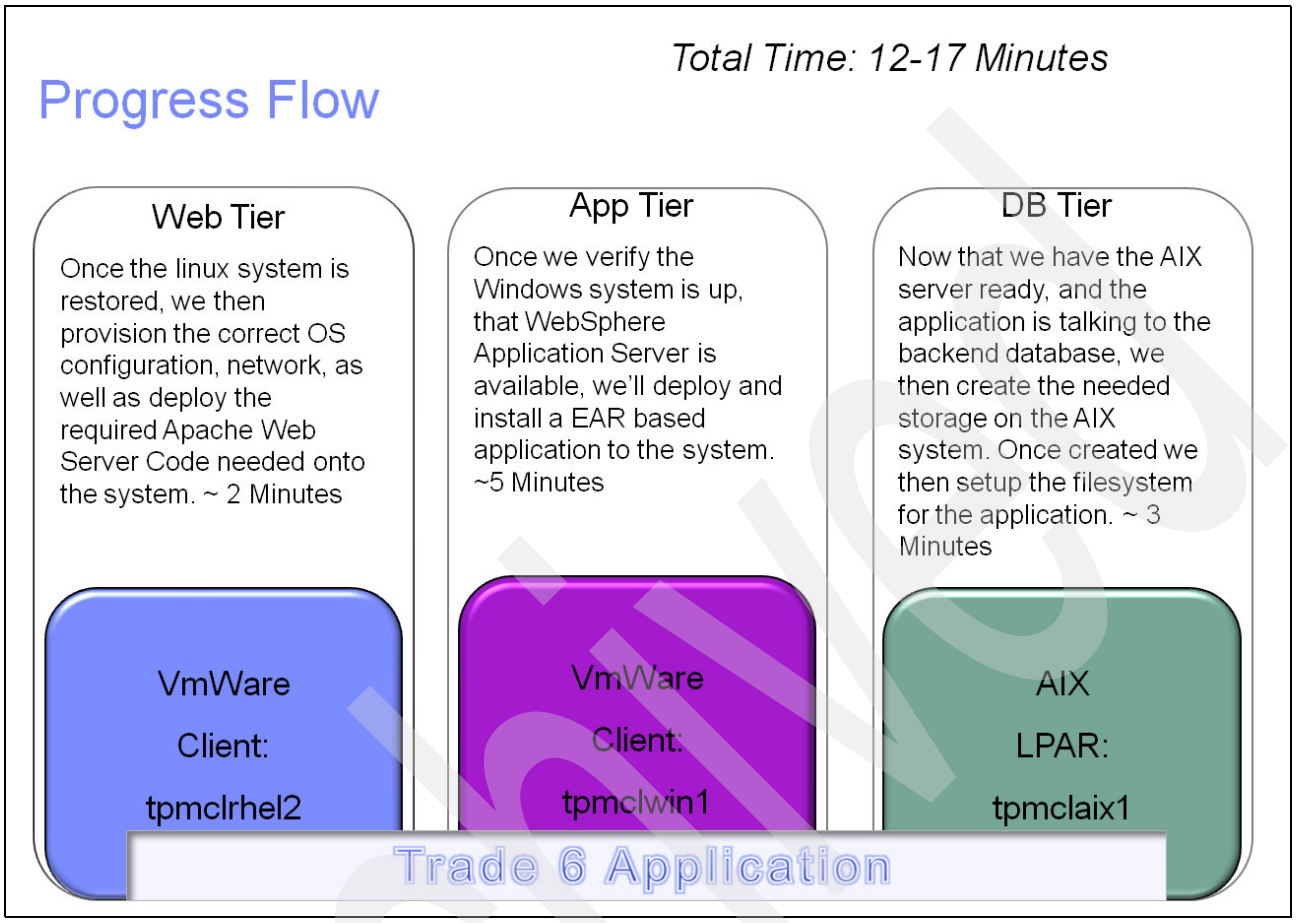


Figure 13-10 Logical solution design

13.5.2 Environment prior to end-to-end provisioning workflow

Prior to provisioning SAN-attached disk volumes we validate that no SAN-attached disks are attached to the system. A quick look into Tivoli Storage Productivity Center shows that there are no SAN volumes to our AIX LPAR. See Figure 13-11 on page 392.

Archive

In the figure, the disk is only mapped to an individual mdisk and not connected to any volumes on the right.

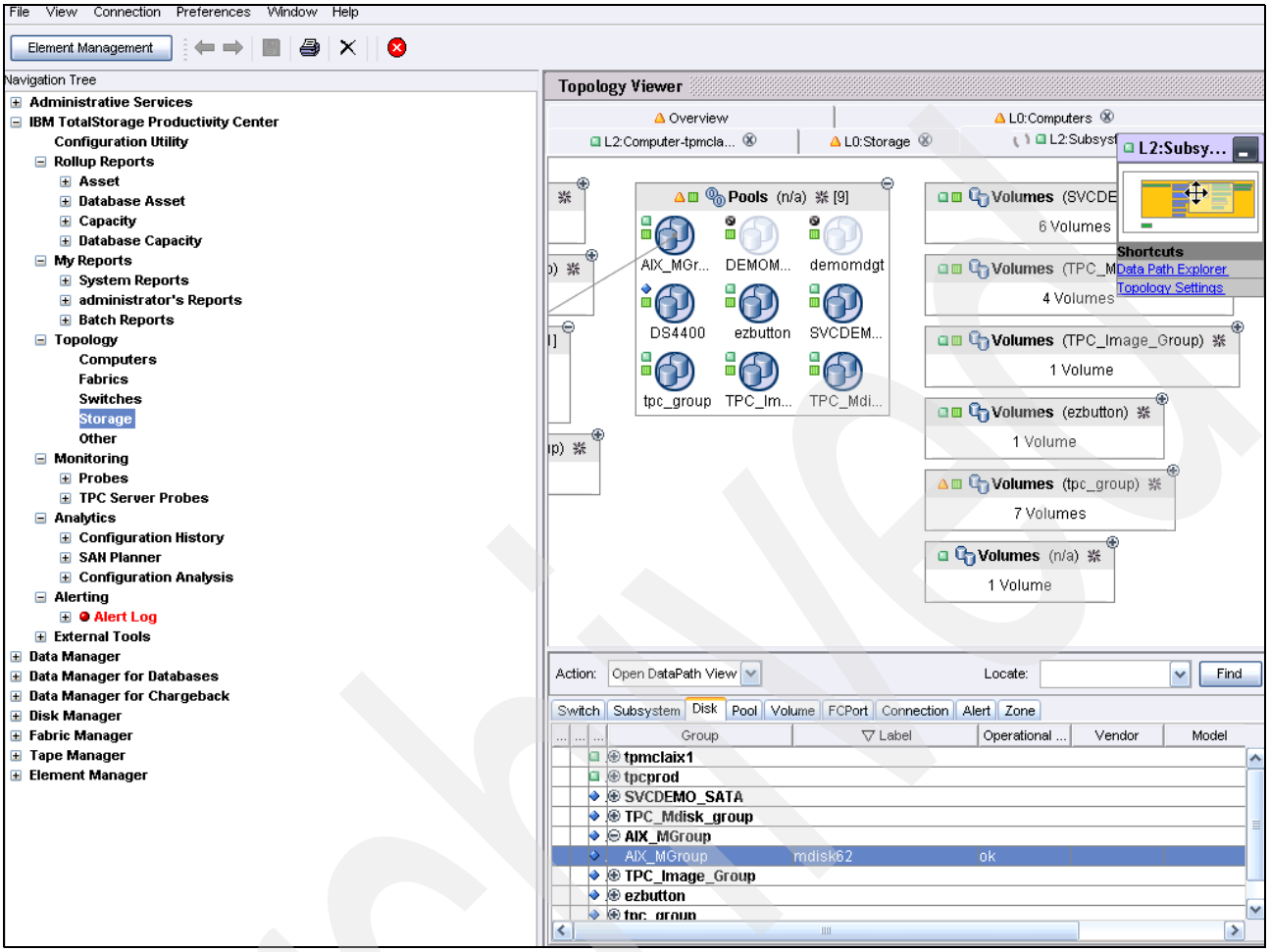


Figure 13-11 Tivoli Storage Productivity Center Validation: No disk mapped to hosts

In addition, a check of the actual AIX System shows that only internal disks are available on the system. See Figure 13-12.

```
root@tpmclaix1 / # df
Filesystem    512-blocks      Free %Used    Iused %Iused Mounted on
/dev/hd4          1474560    642304   57%     2109     3% /
/dev/hd2          8192000   1991160   76%    65620    23% /usr
/dev/hd9var       4194304   2319424   45%      764     1% /var
/dev/hd3          5079040   4696320    8%      536     1% /tmp
/dev/hd1           425984    314768   27%      834     3% /home
/proc                 -         -     -        -      -  /proc
/dev/hd10opt      8192000   4237904   49%     3268     1% /opt
/dev/datavlm     24248320  24243792    1%      101     1% /repository
root@tpmclaix1 / # lsvg
rootvg
datavg
root@tpmclaix1 / #
```

Figure 13-12 Check of the AIX system shows no SAN Volumes are available

13.5.3 Triggering the end-to-end provisioning workflow

The provisioning of the environment is started through a web service call from an external application front end. See Figure 13-13.



Figure 13-13 External application starting the end-to-end provisioning workflow

After the external web application triggers the workflow in Tivoli Provisioning Manager, the end-to-end build process starts. See Figure 13-14 on page 394.

Archived

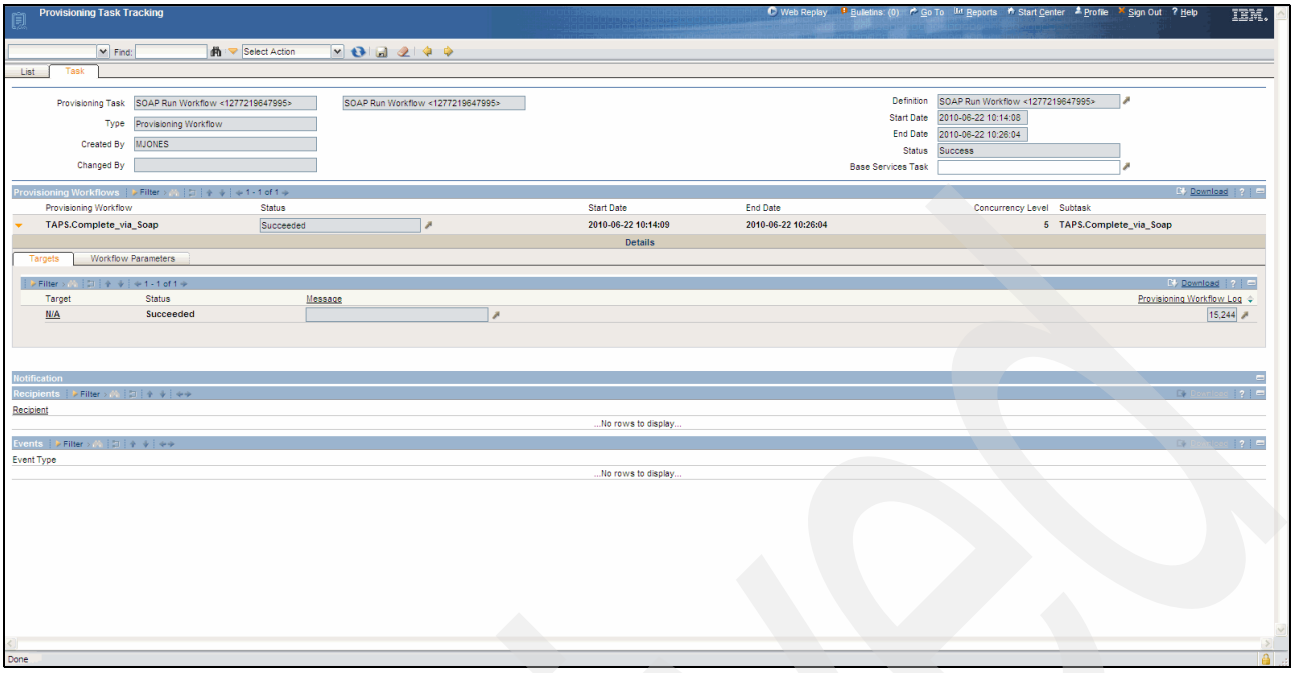


Figure 13-14 Tivoli Provisioning Manager workflow triggered

The workflow steps through the individual tasks that are required to build the environment. In addition, it provides details about any errors that occur as the workflow progresses. See Figure 13-15.

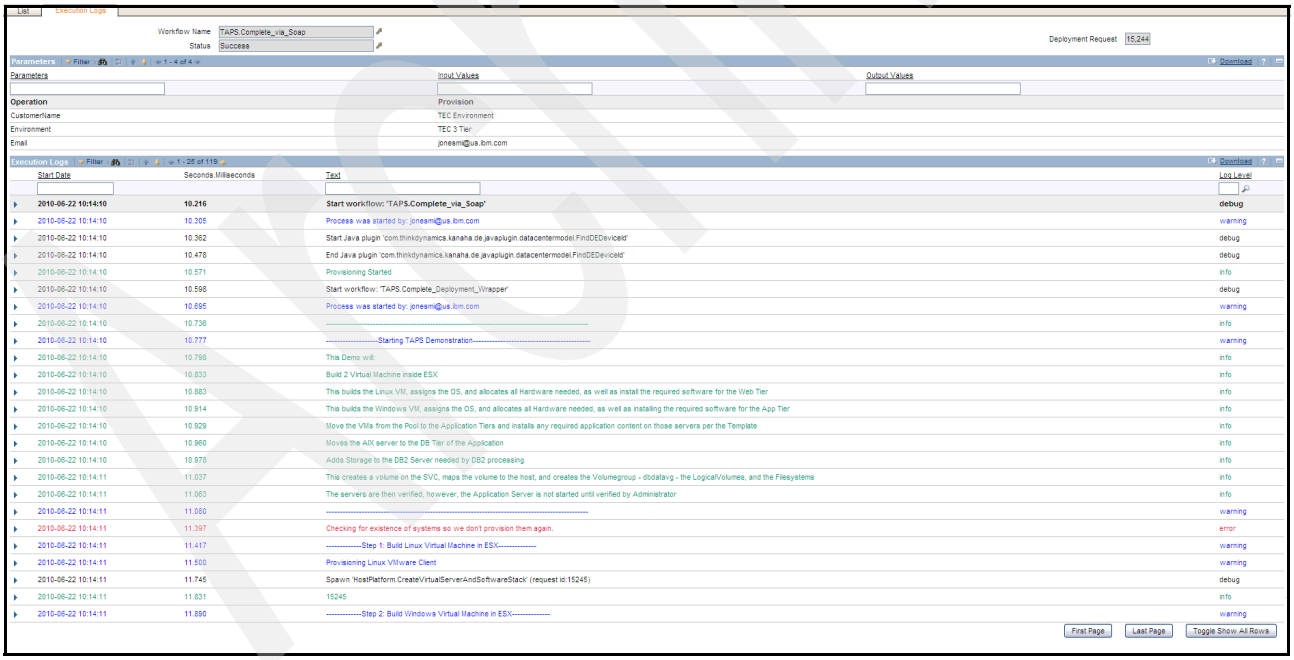


Figure 13-15 Workflow status details

13.5.4 Viewing the environment after the workflow has completed

Validation of Tivoli Storage Productivity Center shows that a new volume has been created and allocated to our AIX system. See Figure 13-16.

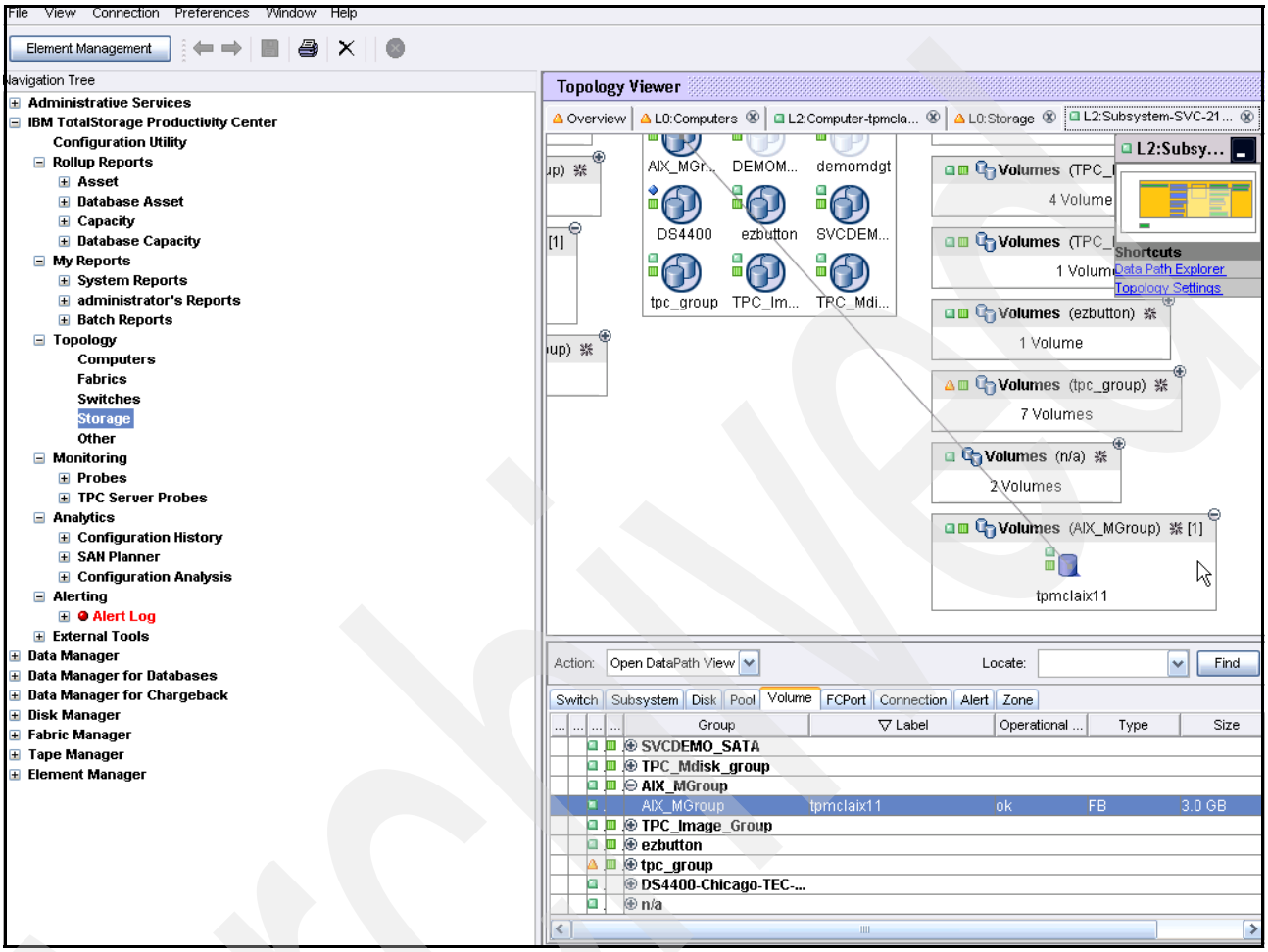


Figure 13-16 Disk is now allocated to system

Archive

Logging in to the AIX System and looking at the file system configuration also shows a new volume group has been created and a SAN-attached disk has been allocated. See Figure 13-17.

```
root@tpmclaix1 / # df -k
Filesystem    1024-blocks      Free %Used    Iused %Iused Mounted on
/dev/hd4           737280    321148   57%     2118     3% /
/dev/hd2          4096000    995580   76%    65620    23% /usr
/dev/hd9var       2097152   1159700   45%      765     1% /var
/dev/hd3          2539520   2348160    8%      536     1% /tmp
/dev/hd1           212992    157384   27%      834     3% /home
/proc                   -         -    -        -     -  /proc
/dev/hd10opt      4096000   2118952   49%     3268     1% /opt
/dev/datavlm     12124160  12121896    1%      101     1% /repository
/dev/san1         1703936   1703348    1%        4     1% /data
root@tpmclaix1 / # lsvg
rootvg
datavg
sanvg
root@tpmclaix1 / # lspv |grep vpath
vpath5          none                                None
vpath6          none                                None
vpath7          none                                None
vpath8          none                                None
vpath9          none                                None
vpath10         none                                None
vpath11         none                                None
vpath12         none                                None
vpath13         00cbd4acf6048f52                    sanvg           active
root@tpmclaix1 / #
```

Figure 13-17 AIX system has SAN-attached disk through automated provisioning processes

Archived

After verifying that the Tivoli Provisioning Manager Integration with Tivoli Storage Productivity Center has successfully completed, we ensure that the end-to-end application has successfully been provisioned. See Figure 13-18.

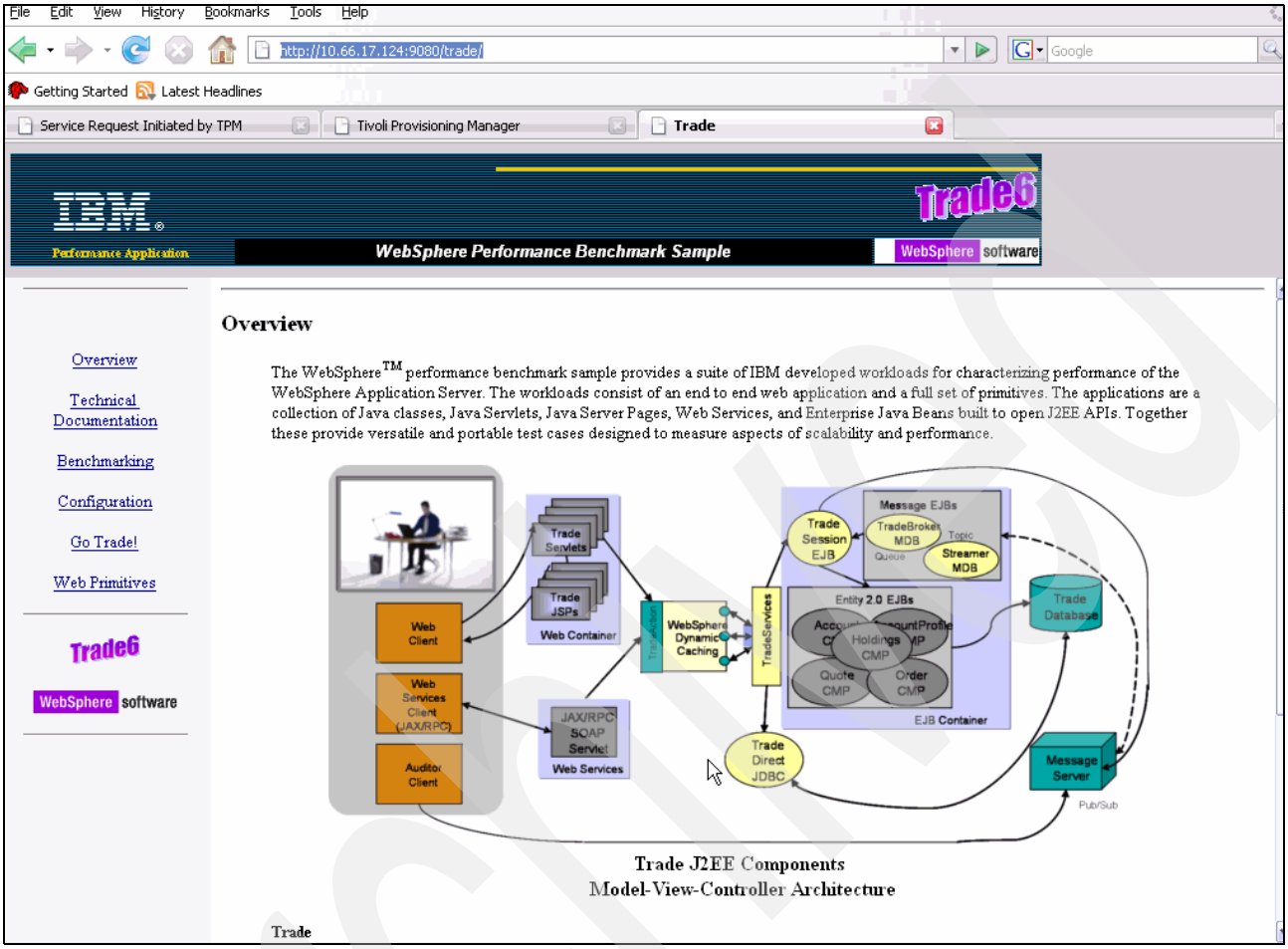


Figure 13-18 Validating that the Trade application has been built successfully

Archive

13.6 Summary

Figure 13-19 shows the summary of this scenario.



Figure 13-19 Quick summary

Archived

14

Tivoli Provisioning Manager discovery of TADDM discovered CIs

This chapter provides information about using TADDM discovered CIs with Tivoli Provisioning Manager discovery. We describe how Tivoli Provisioning Manager can discover software and hardware components by using TADDM and how to manage discovered resources from Tivoli Provisioning Manager.

This chapter contains the following topics:

- 14.1, “Scenario description” on page 400
- 14.2, “Products involved” on page 400
- 14.3, “Benefits” on page 400
- 14.4, “Implementation steps” on page 400
- 14.5, “Summary” on page 404

© Copyright IBM Corp. 2011. All rights reserved.

14.1 Scenario description

Tivoli Provisioning Manager is used for provisioning machines with the necessary operating system, required additional software, and virtualization provisioning. Tivoli Application Dependency Discovery Manager (TADDM) is used for discovering the live nodes in a corporate network with various levels of details (L1, L2, L3 and more). Tivoli Provisioning Manager too has to discover a node before it can provision it. The scenario in this chapter describes the method used for importing TADDM discovered configuration items (CIs) into Tivoli Provisioning Manager, so that Tivoli Provisioning Manager data center model (DCM) contains more details of data and a discovery is not required to run from Tivoli Provisioning Manager while provisioning. More details about the benefits of this solution are in 14.3, “Benefits” on page 400.

14.2 Products involved

The following products are involved in this integration:

- IBM Tivoli Provisioning Manager V 7.2
- IBM Tivoli Application Dependency Discovery Manager 7.2.0.1

See the following web resources:

- Tivoli Provisioning Manager Installation Guide

  http://publib.boulder.ibm.com/infocenter/tivihelp/v38r1/index.jsp?topic=/com.ibm.tivoli.tpm.ins.doc/book/bookhtmlcover.html

- TADDM

  http://publib.boulder.ibm.com/infocenter/tivihelp/v10r1/index.jsp?topic=/com.ibm.taddm.doc_7.2/plugin_files/cmdb_relnotes.html

14.3 Benefits

The benefits of this solution are as follows:

- Ease of provisioning through Tivoli Provisioning Manager because all IT elements are discovered and stored in Tivoli Provisioning Manager DCM.
- No need to run discovery from Tivoli Provisioning Manager.
- L2 and L3 discovery information can be obtained from TADDM.
- Relationship among CIs can be established in Tivoli Provisioning Manager, which is not possible otherwise

14.4 Implementation steps

Perform the following steps to implement this solution:

1. Click **Go To** → **Discovery** → **Provisioning Discovery** → **Discovery Configurations**.
2. Select the **Tivoli Application Dependency Discovery Manager Discovery configuration.** See Figure 14-1 on page 401.

Archive

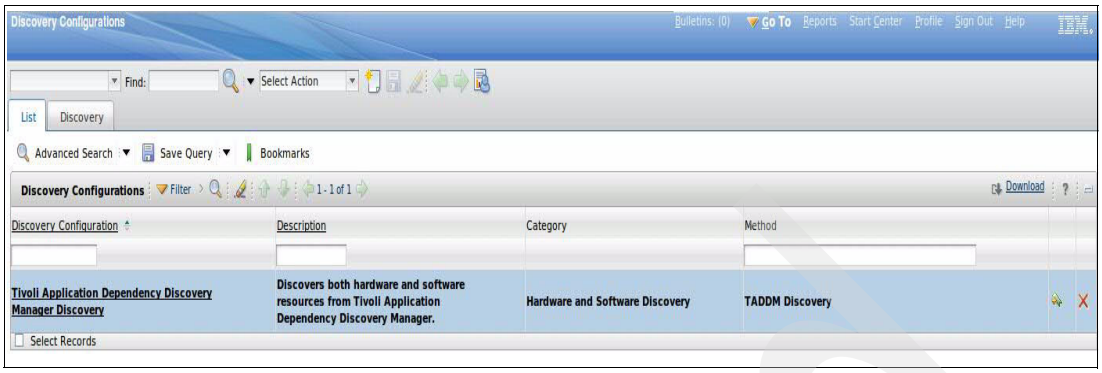


Figure 14-1 Tivoli Application Dependency Discovery Manager Discovery Configuration

3. Type the name and description for the discovery.
4. In the Category field, type `Hardware Discovery` or `Software Discovery`.
5. In the Method field, type `TADDM Discovery`.
6. Click **Next**.
7. Click the **Server Information** tab, and provide the following TADDM server information, as shown in Figure 14-2:
   a. Name: The name of the TADDM server that will perform the discovery
   b. Port Number: The port number of the server
   c. User ID and Password: The credentials of the TADDM server

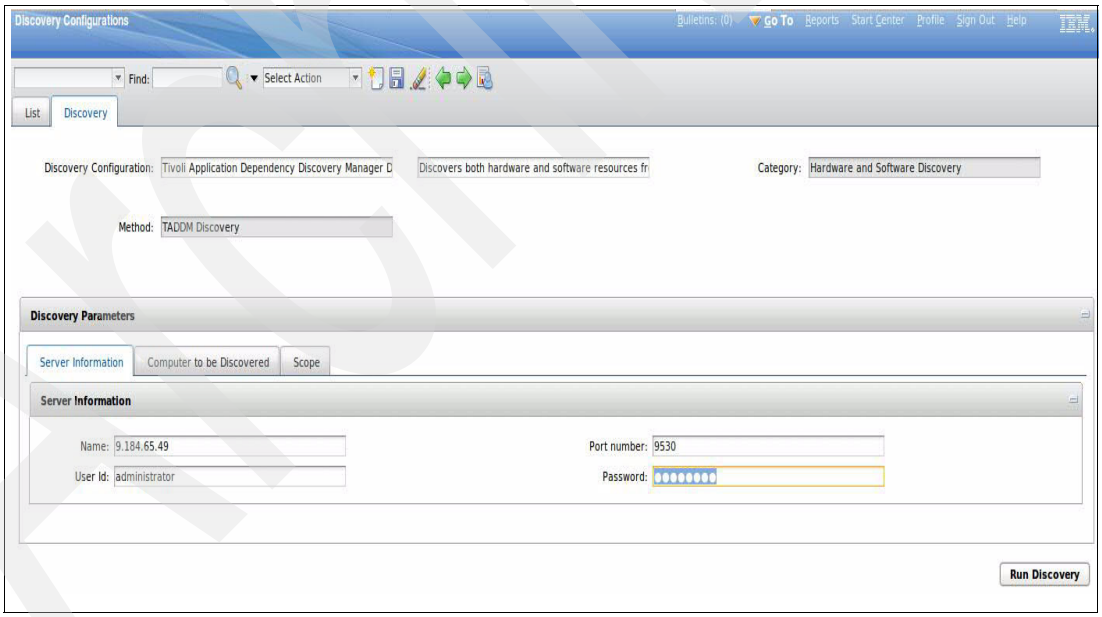


Figure 14-2 Configuring the discovery

8. In the Computer to be Discovered tab, specify the **Fully Qualified Host Name** of the computers you want to discover. All the hosts that you specify in this list will be discovered on the TADDM server.

Archive

Notes:

- If you want to discover all the resources on the TADDM server, do not specify any hosts in the discovery list. If you leave the list empty, all resources on the TADDM server will be discovered.
- Before performing any software distribution operation on the computers imported from TADDM, you must manually add the service access point to them, because on the computers imported from TADDM, the service access point is not automatically created.

9. In the Scope tab, select the appropriate check boxes to determine what software information will be discovered on the TADDM servers:
   - Installed Software Components. Installed software components are software installations that are registered under the operating system of a computer. For example, on a Windows computer, they are registered in the registry editor, and for Linux, in the RPM package manager. If you select this option, all software installations that are in the operating systems registry for that computer are discovered.

     Note: This option differs from the other options. This option discovers only basic details about the software installation in the registry. The other options discover composite application information, including detailed configuration information.

   - WebSphere Application Server. Discover WebSphere Application Server information such as the installation profile and path, the WebSphere server instance, version, and type, and the WebSphere node and cell.
   - Oracle. Discover Oracle information such as server and database instances, the product version, the home installation path and name, its software ID, and its port and base path. After the discovery is complete, the Oracle server instance can be started or stopped from provisioning if the appropriate credentials to access it are set.
   - Web Server. Discover three types of web servers: IBM HTTP Server, Apache Server, and Microsoft IIS. Various configuration details can be discovered for each web server. See “Related publications” on page 805 for more details.
   - WebLogic Server. Discover WebLogic such as server instances, product version, installation path, J2EE application names and file paths, and the BEA home.
   - DB2. Discover DB2 information such as server instances, ports, the installation path, the `DB2.ESE` and `DB2.ADCL` files, and the following parameters: uninstall_delete_users, db2_product_key default_DB2INSTANCE, das_group_name, das_user_name, fence_group_name, fence_group_id, fence_user_name, instance_user_name, instance_port, db2_auth_type, db2_word_width, db2_auto_start
   - VMWare ESX. Discover information about the virtual computers in your environment.
   - Business Applications. Discover information such as the business applications installed, their functional groups and application tier numbers.
   - Computer Collections. Discover information about how the computers are grouped in your environment.
10. Click **Save.**
11. Click **Run Discovery**. The Run Discovery panel opens (Figure 14-3 on page 403).
12. Click **Submit**.

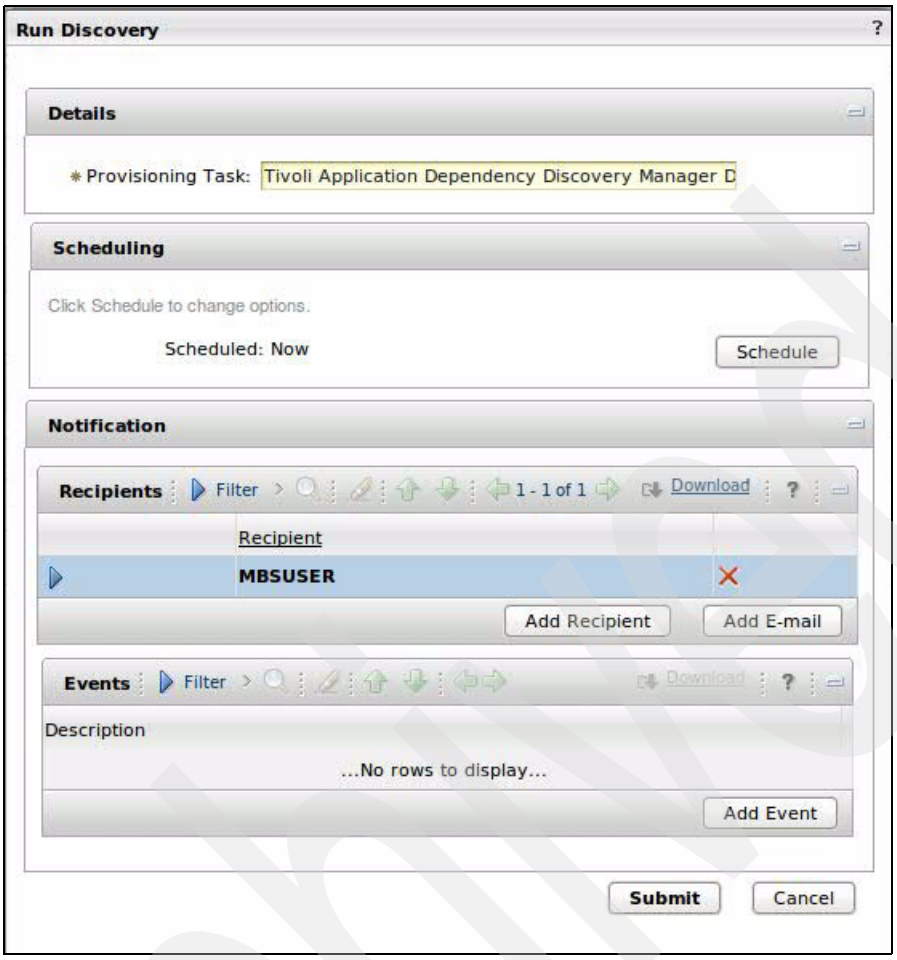


Figure 14-3 Submitting discovery

The discovery can take time, depending on the number of computers to be discovered and the level of details. After the discovery is completed, view the computer by clicking **Go To** → **IT Infrastructure** → **Provisioning Infrastructure** → **Provisioning Computer**. See Figure 14-4.

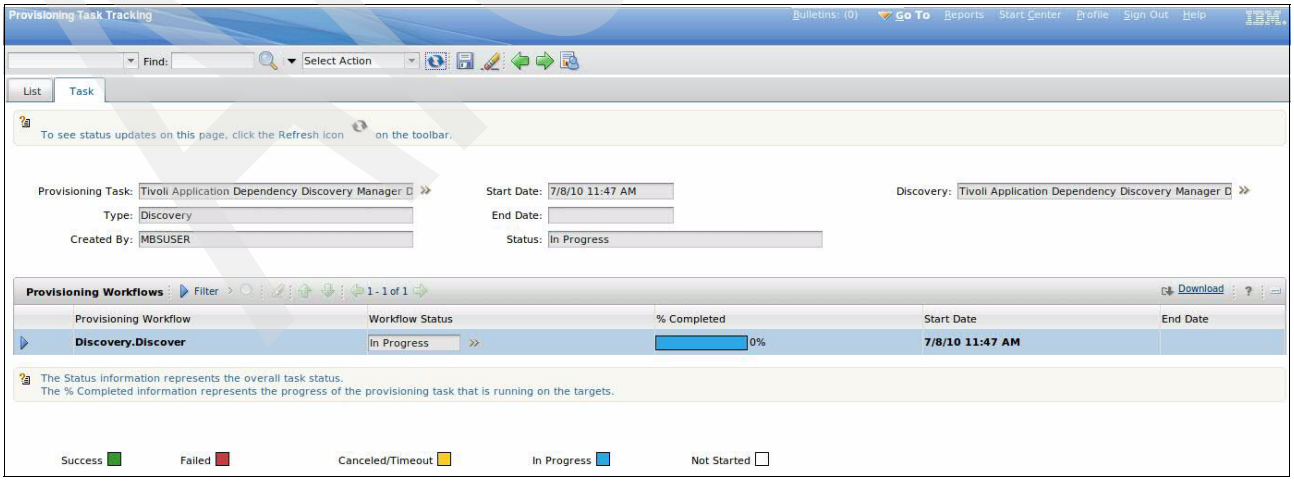


Figure 14-4 Discovery in progress

Selecting a computer will list all the details including hardware and software information, operating system running on the server, software installed, CPU, memory, disk details. We are now ready to do any provisioning task on the discovered resources.

Note: The level of details available depends on whether a system was discovered with L1, L2 or L3 discovery from TADDM.

14.5 Summary

Figure 14-5 shows the quick summary of the solution.

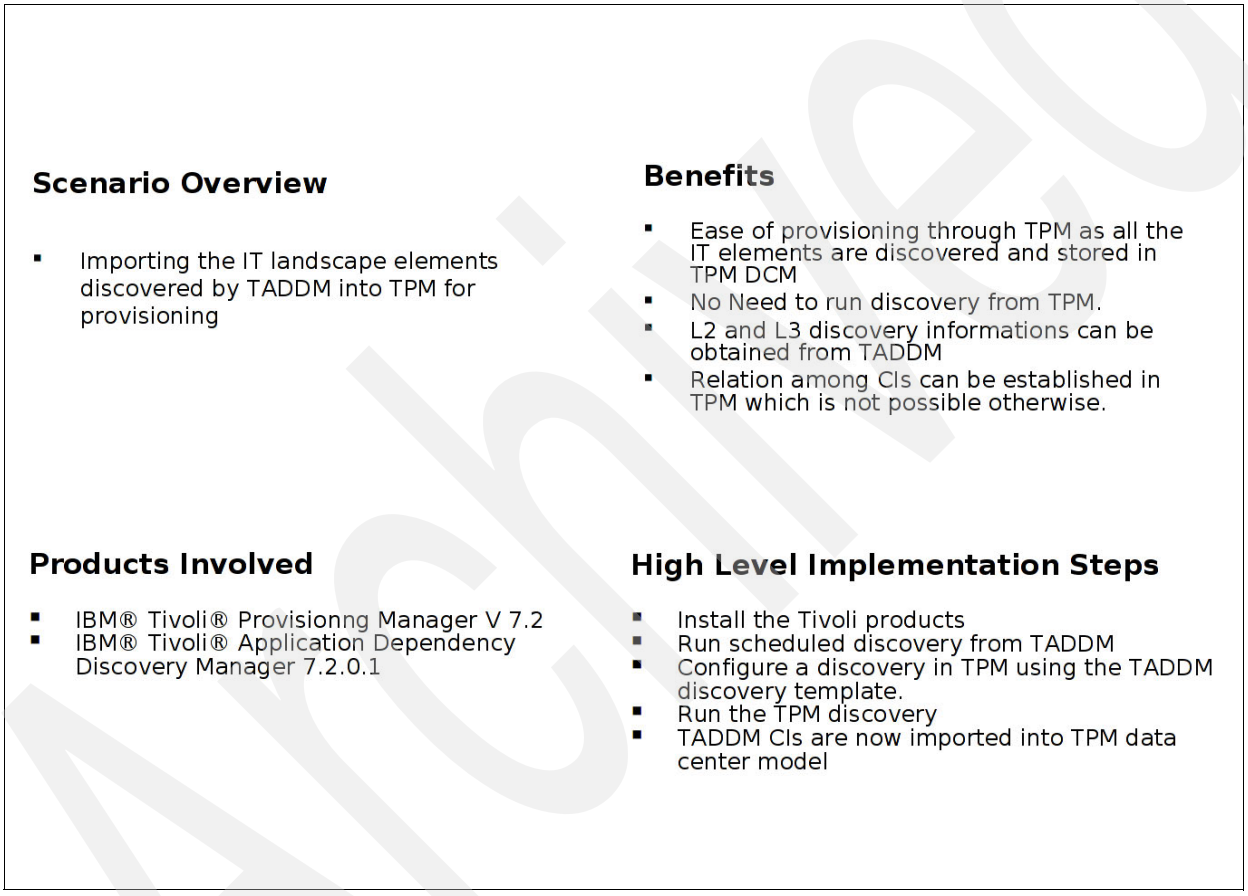


Figure 14-5 Quick summary

Archived

15

Tivoli Provisioning Manager and Tivoli Identity Manager integration

This chapter has details about the Tivoli Provisioning Manager and Tivoli Identity Manager integration scenario. It describes benefits, architecture, implementation, and a walk-through of the integration scenario.

This chapter contains the following topics:

- 15.1, “Scenario introduction” on page 406
- 15.2, “Benefits” on page 406
- 15.3, “Products involved” on page 406
- 15.4, “Architecture of the scenario” on page 406
- 15.5, “Implementation of the scenario” on page 408
- 15.6, “Scenario walk-through” on page 410
- 15.7, “Summary” on page 412

© Copyright IBM Corp. 2011. All rights reserved.

15.1 Scenario introduction

Tivoli Provisioning Manager is used for provisioning machines with the necessary operating system and required additional software. Tivoli Identity Manager is used for managing the identities and the accounts on the machines. To enable Tivoli Identity Manager to manage the user accounts on the Tivoli Provisioning Manager provisioned machines, the provisioned machines must be registered in Tivoli Identity Manager. The integration scenario described in this chapter helps in addressing this use case.

15.2 Benefits

The benefits of Tivoli Provisioning Manager and Tivoli Identity Manager integration are as follows:

- User identities and accounts can be managed from the moment that the machine is provisioned.
- Identity and accounts management can be automated.
- Management of identities and accounts are centralized by using Tivoli Identity Manager features.
- Enables management of accounts on large scale machines provisioned in enterprise lab, data center, or cloud environment.

15.3 Products involved

The following products are used for implementing the scenario:

- Tivoli Provisioning Manager V7.2

  See the installation and configuration information for Tivoli Provisioning Manager 7.2:

  http://publib.boulder.ibm.com/infocenter/tivihelp/v38r1/index.jsp?topic=/com.ibm.tivoli.tpm.ins.doc/book/bookhtmlcover.html

- Tivoli Identity Manager V5.1

  See the installation and configuration information for Tivoli Identity Manager 5.1:

  http://publib.boulder.ibm.com/infocenter/tivihelp/v2r1/index.jsp?topic=/com.ibm.itim.doc/top/landing-install.html

15.4 Architecture of the scenario

The Tivoli Provisioning Manager and Tivoli Identity Manager integration consists of two major components (Figure 15-1 on page 407):

- Tivoli Provisioning Manager automation package
- Tivoli Identity Manager solution.

The Tivoli Provisioning Manager workflow invokes an application programming interface (API) from Tivoli Identity Manager solution to register provisioned machine on Tivoli Identity Manager.

Archived

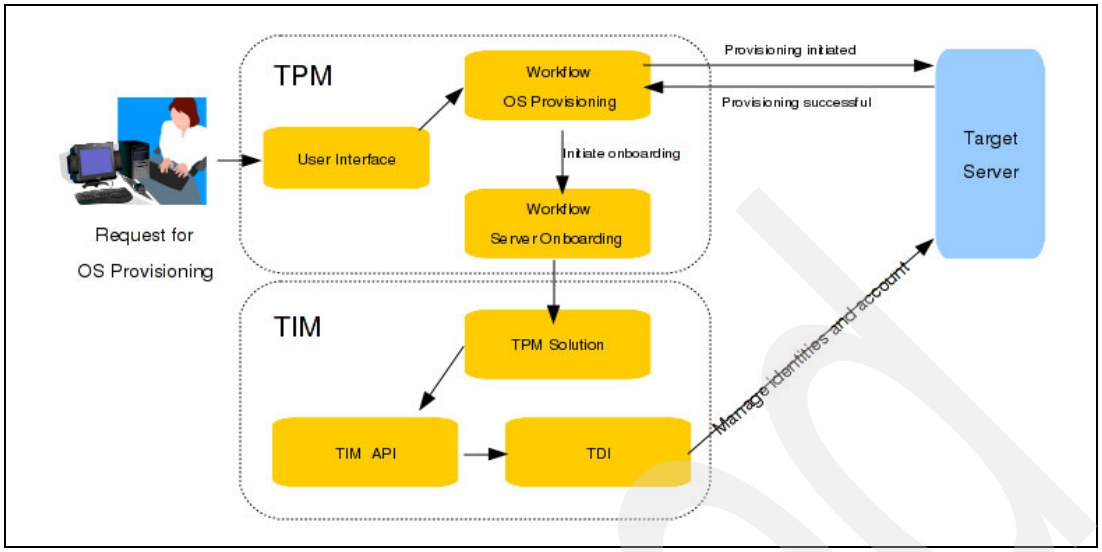


Figure 15-1 Tivoli Identity Manager Tivoli Provisioning Manager Integration architecture

Figure 15-2 shows integration logic.

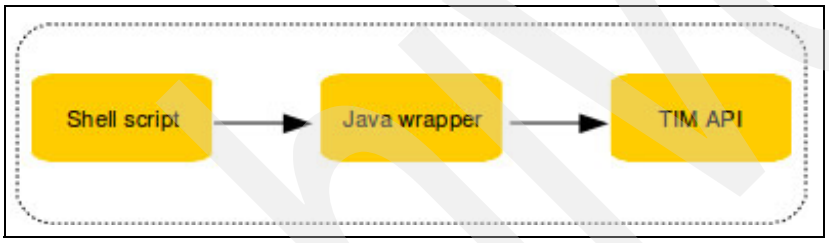


Figure 15-2 Integration logic

Tivoli Provisioning Manager automation package (tcdriver) contains a workflow which can be directly called from another Tivoli Provisioning Manager workflow, invoked from other products like Tivoli Service Automation Manager, or run manually from Tivoli Provisioning Manager UI. The Tivoli Provisioning Manager automation package is installed on Tivoli Provisioning Manager server.

Tivoli Identity Manager solution is composed of a Java wrapper module. As Figure 15-1 shows, Tivoli Provisioning Manager workflow executes the shell script that in turn invokes the Tivoli Identity Manager Client API though the Java wrapper module. The Tivoli Identity Manager solution can reside on any other intermediate server or on Tivoli Identity Manager server itself.

Tivoli Identity Manager manages user identities and accounts on the provisioned UNIX servers by using IBM Tivoli Directory Integrator.

Tivoli Directory Integrator: Tivoli Directory Integrator is a generic data integration tool that is used to address problems that require custom coding and more resources than traditional integration tools. It can move, transform, harmonize, propagate, and synchronize data across otherwise incompatible systems. For more information, see “Chapter 2 Integration Components” in *Integration Guide for IBM Tivoli Service Request Manager V7.1*, SG24-7580. This book details the Tivoli Directory Integrator architecture and components, and presents several integration scenarios where Tivoli Directory Integrator is a key component of the integration.

15.5 Implementation of the scenario

This section provides implementation details of Tivoli Provisioning Manager automation package, Tivoli Identity Manager solution. It also lists the considerations of current implementation.

15.5.1 Tivoli Provisioning Manager automation package

The Tivoli Provisioning Manager workflow from integration automation package validates input parameters and makes use of a scriptlet (shell script) to invoke Java APIs exposed by Java wrapper module, and passes all the required parameters to register machine.

The input parameters to the workflow are as follows:

- User name of WebSphere Application Server administrator at Tivoli Identity Manager server
- Password of WebSphere Application Server administrator user at Tivoli Identity Manager server
- IP address of WebSphere Application Server on Tivoli Identity Manager server
- Host name of server (Tivoli Identity Manager or intermediate) on which Tivoli Identity Manager solution is installed
- Absolute path where Tivoli Identity Manager Solution is extracted (on Tivoli Identity Manager or intermediate server)
- User name of Tivoli Identity Manager Administrator
- Password of Tivoli Identity Manager Administrator
- IP address of provisioned machine
- Host name of provisioned machine
- User name of provisioned machine
- Password of provisioned machine
- Platform type (AIX, Linux, Windows and so on) of provisioned machine

The workflow returns zero value after successful workflow execution and non-zero value in case of failure.

Installing and configuring automation package (tcdriver)

Perform the following steps to install and configure the automation package:

1. Download the `TPM_TIM.tcdriver` package by using the instructions provided in Appendix A, “Additional material” on page 803 and copy it to the `<TIO_HOME>\drivers` folder on the Tivoli Provisioning Manager server. Note that `TIO_HOME` is the path where the Tivoli Provisioning Manager server is installed.
2. Change directory to `<TIO_HOME>\tools` directory.
3. Install the `tcdriver` as follows, depending on your operating system:
   - On Windows

     ```
     'tc-driver-manager.cmd fid TPM_TIM'
     ```

   - On UNIX or Linux

     ```
     ./tc-driver-manager.sh fid TPM_TIM
     ```

Note: The tcdriver can be uninstalled by using the following command:

```
tc-driver.manager.cmd fud TPM_TIM
```

4. Restart the Tivoli Provisioning Manager server using the following command:
   - On Windows

     ```
     <TIO_HOME>\tools\tio.cmd stop -t tpm
     <TIO_HOME>\tools\tio.cmd start -t tpm
     ```

   - On UNIX or Linux

     ```
     <TIO_HOME>/tools/tio.sh stop -t tpm
     <TIO_HOME>/tools/tio.sh start -t tpm
     ```

5. Discover Tivoli Identity Manager server by using Tivoli Provisioning Manager UI:
   a. Click **Go To** → **Discovery** → **Provisioning Discovery** → **Discovery Wizards**,
   b. Click **Computer Discovery** wizard from the list.
   c. Enter discovery parameters such as IP address range (IP address of Tivoli Identity Manager server), credentials as root and <password of root user> on Tivoli Identity Manager server.
   d. Click the **Review and Submit** link and submit the discovery request by clicking **Submit**.

      After discovery workflow is completed, click **Go To** → **Deployment** → **Provisioning computers** to show the host name of the Tivoli Identity Manager server.
6. Run **`TPM_TIM_Create_Service`** workflow to register the service on Tivoli Identity Manager server.

15.5.2 Tivoli Identity Manager solution

Tivoli Identity Manager exposes its public APIs for performing all the identity management operations. To avoid the complexity of using Tivoli Identity Manager APIs and to adapt to an environment that supports Java, a Java wrapper module is written over these APIs.

The Java wrapper module exposes Java APIs that can be invoked through other applications. In case of Tivoli Provisioning Manager and Tivoli Identity Manager integration scenario the Java APIs are invoked from the script in Tivoli Provisioning Manager workflow.

The Java wrapper module invokes the Tivoli Identity Manager APIs to perform Tivoli Identity Manager operations. It works on the concept of *remote method invocation (RMI)* and has a top-level class that accepts the required parameters to get ticket (platform, subject) and information about the service (machine) to be registered. This module takes care of all the protocols for acquiring tickets and invoking appropriate APIs to register the service.

This solution supports registering Windows, Linux, and AIX machines as a service on the Tivoli Identity Manager server.

Archived

15.5.3 Installing and configuring the Tivoli Identity Manager solution

Perform the following steps to install and configure Tivoli Identity Manager solution:

1. Download Tivoli Identity Manager solution using the instructions provided in Appendix A, “Additional material” on page 803 and extract its contents to a folder, which we refer to as `TIM_SOLUTION_HOME`.
2. From Tivoli Identity Manager server, copy the `enrole.properties` file from the `<TIM _HOME>/data` folder to `TIM _SOLUTION_HOME` folder. Also copy `sas.client.properties` and `ssl.client.properties` files from `<WAS-HOME>/profiles/<TIM-PROFILE>/properties` to `TIM_SOLUTION_HOME`. Note the TIM_HOME, WAS_HOME, TIM_PROFILE are the environment variables from Tivoli Identity Manager server.
3. Update the `data.jar` file with the copied three properties files by using the **`jar update`** command (for example, **`jar uf data.jar *.properties`**).

Important: Whenever these properties files are changed (for example security configuration changes or data structure change) in Tivoli Identity Manager server, you must copy them to `TIM_SOLUTION_HOME` and update `data.jar` file with these properties files.

15.5.4 Considerations

Consider the following information when you use this solution:

- Current implementation supports only registration of the provisioned machine. Removing the registration of the machine is not yet implemented.
- As the parameters are passed from the workflow to the Java wrapper module and then to Tivoli Identity Manager API, any change in parameters requires modification of the workflow and the Tivoli Identity Manager solution.

15.6 Scenario walk-through

The following steps walk through the scenario for registering a provisioned machine to Tivoli Identity Manager server:

1. The client sends machine provisioning request to Tivoli Provisioning Manager server (client is another workflow, UI or another application).
2. Tivoli Provisioning Manager server provisions the machine.
3. Tivoli Provisioning Manager and Tivoli Provisioning Manager integration workflow is invoked to register provisioned machine on Tivoli Identity Manager server. See Figure 15-3 and Figure 15-4 on page 411.

Archive

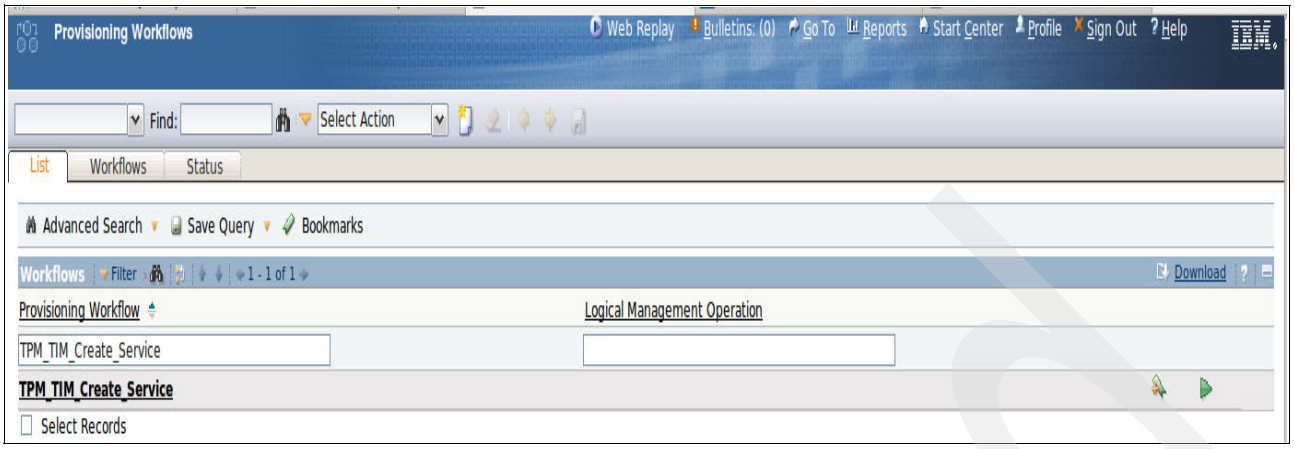


Figure 15-3 Integration workflow

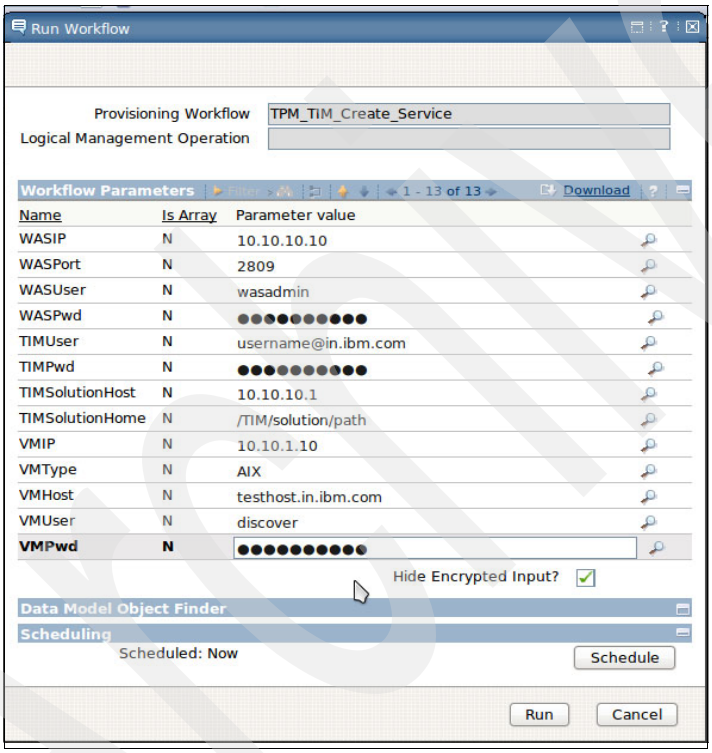


Figure 15-4 Running the workflow manually

4. Workflow executes the scriptlet (shell script).
5. The shell script calls Java API from the Tivoli Identity Manager solution.
6. Java wrapper module that is included with the Tivoli Identity Manager solution invokes the Tivoli Identity Manager API for registering the machine.
7. After execution of Tivoli Identity Manager API, the provisioned machine is registered on the Tivoli Identity Manager server.

15.7 Summary

Figure 15-5 shows the quick summary of this integration.

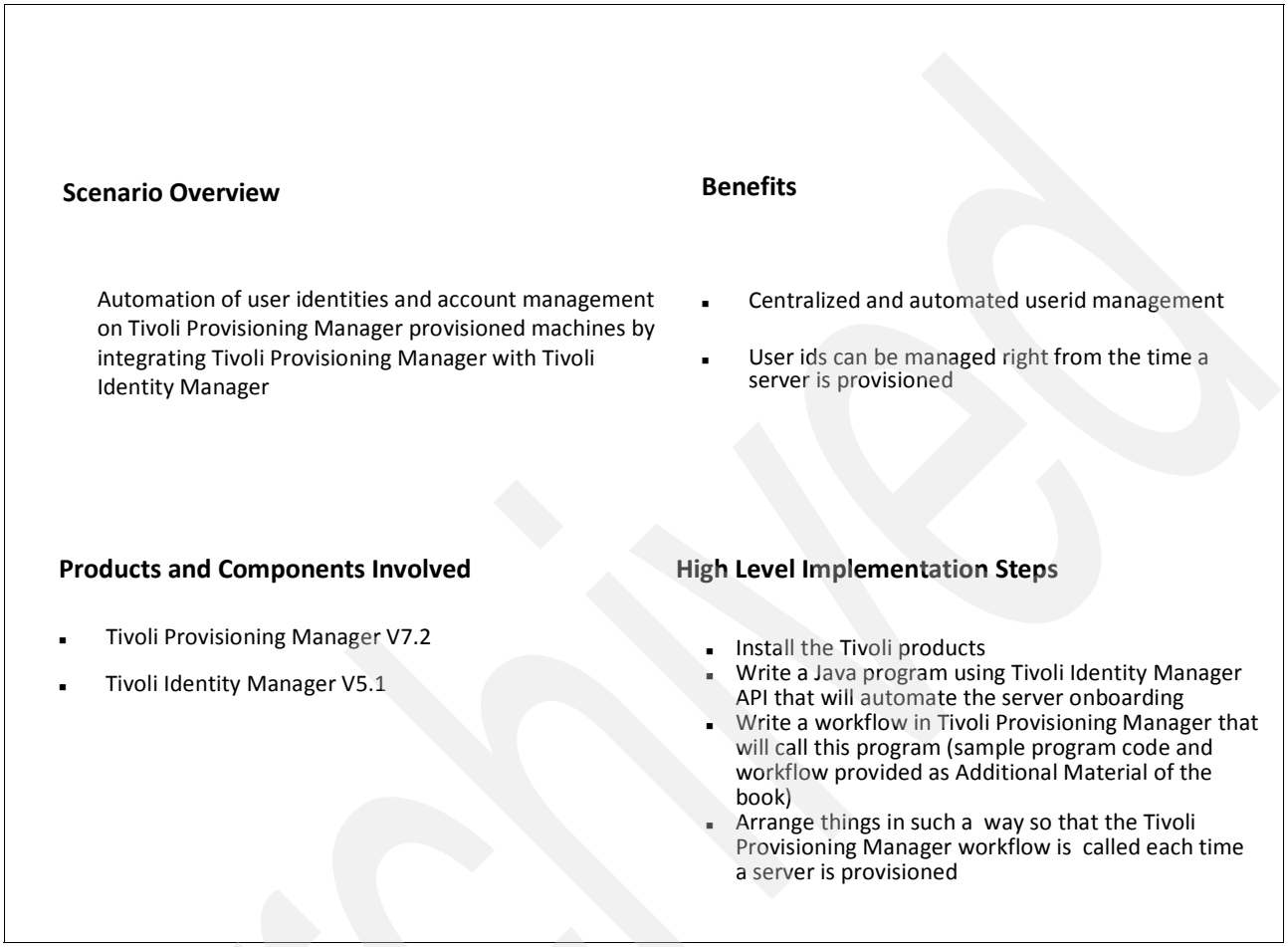


Figure 15-5 Quick summary

Archived

Part 5

IBM Service Management integration scenarios

In this part, we cover the IBM Service Management integration scenarios integration scenarios.

This part contains the following chapters:

- Chapter 16, “Taking automatic actions based on predefined policies” on page 415
- Chapter 17, “Asset and CI integration” on page 477
- Chapter 18, “Rational Asset Manager and Tivoli Change and Configuration Management Database integration” on page 545
- Chapter 19, “User and password management with Tivoli Identity Manager and Tivoli Service Request Manager” on page 595
- Chapter 20, “Employee onboarding” on page 623
- Chapter 21, “TADDM L2 discovery using IBM Tivoli Monitoring agents” on page 641
- Chapter 22, “Managing business services in an integrated environment” on page 661
- Chapter 23, “CCMDB V7.2.1: New launch-in-context technology” on page 699

© Copyright IBM Corp. 2011. All rights reserved.

Archived

16

Taking automatic actions based on predefined policies

This chapter provides an integration scenario for taking automatic actions based on predefined policies, when a problem is detected. This integration scenario is based on ITIL best practices, and covers the most commonly used solutions in this type of integration.

This chapter contains the following topics:

- 16.1, “Scenario overview” on page 416
- 16.2, “Products involved” on page 416
- 16.3, “Benefits” on page 417
- 16.4, “Architectural diagram of the integration” on page 417
- 16.5, “Implementation steps” on page 418
- 16.6, “Scenario walk-through” on page 474
- 16.7, “Summary” on page 476

© Copyright IBM Corp. 2011. All rights reserved.

16.1 Scenario overview

This scenario describes how to take automatic actions and provide fixes based on predefined policies, when a problem is detected.

Following event analysis and enrichment, one of the following actions is automatically triggered:

- Create incident ticket using Tivoli Netcool/OMNIbus: Tivoli Service Request Manager integration
- Create request for change using Tivoli Netcool/Impact: Change and Configuration Management Database integration
- Run workflow using Tivoli Netcool/Impact: Tivoli Provisioning Manager integration

Tivoli Netcool/Impact acts as a decision maker and chooses the predefined action referring to its mapping table, as shown in Table 16-1.

Table 16-1 Overview of event mapping

Alert group	Action	Description
ITM_NT_Process	Create incident ticket	Incident ticket is automatically created after maintenance window check. If incident happens outside of maintenance window, event is enriched through a mapping table. Ticket number appears in Tivoli Netcool/OMNIbus Active Event List.
ITM_DB_Connection_Pools	Create request for change	Request for change is automatically created and CCMDB response plan is triggered by CCMDB escalation to fulfill change details. Request for change (RFC) number appears in Tivoli Netcool/OMNIbus Active Event List.
ITM_Garbage_Collection_Analysis	Run workflow	Workflow runs automatically and its identification number and status appears in Tivoli Netcool/OMNIbus Active Event List.

Note: These event mapping examples are not based on best practices.

16.2 Products involved

The products and product components involved in this scenario are as follows:

- IBM Tivoli Netcool/OMNIbus V7.2.1
- IBM Tivoli Netcool/OMNIbus Probe for Tivoli EIF
- IBM Tivoli Netcool/OMNIbus Gateway for Tivoli Service Request Manager
- IBM Tivoli Netcool/Impact V5.1.1
- IBM Tivoli Monitoring V6.2.2 FP2
- IBM Tivoli Monitoring and Tivoli Event Synchronization

Archived

- IBM Tivoli Service Request Manager V 7.2.0.1
- IBM Tivoli Change and Configuration Management Database V7.2.0.1
- IBM Tivoli Provisioning Manager V7.2

16.3 Benefits

This scenario has several benefits:

- For Operations Managers:
  - Speeds mean-time-to-resolution by event enrichment.
  - Improves decision making by predefined policies and minimum human intervention.
  - Maximizes operational staff productivity.
- For Change Managers:
  - Improves change management quality by predefined responses and change procedures.
- For IT Architects:
  - Provides a fully integrated solution for automated event management.

16.4 Architectural diagram of the integration

Figure 16-1 shows the architectural diagram of this integration.

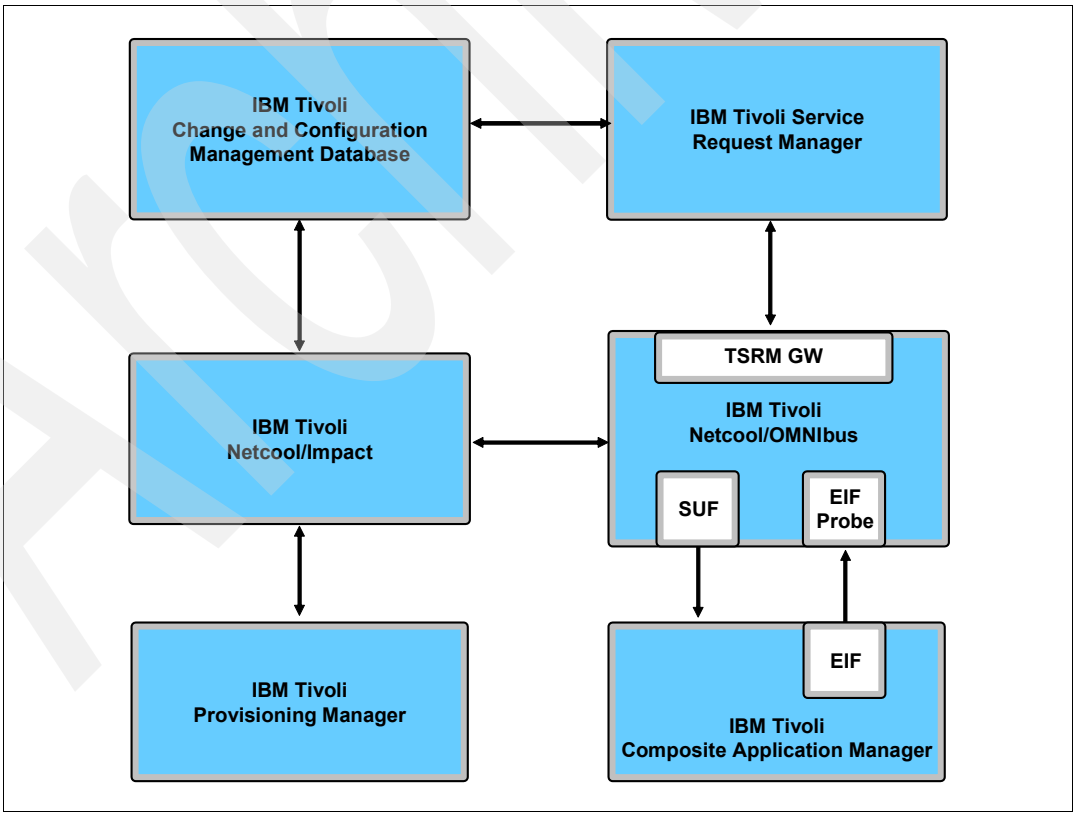


Figure 16-1 Architectural diagram of data integration

Archived

The diagram describes the system interactions, as follows:

- Event Integration Facility (EIF) interacts with EIF Probe which is connected to Tivoli Netcool/OMNIbus, to send monitoring situation details.
- Tivoli Netcool/Impact interacts with Tivoli Netcool/OMNIbus to read Object Server `alerts.status` table and to update this table with data processed by Tivoli Netcool/Impact policies.
- Tivoli Service Request Manager Gateway interacts with Tivoli Service Request Manager to create Incident Ticket and to get the ticket number.
- Tivoli Netcool/Impact interacts with CCMDB to create Request for Change and to get the RFC number. This interaction is also used for event enrichment as event enrichment mapping table resides in CCMDB.
- Tivoli Netcool/Impact interacts with Tivoli Provisioning Manager to run workflow and get the workflow status after its execution.
- Situation Update Forwarder (SUF) interacts with Tivoli Monitoring/Tivoli Composite Application Manager to update situation status.

16.5 Implementation steps

In this section, we discuss the steps required to implement this scenario.

16.5.1 Probe for Tivoli EIF

The *Tivoli Netcool/OMNIbus Probe Tivoli EIF Linux English* product enables IBM Tivoli Netcool/OMNIbus to receive alerts from other products that use EIF. In this integration, we use IBM Tivoli Monitoring and IBM Tivoli Composite Application Manager to send monitoring events.

Requirements

The following items must be installed:

- Object Server
- The probe-nonnative-base-3_0 package
- The Java 1.5 (supplied with Tivoli Netcool/OMNIbus) package

Installation

In this scenario, we use IBM Tivoli Netcool/OMNIbus V7.2.1. The instructions in this section for installing the IBM Tivoli Netcool/OMNIbus Probe for Tivoli EIF apply to this version.

Starting with Tivoli Netcool/OMNIbus V7.3, IBM Tivoli Netcool/OMNIbus Probe for Tivoli EIF is installed with the same installer as the core Tivoli Netcool/OMNIbus product. See “IBM Tivoli Netcool EIF Probe 9.0” on page 306 for more information.

Archived

Perform the following steps to install the probe:

1. Install the patches for non-native probe and EIF probe. For each patch, run the patch utility script:

   ```
   $OMNIHOME/install/nco_patch -install <patchname>
   ```

   In the script, <patchname> specifies the name of the patch you downloaded.

2. Copy the properties file:

   ```
   cp <install_dir>/tivoli_eif.props $OMNIHOME/etc/PROBE_EIF.props
   ```

3. Edit the RulesFile and PortNumber as shown in Example 16-1

   Example 16-1 PROBE_EIF.props example file

   ```
   RulesFile : '$OMNIHOME/probes/linux2x86/tivoli_eif.rules'
   PortNumber : 9998
   ```

4. Run the SQL provided by EIF probe to update the alert.status table with new fields needed for the data coming from IBM Tivoli Monitoring:

   ```
   $OMNIHOME/bin/nco_sql -user <username> -password <password> -server <obj_serv>
   < $OMNIHOME/probes/linux2x86/tec_db_update.sql
   ```

   The command has the following definitions:

   username Specifies the name of user authorized to run Tivoli Netcool/OMNIbus

   password Specifies the password associated with this user

   obj_serv Specifies the name of ObjectServer instance to which the gateway will connect

   We use the following command:

   ```
   $OMNIHOME/bin/nco_sql -user netcool -password netcool -server NCOMS <
   $OMNIHOME/probes/linux2x86/tec_db_update.sql
   ```

   Note: In this scenario, we use Process Agent. Because probe calls the correct version of Java when started through Process Agent, we add JAVA_HOME to the nco_p_tivoli_eif file and change the **exec** command line as follows:

   ```
   JAVA_HOME=/opt/ibm/netcool/platform/linux2x86/jre_1.5.6/jre
   exec $OMNIHOME/probes/$ARCH/nco_p_nonnative $JAVA_HOME/bin/java
   $NCO_JPROBE_JAVA_FLAGS -cp $CLASSPATH $NCO_JPROBE_JAVA_XFLAGS
   -DOMNIHOME="$OMNIHOME" $PROGRAM "$@"
   ```

For more information, see *IBM Tivoli Netcool/OMNIbus Probe for Tivoli EIF Reference Guide*, SC23-6072.

16.5.2 Tivoli Event Synchronization

The IBM Tivoli Monitoring V6.2.2 Tools, Multiplatform, English product is an IBM Tivoli Monitoring component that is used to send Tivoli Netcool/OMNIbus event updates back to Tivoli Enterprise Monitoring Server.

Installation

Installation of the Tivoli Event Synchronization is as follows:

1. Run the `ESync2200<operating_system>.bin` file in the `tec` subfolder.

   We used the following command:

   `./ESync2200Linux.bin`

   The welcome window opens as shown in Figure 16-2.

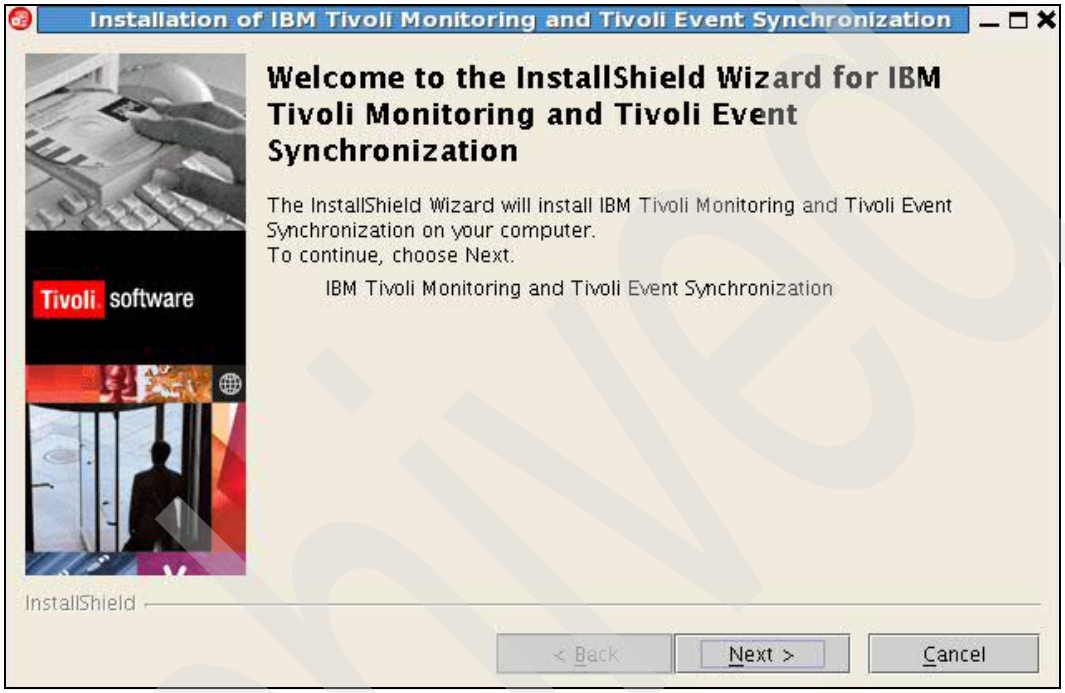


Figure 16-2 Welcome window

2. Accept the terms of license agreement as shown in Figure 16-3 on page 421.

Archive

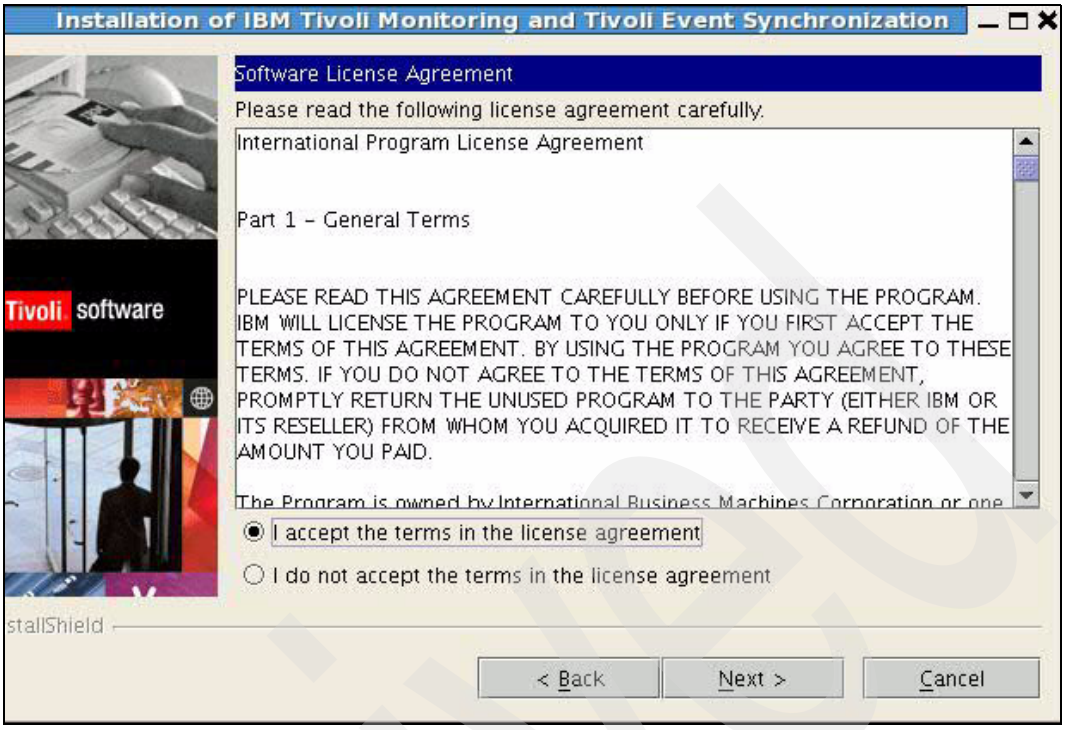


Figure 16-3 License Agreement

3. Specify the installation directory as shown in Figure 16-4.

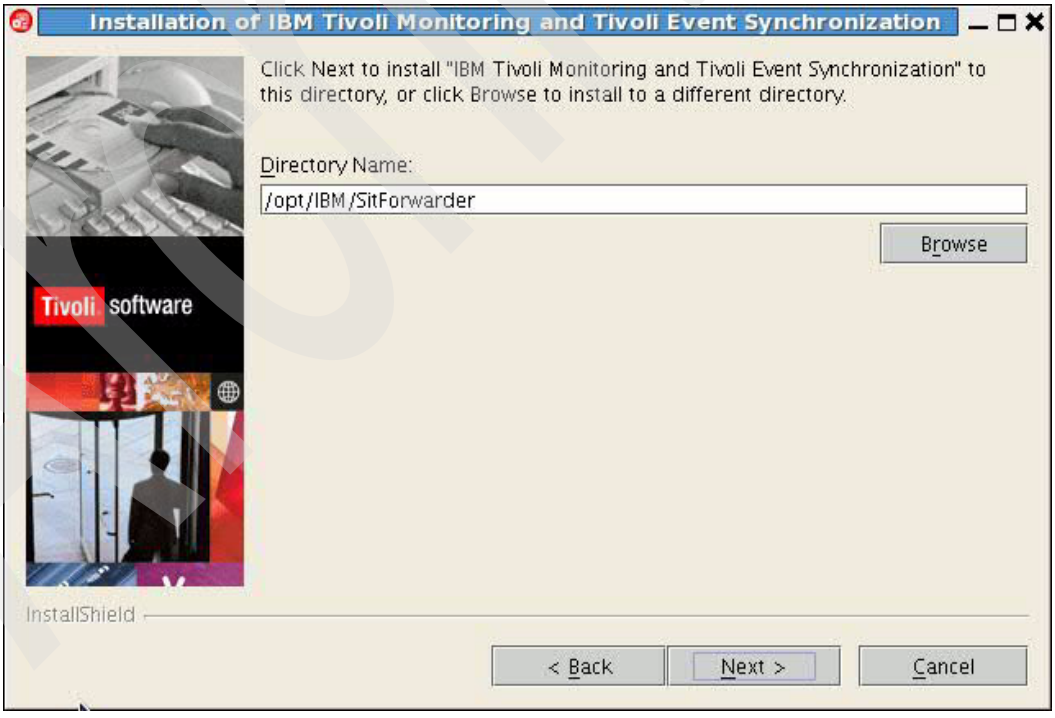


Figure 16-4 Directory Name

4. Set the configuration values, as shown in Figure 16-5 on page 422 and in Figure 16-6 on page 422.

Tip: The debug level can be set to **med** or **verbose** to increase log details.

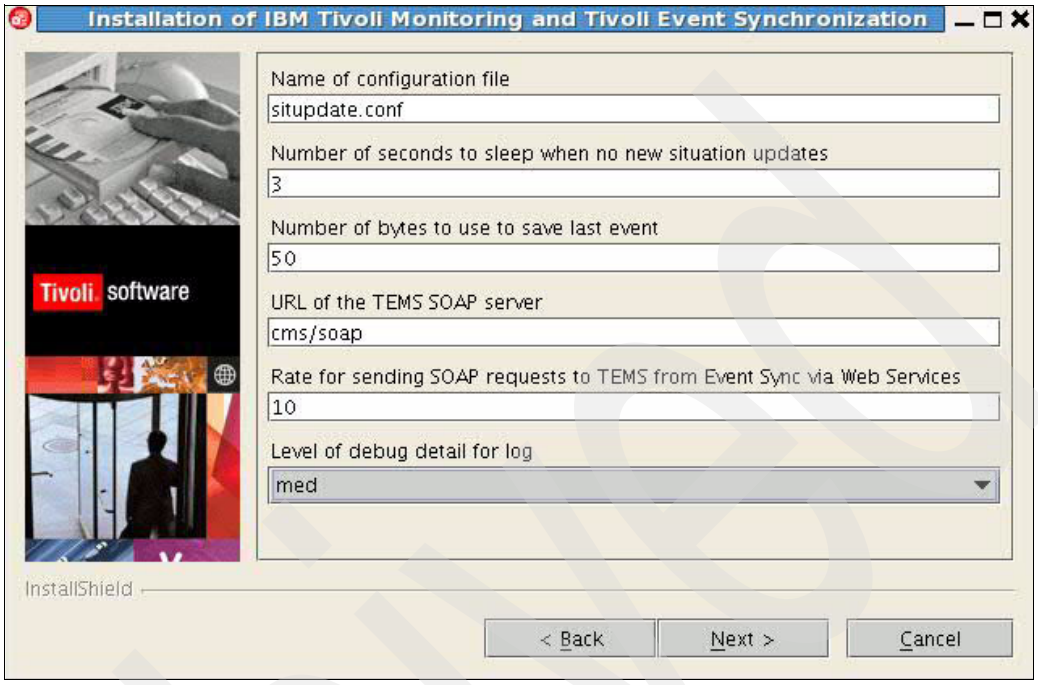


Figure 16-5 Configuration 1

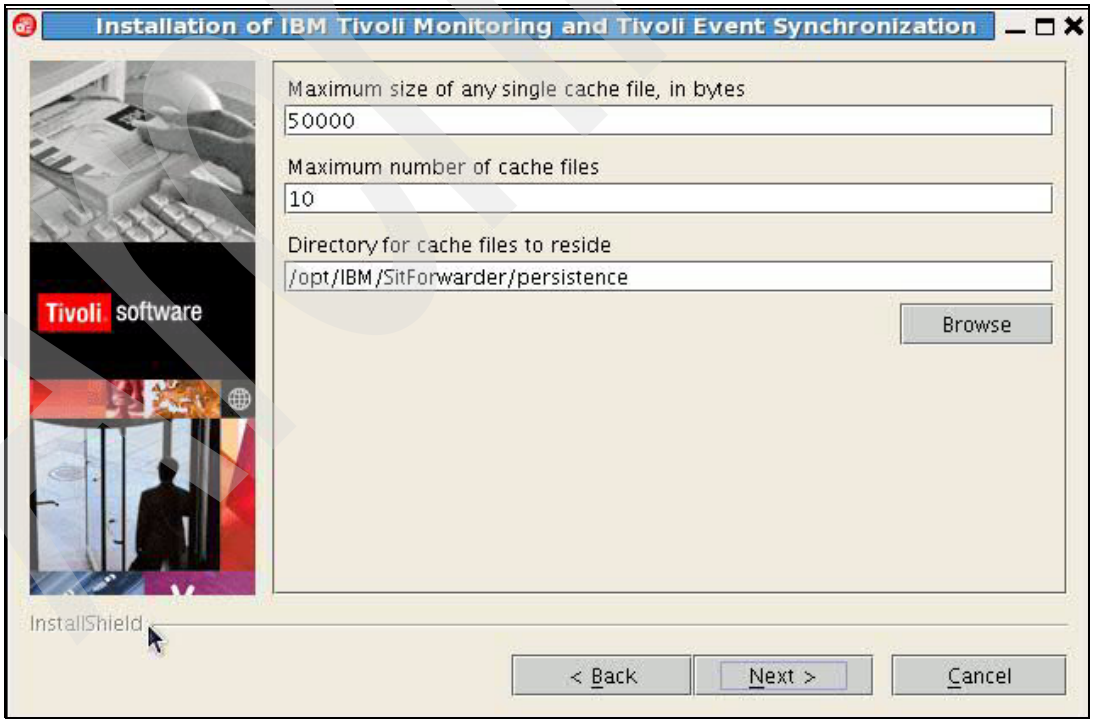


Figure 16-6 Configuration 2

5. Set Tivoli Enterprise Monitoring Server access information as shown in Figure 16-7 on page 423.

Note: Be sure that you use the fully qualified host name that matches the ITMHostname field in the Event List.

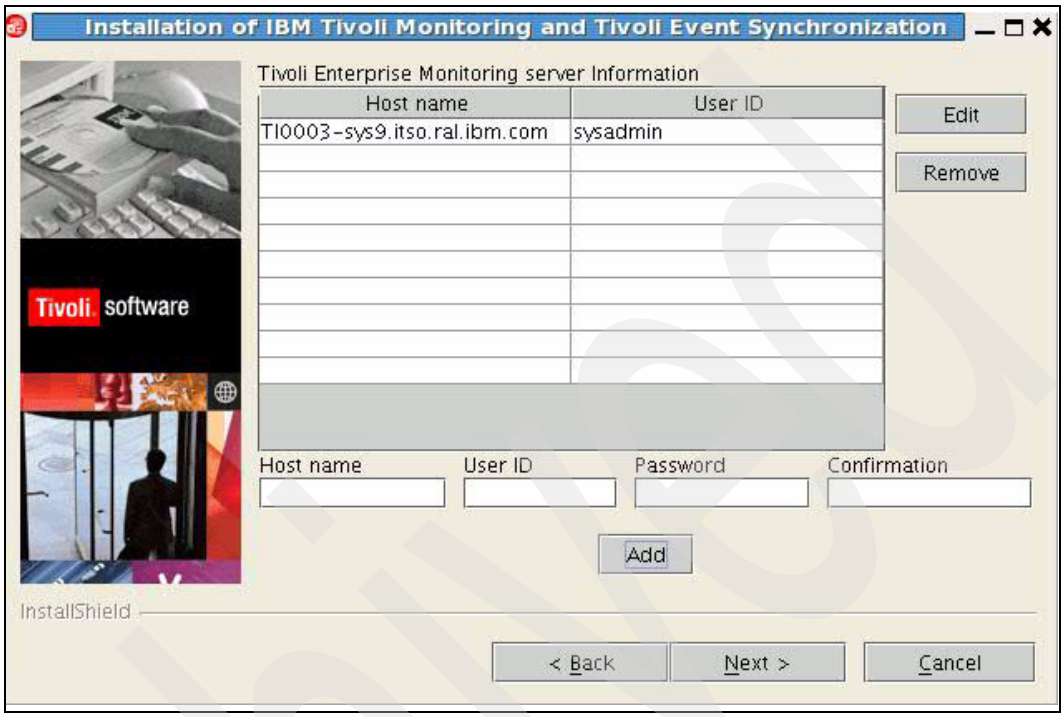


Figure 16-7 Tivoli Enterprise Monitoring Server Configuration

6. Click **Next** to proceed and then click **Finish** to exit the installation wizard.

Configuration

Perform the following steps to configure the Tivoli Event Synchronization:

1. Configure the object server settings in the `$OMNIHOME/etc/NCOMS.props` file as shown in Example 16-2.

Example 16-2 NCOMS.props example for PA credentials

```
PA.Password: 'EHELAGBFBOFPFIGK'
PA.Username: 'root'
```

Notes:

- To run the event synchronization program from SQL automation scripts for sending synchronization events to the monitoring server, the Tivoli Netcool/OMNIbus event server must be running under process control.
- If the password is not blank, it should be encrypted by the following encryption utility and then PA.Password should be set to this encrypted value:.

  $OMNIHOME/nco_pa_crypt

2. Restart the object server:

```
$OMNIHOME/bin/nco_pa_stop -process <obj_serv>
$OMNIHOME/bin/nco_pa_start -process <obj_serv>
```

3. Update the object server database schema by running the following command:

```
$OMNIHOME/bin/nco_sql -U <username> -P <password> -S <obj_serv> <
<path_to_file>/itm_proc.sql
$OMNIHOME/bin/nco_sql -U <username> -P <password> -S <obj_serv> <
<path_to_file>/itm_db_update.sql
$OMNIHOME/bin/nco_sql -U <username> -P <password> -S <obj_serv> <
<path_to_file>/itm_sync.sql
```

4. Update the probe rules by replacing the original rules file with the file that is provided by event synchronization as follows:

```
cp <suf_install_dir>/omnibus/tivoli_eif.rules $OMNIHOME/probes/<arch>
```

   The command has the following definitions:

   suf_install_dir Specifies the folder where Tivoli Event Synchronization is installed.

   arch Specifies the architecture of server where EIF probe is installed.

5. Restart the EIF probe:

```
nco_pa_stop -user netcool -password netcool -process ProbeEIF
nco_pa_start -user netcool -password netcool -process ProbeEIF
```

6. Configure event synchronization to send error events to the object server. Edit the <suf_install_dir>/omnibus/errorevent.conf file as shown in Example 16-3.

   Example 16-3 errorevent.conf file example for EIF probe

```
ServerName=nci51demo.nci.ibm.com
ServerPort=9998
```

   The example uses the following settings:

   ServerName Specifies the name of server where EIF probe is running.

   ServerPort Specifies the port listened by EIF probe.

7. Test the Tivoli Enterprise Monitoring Server connection by running the following command (and see Example 16-4):

```
<suf_install_dir>/bin/test.sh
```

   Example 16-4 Testing connectivity

```
[root@nci51demo bin]# ./test.sh
Successfully connected to Tivoli Enterprise Monitoring Server
ti0003-sys9.itso.ral.ibm.com
```

8. Run Situation Update Forwarder (SUF) by running the following command:

```
<suf_install_dir>/bin/startSUF.sh
```

For more information, see *IBM Tivoli Monitoring Installation and Setup Guide*, GC32-9407-03.

Archived

16.5.3 Event forwarding to Tivoli Netcool/OMNIbus

This step enables IBM Tivoli Monitoring to forward events to Tivoli Netcool/OMNIbus Probe for Tivoli EIF.

The configuration steps are as follows:

1. Configure Hub TEMS by running the following command:

   ```
   $CANDLEHOME/bin/itmcmd config -S -t <tems_name>
   ```

   In the command, `tems_name` is the name of monitoring server

2. Enter the EIF server host name and port during configuration as shown in Example 16-5.

   Example 16-5 Event Integration Facility settings

   ```
   Tivoli Event Integration Facility? [1=YES, 2=NO] (Default is: 1):
   EIF Server Host Name or type 0 for "none" :(Default is: 0): nci51demo.nci.ibm.com
   EIF Port? (Default is: 0): 9998
   ```

3. Restart Hub TEMS:

   ```
   $CANDLEHOME/bin/itmcmd server stop <tems_name>
   $CANDLEHOME/bin/itmcmd server start <tems_name>
   ```

16.5.4 Creating the mapping table and populating the data in CCMDB

In this step, we explain how to create the mapping table and populate the data in CCMDB.

The mapping table contains data for classification. Through this table, incident ticket and request for change is mapped to the correct item such as classification and owner group.

Other CCMDB data such as Change Window Calender, Configuration Item, and Response Plan have an important role for the scenario because they provide required information to other components.

Creating Change Window Calendars

We start by creating Change Window Calendars. These calenders specify maintenance windows that we check before incident ticket creation.

Perform the following steps:

1. With administrative permissions, log on to the CCMDB server.
2. In the Start Center window, click **Go To** → **Change** → **Change Window Calendars**.
3. Create a new Change Window Calendar:
   a. Enter a Change Window Calendar Name, for example: `WINDOW1`
   b. Enter description, for example: `Change Window 1`
   c. Enter Start Date, for example 4/1/10
   d. Enter End Date, for example 4/1/11
   e. Save Calendar.
4. Click **Select Action** → **Schedule Change Windows**.
5. Change Duration to 8:00.
6. Click the first **option** button. Enter `Every 1 day(s)` at `00:00` time.

Archived

7. Click **Preview**. The Date Preview list (Figure 16-8) shows date information, similar to the following values:

```
4/2/10 00:00:00
4/3/20 00:00:00
```

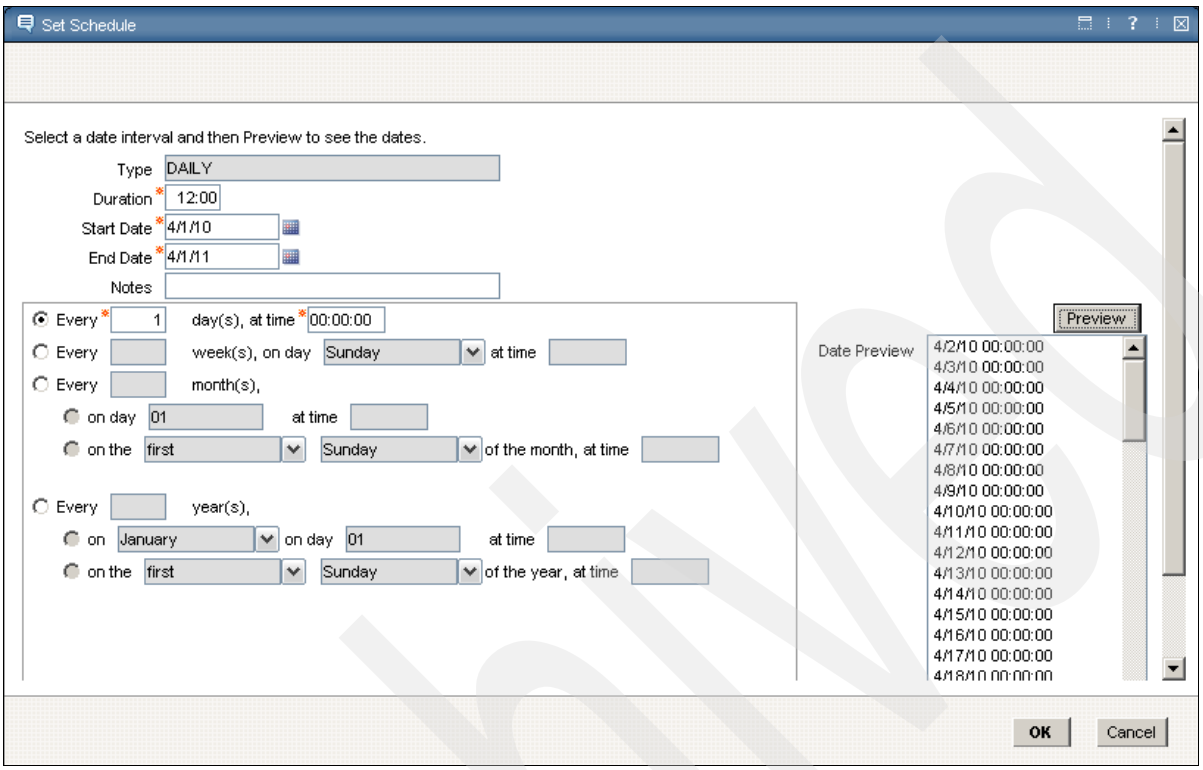


Figure 16-8 Set Schedule Window1

8. Click **OK** to close the window.
9. Click **Save** again to save the changed window. See Figure 16-9.

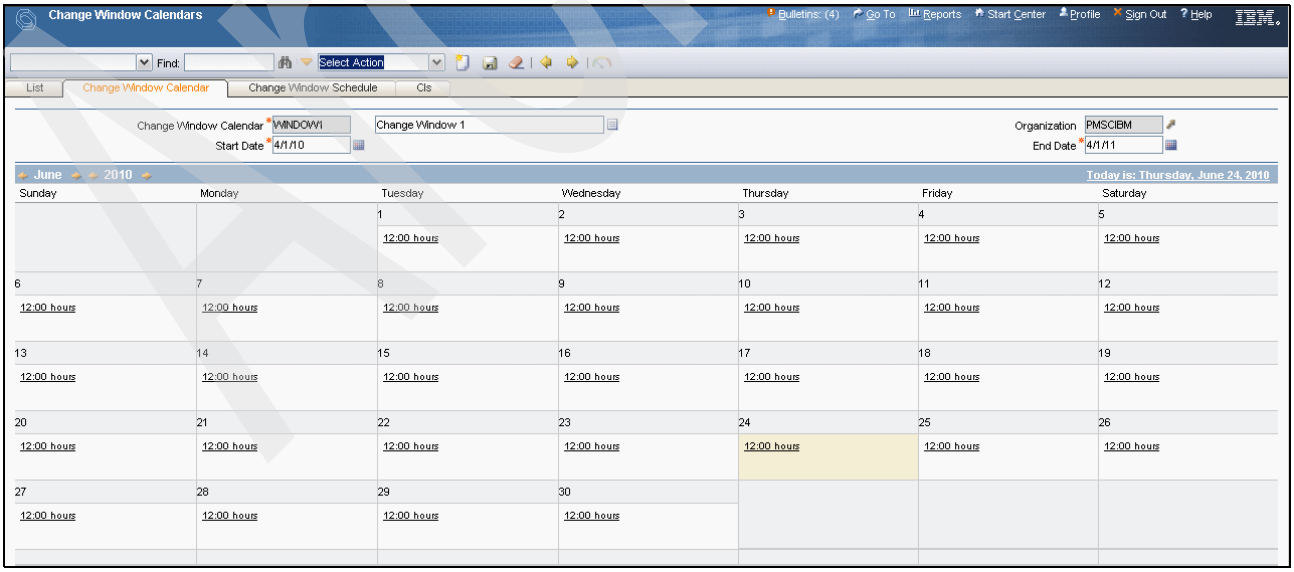


Figure 16-9 Change Window Calendars

10. Create another new Change Window Calendar.
    a. Enter Change Window Calendar Name, for example: `WINDOW2`
    b. Enter description, for example: `Change Window 2`
    c. Enter Start Date, for example `4/1/10`
    d. Enter End Date, for example `4/1/11`
    e. Save the Calendar.
11. Click **Select Action** → **Schedule Change Window**s.
12. Change Duration to `8:00`.
13. Click the first **option** button. Enter `Every 1 day(s)` at time `12:00`.
14. Click **Preview**. The Date Preview list shows information similar to the following values:
    – `4/2/10 12:00:00`
    – `4/3/20 12:00:00`
15. Click **OK** to close dialog box.
16. Click **Save** again to save change window.

Creating Configuration Items

Perform the following steps to create the Configuration Items:

1. Log on to the CCMDB server with administrative permission.
2. In the Start Center window, click **Go To** → **IT Infrastructure** → **Configuration Items**.
3. Click **New CI** in the toolbar to create a new configuration item.
4. Set Configuration Item Number field to `TI003-SYS3`.
5. Set Configuration Item Name field to `TI0003-SYS3`.
6. Set Classification field to `CIROOT \ CI.COMPUTERSYSTEM \ CI.WINDOWSCOMPUTERSYSTEM`.
7. In the Specifications section, set the Alphanumeric Value field for `COMPUTERSYSTEM_NAME` to `TI0003-SYS3`.
8. Set Service field to `SHIPPING`.
9. Set CI Location field to `FACILITY`.
10. Set Change Window to `WINDOW1`.
11. Set CI Owner to a valid person, for example `MKIPEL`.
12. Save the Configuration Item by clicking **Save**. See Figure 16-10 on page 428.

Archived

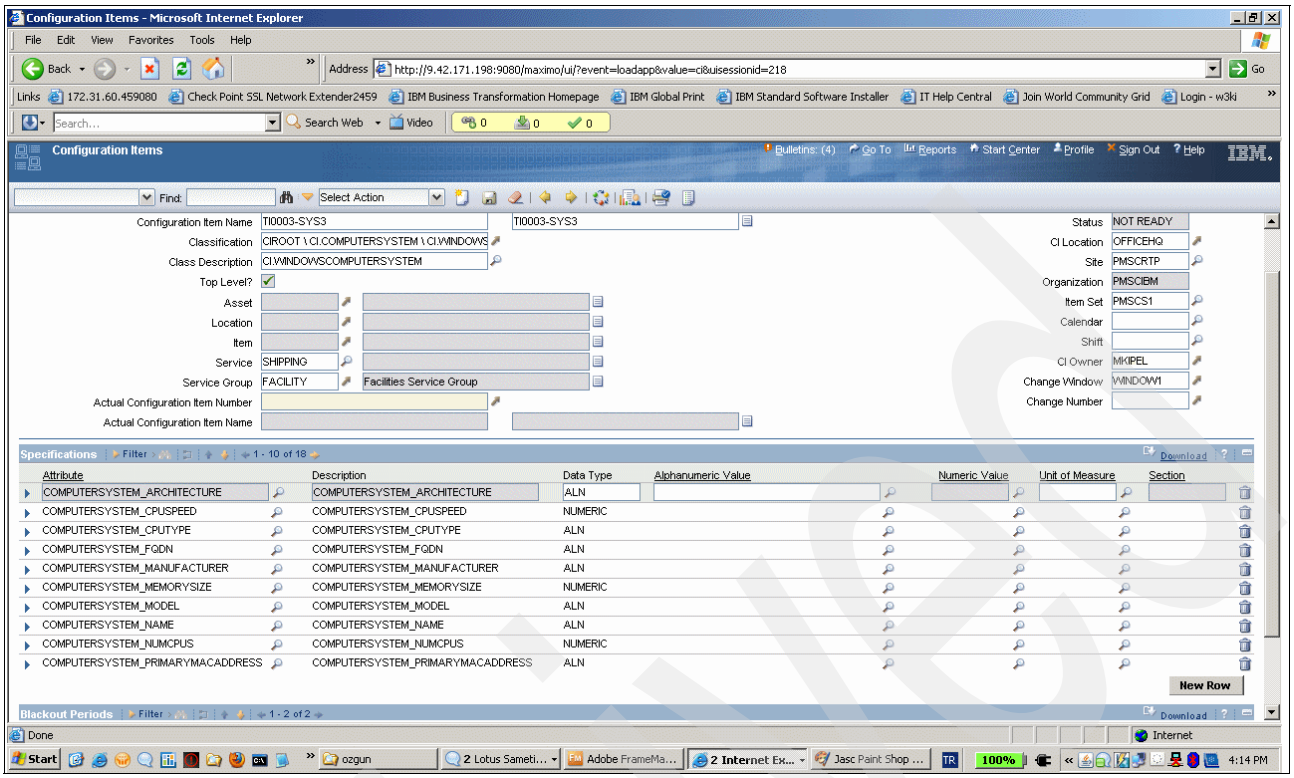


Figure 16-10 Create Configuration Item

13. Repeat steps 3 on page 427 - 12 on page 427 to enter additional Configuration Items, listed in Table 16-2.

Table 16-2 Configuration Item list

Configuration Item number and name	Classification	Service	CI location	CI Owner	Change Window	COMPUTERSYSTEM_NAME
TI003-SYS3	CIROOT \ CI.COMPUTERSYSTEM \ CI.WINDOWSCOMPUTERSYSTEM	SHIPPING	OFFICEHQ	BENJAMINW	WINDOW1	TI003-SYS3
TI003-SYS14	CIROOT \ CI.COMPUTERSYSTEM \ CI.WINDOWSCOMPUTERSYSTEM	SUPPLYCH	OFFICEHQ	FINLEY	WINDOW2	TI003-SYS14
was60	CIROOT \ CI.COMPUTERSYSTEM \ CI.LINUXCOMPUTERSYSTEM	DISTRIBY	WEST	AMAN	WINDOW1	was60

Creating the Impact table

Perform the following steps to create the Impact table:

1. With administrative permission, log on to the CCMDB server.
2. In the **Start Center** window, click **Go To** → **System Configuration** → **Platform Configuration** → **Database Configuration**.
3. Create a new object:
   a. Click **New Object**, on the toolbar.
   b. Set Object field to `IMPACTMAP`.
   c. Set Object description to `Impact Map`.
   d. Check Main Object check box.
   e. Click **Save Object,** on the toolbar. See Figure 16-11.

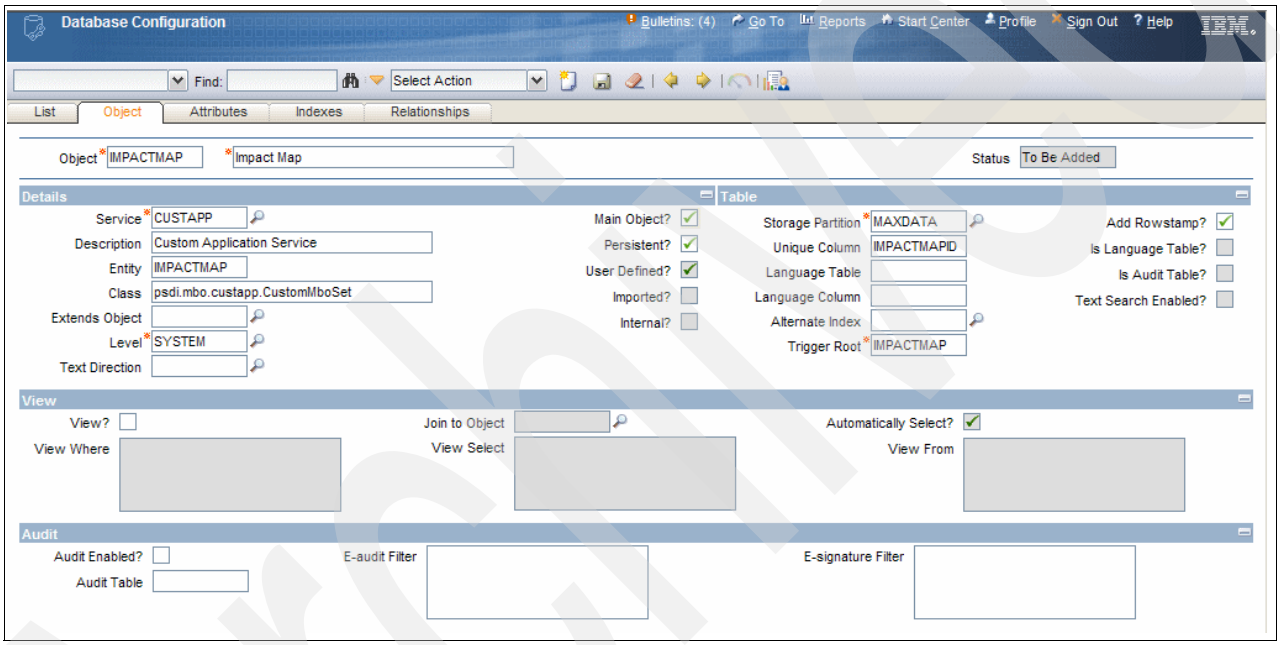


Figure 16-11 Create IMPACTMAP Object

4. Click the **Attributes** tab. See Figure 16-12 on page 430.
5. Click **Mark Row for Delete** icon on the `DESCRIPTION` row.
6. Check that the status of the DESCRIPTION row indicates `DELETE`.
7. Click **Save Object,** on the toolbar.
8. Click **Mark Row for Delete** icon on the DESCRIPTION_LONGDESCRIPTION row.
9. Check that the status of the DESCRIPTION_LONGDESCRIPTION row indicates `DELETE`.
10. Click **Save Object**, on the toolbar.
11. Click **New Row**.
12. Set Attribute field to `Changeclassification`.
13. Click **Save Object**, on the toolbar.

Archived

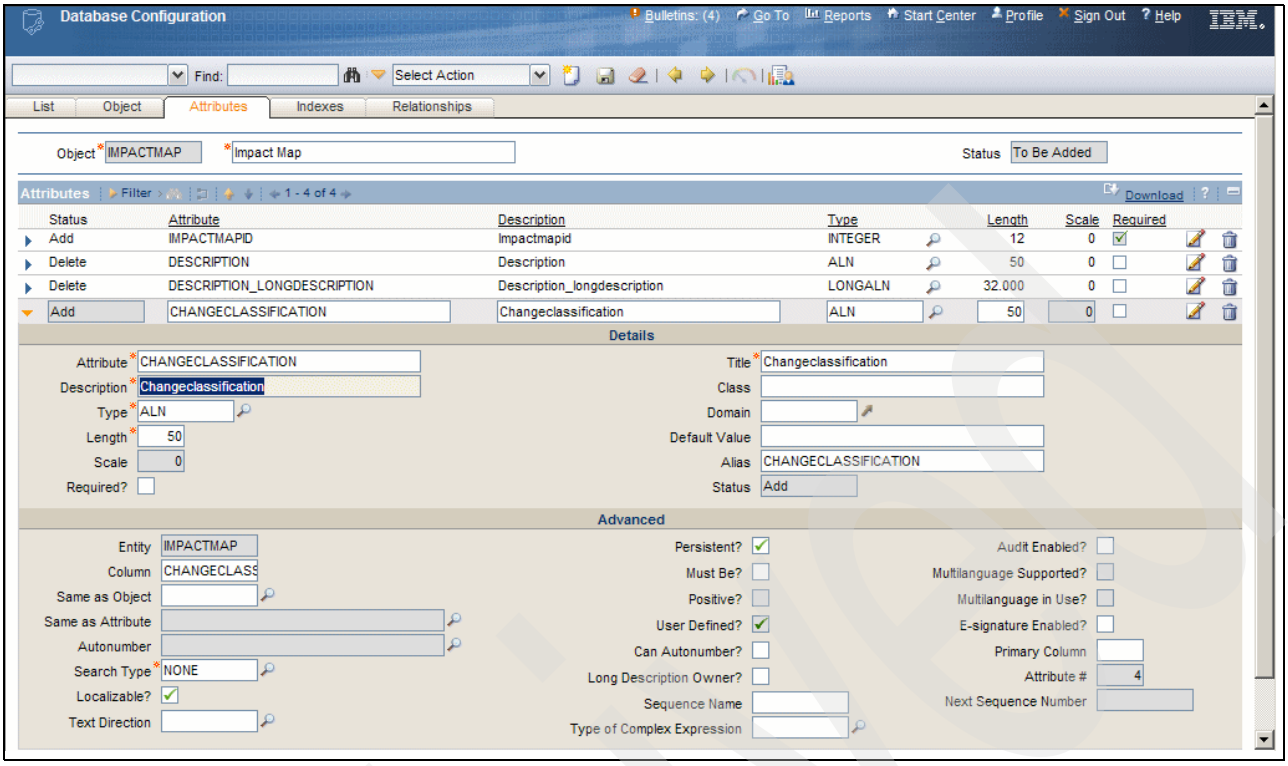


Figure 16-12 ChangeClassification Attribute

14. Click **New Row**.

Archive

15. Set Attribute field to `incidentclassification`. See Figure 16-13.
16. Click **Save Object**, on the toolbar.

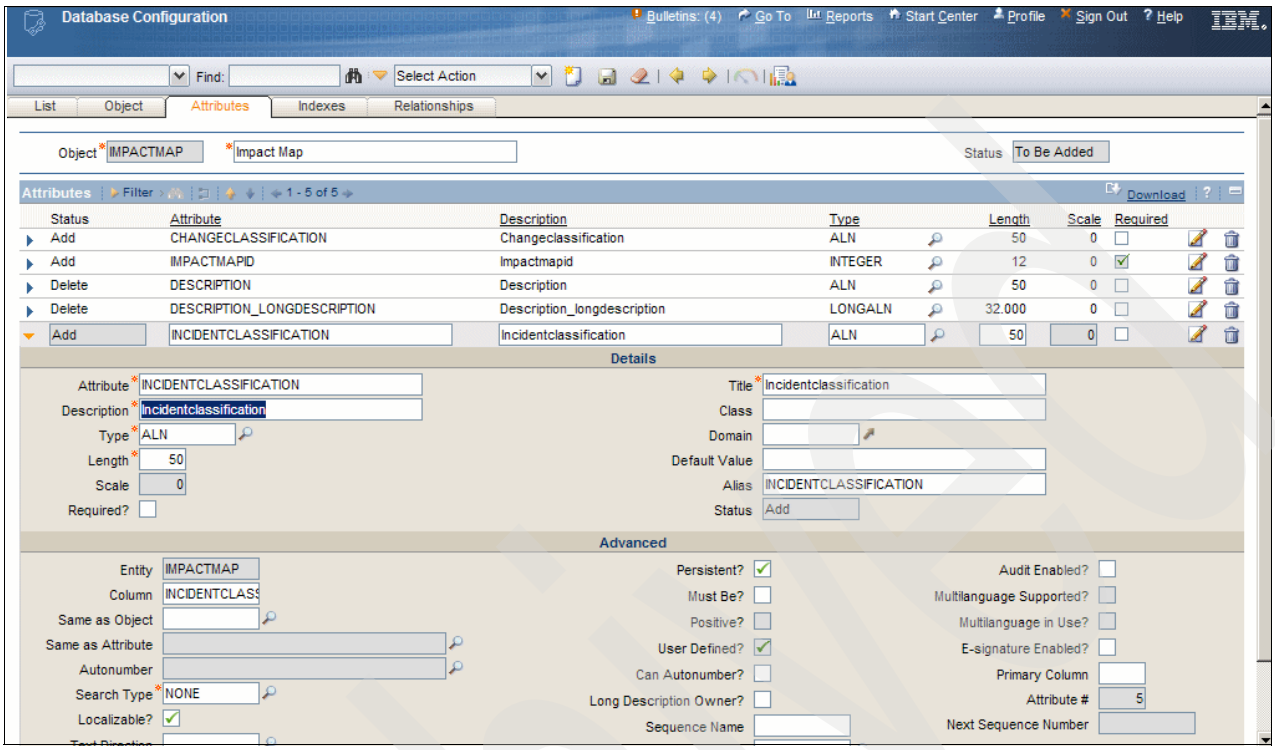


Figure 16-13 incidentClassification Attribute

17. Click **New Row**.

Archive

18. Set Attribute field to `impact`. See Figure 16-14.
19. Set Same as Object field to `TICKET`.
20. Set Same as Field field to `IMPACT`.
21. Click **Save Object**, on the toolbar.

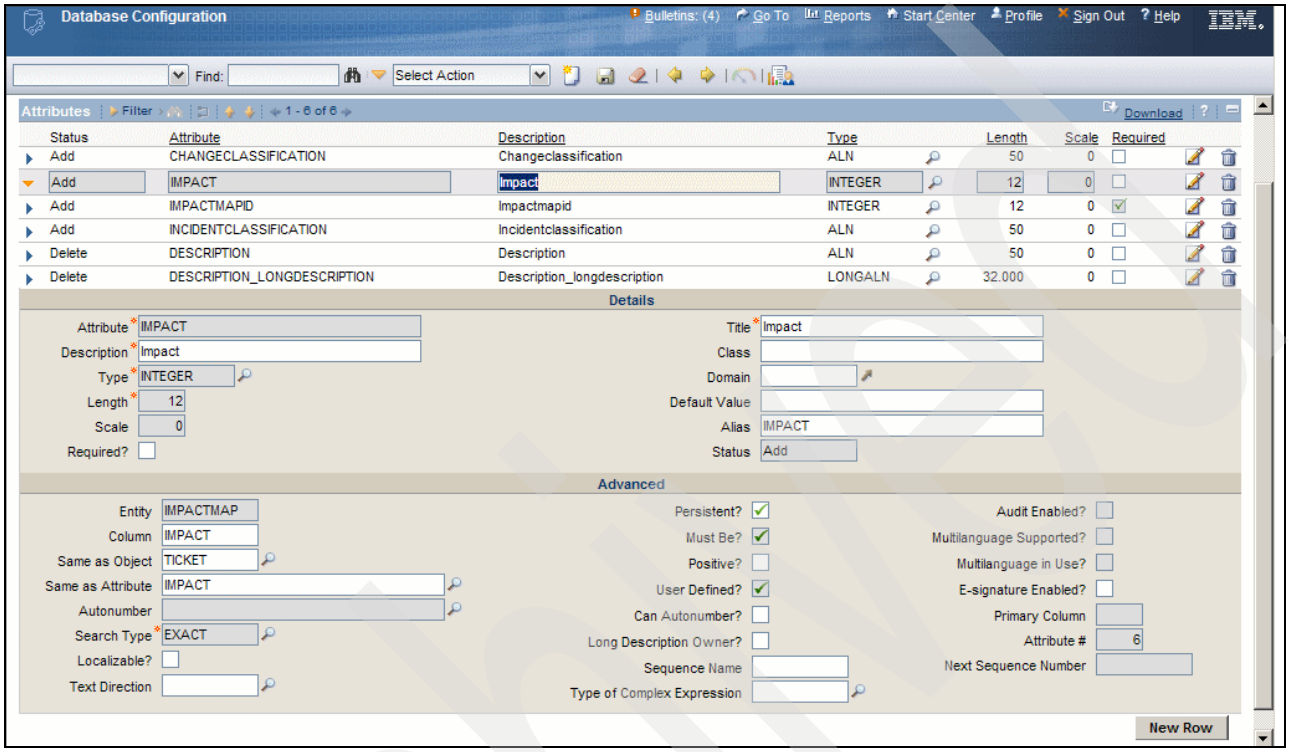


Figure 16-14 Impact Attribute

22. Click **New Row**.

Archive

23. Set Attribute field to `urgency`. See Figure 16-15.
24. Press **Tab** key.
25. Set Same as Object field to `TICKET`.
26. Set Same as Field field to `URGENCY`.
27. Click **Save Object**, on the toolbar.

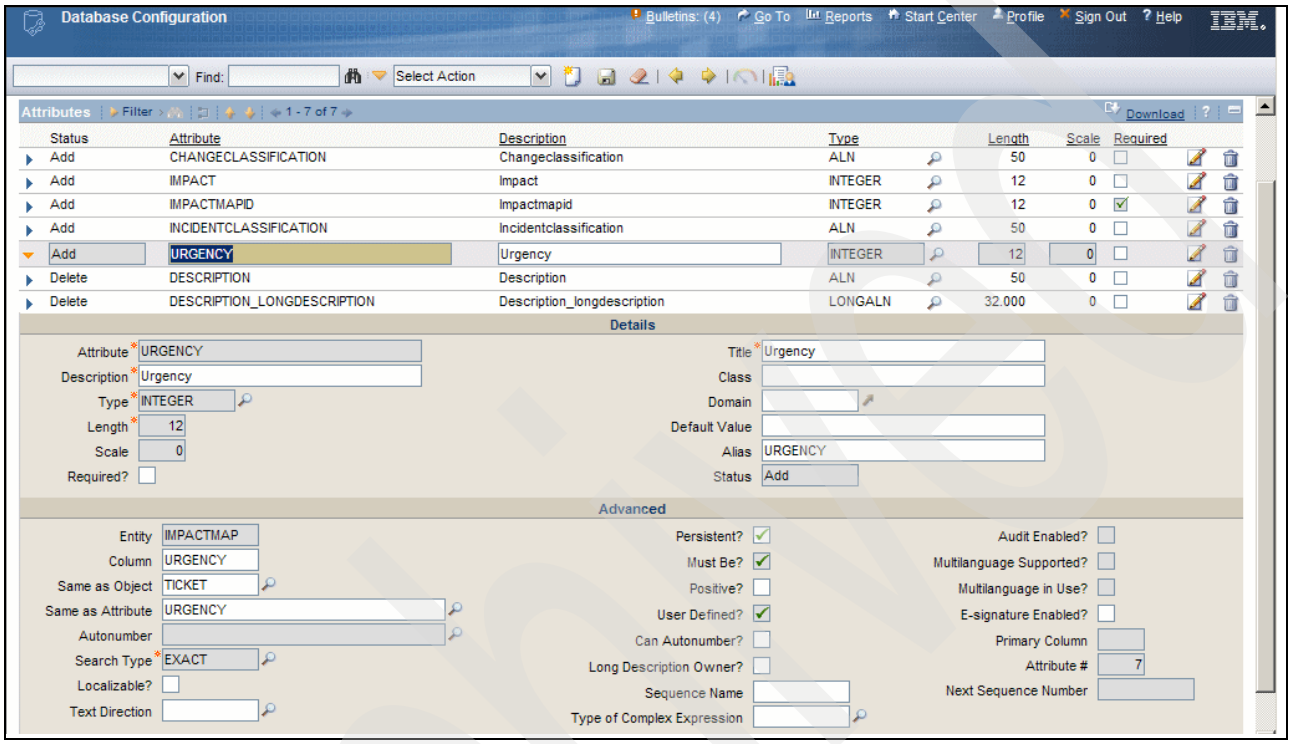


Figure 16-15 Urgency Attribute

28. Click **New Row**.

Archive

29. Set Attribute field to `ownergroup`. See Figure 16-16.
30. Press **tab.**
31. Set Same as Object field to `PERSONGROUP`.
32. Set Same as Field field to `PERSONGROUP`.
33. Click **Save Object**, on the toolbar.

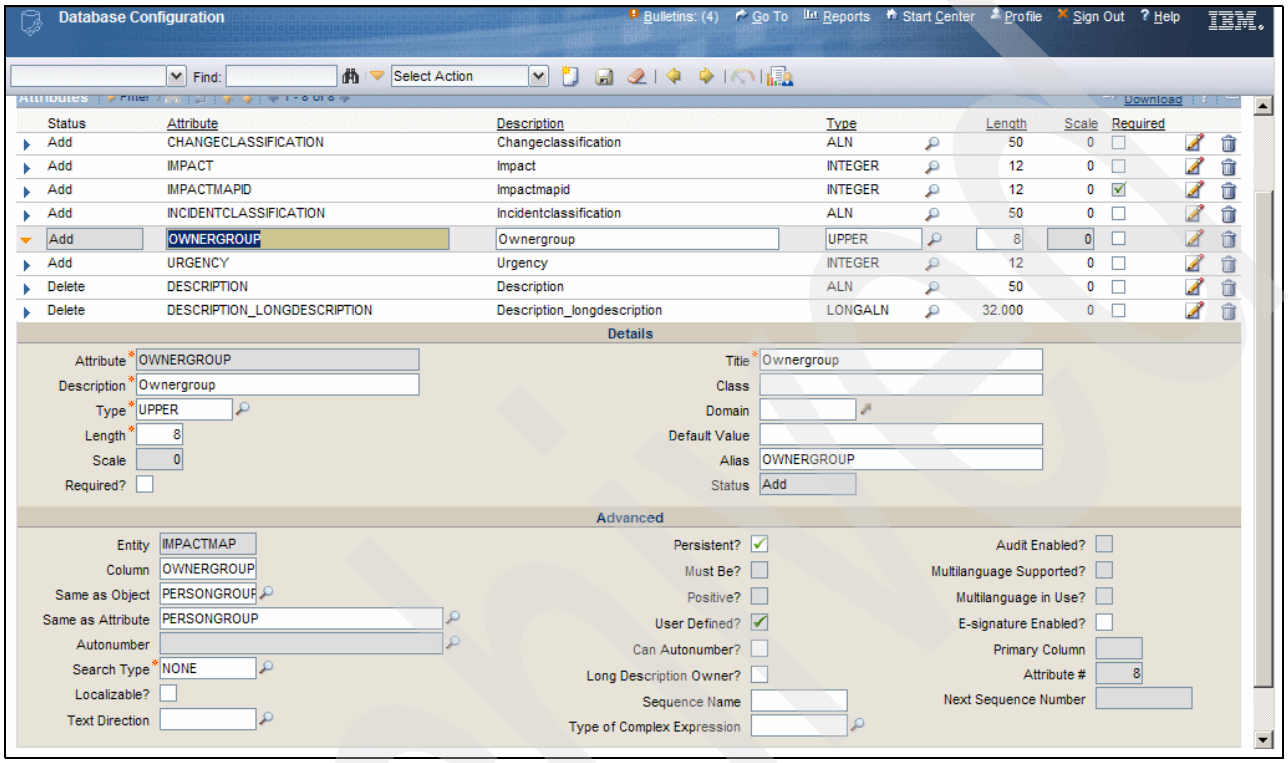


Figure 16-16 Ownergroup Attribute

34. Click **New Row**.

Archive

35. Set Attribute field to `classificationid`. See Figure 16-17.
36. Press **tab.**
37. Set Same as Object field to `PERSONGROUP`.
38. Set Same as Field field to `PERSONGROUP`.
39. Click **Save Object**, on the toolbar.

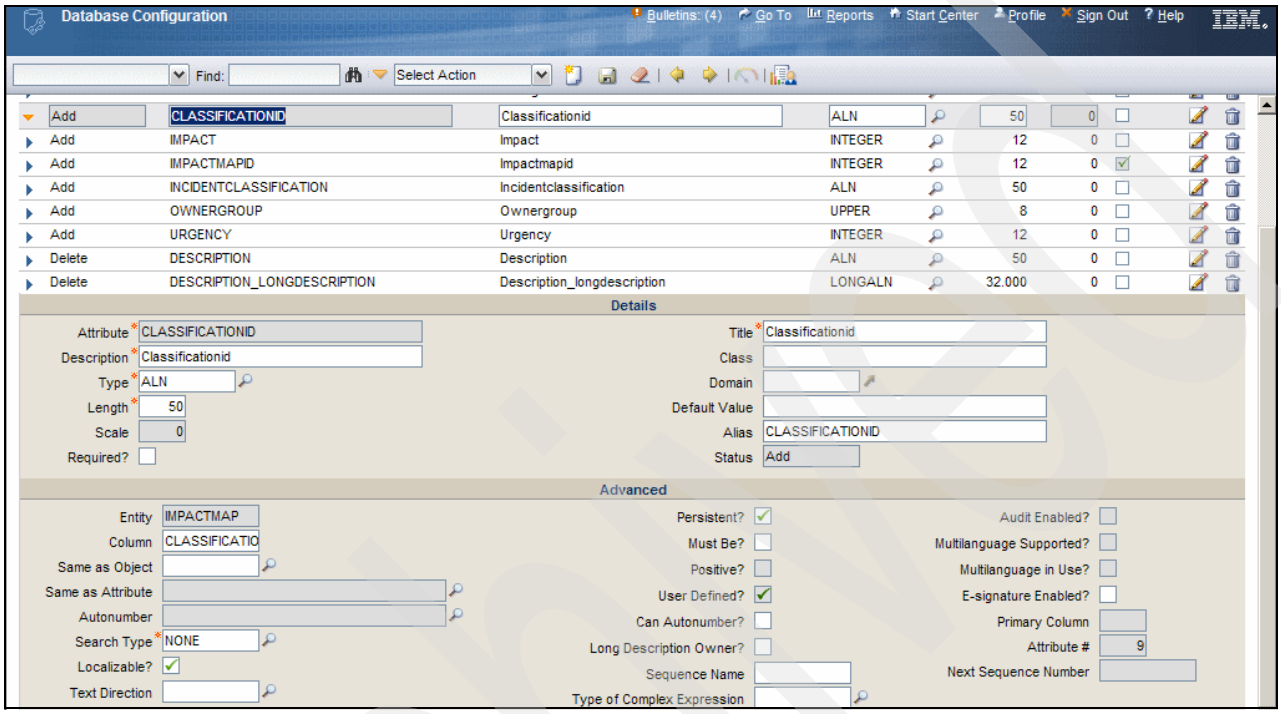


Figure 16-17 Classificationid Attribute

40. Click **List** tab.
41. Click **Select Action** → **Manage Admin Mode.**
42. Click **Turn Admin Mode On**.
43. In the Electronic Signature Authentication window, enter password of the currently logged on administrative user.
44. Enter a reason for change, for example: `Impact Table Creation`.
45. Click **OK**.
46. Click **OK** again to close system message dialog box.

47. Wait for a couple of minutes, and then click **Refresh Status**. See Figure 16-18.
48. When you see the message `Successfully Set Admin Mode ON`, click **Close**.

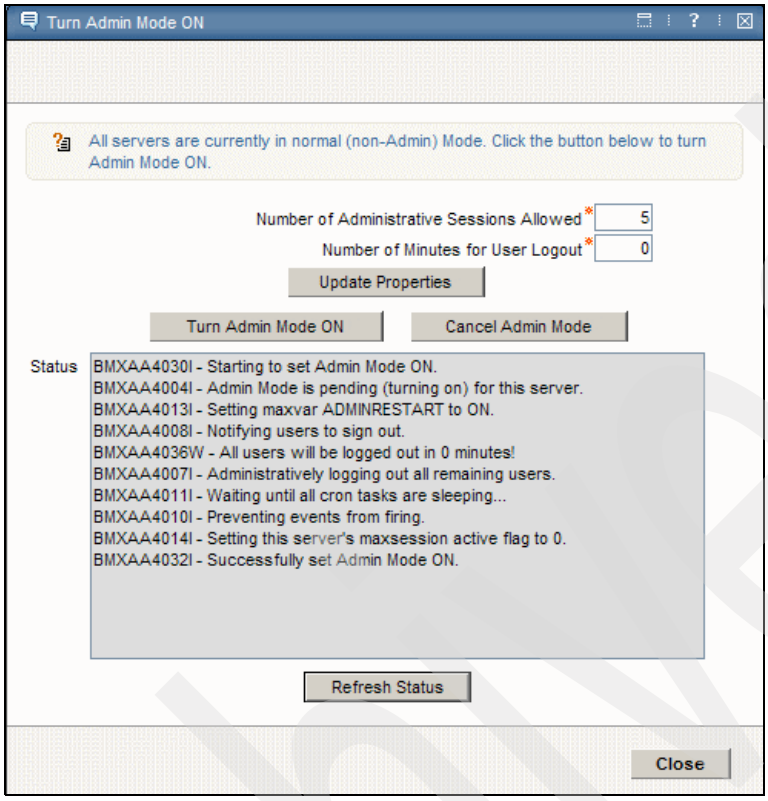


Figure 16-18 Turning Admin Mode ON

49. Click **Select Action** → **Apply Configuration Changes**.
50. In the Structural Database Configuration dialog box, select the **Do you have a Current Backup** check box.
51. Click **Start Configuring the Database**.
52. In the Electronic Signature Authentication dialog box, enter the password of the currently logged on administrative user.
53. Enter a reason for change, for example: `Impact Table Creation`.
54. Click **OK**.
55. Click **OK** again to close system message dialog box.
56. Wait for a couple of minutes, then click **Refresh Status**.
57. Click **OK**.
58. Click **Select Action** → **Manage Admin Mode**
59. Click **Turn Admin Mode OFF**.
60. Wait for a couple of minutes, and then click **Refresh Status**.

61. Click **Close** to close dialog box. See Figure 16-19.

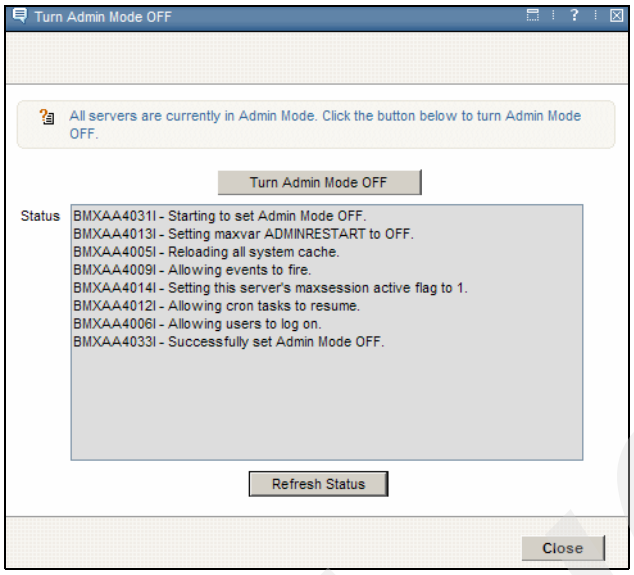


Figure 16-19 Turning Admin Mode off

Creating the Impact Application

Perform the following steps to create the Impact Application:

1. Log on to the CCMDB server with administrative permission.
2. In the Start Center window, click **Go To** → **System. Configuration** → **Platform Configuration** → **Application Designer.**
3. Click **New Application Definition**, on the toolbar.
4. Set Application field to `IMPACTMAP`.
5. Set Description field to `Impact Map`.
6. Set Main Object field to `IMPACTMAP`
7. Set Key Attribute field to `IMPACTMAPID`
8. Set Module Name to `SD` (Service Desk).
9. Click **Single Page App**. See Figure 16-20.

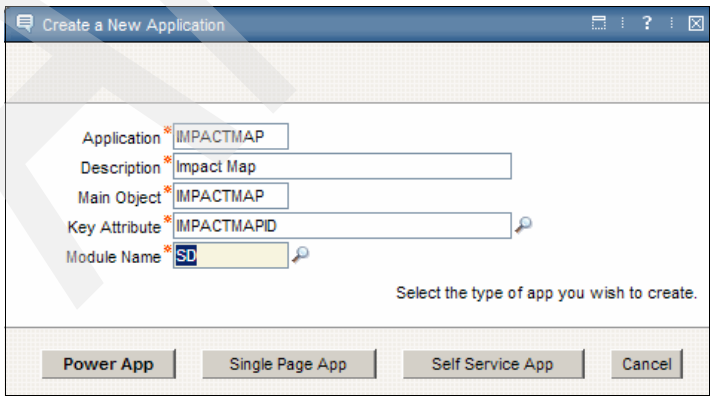


Figure 16-20 Create the Impactmap application

Archived

10. In the application designer application, drag the `Classificationid` field to the `Impact` field. See Figure 16-21.

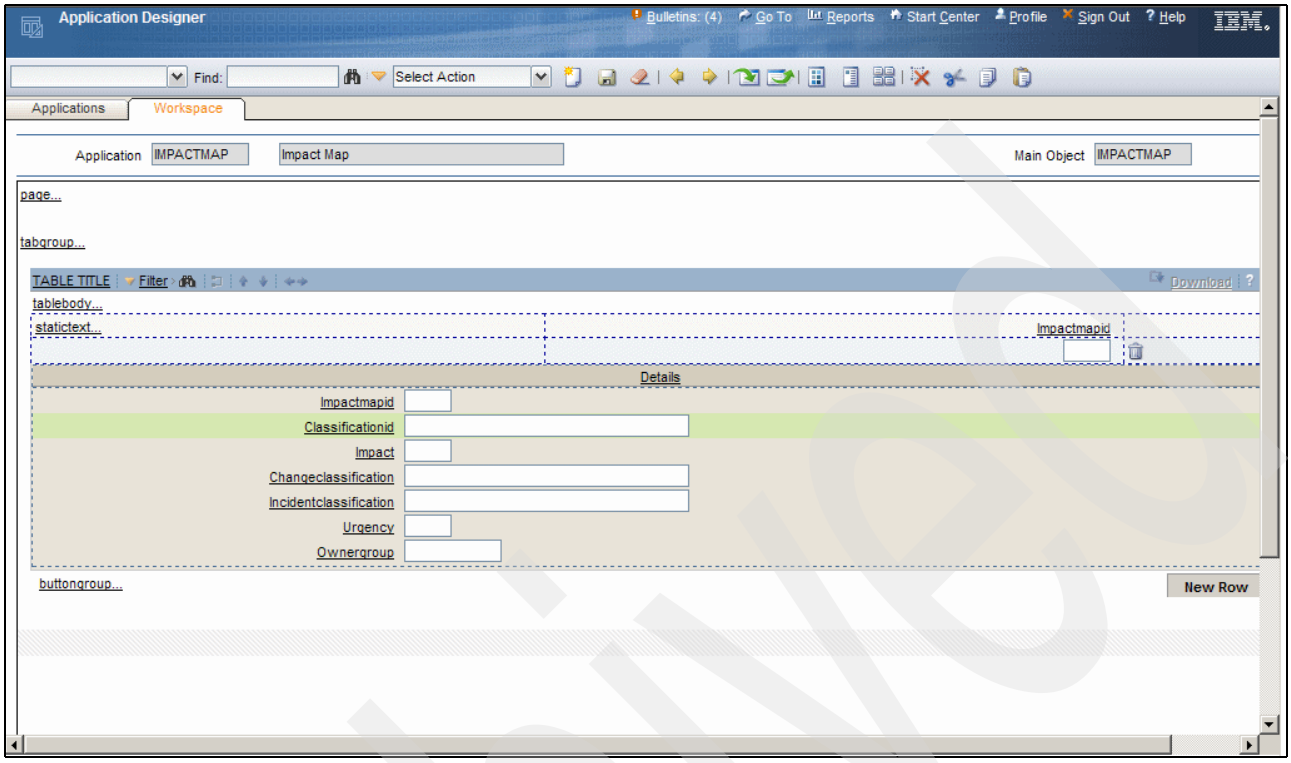


Figure 16-21 Drag and Drop Classification ID

11. Click the **Control Palette**, on the toolbar.

12. Drag the **Table Column icon** from the Control Palette to the first column of table. Perform this step five more times. See Figure 16-22.

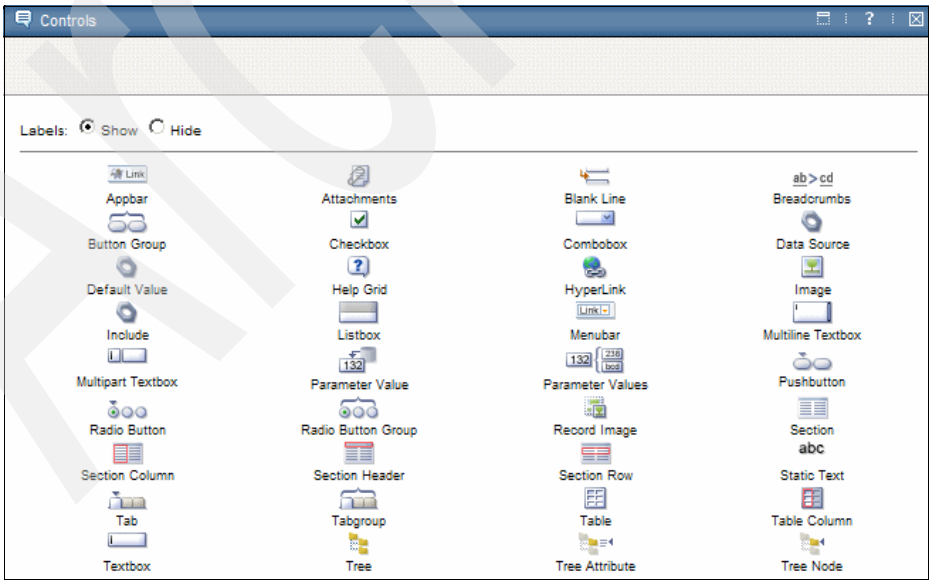


Figure 16-22 Application Designer Control Palette

13. Click the **X** icon on the top right corner of Control Palette to close.

14. Right-click the first table column and click the **Properties** menu item. See Figure 16-23.

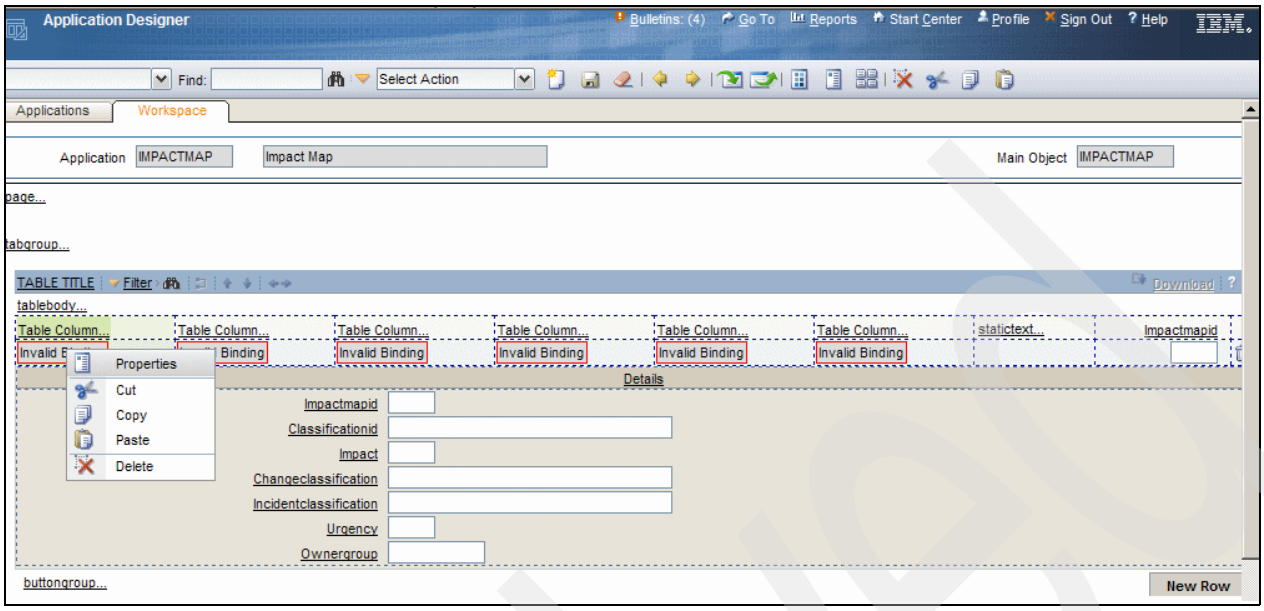


Figure 16-23 Edit Properties Menu

15. In the Table Column Properties dialog box, set the Attribute field to `CLASSIFICATIONID` (press the Tab key to refresh the designer canvas). See Figure 16-24.

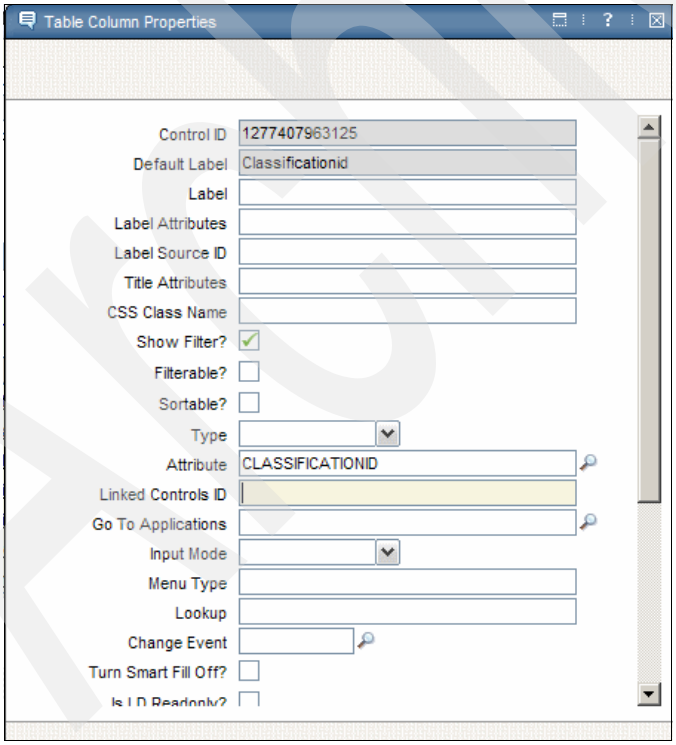


Figure 16-24 Set classificationid attribute.

16. Click the **second** table column.

17. In the Table Column Properties dialog box, set the Attribute field to `CHANGECLASSIFICATION` (press the Tab key to refresh the designer canvas).

18. Click the **third** table column.
19. In the Table Column Properties dialog box, set the Attribute field to `INCIDENTCLASSIFICATION` (press the Tab key to refresh the designer canvas).
20. Click the **fourth** table column.
21. In the Table Column Properties dialog box, set the Attribute field to `IMPACT` (press the Tab key to refresh the designer canvas)
22. Click the **fifth** table column.
23. In the Table Column Properties dialog box, set the Attribute field to `URGENCY` (press the Tab key to refresh the designer canvas)
24. Click the **sixth** table column.
25. In the Table Column Properties dialog box, set the Attribute field to `OWNERGROUP` (press the Tab key to refresh the designer canvas)
26. Click **Save Application Definition**, on the toolbar.

Populating the Impact table

Perform the following steps to populate the Impact table:

1. Log on to the CCMDB server with administrative permission.
2. In the Start Center window, click **Go To** → **Service Desk** → **Impact Map**.
3. Click the **New Row**.
4. Set ClassificationId to `ITM_DB_CONNECTION_POOLS`.
5. Set ChangeClassification to `J2EEMDDB`.
6. Set IncidentClassification to `21020307`.
7. Set Impact to `1`.
8. Set Urgency to `1`.
9. Set Ownergroup to `HARDWARE`.
10. Click **Save IMPACTMAP**, on the toolbar. Figure 16-25.

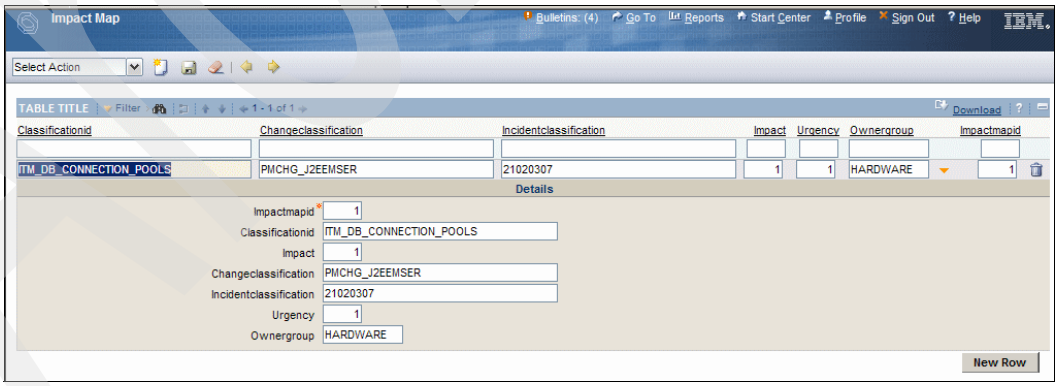


Figure 16-25 IMPACTMAP

11. Repeat the steps 1-10 to enter the values in Table 16-3 on page 441.

Archived

Table 16-3 Impact mapping

ClassificationID	Change Classification	Incident Classification	Impact	Urgency	Ownergroup
ITM_DB_Connection_Pools	J2EEMDDB	21020307	1	1	HARDWARE
ITM_Thread_Pools	J2EEMSRA	21020307	2	2	HARDWARE
ITM_Garbage_Collection_Analysis	J2EEMSRA	21020307	3	2	HARDWARE
ITM_NT_Process	4010304	1010103	2	1	HARDWARE

Column headings are as follows:

ClassificationID Specifies the alerts.status.AlertGroup value, which we map.

ChangeClassification Specifies CLASSSTRUCTURE.CLASSIFICATION.ID for RFC.

IncidentClassification Specifies CLASSANCESTOR.CLASSIFICATIONID for incident ticket.

Impact Specifies impact of incident. Value varies in 1 - 5 range.

Urgency Specifies urgency of incident. Value varies in 1 - 5 range.

Ownergroup Specifies ownergroup of RFC or incident ticket.

Creating the Actions

Perform the following steps to create the Actions. These actions are required to fulfill automatically RFC details.

1. Log on to the CCMDB server with administrative permission.
2. In the Start Center window, click **Go To** → **System Configuration** → **Platform Configuration** → **Actions**.
3. Click **New Action**, on the toolbar.
4. Set Action field to `CHANGEDESCRIPTION1`.
5. Set Object field to `WOCHANGE`.
6. Set Type field to `Set Value`.
7. Set Value field to `Need of more powerful machine as it is under powered DB server`.
8. Set Parameter/Attribute field to `description`.
9. Save the action, as shown in Figure 16-26.

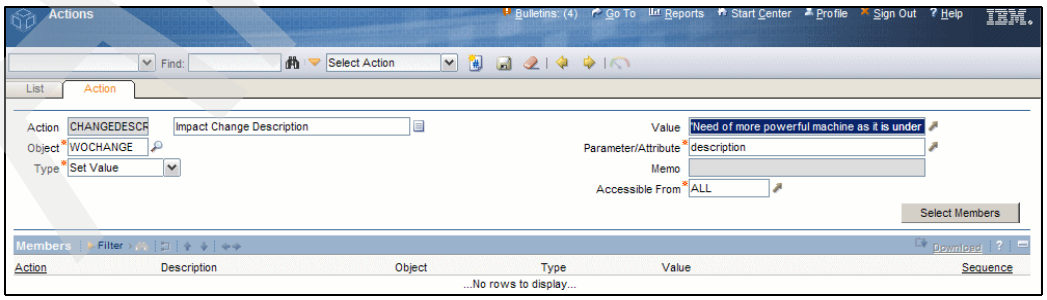


Figure 16-26 New Action

10. Create the actions in Table 16-4.

Table 16-4 Actions

Action	Object	Type	Value	Parameter
CHANGEDESCRIPTION1	WOCHANGE	Set Value	'Need of more powerful machine as it is under powered DB server'	description
CHANGEREASONFORCHANGE1	WOCHANGE	Set Value	'HIGH DB2 UTIL'	reasonforchange
CHANGERISK1	WOCHANGE	Set Value	'No Risk'	risk
CHANGEESTDUR1	WOCHANGE	Set Value	2	estdur
CHANGEJUSTIFYPRIORITY1	WOCHANGE	Set Value	'BUSINESS CRITICAL'	justifypriority
CHANGEVERIFICATION1	WOCHANGE	Set Value	'VERIFIED'	verification
CHANGEWOPRIORITY1	WOCHANGE	Set Value	1	wopriority

11. Add all actions created with type **Set Value** in the previous step to CHANGEGROUP1 as member.

Creating the Escalations

Perform the following steps to create the Escalations, which are required to periodically run specific response plans:

1. Log on to the CCMDB server with administrative permission.
2. In the Start Center window, click **Go To** → **System Configuration** → **Platform Configuration** → **Escalations**
3. Click **New Escalation**, on the toolbar.
4. Set **Applies To** field to `WOCHANGE`
5. Set Schedule field as follows:

   `2m,*,*,*,*,*,*,*,*,*`
6. Click **New Row**, in the *Actions section*.
7. Select the **Apply Response Plan** Action
8. Save the escalation.
9. Click **Select Action** → **Validate**
10. Click **Select Action** → **Activate/Deactivate Escalation**. See Figure 16-27 on page 443.

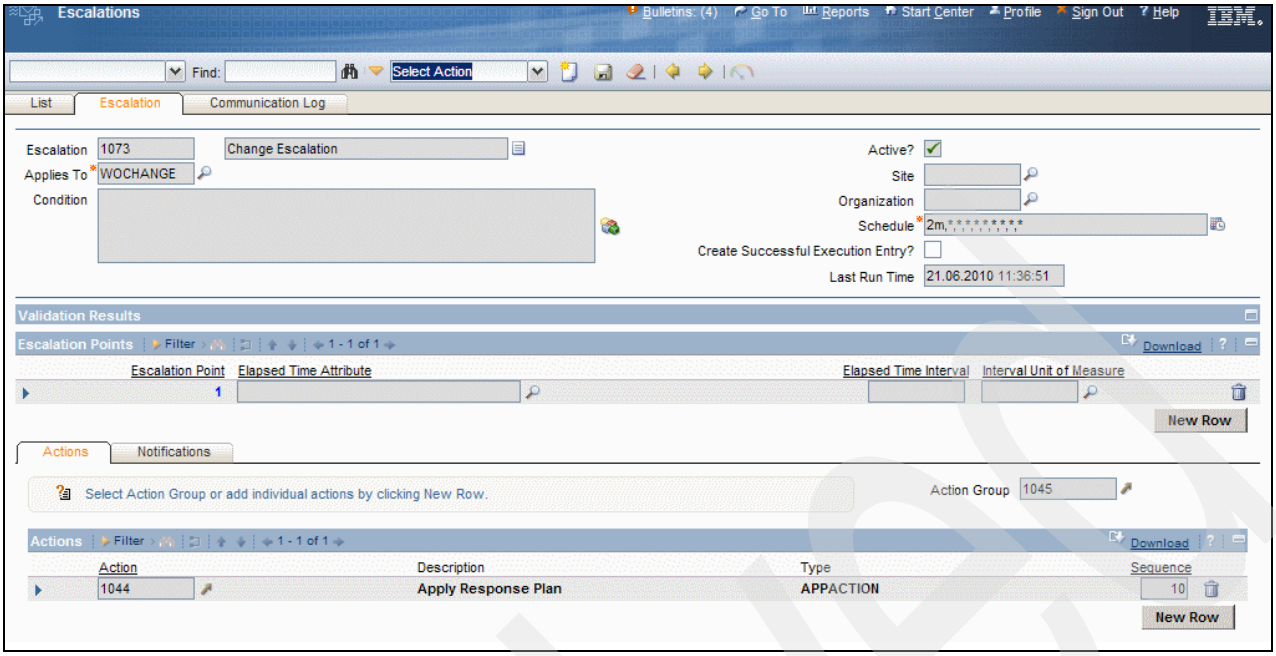


Figure 16-27 Escalations

Creating the Response Plans

Perform the following steps to create the Response Plans, which are containers that contain actions.

1. Log on to the CCMDB server with administrative permission.
2. In the Start Center window, click **Go To** → **Service Level** → **Response Plans**.
3. Click **Create New Response Plan**, on the toolbar.
4. Set Response Plan field to `IMPACTRP1`.
5. Set Ranking field to `10`.
6. Set Description to `Response Plan for IMPACT RFC.`
7. Click the **Conditions** tab.
8. Set Classification field to `PMCHG \ PMCHGSFW \ MDLEWARE \ J2EEMDDB`
9. Click **Response Actions** Tab.
10. Click **New Row** on the Actions section.
11. Enter Action `Change Description 1.`
12. Repeat steps 9-10 for actions in Table 16-4 on page 442.
13. Save Response Plan.
14. Click **Select Action** → **Change Status.**
15. Select Active as New Status. Click **OK**. See Figure 16-28 on page 444.

Active

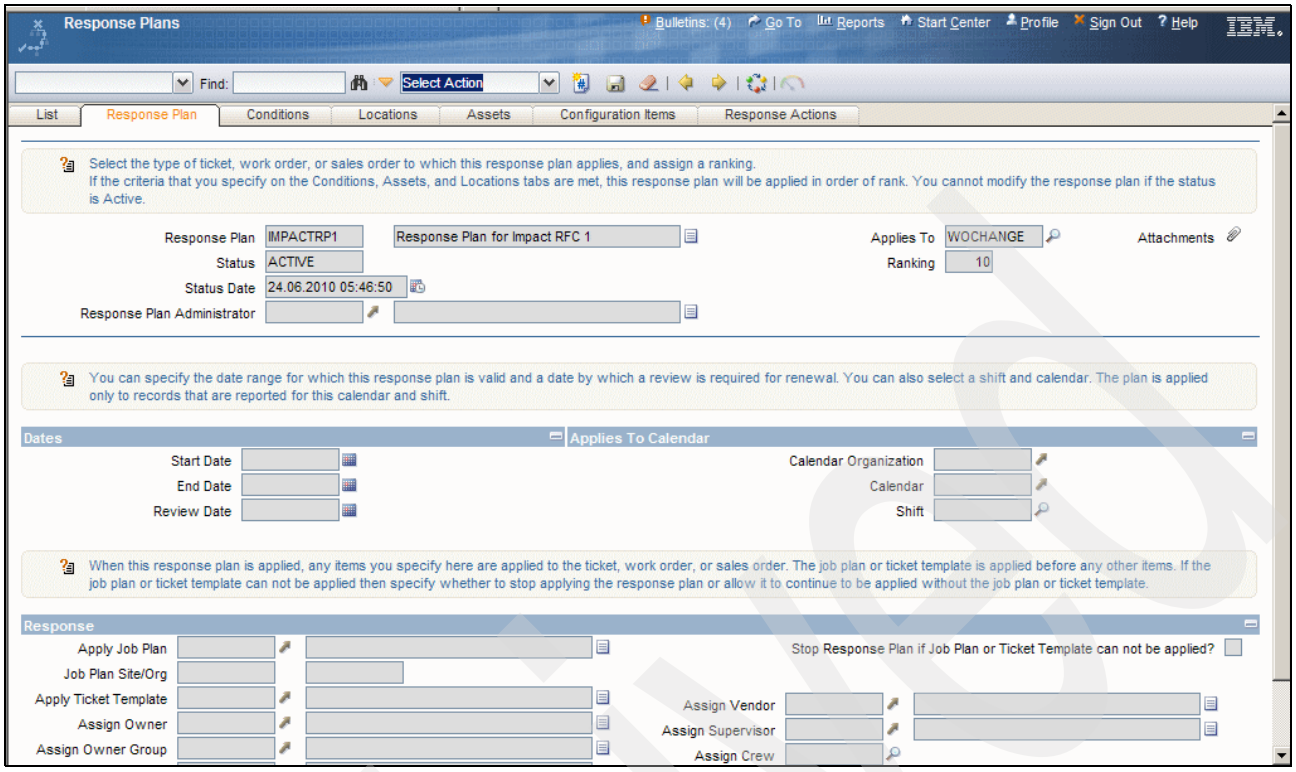


Figure 16-28 Response Plans

Creating the Work Flows

Perform the following steps to create the Work Flows:

1. Log on to the CCMDB server with administrative permission.
2. In the Start Center window, click **Go To** → **System Configuration** → **Platform Configuration** → **Workflow Designer**
3. Click **New Process**, on the toolbar.
4. Set Process field to `CHANGERP`.
5. Set Object Field to `WOCHANGE`.
6. Click the **Connect Nodes** icon.
7. When the mouse pointer turns into a yellow pen icon, click the **Start1** Icon on the workflow canvas.
8. Drag a line and drop on `STOP2` icon.
9. Double Click the line between `START1` and `STOP2` icon.
10. Enter `action name` in the Action field.
11. Click **OK** to close dialog box. SeeFigure 16-29 on page 445.

Archive

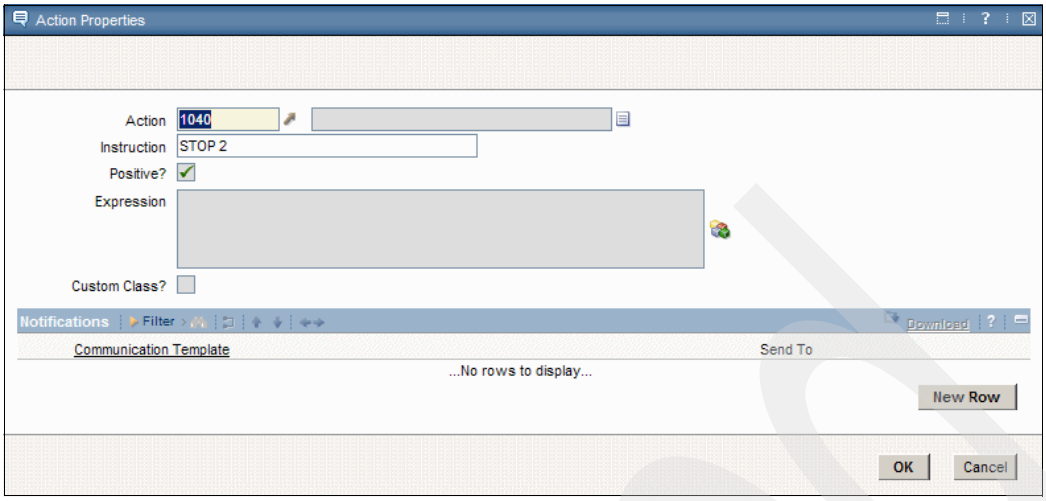


Figure 16-29 Set Workflow Action

12. Click **Select Action** → **Enable Process**
13. Click **Select Action** → **Activate Process.** See Figure 16-30.

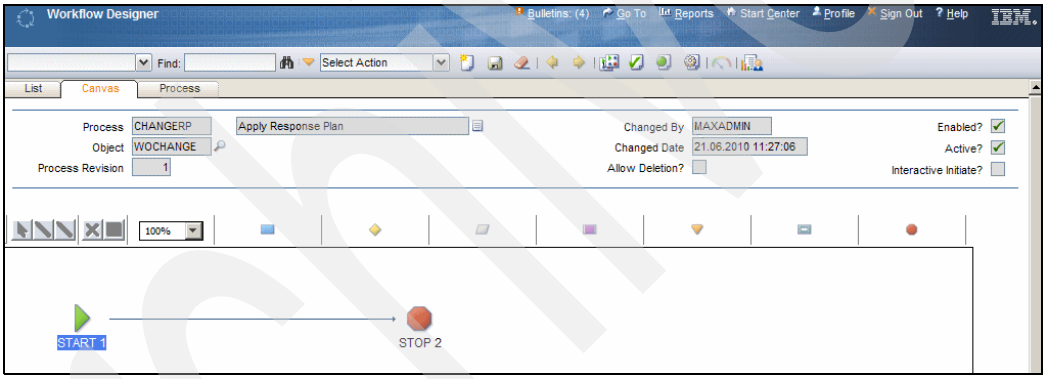


Figure 16-30 CHANGERP Workflow.

16.5.5 Tivoli Netcool/OMNIbus additional fields

For event enrichment, we must add more fields to the `alerts.status` table on Tivoli Netcool/Impact Object Server:

1. Create an SQL file that contains *add* queries, as shown in Example 16-6.

Example 16-6 impactFields.sql file

```
alter table alerts.status add CI VARCHAR(100);
alter table alerts.status add AffectedPerson VARCHAR(50);
alter table alerts.status add AffectedService VARCHAR(50);
alter table alerts.status add AssetLocSiteID VARCHAR(50);
alter table alerts.status add Impact INTEGER;
alter table alerts.status add Urgency INTEGER;
alter table alerts.status add OwnerGroup VARCHAR(50);
alter table alerts.status add HierarchyPath VARCHAR(50);
alter table alerts.status add Action VARCHAR(50);
go
```

Archived

2. Run the SQL file as shown in Example 16-7.

Example 16-7 Running the ImpactFields.sql file

```
[netcool@nci51demo sql]$ nco_sql -user netcool -password netcool <
ImpactFields.sql
(0 rows affected)
```

16.5.6 Configuring Tivoli Service Request Manager to initialize application server

Perform the following steps to configure IBM Tivoli Service Request Manager to receive alerts from Tivoli Netcool/OMNIbus and web services requests from Tivoli Netcool/Impact:

1. Log on to the CCMDB server with administrative permission.
2. In the Start Center window, click **Go To** → **System Configuration** → **Platform Configuration** → **System Properties**
3. Configure Web Application URL and Integration Global Directory to match the address of your IBM Tivoli Service Request Manager application server:
   a. Navigate to `mxe.int.webappurl`.
   b. In the Global Value field, set the application server URL:

      `http://<server_ip>/meaweb`

   c. Select the value next to the **mxe.int.webappurl** check box.
   d. Save the property and click **Live Refresh**.
   e. Navigate to `mxe.int.globaldir`.
   f. In the Global Value field, set the path of global directory to:

      `<websphere_application_installation_path>\profiles\AppSrv01\axis2`

   g. Select the **mxe.int.globaldir** check box value.
   h. Save the property and click **Live Refresh**.
4. Configure Tivoli Service Request Manager to receive journal entries:
   a. Click menu **Go To** → **Integration** → **Object Structures.**
   b. Search for specify Object Structure **MXOSINCIDENT**.
   c. With the selected Object Structure, go to **Action** menu and select **Generate Schema/View XML** option.
   d. Confirm the system message.

16.5.7 Tivoli Netcool/OMNIbus Gateway for Tivoli Service Request Manager

This product, IBM Tivoli Netcool/OMNIbus Gateway for Tivoli Service Request Manager (`nco-g-tsrm-1_1`) for Linux English, integrates Tivoli Netcool/OMNIbus with Tivoli Service Request Manager to open incidents using object server alerts.

Archived

Requirements

The following items must be installed:

- Object Server
- The omnibus-linux2x86-gateway-libngjava-1_0 patch
- The omnibus-linux2x86-gateway-libtal-1_2 patch
- The omnibus-linux2x86-gateway-nco-g-tsrm-1_1 patch

Installation

Perform the following steps:

1. Install the patches by running the following patch utility script for each patch:

   ```
   $OMNIHOME/install/nco_patch -install <patchname>
   ```

   In the utility, `patchname` specifies the name of the patch you downloaded in each case.

2. Install TAL API automations and gateway patches:

   ```
   $OMNIHOME/bin/nco_sql -U <username> -P <password> -S <obj_serv> <
   $OMNIHOME/gates/tal/tal_automations.sql
   $OMNIHOME/bin/nco_sql -U <username> -P <password> -S <obj_serv> <
   $OMNIHOME/gates/tsrm/tsrm.sql
   ```

   We use the following commands:

   ```
   $OMNIHOME/bin/nco_sql -user netcool -password netcool -server NCOMS <
   $OMNIHOME/gates/tal/tal_automations.sql
   $OMNIHOME/bin/nco_sql -user netcool -password netcool -server NCOMS <
   $OMNIHOME/gates/tsrm/tsrm.sql
   ```

3. Copy the properties file (`NCO_GATE.props`) to the `$OMNIHOME/etc` folder:

   ```
   cp <install_dir>/NCO_GATE.props $OMNIHOME/etc/GATE_TSRM.props
   ```

Configuration

Perform the following steps:

1. Edit the `GATE_TSRM.props` file to configure Tivoli Service Request Manager parameters:
   a. Copy `NCO_GATE.props` file to the `$OMNIHOME/etc` folder:

      ```
      cp <install_dir>/NCO_GATE.props $OMNIHOME/etc/GATE_TSRM.props
      ```

   b. Edit the properties file as shown in Example 16-8.

Example 16-8 GATE_TSRM.props

```
Gate.TSRM.UserName      :       'maxadmin'
Gate.TSRM.Password      :       'maxadmin'
Gate.TSRM.PollTime      :       30
Gate.TSRM.CreateTicketTemplate  :       '$OMNIHOME/gates/tsrm/create.xml.template'
Gate.TSRM.UpdateTicketTemplate  :       '$OMNIHOME/gates/tsrm/update.xml.template'
Gate.TSRM.QueryTicketTemplate   :       '$OMNIHOME/gates/tsrm/query.xml.template'
Gate.TSRM.JournalTemplate   :       '$OMNIHOME/gates/tsrm/journal.xml.template'
Gate.TSRM.URLList       :       'http://9.42.171.198/meaweb/os/MXOSINCIDENT'
Gate.TSRM.TicketObjectName    :   'MXOSINCIDENT'
Gate.TSRM.TimeZone :    ''
Name         : 'GATE_TSRM'
Gate.RdrWtr.TblReplicateDefFile : '$OMNIHOME/gates/tsrm/tsrm.rdrwtr.tblrep.def'
Gate.MapFile                            : '$OMNIHOME/gates/tsrm/tsrm.map'
Gate.StartupCmdFile                     : '$OMNIHOME/gates/tsrm/tsrm.startup.cmd'
Gate.TAL.NBScriptFileName               : '$OMNIHOME/gates/tsrm/tsrm.script'
Gate.TAL.TTFilterClause              : 'LogTicket=1'
```

Archived

2. Create a tool named *Create Ticket*. This tool changes the key field LogEvent to 1 (shown in Figure 16-31) so that IBM Tivoli Netcool/OMNIbus Gateway for Tivoli Service Request Manager gateway identifies and sends this alert forward.

   Perform the following steps:

   a. Open `nco_config` with administrative permission.

   b. Access the **Navigator** menu and select the server name.

   c. With administrative permission, connect at NCOMS.

   d. Click **Menu** → **Tools** option.

   e. Create a tool called `CreateTicket`. See Figure 16-31.

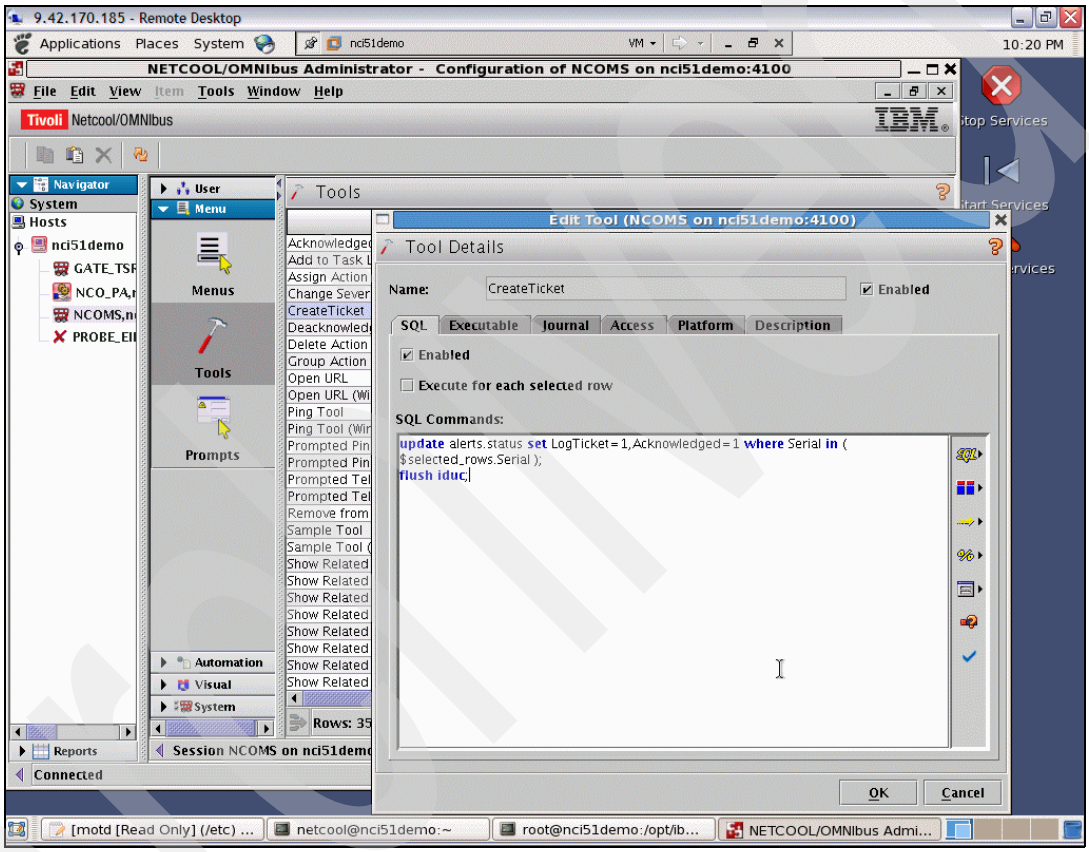


Figure 16-31 Creating a tool in nco_config

   f. Go to **Menu** → **Menus** → **AlertsMenu**.

   g. Create a menu called `CreateTicket` that is associated with the CreateTicket tool.

3. Create an automation using triggers that will close an event when the corresponding incident has been closed:

   a. Open `nco_config` with administrative permission.

   b. Access **Navigator** menu and select NCOMS.

   c. With administrative permission, connect at NCOMS.

   d. Select the **Automation** → **Triggers** option.

   e. Create a timer trigger. See Figure 16-32 on page 449.

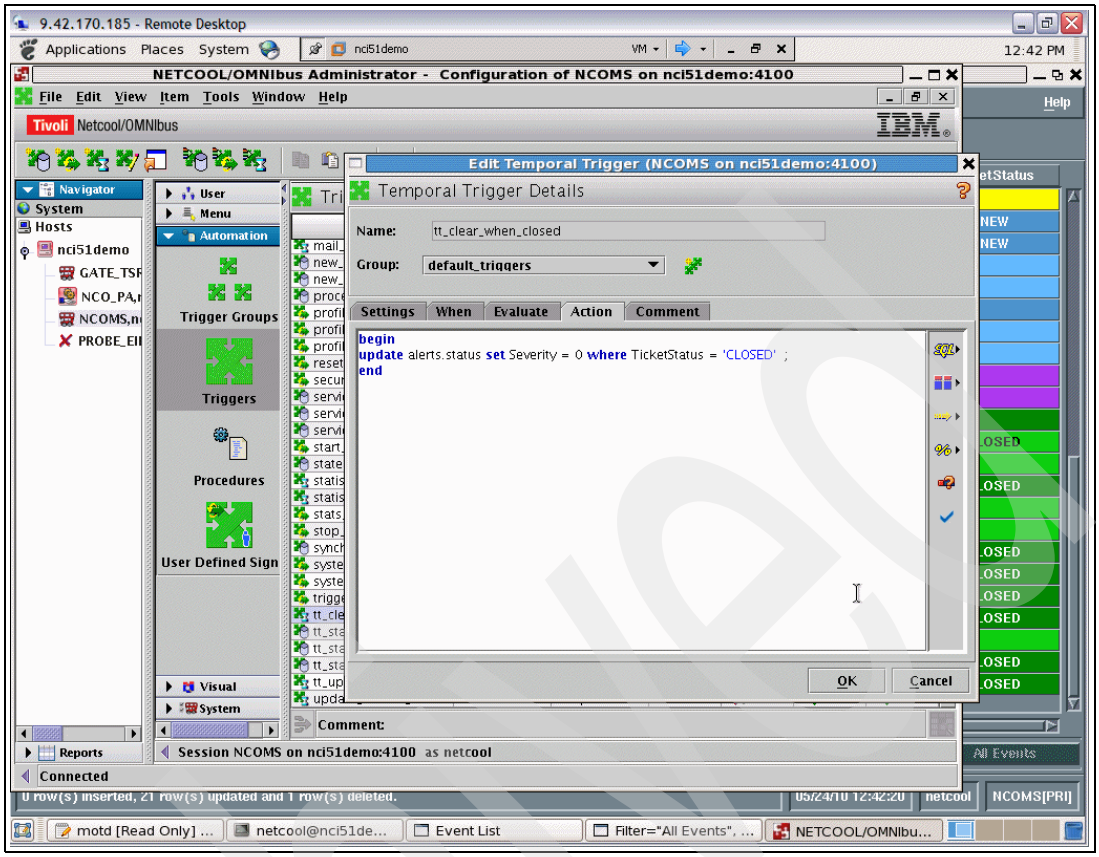


Figure 16-32 Creating a trigger in nco_config

Tip: In the trigger configuration in Figure 16-32 on page 449, shows the following update clause:

"`update alerts.status set Severity = 0 where Severity > 0 and TicketStatus = 'Closed';`"

This clause can also be written as follows, which prevents the update from firing repetitively on the same event:

"`update alerts.status set Severity = 0 where Severity > 0 and TicketStatus = 'Closed';`"

4. Edit the `$OMNIHOME/gates/tsrm/tsrm.map` file to configure field mapping between `alerts.status` table fields and incident object fields. See Example 16-9.

Example 16-9 The tsrm.map example file

```
CREATE  LOOKUP SeverityTable (
        {3 , 3},
        {4 , 2},
        {5 , 1}  )
DEFAULT = 4 ;

CREATE  LOOKUP StatusTable (
        {0 , 'RESOLVED'})
DEFAULT = '' ;

CREATE MAPPING StatusMap
(
      'DESCRIPTION'    =      '@Identifier' ON INSERT ONLY,
```

```
        'DESCRIPTION_LONGDESCRIPTION'       =  '@Summary' ON INSERT ONLY,
        'EXTERNALSYSTEM'       =       'EVENTMANAGEMENT',
        'REPORTEDBY'     =      'NETCOOL AUTOMATION',
        'REPORTDATE'       =    TO_TIME('@FirstOccurrence') ON INSERT ONLY,
        'REPORTEDPRIORITY' =    Lookup('@Severity','SeverityTable'),
        'STATUS'         =    Lookup('@Severity', 'StatusTable'),
        'TTNumber'         =    '@TTNumber',
        'CINUM'    =      '@CI',
        'AFFECTEDPERSON' = '@AffectedPerson',
        'COMMODITY'   =   '@AffectedService',
        'ASSETSITEID' =   '@AssetLocSiteID',
        'IMPACT'   =    '@Impact',
        'URGENCY'     =       '@Urgency',
        'OWNERGROUP'  =   '@OwnerGroup',
        'HIERARCHYPATH'    =   '@HierarchyPath'
);

CREATE MAPPING JournalMap
(
    'Chrono'        =       '@Chrono',
    'CREATEDATE'        =   TO_TIME('@Chrono'),
    'DESCRIPTION'       =   'NETCOOL JOURNAL ENTRY',
    'DESCRIPTION_LONGDESCRIPTION'       =   TO_STRING('@Text1') + TO_STRING('@Text2') +
TO_STRING('@Text3')
);
```

Tip: Observe the maximum field length in TSRM. If the length is not respected, the ticket cannot be opened and an error is generated in the `GATE_TSRM.log` file.

For more information n see *IBM Tivoli Netcool/OMNIbus Gateway for TSRM*, SC27-2703.

16.5.8 Tivoli Netcool/OMNIbus Process Agent control

Process Agent is a program that manages processes and services in a process control system. We use Process Agent for managing the object server, probe for EIF, and Tivoli Service Request Manager gateway.

Process agent is configured by the `$OMNIHOME/etc/nco_pa.conf` configuration file as shown in Example 16-10.

Example 16-10 The nco_pa.conf example file

```
nco_process 'MasterObjectServer'
{
        Command '$OMNIHOME/bin/nco_objserv -name NCOMS -pa NCO_PA' run as 0
        Host            =       'nci51demo'
        Managed         =       True
        RestartMsg      =       '${NAME} running as ${EUID} has been restored on ${HOST}.'
        AlertMsg        =       '${NAME} running as ${EUID} has died on ${HOST}.'
        RetryCount      =       0
        ProcessType     =       PaPA_AWARE
}

nco_process 'GateTSRM'
{
    Command '$OMNIHOME/bin/nco_g_tsrm -name GATE_TSRM -propsfile
$OMNIHOME/etc/GATE_TSRM.props' run as 0
    Host            =       'nci51demo'
```

```
    Managed         =       True
    RestartMsg      =       '${NAME} running as ${EUID} has been restored on ${HOST}.'
    AlertMsg        =       '${NAME} running as ${EUID} has died on ${HOST}.'
    RetryCount      =       3
    ProcessType     =       PaPA_AWARE

}

nco_process 'ProbeEIF'
{
    Command '$OMNIHOME/probes/nco_p_tivoli_eif -name PROBE_EIF -propsfile
$OMNIHOME/etc/PROBE_EIF.props' run as 0
    Host            =       'nci51demo'
    Managed         =       True
    RestartMsg      =       '${NAME} running as ${EUID} has been restored on ${HOST}.'
    AlertMsg        =       '${NAME} running as ${EUID} has died on ${HOST}.'
    RetryCount      =       3
    ProcessType     =       PaPA_AWARE
}

nco_service 'Core'
{
        ServiceType     =       Master
        ServiceStart    =       Non-Auto
        process 'MasterObjectServer' NONE
        process 'GateTSRM' 'MasterObjectServer'
        process 'ProbeEIF' 'MasterObjectServer'
}

nco_service 'InactiveProcesses'
{
        ServiceType     =       Non-Master
        ServiceStart    =       Non-Auto
}

nco_routing
{
        host 'nci51demo' 'NCO_PA' 'user' 'password'
}
```

After editing the configuration file perform the following steps:

1. Start the Process Agent:

   ```
   sudo $OMNIHOME/bin/nco_pad -name NCO_PA
   ```

2. Start ProbeEIF:

   ```
   nco_pa_start -user netcool -password netcool -process ProbeEIF
   ```

3. Start Tivoli Netcool/OMNIbus Gateway for Tivoli Service Request Manager gateway:

   ```
   nco_pa_start -user netcool -password netcool -process GateTSRM
   ```

For more information about Process Agent, see *IBM Tivoli Netcool/OMNIbus Administration Guide*, SC23-6371.

Archived

16.5.9 The Tivoli Netcool/Impact project

In this step, we create a new Tivoli Netcool/Impact project and include the following data models:

- Data sources
- Data types
- Data items
- Links

Creating a new project

Perform the following steps to create a new project:

1. Log on to the Netcool/Impact server with administrative permission.
2. On the Projects tab, click **New Project**.
3. Give a name to the project and click **OK** without adding any member, as shown in Figure 16-33.

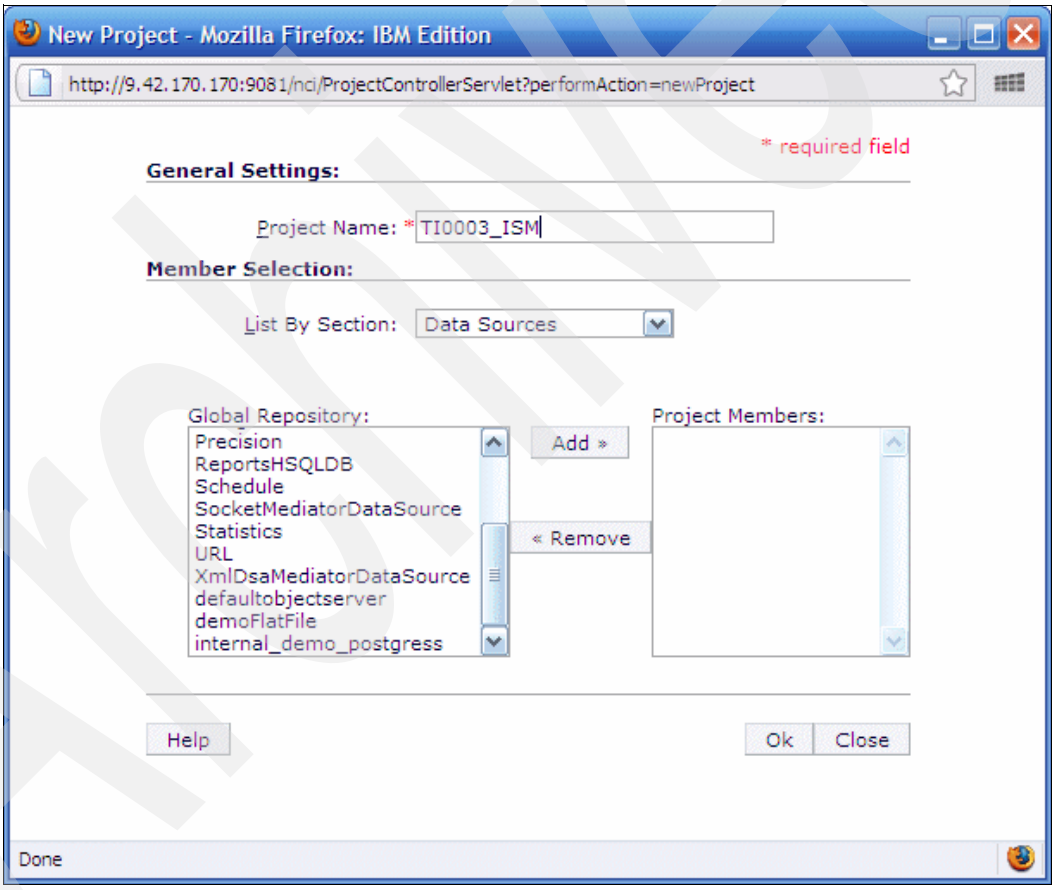


Figure 16-33 New Project

Creating data sources

Perform the following steps:

1. Inside the project, go to **Data Sources and Types** section and click **New Data Source** after selecting **DB2** as source type.
2. Specify the connection settings for MAXDB as shown in Figure 16-34 on page 453.

Archived

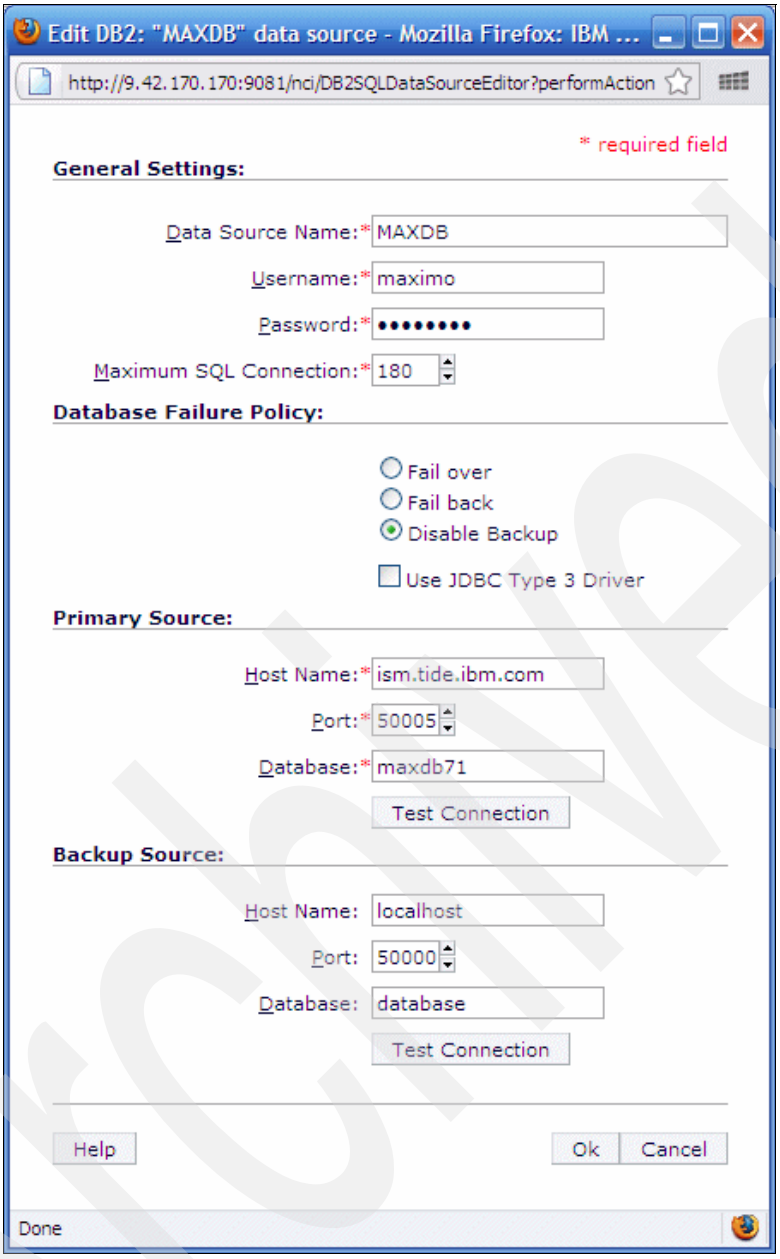


Figure 16-34 Data source settings for MAXDB

3. Click **Test Connection** to check the database connection. See Figure 16-35.

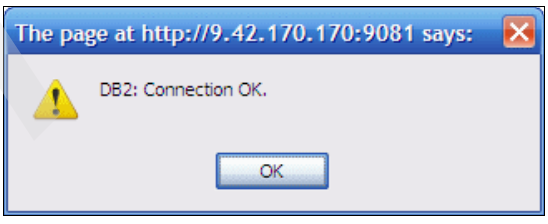


Figure 16-35 Connection test

4. Click **OK** to return to the project.

Archived

5. Select **ObjectServer** as source type and click **New Data Source**.
6. Specify the connection settings for Object Server as shown in Figure 16-36.

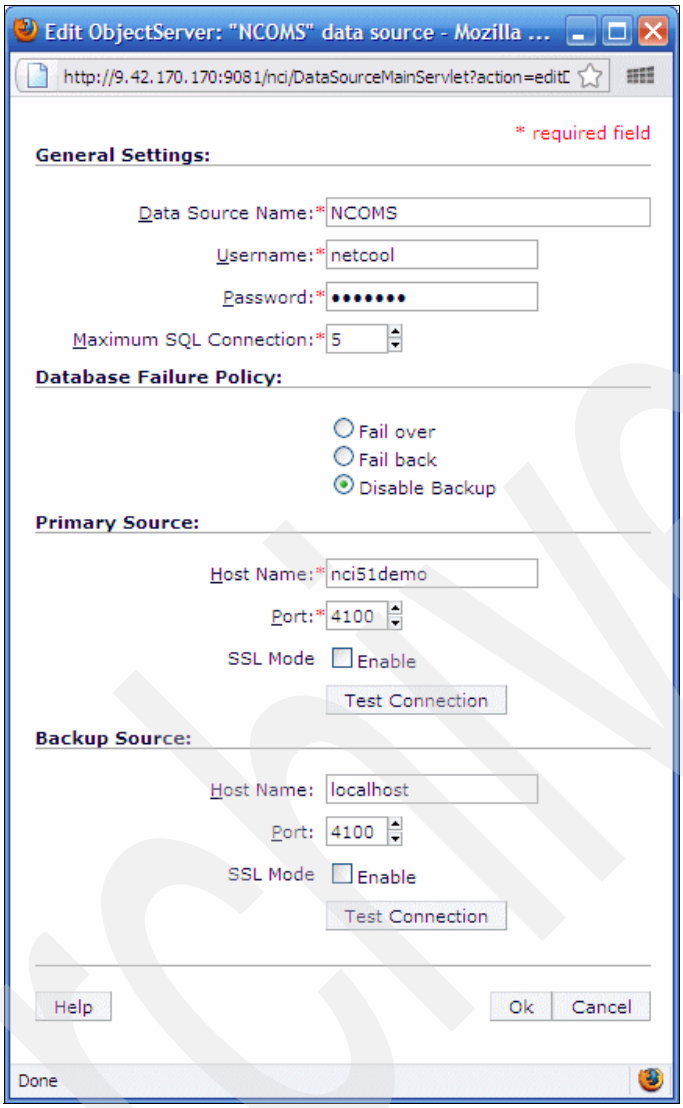


Figure 16-36 Data source settings for NCOMS

7. Test the connection and click **OK** to return to project.

Creating data types

Follow these steps to create the Data Types:

1. Inside the project, go to the Data Sources and Types section and click **Create a new data type of MAXDB**, next to MAXDB.
2. Set Data Type Name field to `CLASSANCESTOR`.
3. Select **MAXDB** in the drop-down Data Source Name menu.
4. Set Base Label to `CLASSANCESTOR` and click **Refresh**.

Archived

5. Set primary key to `CLASSANCESTORID` by checking **Key Field** in the same row.

   Figure 16-37 shows the data type settings.

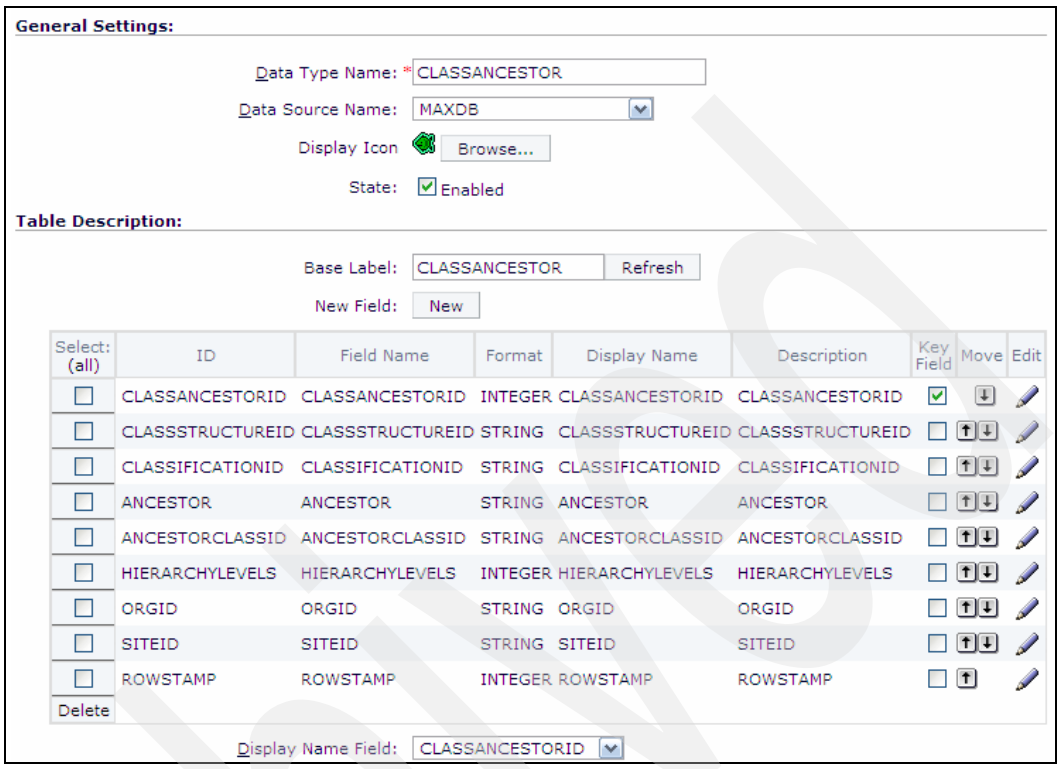


Figure 16-37 Data type example for CLASSANCESTOR table

6. Click **Save**, and close the edit window.
7. Repeat the steps 1 - 6 to enter the values in Table 16-5.

Table 16-5 Data types

Data Type Name	Data Source Name	Base Label	Key Field
CLASSSTRUCTURE	MAXDB	CLASSSTRUCTURE	CLASSSTRUCTUREID
CISPEC	MAXDB	CISPEC	CISPECID
CI	MAXDB	CI	CIID
IMPACTMAP	MAXDB	IMPACTMAP	IMPACTID
PMCHGCWTP	MAXDB	PMCHGCWTP	PMCHGCWTPID
NCOMS_AllEvents	NCOMS	alerts.status	Identifier

Notes:

- When there is a change in the source table which data type refers to, you should refresh and save the data type.
- The CLASSANCESTOR table must be sorted by HIERARCHYLEVELS for a correct HierarchyPath assignment. Edit the Data Filter and Ordering part for this data type.

Archived

Creating the links

Perform the following steps to create the Links:

1. In the project, go to the Data Sources and Types section and click **CISPEC** data type.
2. Click the **Dynamic Links** tab.
3. Click **New Link By Filter**.
4. Select **CI** as the Target Data Type.
5. Set the Filter into target Data Type value to:

   ```
   CINUM='%CINUM%'
   ```

6. Click **OK** to go back to data type window.
7. Click **Save,** and close this edit window.
8. Repeat steps 1 - 6 to enter the values in Table 16-6.

Table 16-6 Creating the Links

Source Data Type	Target Data Type	Filter into target Data Type
CISPEC	CI	CINUM='%CINUM%'
IMPACTMAP	CLASSANCESTOR	CLASSIFICATIONID='%INCIDENTCLASSIFICATION%'
IMPACTMAP	CLASSSTRUCTURE	CLASSIFICATIONID='%CHANGECLASSIFICATION%'

16.5.10 Tivoli Netcool/Impact Enrichment Policy for Incident Ticket

Perform the following steps to create an enrichment policy that enriches the event by using data gathered from CCMDB:

1. In the project, click **Template** → **Custom** and go to **Policies** → **New Policy**.
2. Edit the policy, as shown in Example 16-11.

Example 16-11 TSRM_Incident policy

```
@CI="Unknown";
@AssetLocSiteID="Unknown";
@AffectedService="Unknown";
@AffectedPerson="Unknown";
@Impact=1;
@Urgency=1;
@OwnerGroup="Unknown";

log("Start policy: TSRM_Incident");
CISPEC = GetByFilter('CISPEC', "(ALNVALUE='"+EventContainer.TECHostname+"')", false);

if (length(CISPEC) > 0)
    {
    CI=GetByLinks({"CI"},null,1,CISPEC);

    if (length(CI) > 0)
        {
        @CI =""+CI[0].CINUM;
        @AffectedService=""+CI[0].SERVICE;
        @AffectedPerson=""+CI[0].PERSONID;
        @AssetLocSiteID=""+CI[0].ASSETLOCSITEID;
        PMCHGCWTP=GetByFilter('PMCHGCWTP', "(CALNUM='"+CI[0].PMCHGCWNUM+"')",false);
        WindowNum=length(PMCHGCWTP);
```

```
        if(WindowNum > 0)
            {
            Counter=0;
            Maintenance=0;
            EventTime=String(LocalTime(EventContainer.FirstOccurrence, "yyyy-MM-dd
HH:mm:ss.S"));

            while(Counter < WindowNum)
                {
                StartTime=String(PMCHGCWTP[Counter].STARTTIME);
                EndTime=String(PMCHGCWTP[Counter].ENDTIME);

                 if((EventTime>=StartTime)&&(EventTime<=EndTime))
                    {
                      Maintenance=1;
                      Counter=WindowNum;
                    }
                Counter=Counter+1;
                }

            if(Maintenance==1)
                {
                @Action="System is in maintenance mode, no action taken";
                }
            else
                {
                @Action="Incident ticket is created";
                @LogTicket=1;
                }
            @Acknowledged=1;
            }
        }
    }

IMPACT= GetByFilter('IMPACTMAP', "(CLASSIFICATIONID='"+EventContainer.AlertGroup+"')",
false);
        if(length(IMPACT) > 0)
            {
            PATH=GetByLinks({"CLASSANCESTOR"},null,10,IMPACT);
            log(PATH);
            PathLength=length(PATH);
            Counter=0;
            Path="";
            while(Counter<PathLength)
                {
                if(Counter>0)
                    {
                    Path=" \\ "+Path;
                    }
                Path=PATH[Counter].ANCESTORCLASSID+Path;
                Counter=Counter+1;
                }
            log("Classification Path is "+Path);

            @Impact=IMPACT[0].IMPACT;
            @Urgency=IMPACT[0].URGENCY;
            @OwnerGroup=""+IMPACT[0].OWNERGROUP;
            @HierarchyPath=Path;
            }
```

Archived

```
ReturnEvent(EventContainer);

log("!!!!!!! End Of Executing Policy");
```

3. Rename the policy as **TSRM_Incident** and save it.

16.5.11 CCMDB Web Services

Perform the following steps to create web services in CCMDB:

1. Log on to the CCMDB server with administrative permission.
2. Navigate to **Go To** → **System Configuration** → **Platform Configuration** → **WebServices Library**.
3. Navigate to **Select Actions** → **Create Web Service** → **Create WS from an Enterprise Service**.
4. Select the source name **EXTSYSTDI_MXChangeCreate** and click **Create**.
5. In the List tab, go to **EXTSYSTDI_MXChangeCreate**.
6. Navigate to **Select Action** → **Deploy Web Service**. The following message is displayed:

   `Web Service is deployed for service EXTSYSTDI_MXChangeCreate`
7. In the Operations section, click **GenerateSchema/View XML**.
8. In the View XML window, select and copy the XML data to a text editor.

Creating endpoint for Web Services

Perform the following steps to create the Web Services:

1. Navigate to **Integration** → **Endpoint**.
2. Create a new entry named `IMPACT`.
3. Set Handler value to **WEBSERVICE**.
4. Obtain the endpoint URL from the following file:

   `<websphere_application_installation_path>\profiles\AppSrv01\axis2\wsdl\EXTSYSTDI_MXChangeCreate.wsdl`

   The URL is at the end of the file, and is similar to the following URL:

   `http://9.42.171.198/meaweb/services/EXTSYSTDI_MXChangeCreate`
5. In the Property field, specify the endpoint URL from the previous step.
6. Obtain the soapAction value from the same WSDL file. The value is close to the end of the file on a line similar to the following line:

   `<soap12:operation soapAction="urn:processDocument" style="document" />`
7. Set the SERVICENAME value to the service name in the WSDL file. For `EXTSYSTDI_MXChangeCreate`, use `EXTSYSTDI_MXChangeCreateService`.
8. Save your changes.

Archived

16.5.12 Tivoli Netcool/Impact Web Services DSA Policy for RFC

In this step, we compile WSDL file and create a web services policy that invokes CCMDB web service to create a request for change.

Compiling WSDL file

Perform the following steps to compile the WSDL file

1. Copy the following `wsdl` and `schema` folders from CCMDB server to Tivoli Netcool/Impact server:

   ```
   <websphere_application_installation_path>\profiles\AppSrv01\axis2\wsdl
   <websphere_application_installation_path>\profiles\AppSrv01\axis2\schema
   ```

2. Copy the following `.xsd` files to the `wsdl` folder in your Tivoli Netcool/Impact server:

   ```
   schema/common/mos/MXOSCHANGE.xsd
   schema/common/mbo/wochange.xsd
   schema/common/meta/MXMeta.xsd
   schema/service/MXOSCHANGEService.xsd
   ```

3. Edit the following `.wsdl` and `.xsd` files and replace the `xsd:import` and `xsd:include` paths according to the new paths in your Tivoli Netcool/Impact server as shown in Example 16-12.

   Example 16-12 Changes in wsdl and xsd files

   ```
   wsdl/EXTSYSTDI_MXChangeCreate.wsdl:

   <xsd:import namespace="http://www.ibm.com/maximo"
   schemaLocation="MXOSCHANGEService.xsd"/>

   schema/service/MXOSCHANGEService.xsd:

   <xsd:include schemaLocation="MXMeta.xsd"/>
   <xsd:include schemaLocation="MXOSCHANGE.xsd"/>
   <xsd:include schemaLocation="wochange.xsd"/>

   schema/common/mos/MXOSCHANGE.xsd:

   <xsd:include schemaLocation="MXMeta.xsd"/>

   schema/common/mbo/wochange.xsd:

   <xsd:include schemaLocation="MXMeta.xsd"/>
   ```

4. Run the **`nci_compilewsdl`** command to compile the WSDL file and to produce a set of Java class files that contain a programmatic representation of the WSDL data:

   ```
   $IMPACT_HOME/bin/nci_compilewsdl <package_name> <wsdl_file> <destination>
   ```

   The command has the following definitions:

   package_name Specifies name of the `.jar` file

   wsdl_file Specifies the location of the WSDL file on the local file system.

   destination Specifies the directory to which to copy the generated `.jar` file.

Archived

We use the following command:

```
$IMPACT_HOME/bin/nci_compilewsdl EXTSYSTDI_MXChangeCreate
/home/netcool/wsdl/EXTSYSTDI_MXChangeCreate.wsdl $NCHOME/wslib
```

Notes:

- The Web Services Wizard is also capable of compiling the WSDL file. Instead of compiling the WSDL file as described in the previous step, you can select **Provide a package name for the new client stub** and enter `Package Name` as in Figure 16-39 on page 461.
- Current AXIS (Apache AXIS is an open source, XML-based web service framework) tools used in Netcool/Impact are unable to compile the WSDL files generated. After Netcool/Impact moves to the newer version of AXIS tools, the compilation of the WSDL file through URL will be supported.

Creating the Web Services Policy

We use Web Services Wizard to create our policy, as follows:

1. In the project, go to **Wizards** → **Web Services.**
2. Enter `CCMDB_RFC` as the policy name, as shown in Figure 16-38.

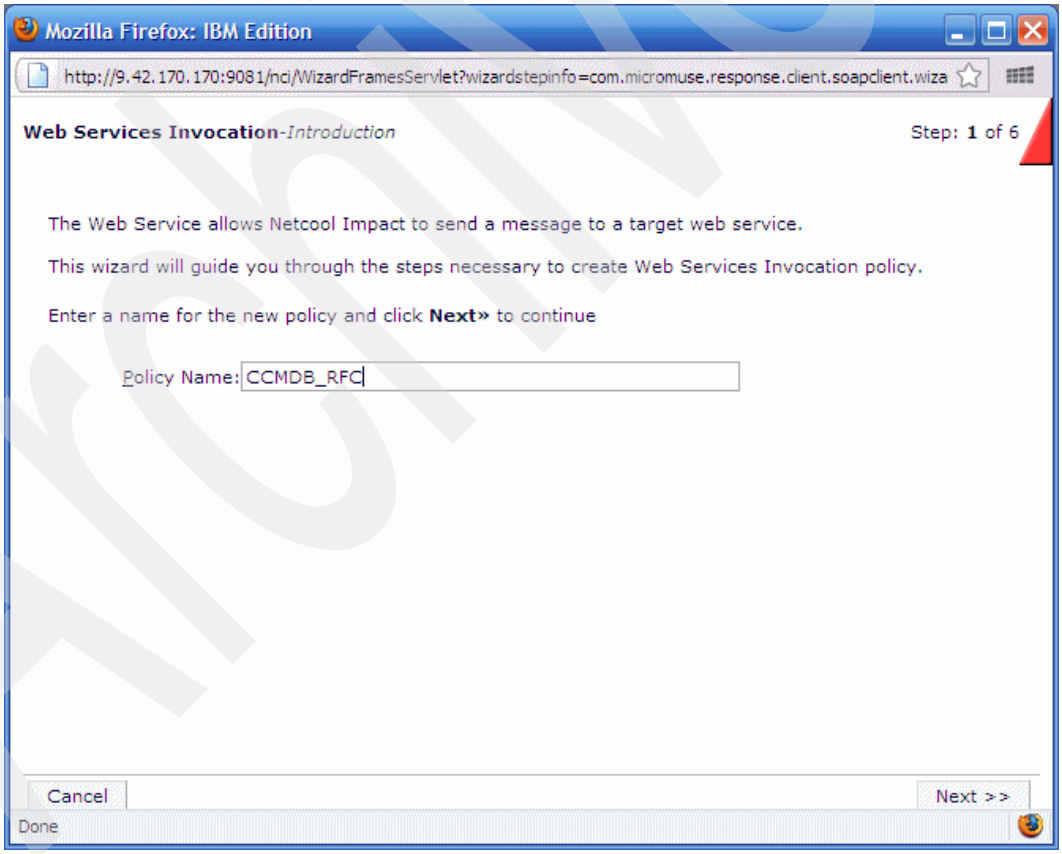


Figure 16-38 Introduction

Archive

3. Enter the path to WSDL and select the compiled `.jar` file, as shown in Figure 16-39.

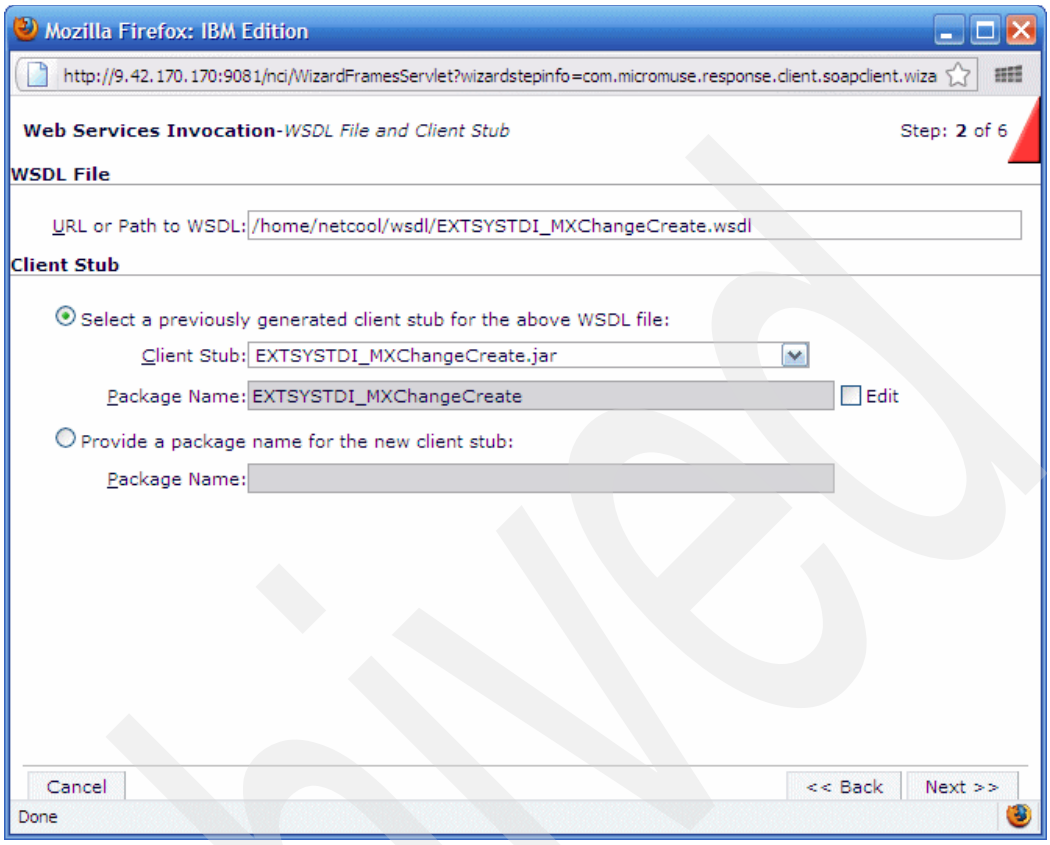


Figure 16-39 WSDL file and Client Stub

Archived

4. Select the web service, port type, and method, as shown in Figure 16-40.

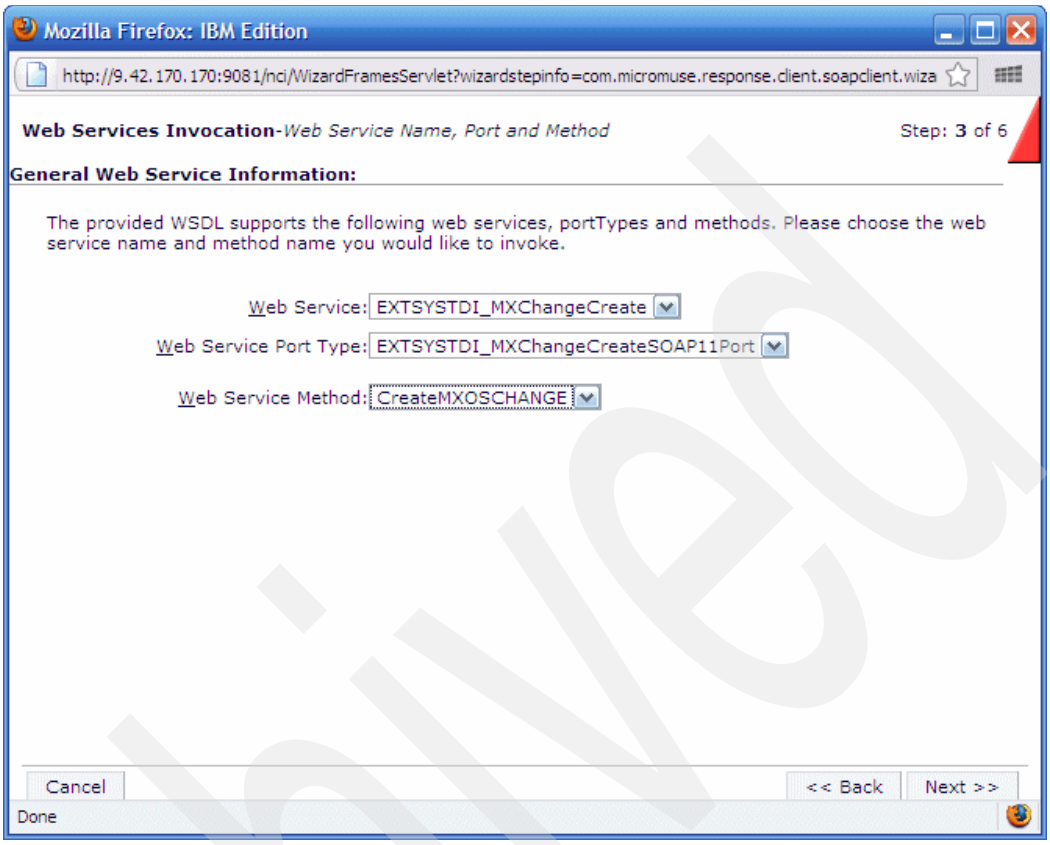


Figure 16-40 Web Service name, port, and method

Archived

5. Expand the parameters and click **OK** after entering 1 as collection type size, as shown in Figure 16-41.

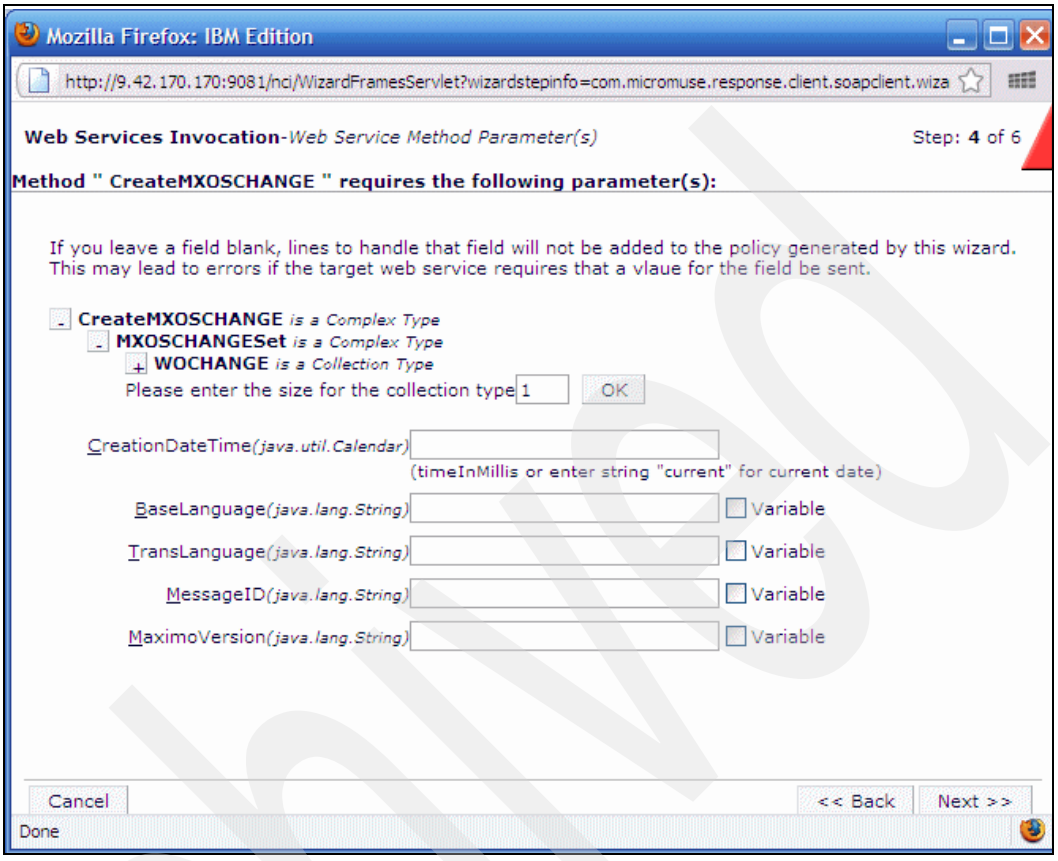


Figure 16-41 Web Service method parameters

Archive

6. Expand the parameters you want to add and fill in the default values. See Figure 16-42.

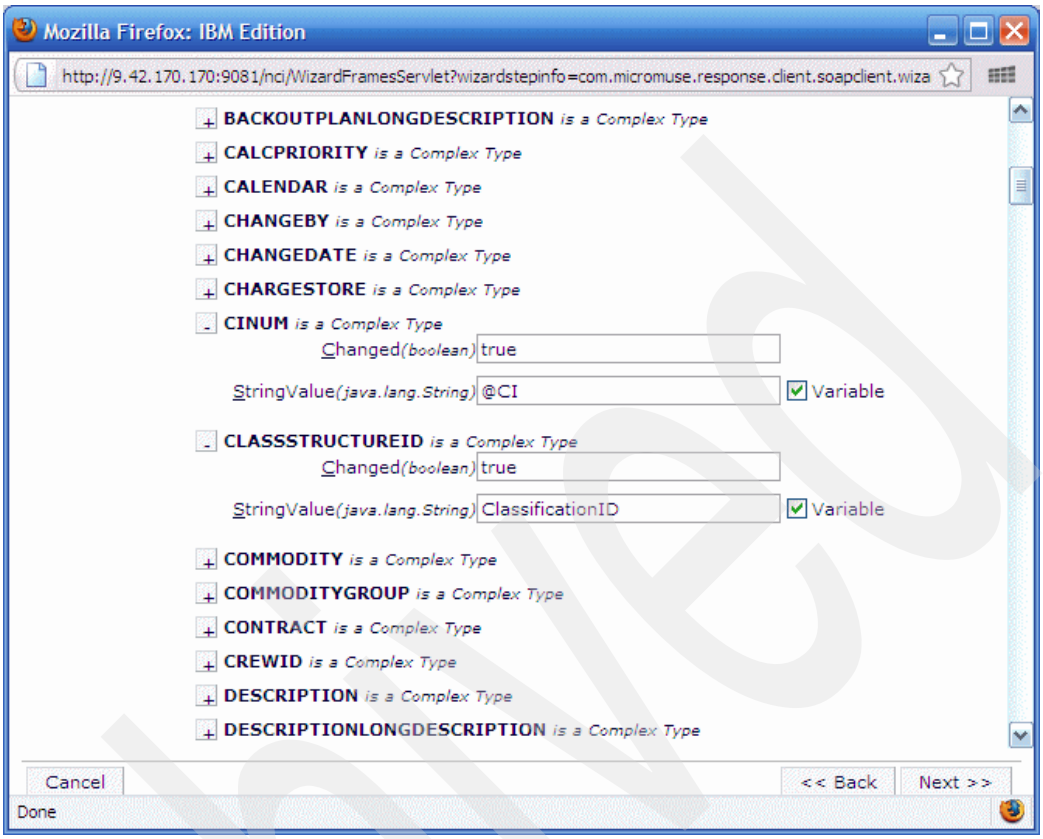


Figure 16-42 Web Service method parameters

Archived

7. Below the form, select **AddChange** as Action and fill in the other fields. See Figure 16-43.

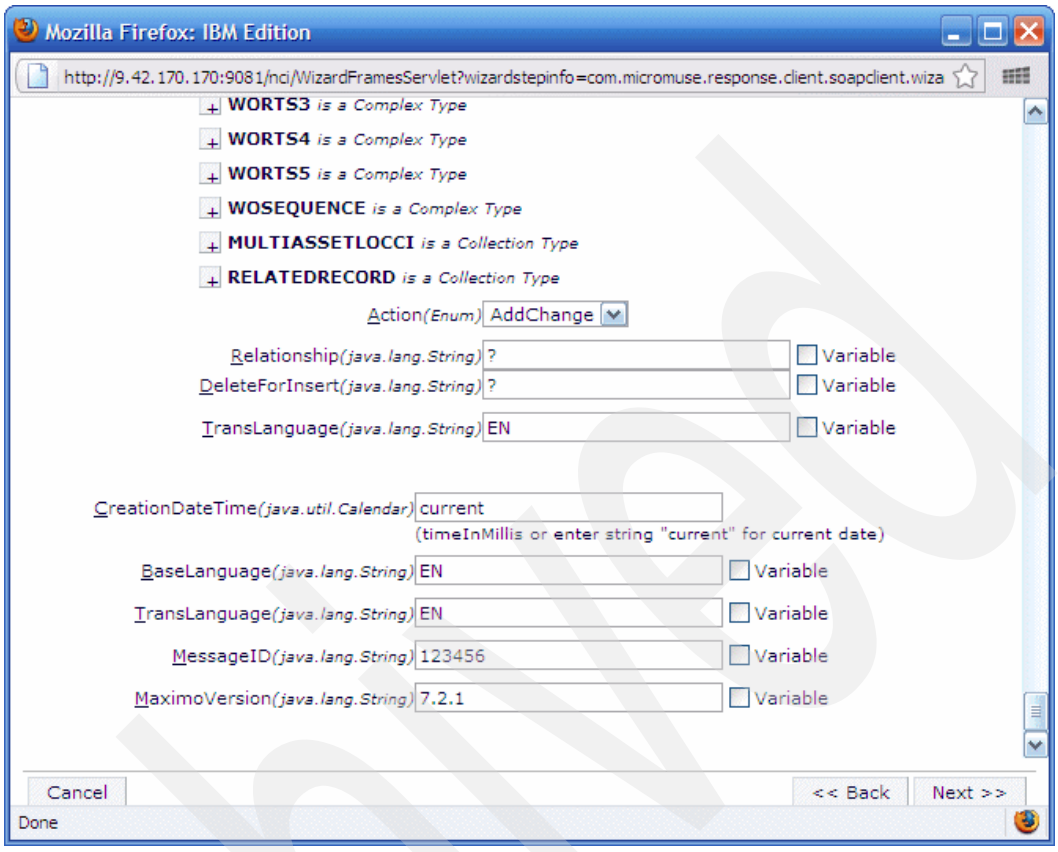


Figure 16-43 Web Service method parameters

Archived

8. Check the End Point URL, as shown in Figure 16-44.



Figure 16-44 Web Service End Point

Archived

9. Click **Finish** to exit the wizard and to generate a policy, as shown in Figure 16-45.



Figure 16-45 Summary and Finish

10. Newly created CCMDB_RFC policy is automatically open for editing. Add the section in Example 16-13 on page 468 to the beginning of the policy.

Archived

Example 16-13 Beginning of policy

```
@CI="Unknown";
@AssetLocSiteID="Unknown";
@OwnerGroup="Unknown";

log("Start policy: CCMDB_RFC");
CISPEC = GetByFilter('CISPEC', "(ALNVALUE='"+EventContainer.TECHostname+"')",
false);

if (length(CISPEC) > 0)
    {
    CI=GetByLinks({"CI"},null,1,CISPEC);

    if (length(CI) > 0)
        {
        @CI =""+CI[0].CINUM;
        @AssetLocSiteID=""+CI[0].ASSETLOCSITEID;
        }
    }

IMPACT= GetByFilter('IMPACTMAP',
"(CLASSIFICATIONID='"+EventContainer.AlertGroup+"')", false);
        if(length(IMPACT) > 0)
            {
            ChangeClassification=GetByLinks({"CLASSSTRUCTURE"},null,1,IMPACT);
            ClassificationID=""+ChangeClassification[0].CLASSSTRUCTUREID;
            @OwnerGroup=""+IMPACT[0].OWNERGROUP;
         }
```

11. To the end of the policy, add the section in Example 16-14.

Example 16-14 End of policy

```
@Action="RFC "+Extract(Extract(WSInvokeDLResult,5,"<"),1,">")+" is created";
@Acknowledged=1;
```

The final policy is shown in Example 16-15.

Example 16-15 CCMDB_RFC policy

```
//This policy generated by Impact Wizard.

//This policy is based on wsdl file at /home/netcool/wsdl/EXTSYSTDI_MXChangeCreate.wsdl

@CI="Unknown";
@AssetLocSiteID="Unknown";
@OwnerGroup="Unknown";

log("Start policy: CCMDB_RFC");
CISPEC = GetByFilter('CISPEC', "(ALNVALUE='"+EventContainer.TECHostname+"')", false);

if (length(CISPEC) > 0)
    {
    CI=GetByLinks({"CI"},null,1,CISPEC);

    if (length(CI) > 0)
        {
```

Archived

```
        @CI =""+CI[0].CINUM;
        @AssetLocSiteID=""+CI[0].ASSETLOCSITEID;
        }
    }

IMPACT= GetByFilter('IMPACTMAP', "(CLASSIFICATIONID='"+EventContainer.AlertGroup+"')",
false);
        if(length(IMPACT) > 0)
            {
            ChangeClassification=GetByLinks({"CLASSSTRUCTURE"},null,1,IMPACT);
            ClassificationID=""+ChangeClassification[0].CLASSSTRUCTUREID;
            @OwnerGroup=""+IMPACT[0].OWNERGROUP;
            }

//Specify package name as defined when compiling WSDL in Impact
WSSetDefaultPKGName('EXTSYSTDI_MXChangeCreate');

//Specify parameters
CreateMXOSCHANGEDocument=WSNewObject("com.ibm.www.maximo.CreateMXOSCHANGEDocument");
_CreateMXOSCHANGE=WSNewSubObject(CreateMXOSCHANGEDocument,"CreateMXOSCHANGE");

_MaximoVersion = '7.2.1';
_CreateMXOSCHANGE['MaximoVersion'] = _MaximoVersion;
_MessageID = '123456';
_CreateMXOSCHANGE['MessageID'] = _MessageID;
_TransLanguage = 'EN';
_CreateMXOSCHANGE['TransLanguage'] = _TransLanguage;
_BaseLanguage = 'EN';
_CreateMXOSCHANGE['BaseLanguage'] = _BaseLanguage;

//Handle special calendar type...
date = WSNewObject("java.util.GregorianCalendar");
_CreationDateTime = date;
_CreateMXOSCHANGE['CreationDateTime'] = _CreationDateTime;

_MXOSCHANGESet = WSNewSubObject(_CreateMXOSCHANGE,"MXOSCHANGESet");

_WOCHANGE_0_ = WSNewSubObject(_MXOSCHANGESet,"WOCHANGE");
_WOCHANGE_0_['TransLanguage'] = 'EN';
_WOCHANGE_0_['DeleteForInsert'] = '?';
_WOCHANGE_0_['Relationship'] = '?';
_WOCHANGE_0_['Action'] =
WSNewEnum('com.ibm.www.maximo.ProcessingActionType','AddChange');

_CINUM = WSNewSubObject(_WOCHANGE_0_,"CINUM");
_CINUM['StringValue'] = @CI;
_CINUM['Changed'] = true;
log(_CINUM['StringValue']);

_CLASSSTRUCTUREID = WSNewSubObject(_WOCHANGE_0_,"CLASSSTRUCTUREID");
_CLASSSTRUCTUREID['StringValue'] = ClassificationID;
_CLASSSTRUCTUREID['Changed'] = true;
log(_CLASSSTRUCTUREID['StringValue']);

_OWNERGROUP = WSNewSubObject(_WOCHANGE_0_,"OWNERGROUP");
_OWNERGROUP['StringValue'] = @OwnerGroup;
_OWNERGROUP['Changed'] = true;
log(_OWNERGROUP['StringValue']);

_SITEID = WSNewSubObject(_WOCHANGE_0_,"SITEID");
```

Archived

```
_SITEID['StringValue'] = @AssetLocSiteID;
_SITEID['Changed'] = true;
log(_SITEID['StringValue']);

WSParams = {CreateMXOSCHANGEDocument};
log(WSParams);

//Specify web service name, end point and method
WSService = 'EXTSYSTDI_MXChangeCreate';
WSEndPoint = 'http://9.42.171.198/meaweb/services/EXTSYSTDI_MXChangeCreate';
WSMethod = 'CreateMXOSCHANGE';

WSInvokeDLResult = WSInvokeDL(WSService, WSEndPoint, WSMethod, WSParams);

log("Response to Web Service call CreateMXOSCHANGE ...: " +WSInvokeDLResult);

@Action="RFC "+Extract(Extract(WSInvokeDLResult,5,"<"),1,">")+" is created";
@Acknowledged=1;

ReturnEvent(EventContainer);

log("!!!!!!! End Of Executing Policy");
```

12. Save your policy after you finish editing.

16.5.13 Tivoli Netcool/Impact policy for Tivoli Provisioning Manager workflow

In this step, we configure SOAP client for and create a new policy that invokes Tivoli Provisioning Manager workflow.

Configuring the SOAP Client

Perform the following steps to configure the SOAP Client:

1. Copy the contents of TIO_HOME/soapclient directory from the provisioning server to a folder created on the Tivoli Netcool/Impact server. We use the following command to copy the SOAP client:

   ```
   scp -r /opt/IBM/tivoli/tpm/soapclient netcool@nci51demo:/home/netcool/SOAP
   ```

2. Edit the `soapcli.sh` file and set parameters as shown in Example 16-16.

   Example 16-16 The soapcli.sh file

   ```
   JAVA_HOME=/opt/ibm/netcool/platform/linux2x86/jre_1.5.6/jre
   TIO_HOME=/home/netcool/SOAP
   export TIO_LIB="$TIO_HOME"/tpmlteSoap/lib
   ```

3. Save the `soapcli.sh` file.

Creating the workflow policy

Perform the following steps to create the workflow policy:

1. In the project, go to **Policies** and click **Template** → **Custom**, and then click **New Policy**.
2. Edit your policy, as shown in Example 16-17 on page 471.

Example 16-17 Workflow policy

```
Hostname=@TECHostname;
Command="/home/netcool/SOAP/tpmlteSoap/soapcli.sh";
Parameters={"maxadmin","maxadmin","http://9.42.171.32:8777/ws/pid/TpmLiteSoapSe
rvice?wsdl","executeDeploymentRequest","Increase_Max_Heap_Size",Hostname,"20"};
TimeOut=60;

Response=JRExecAction(Command, Parameters, False, TimeOut);

WorkflowID=Trim(Extract(Response,1,":"));

Log("Sleeping");
JavaCall("java.lang.Thread",null,"sleep",{60000});
Log("Awake");

Parameters={"maxadmin","maxadmin","http://9.42.171.32:8777/ws/pid/TpmLiteSoapSe
rvice?wsdl","isSuccessful",WorkflowID};

Response=JRExecAction(Command, Parameters, False, TimeOut);

Result=Trim(Extract(Response,1,":"));

if(Result=="true")
    {
    @Action="Provisioning workflow "+WorkflowID+" is succeeded";
    @Acknowledged=1;
    }
else
    {
    @Action="Provisioning workflow "+WorkflowID+" is failed";
    }

ReturnEvent(EventContainer);

log("!!!!!!! End Of Executing Policy");
```

3. Rename the policy to **TPM_Workflow** and save it.

Note: In this example, we used the following values:

- The Increase_Max_Heap_Size workflow, which has two parameters: Hostname and Percentage. This workflow increases the max_heap_size as *Percentage* on the target *Hostname*.
- A value of 60 seconds as the waiting time between workflow execution and workflow result checking steps

Running JRExec server

JRExec is a runnable server component that is used by external commands or scripts from within a Tivoli Netcool/Impact policy. We use this server to run the `soapcli.sh` script.

Run the **`$NCHOME/impact/bin/nci_jrexec`** command to start JRExec server.

Archived

16.5.14 Tivoli Netcool/Impact event mapping

Perform the following steps to set Tivoli Netcool/OMNIbus Event Reader configuration to map specific events to specific policies:

1. In the project, click **OMNIbus EventReader.**
2. Click the **Event Mapping** tab.
3. Click **New**, next to New Mapping.
4. Set Filter Expression to the following value:

   ```
   (AlertGroup = 'ITM_NT_Process') or (AlertGroup = 'ITM_Thread_Pools')
   ```

5. Select **TSRM_Incident** policy as Policy to Run.
6. Check **Active**.
7. Click **OK**.
8. Repeat steps 3 - 7 to enter the values in Table 16-7.

Table 16-7 Event mapping settings

Restriction Filter	Policy Name	Active
AlertGroup = 'ITM_DB_Connection_Pools'	CCMDB_RFC	Yes
ITM_Garbage_Collection_Analysis	TPM_Workflow	Yes

When complete, the event mapping setting looks similar to Figure 16-46.

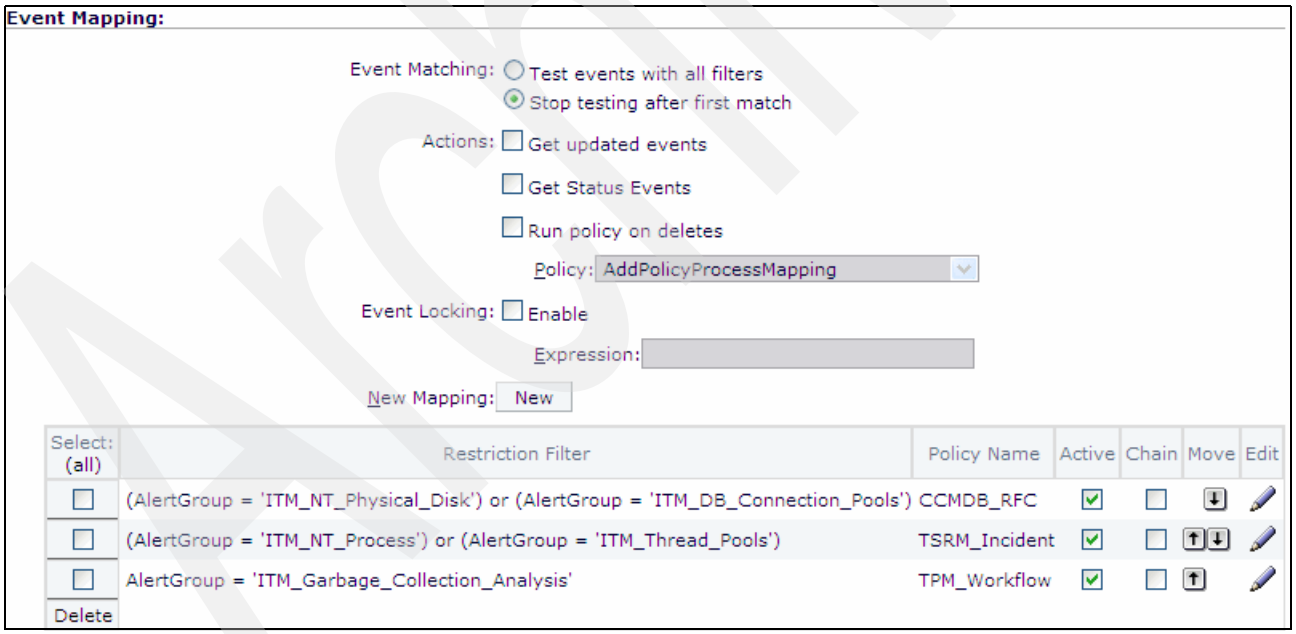


Figure 16-46 Event Mapping

9. Click **OK** to return to project.

Archived

16.5.15 Edit Netcool/OMNIbus event list view

Perform the following steps to edit event list view:

1. Run **`nco_event` command** with administrative permission, as shown in Figure 16-47.

> **AEL:** In the figure, the native event list was used to show the monitor box and event list. You may also use the Web GUI Active Event List (AEL) for this purpose. AEL is the list of events that the user can configure to monitor the environment. With the help of certain tools, the user can work on the stored events and take actions. See Chapter 11, “WebGUI launch to IBM Tivoli Monitoring” on page 303 for details of Web GUI Active Event List (AEL).

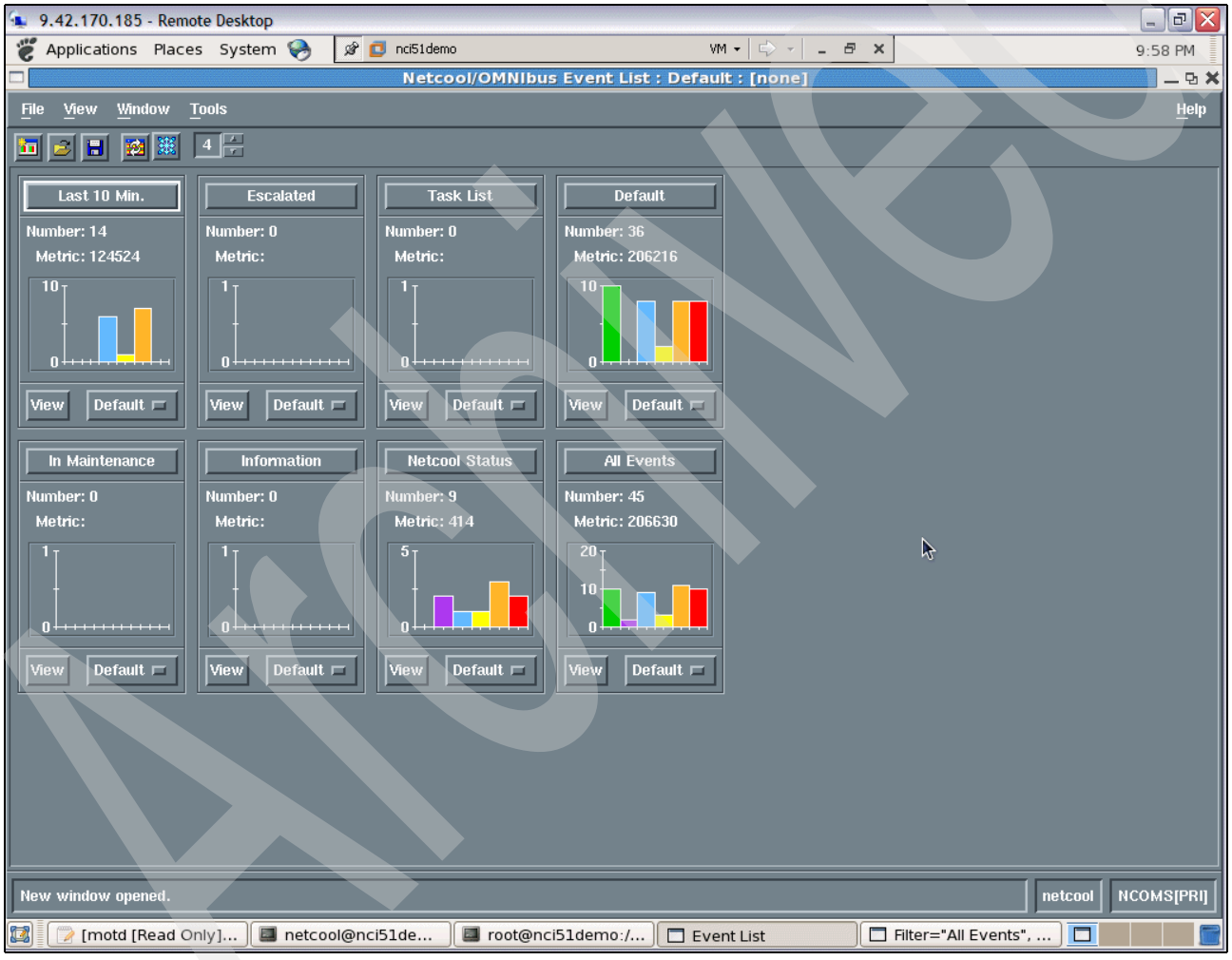


Figure 16-47 The nco_event default view

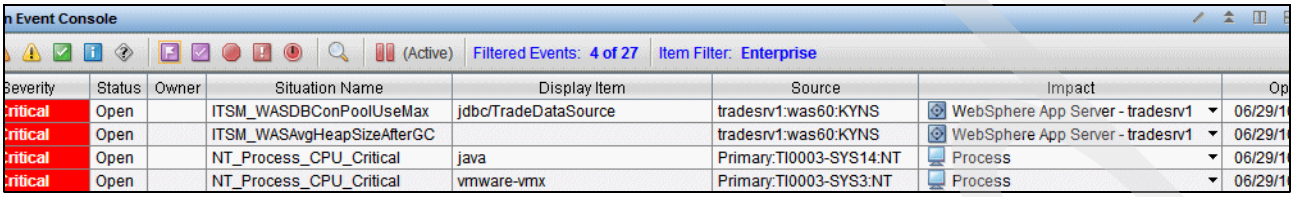
2. Click **View**, in the All Events section.
3. Click **Edit** → **Edit View**.
4. Select **CI**, **Action**, **TTNumber**, and **TicketStatus** attributes and click **Add**.
5. Click **Apply** and then **Close**.

Archive

16.6 Scenario walk-through

In this section, we walk you through the scenario components.

This scenario starts with a Situation Event Console view in Tivoli Enterprise Portal. The operator views four critical situations in the *Open* status as shown in Figure 16-48.



Severity	Status	Owner	Situation Name	Display Item	Source	Impact	Op
ritical	Open		ITSM_WASDBConPoolUseMax	jdbc/TradeDataSource	tradesrv1:was60:KYNS	WebSphere App Server - tradesrv1	06/29/1
ritical	Open		ITSM_WASAvgHeapSizeAfterGC		tradesrv1:was60:KYNS	WebSphere App Server - tradesrv1	06/29/1
ritical	Open		NT_Process_CPU_Critical	java	Primary:TI0003-SYS14:NT	Process	06/29/1
ritical	Open		NT_Process_CPU_Critical	vmware-vmx	Primary:TI0003-SYS3:NT	Process	06/29/1

Figure 16-48 Open Situations in Tivoli Enterprise Portal

In the background, the following operations occur:

1. Event Integration Facility running on IBM Tivoli Monitoring forwards these situations to Tivoli Netcool/OMNIbus Probe for Tivoli EIF.
2. Probe inserts the events in the alerts.status table of the object server after processing according to its rules file.
3. Tivoli Netcool/Impact/OMNIbus Event Reader service retrieves these new inserted events and map them according to Table 16-8.

Table 16-8 Mapping of events

Situation Name	Alert Group	Policy Name
NT_Process_CPU_Critical	ITM_NT_Process	TSRM_Incident
ITSM_WASDBConPoolUseMax	ITM_DB_Connection_Pools	CCMDB_RFC
ITSM_WASAvgHeapSizeAfterGC	ITM_Garbage_Collection_Analysis	TPM_Workflow

4. After mapping process, Tivoli Netcool/Impact Event Processor service runs the mapped policies and returns events back to the Tivoli Netcool/OMNIbus Object Server.

The operator switches to the Active Event List view and then observes that open events are automatically acknowledged after a while, as shown in Figure 16-49.

ITMDisplayItem	Alert Group	CI	Action	TTNumber	TicketStatus	Ac
java	ITM_NT_Process	TI0003-SYS14	Incident ticket is created	1725	QUEUED	Ye
vmware-vmx	ITM_NT_Process	TI0003-SYS3	System is in maintenance mode, no action taken			Ye
c/TradeDataSou	ITM_DB_Connection_Pools	WAS60	RFC C1101 is created			Ye
	ITM_Garbage_Collection_Analysis		Provisioning workflow 1845 is succeeded			Ye

Figure 16-49 Processed Events in Tivoli Netcool/OMNIbus

The background events are as follows:

- First event triggers the incident ticket 1725. Most of the ticket fields are filled by data that is gathered from CCMDB as shown in Figure 16-50.

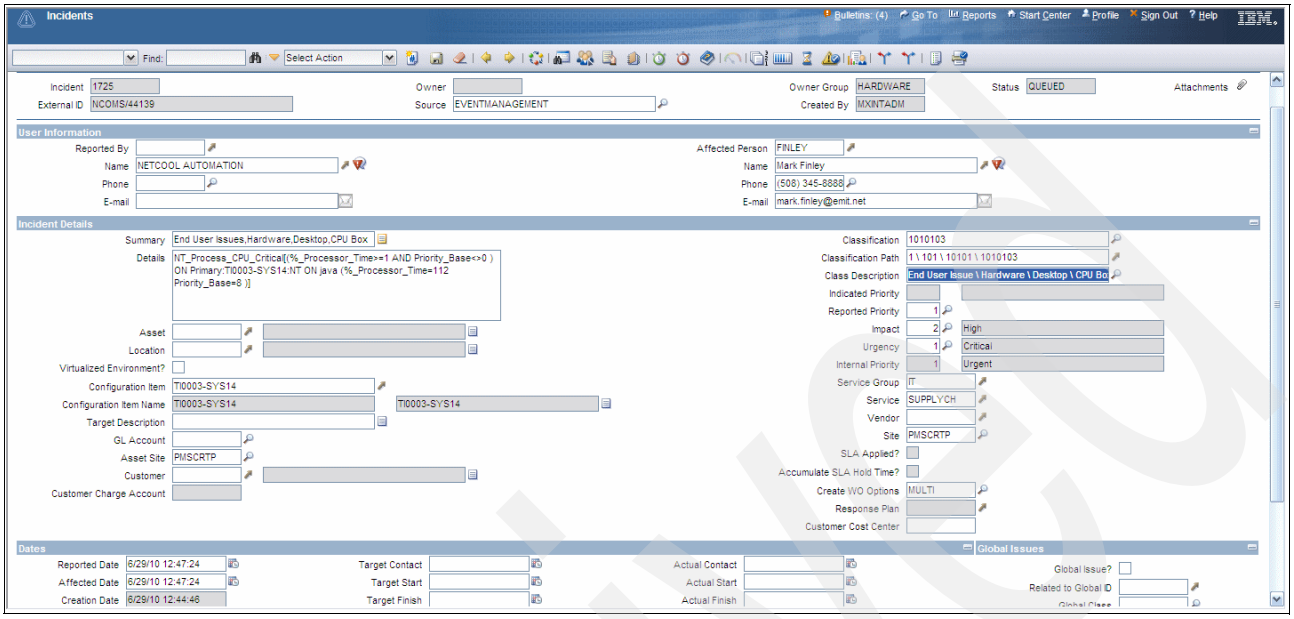


Figure 16-50 Automatic Incident Ticket

- Second event is acknowledged without action because TI0003-SYS3 system is in the maintenance window at the time that event occurs.
- Third event triggers the RFC C1101. Scheduled escalation runs the response plan actions and these actions populate RFC with predefined data. In the end, RFC 1101 is created, as shown in Figure 16-51.

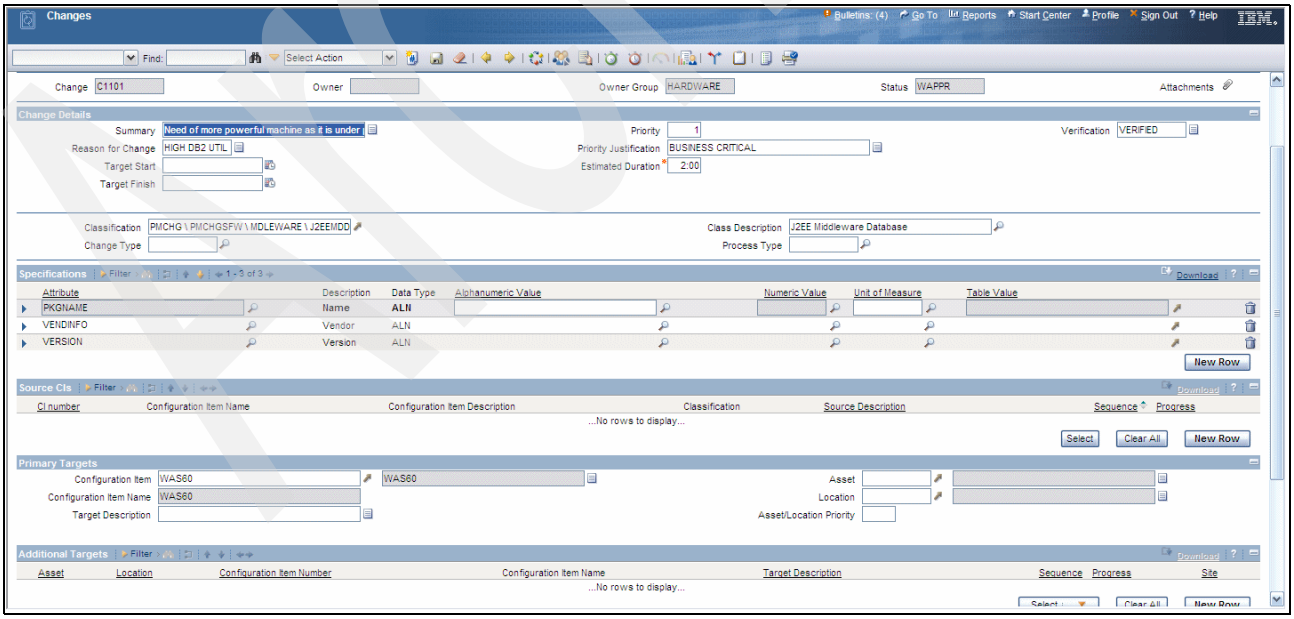


Figure 16-51 Automatic Request for Change

- Fourth event triggers the Tivoli Provisioning Manager workflow. Tivoli Netcool/Impact policy calls the workflow and then waits for one minute to gather the workflow status. If it is successfully completed, event is acknowledged.

The operator switches back to the Situation Event List in Tivoli Enterprise Portal and then observes that situations are acknowledged, as shown in Figure 16-52.

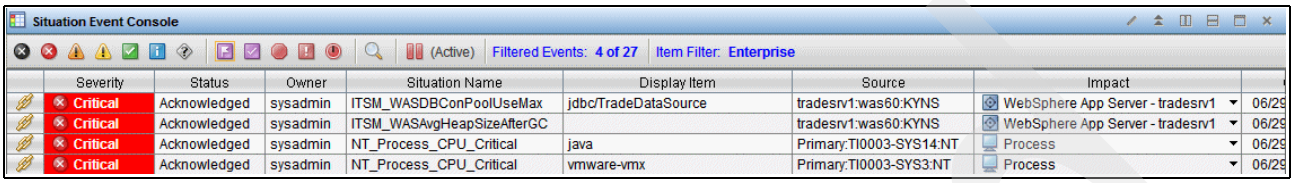


Figure 16-52 Acknowledged Situations in Tivoli Enterprise Portal

16.7 Summary

Figure 16-53 shows the quick summary of this integration.

Scenario Overview

- How IBM is making the event management more autonomous when a problem is detected by:
  - Creating incident ticket
  - Creating request for change
  - Running workflow

Products and Components Involved

- IBM Tivoli Netcool/OMNIbus V7.2.1
- IBM Tivoli Netcool/OMNIbus Probe for Tivoli EIF
- IBM Tivoli Netcool/OMNIbus Gateway for Tivoli Service Request Manager
- IBM Tivoli Netcool/Impact V5.1.1
- IBM Tivoli Monitoring V6.2.2 FP2
- IBM Tivoli Monitoring and Tivoli Event Synchronization
- IBM Tivoli Service Request Manager V 7.2.0.1
- IBM Tivoli Change and Configuration Management Database V7.2.0.1
- IBM Tivoli Provisioning Manager V7.2

Benefits

- Faster mean-time-to-resolution by event enrichment
- Better decision making by predefined policies and minimum human intervention
- More productive operational staff
- Better change management quality by predefined responses and change procedures
- Fully integrated solution for automated event management

High Level Implementation Steps

- Integrate Netcool/OMNIbus and IBM Tivoli Monitoring for event forwarding
- Integrate Netcool/OMNIbus and IBM Tivoli Service Request Manager for ticket forwarding
- Integrate Netcool/Impact and IBM Tivoli CCMDB for RFC creation
- Integrate Netcool/Impact and IBM Tivoli Provisioning Manager for workflow call
- Create Netcool/Impact policies for event enrichment, web service invocation and workflow call

Figure 16-53 Quick summary

Archive

17

Asset and CI integration

This chapter provides information to help you integrate assets and CIs. It has integration steps and describes why you would want to do this integration in your environment. It explains the benefits of integration and also has a case study of a fictional company.

This chapter does not cover installation details of these environments. These details are in the Tivoli products manuals and other IBM RedBooks publications.

This chapter contains the following topics:

- 17.1, “Differences between assets and CIs” on page 478
- 17.2, “Scenario overview and products involved” on page 479
- 17.3, “Benefits” on page 480
- 17.4, “Case study: Fictional IBM RedBooks Company A” on page 481
- 17.5, “Adding a Configuration Item field in the Asset application” on page 510
- 17.6, “Using a link rule to link assets to CIs” on page 515
- 17.7, “Summary” on page 543

© Copyright IBM Corp. 2011. All rights reserved.

17.1 Differences between assets and CIs

An *asset*, according to ITIL is *any resource or capability*, such as management, organization, process, knowledge, people, information, applications, infrastructure, and financial capital.

Configuration items (CI), according to the ITIL, are components of an infrastructure that currently are, or soon will be, *under the control of a configuration management system*. In other words, a CI is any component that must be managed to deliver an IT service.

CIs can be single modules, such as a monitor or tape drive, or more complex items, such as a complete software system. Typical groups of CIs are as follows:

- Models
- People
- Procedures
- Hardware
- Software
- Formal documentation (such as process documentation)

For example a server in production can be both an asset and a CI. A building is typically an asset, but software documentation usually defined as a CI, but not an asset.

Use assets in your environment in the following situations:

- If no requirement exists to implement configuration management, so you do not need to view CIs and their relationships
- If you do not need asset relationships in any of the process implementations (for example, change, incident, problem)

Use only CIs in your environment in the following situations:

- If you want to manage only the operational assets, not the entire life cycle
- If the license compliance is not needed

The term *configuration items* is used in the context of configuration management with an emphasis on how they are physically and logically related. The term *assets* is typically used in the context of financial management with an emphasis on financial attributes.

To understand the difference between assets and CIs we need to understand the life cycle of an asset.

ITIL definition of a life cycle: Life cycle refers to the various stages in the life of an IT service, configuration item, incident, problem, change, and so on. The life cycle defines the Categories for Status and the Status transitions that are permitted. For example:

- The life cycle of an application includes requirements, design, build, deploy, operate, optimize stages.
- The expanded incident life cycle includes detect, respond, diagnose, repair, recover, restore.

The life cycle of a server can include ordered, received, in test, live, disposed, and so on.

Assets have a life cycle that spans procurement and development through operations and retirement. After an *asset is in operation within the IT infrastructure*, it may also be considered as a configuration item.

Archived

For instance, software can be requested and procured. At that stage, it is considered an asset. Later when is becomes operational within the IT infrastructure it will also be considered as a CI.

Figure 17-1 explains the intersection of a CI in the asset life cycle.

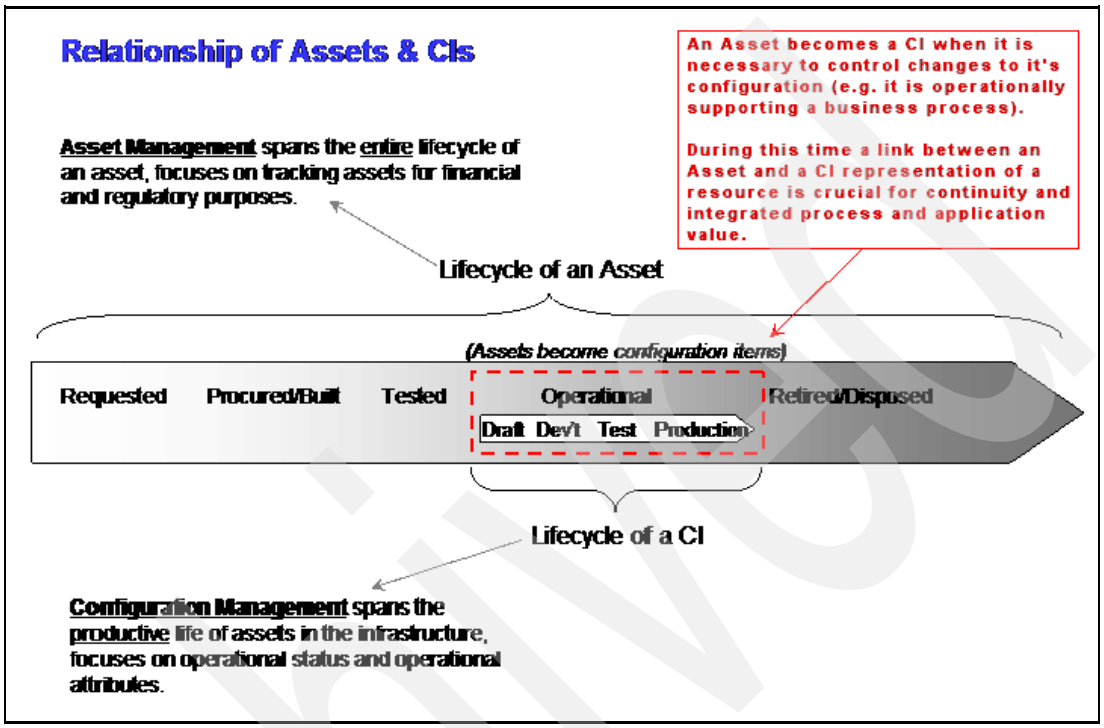


Figure 17-1 Life cycle of an asset and of a CI

After the useful lifetime, this CI is removed from operational status and becomes part of a set of spares or possibly retired and disposed. *Thus, an asset has a longer life cycle than a CI.*

17.2 Scenario overview and products involved

To make this integration, you must have at least the following products and product components (Figure 17-2 on page 480):

- IBM Tivoli Change and Configuration Management Database V7.2.0.1
- IBM Tivoli Asset Management for IT V7.2.0
- IBM Maximo Base Services V7.1.1.6
- IBM Tivoli Application Dependency Discovery V7.2.0

We can create defined rules to integrate the asset you discovered in your environment with the asset you already have in the environment.

We use Tivoli Application Dependency Discovery Manager (TADDM) as a product for automatic discovery but you can use other product if you want to.

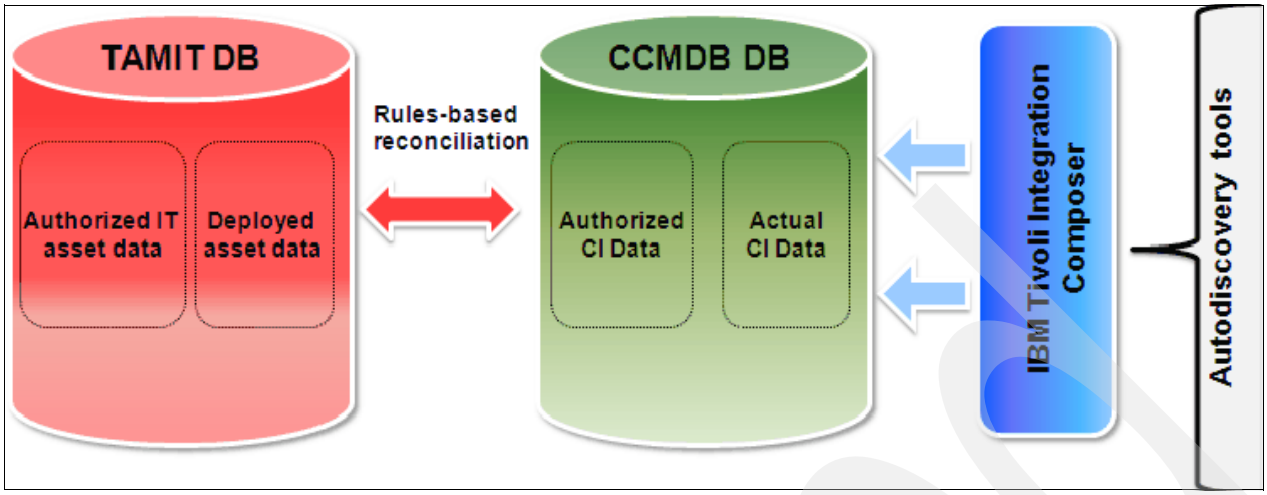


Figure 17-2 Integration overview

17.3 Benefits

CCMDB and IBM Tivoli Asset Manager for IT are complementary products that can be combined to give an organization the most streamlined, cost-effective, and efficient infrastructure that can be deployed to achieve competitive advantage.

Basically, the integration can help ease the work of all employees of your company because, when an asset and a CI are related, fewer fields must be filled, therefore you can improve the environment control because your environment also becomes more consistent.

17.3.1 The challenge of controlling assets and CIs

Many companies make big investments in physical assets and other resources in all their sites. Globalization demands that investments be competitive with other companies and always deliver a good service and product to customers.

However, at the same time that investment is important, economy is important and there is no way to economize if you do not have a technology to manage these thousand of assets with a more meticulous, granular view, and with attention to every detail in the company.

All investment decisions are affected by economic changes, whether or not those decisions have anything to do with buying and selling. If you have control of your environment, knowing where you can invest and which area needs more attention becomes easier.

Change and Configuration Management Database (CCMDB) and IBM Tivoli Asset Management for IT working together have the mechanism to improve the comprehensive management of assets from end to end.

17.3.2 Integration advantages

After implementing the asset and CIs you can more easily manage certain operational aspects of your environment.

With assets, you can create the following items:

- Incidents, problems, service requests, change, and release
- Comparisons of discovered and managed attributes by using assets and deployed assets
- Service level agreements (SLAs)
- Auditing of discovered and managed attributes

CIs provide the following features:

- A vast number of CI types and their relationships can be discovered by TADDM and imported into CCMDB 7.2. This data provides support for impact analysis that can aid in change management, incident management, release management, and problem management processes.
- Configuration management can be used to reconcile CIs and actual CIs. A customer can define which CIs and attributes are to be reconciled. Automated tasks can be scheduled for these audits and reports can be generated.
- CIs can be used for configuration management and operational management, but not for financial management.
- The change and release process can provide automation of implementation tasks, for example, software distribution tasks. Infrastructure services are provided so that customers can develop their own automation.
- Change management provides impact analysis.
- Incident and problem management processes provide launch-in-context to TADDM, which helps in resolution.
- The configuration management process helps audit and control updates to CIs.

When you have an asset and CI integration, you can have all these features and functions, and you will be implementing a complete service management solution.

An advantage of this integration is that all the relationships of CIs are mapped, and therefore, the CCMDB is an invaluable basis for service management processes such as incident, problem, and change management.

If a user raises an incident ticket about an outage on a CI, the CCMDB can provide easy identification of other applications and appliances affected, which services are affected, which other users are affected, and more. This advantage clearly assists in faster incident resolution, clear communication to users and proactive problem anticipation in the services provided.

17.4 Case study: Fictional IBM RedBooks Company A

“Fictional IBM RedBooks Company A” is a multinational corporation headquartered in Sao Paulo, Brazil. The company employs 500 people in its operations. The IT department manages the operating systems of each of the 500 computers that are currently being used by the employees for data entry. The company also has more than 30000 assets from various customers.

IBM RedBooks Company A has offices in Brazil, United States, Canada, and Europe.

Note: All names and references to company and other business institutions used in this chapter are purely fictional. Any match with a real company or institution is coincidental.

Archive

Figure 17-3 shows the management environment in IBM RedBooks Company A. We use this environment to integrate the assets and configuration items.

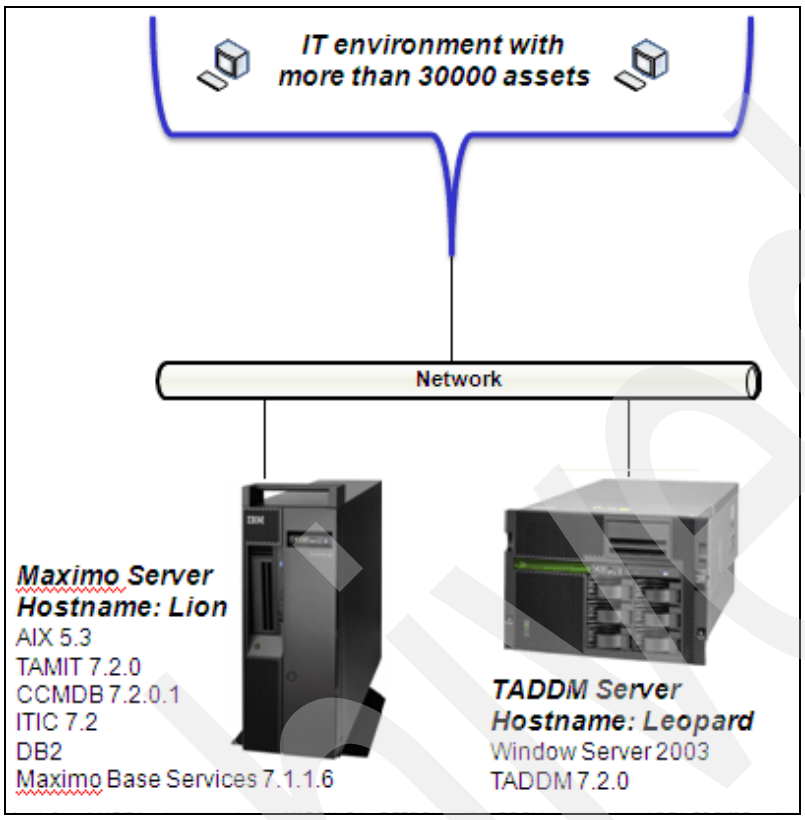


Figure 17-3 IBM RedBooks Company A topology

In this integration, the first thing is to configure is the link rules, so that we can import the CIs into the Tivoli Process Automation Engine (Maximo) Server.

17.4.1 Linking assets to CIs using GUID

This scenario for linking CIs to assets uses the Globally Unique Identifier (GUID), which is generated for both deployed asset data and actual CI data that is imported into Asset Management for IT. We have to use the GUID because at IBM RedBooks Company A, the CIs are imported using TADDM.

When discovered data is imported from TADDM into CCMDB through Integration Composer, a GUID is assigned to the actual CI records. When discovered data is imported from TADDM into Asset Management for IT through Integration Composer, a GUID is assigned to the deployed asset records.

This scenario automates the link between CIs and assets based on the GUIDs of the associated actual CIs and deployed assets, respectively.

Linking CIs to assets with a GUID involves the following steps:

1. Configuring the reconciliation, which includes the following tasks:
   a. Creating a link rule.
   b. Creating a reconciliation task.

c. Specifying the relationship between the actual CI and the deployed asset.

d. Configuring the escalation that will add the asset number of the asset to the CI.

2. Configuring the escalation, which includes the following tasks:

a. Creating the action that specifies how to add the asset number.

b. Creating the escalation that schedules and performs the action.

Configuring the reconciliation for linking CIs to assets with a GUID

To configure the reconciliation for linking assets to deployed assets, create a link rule and a reconciliation task.

Use the Reconciliation module applications in Tivoli Asset Management for IT to evaluate data about information technology devices and networks and view the results of the evaluation. You use these applications to configure a background process that reconciles objects from one data set (Data Set 1) with objects in another data set (Data Set 2).

The reconciliation process identifies successful matches and discrepancies and variances between the two sets of data. You can use these applications to reconcile data about assets or data about configuration items.

Perform the following steps:

1. On the navigation bar in Asset Management for IT, click **Go To** → **Administration** → **Reconciliation** → **Link Rules**. See Figure 17-4.

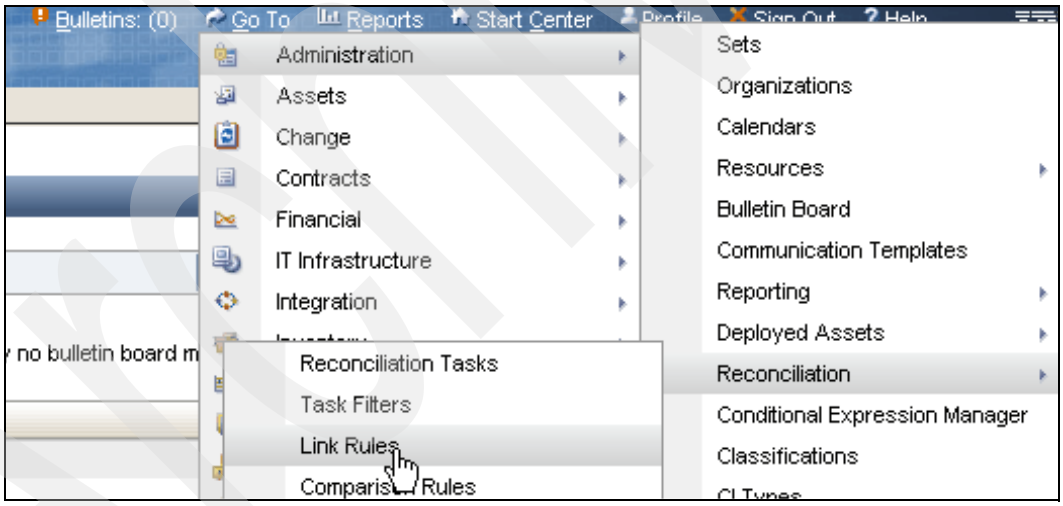


Figure 17-4 Accessing Link Rules application

2. On the toolbar in the Link Rules application, click **New Link Rule**. See Figure 17-5 on page 484.

Archived

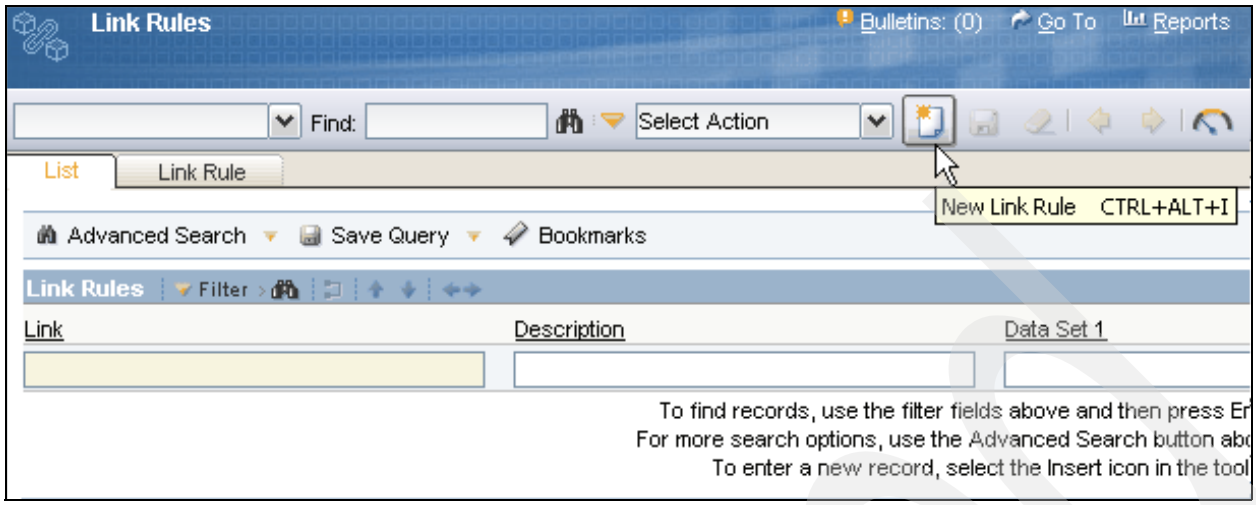


Figure 17-5 New Link Rule

3. In the Link field, type the name that you want to assign to the link rule, for example `ASSET2DPA_SERIALNUM`. This name must be unique.
4. In the Description field, type a brief description of the link rule.
5. In the Type of Reconciliation table window, in the Data Set 1 field, click **Select Value** and select **ASSET**.
6. In the Data Set 2 field, click **Select Value** and select **DEPLOYEDASSET**. See Figure 17-6.

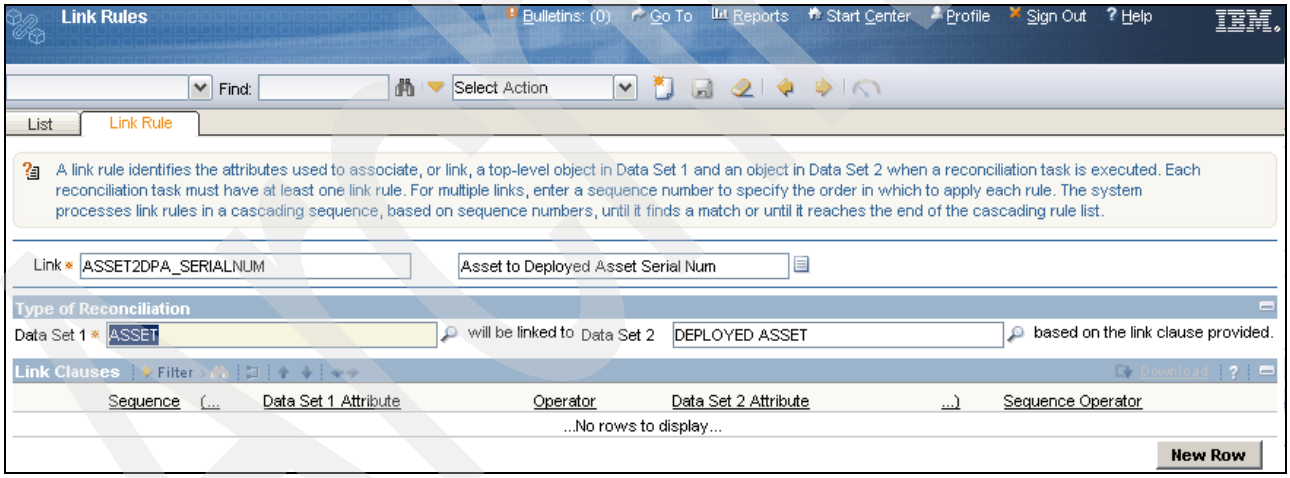


Figure 17-6 Link Rules window

7. To add a link rule clause, in the Link Clauses table window, click **New Row**. See Figure 17-7 on page 485.

> **Note:** The application generates a sequence number for each new row that is created. By default, the application creates sequence numbers in increments of ten in ascending order. You may modify the numbers as needed or add new clauses with sequence numbers that fall between already-existing numbers. When you save the record, the application sorts the clauses in ascending numerical order.

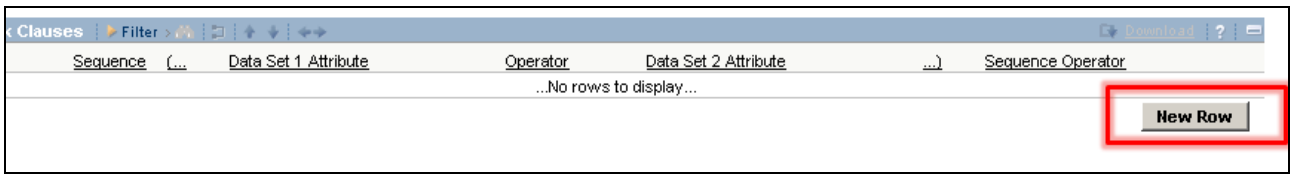


Figure 17-7 Link Clauses

8. If appropriate, in the **(...** field (open parenthesis), enter an open parenthesis character.
9. In the Data Set 1 Object field, click **Select Value** and select **ASSET**. The value ASSET populates the Data Set 1 Object field. See Figure 17-8.

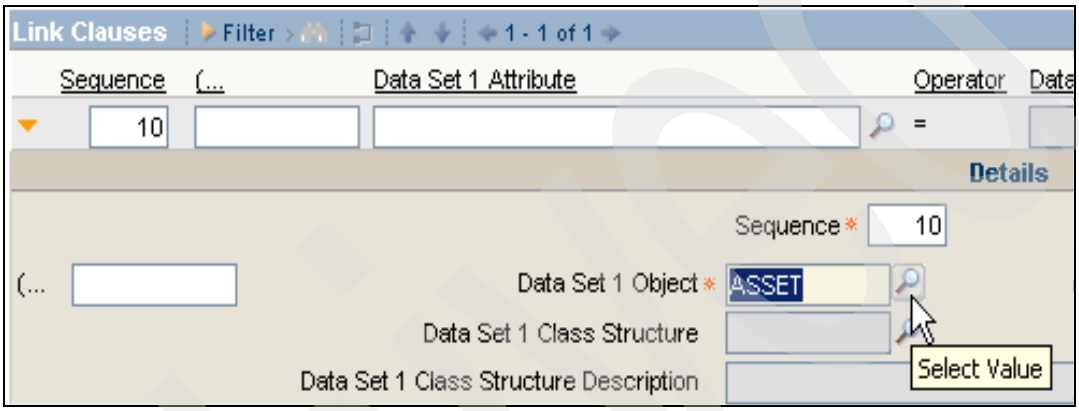


Figure 17-8 Data Set 1 Configuration

10. In the Data Set 1 Attribute field, click **Select Value** and select the attribute that is appropriate for the asset. The value selected populates the Data Set 1 Attribute field, and the application displays the title of the attribute in the Data Set 1 Attribute Title field.
11. In the Data Set 2 Object field, click **Select Value** and **DEPLOYEDASSET** from the Select Value dialog box. The DEPLOYEDASSET value populates the Data Set 2 Object field.
12. In the Data Set 2 Attribute field, click **Select Value** and select the deployed asset attribute that corresponds to the attribute you selected for the ASSET. The value selected populates the Data Set 2 Attribute field, and the application displays the title of the attribute in the Data Set 2 Attribute Title field.

13. If appropriate, in the **...)** field (close parenthesis), enter a close parenthesis character. See Figure 17-9.

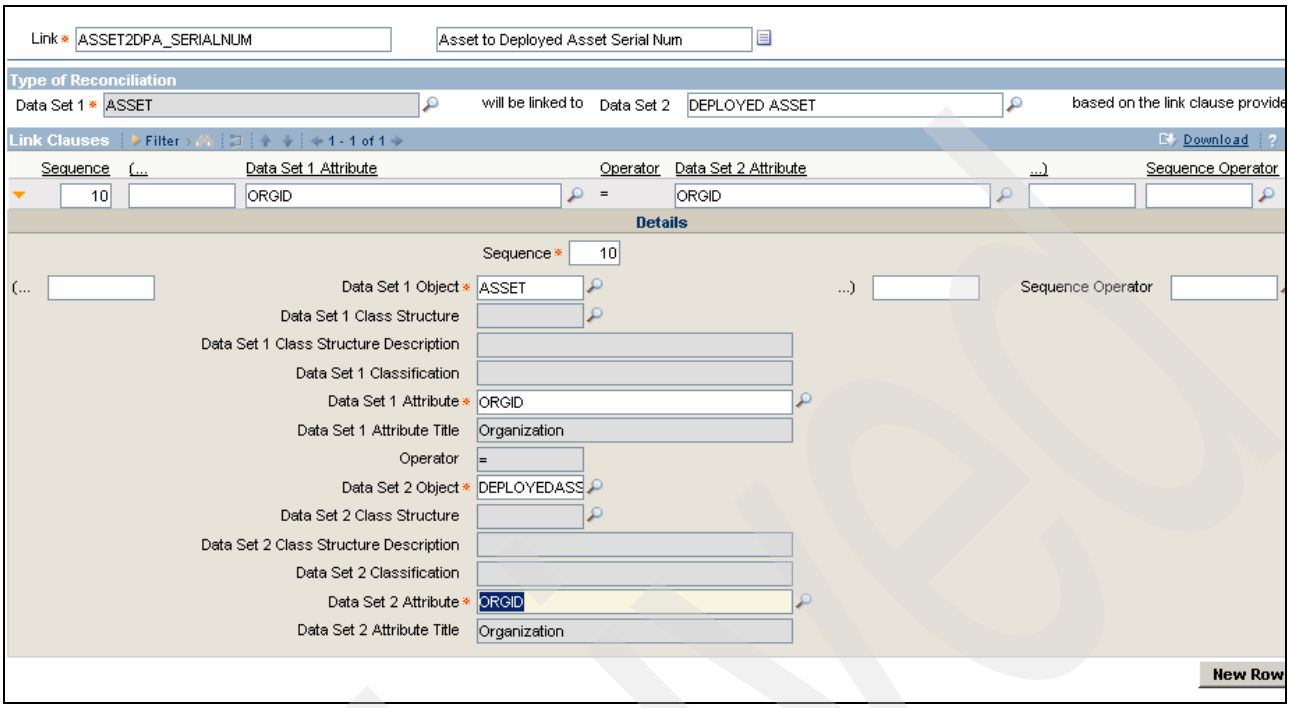


Figure 17-9 Link Clauses configuration

14. If you want to add more link rule clauses to the link rule, perform the following steps:

    a. In the Sequence Operator field, click **Select Value**. A Select Value dialog box opens (Figure 17-10 on page 487).

    b. In the dialog box, click to select the appropriate sequence operator. The Select Value dialog box closes, and the selected operator populates the Sequence Operator field.

    When more than one link rule clause exists, the sequence operator is required. It specifies how the current clause relates to the next clause in the sequence.

Archive

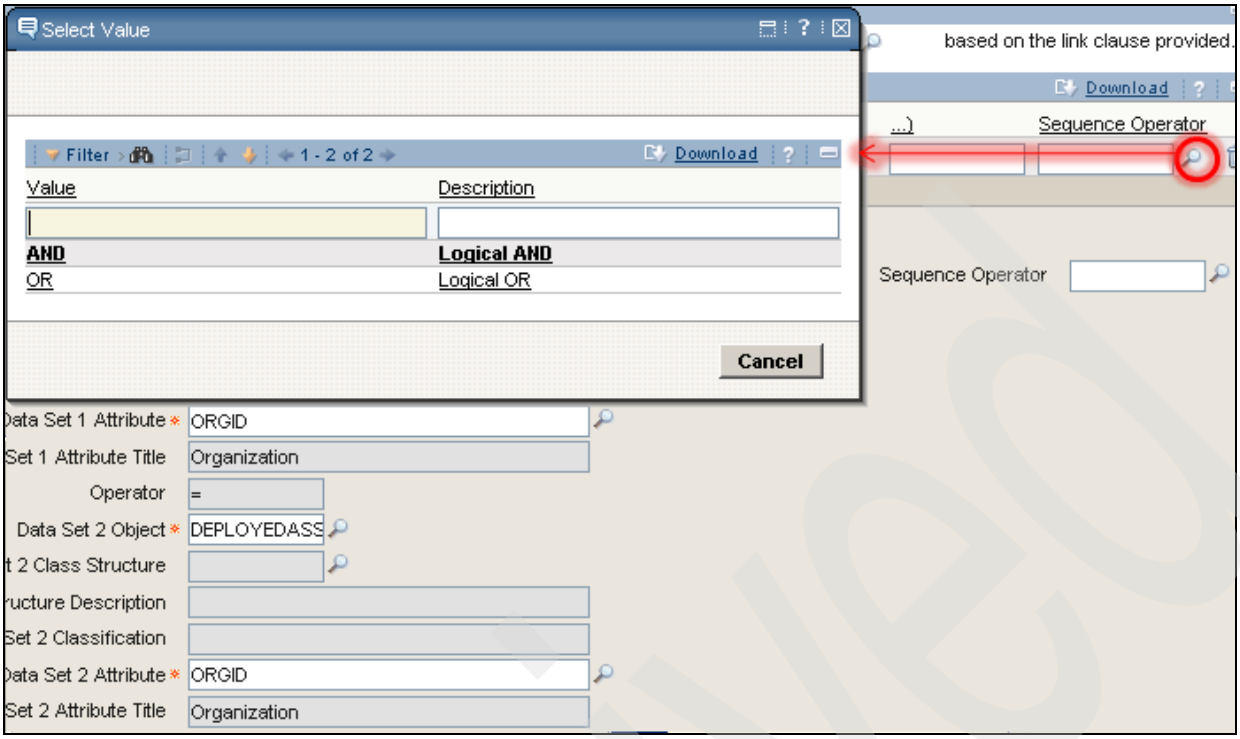


Figure 17-10 Adding a new condition

15. If appropriate, create additional clauses to define the link rule for CI-to-asset linking.
16. Click **Save** to save the Link Rule. See Figure 17-11.



Figure 17-11 Saving the Link Rule

Creating the reconciliation task

The reconciliation task combines the elements required for the reconciliation into a specific task that you can execute. A task consists of three possible components:

- A task filter (optional)
- One or more link rules (required)
- One or more comparison rules (optional)

For the CI-to-asset linking, only a link rule is required.

Archive

In the reconciliation task, you also define the schedule for running the reconciliation, and you activate the schedule:

1. On the navigation bar in Asset Management for IT, click **Go To** → **Administration** → **Reconciliation** → **Reconciliation Tasks**. See Figure 17-12.

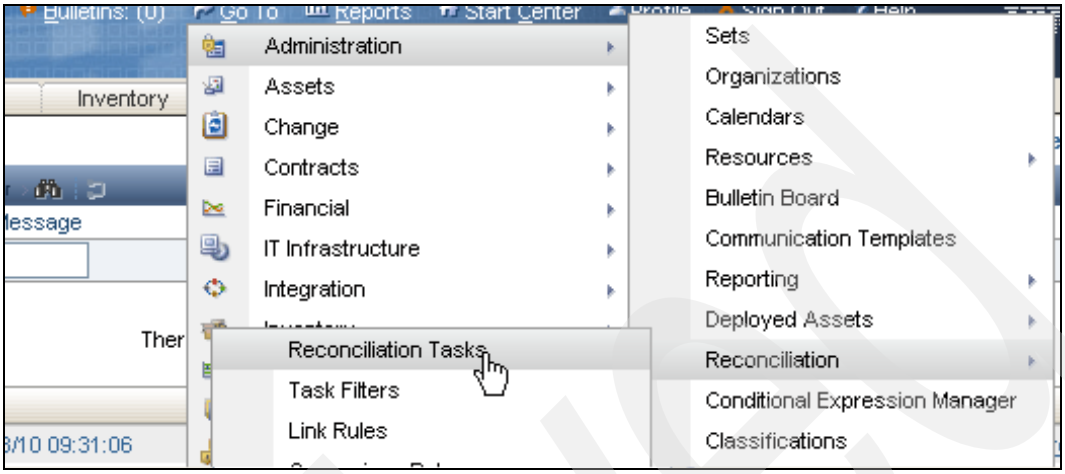


Figure 17-12 Reconciliation Task application

2. On the toolbar in the Reconciliation Tasks application, click **New Reconciliation Task** icon. See Figure 17-13.

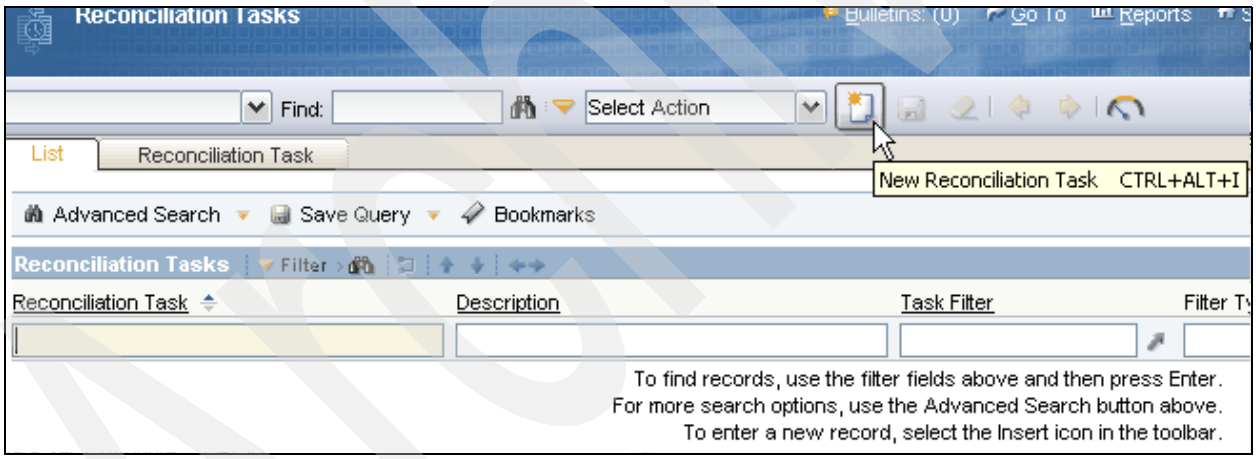


Figure 17-13 New Reconciliation Task

3. In the Reconciliation Task field, type the name that you want to assign to the reconciliation task, for example `ASSET2DPA_RECONTASK`. This name must be unique.
4. In the Description field, type a brief description of the link rule.

Archived

5. In the Type of Reconciliation table window, in the Data Set 1 field, click **Select Value** and select **ASSET**. See Figure 17-14.

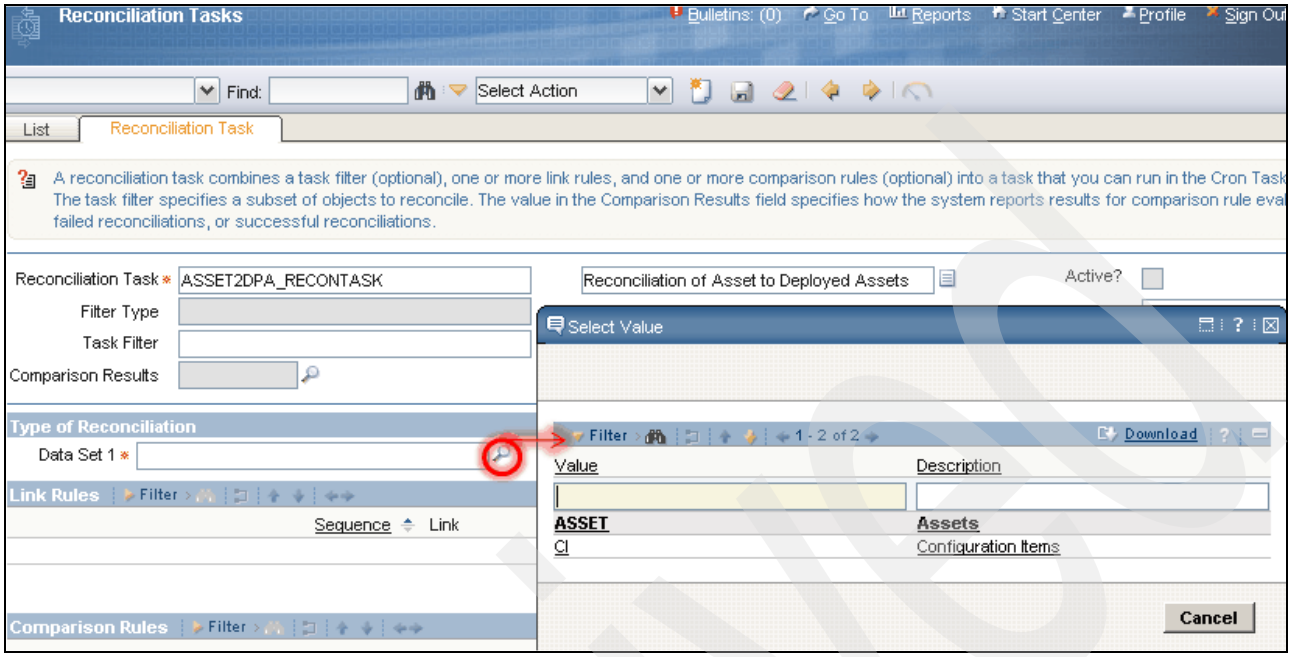


Figure 17-14 Setting the Data Set

6. In the Data Set 2 field, click **Select Value** and select **DEPLOYEDASSET**.
7. In the Link Rules table window, click **Select Link Rule** and, from the Select Link Rule window, select the link rule that you created for linking assets to deployed assets. See Figure 17-15.

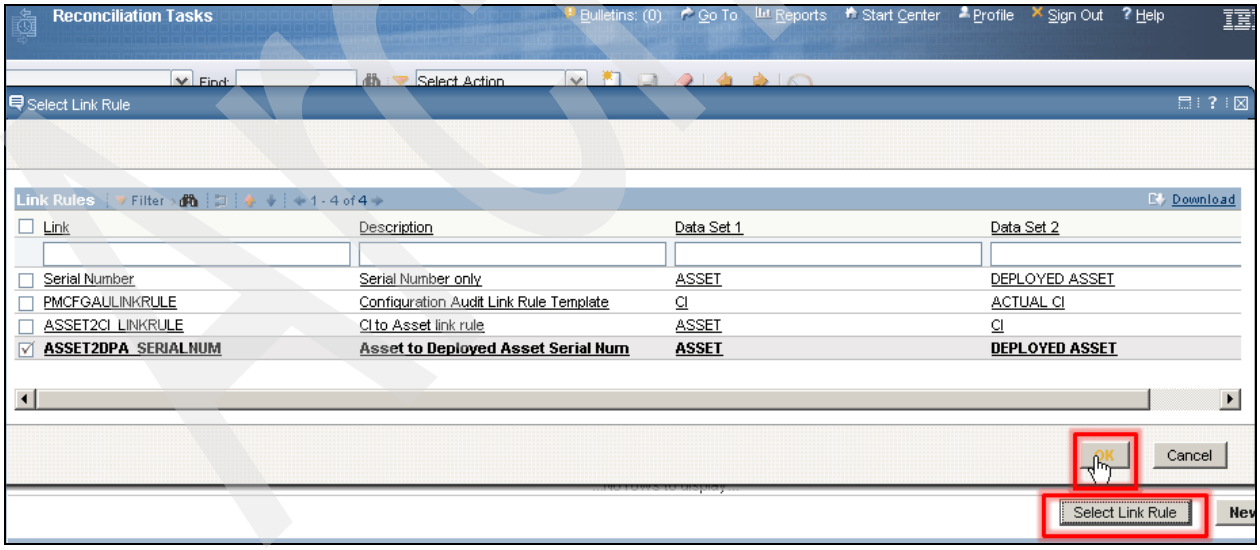


Figure 17-15 Selecting a Link Rule

8. To schedule execution of the reconciliation task, complete the following steps:
   a. In the Schedule field, click **Set Schedule**.
   b. In the Select Value window, set the schedule for executing the reconciliation task, and click **OK**. See Figure 17-16.

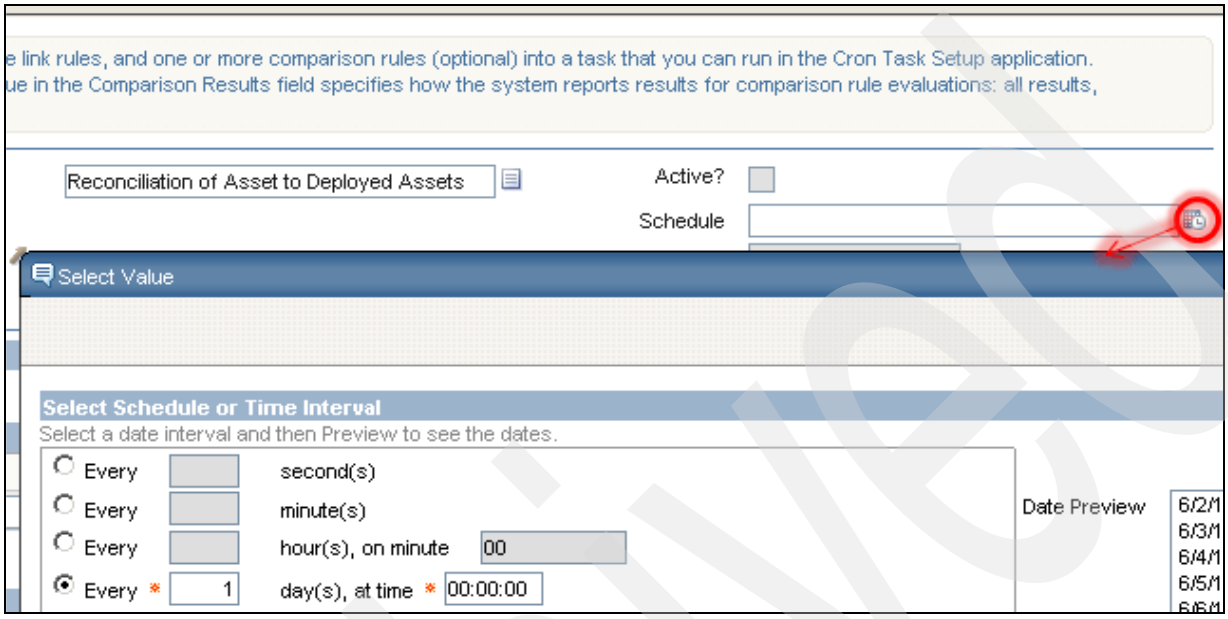


Figure 17-16 Set the Schedule

   c. Save the record.
   d. Click **Save Reconciliation Task**.
9. To activate your Reconciliation Task, perform the following steps:
   a. Click **Go To** → **System Configuration** → **Cron Task Setup**, and then look for ReconciliationCronTask.
   b. In the Cron Task Instance find the appropriate name for the Reconciliation Task you have just created. If it is not Active, select the check box (to set it to Active). See Figure 17-17 on page 491.

Archived

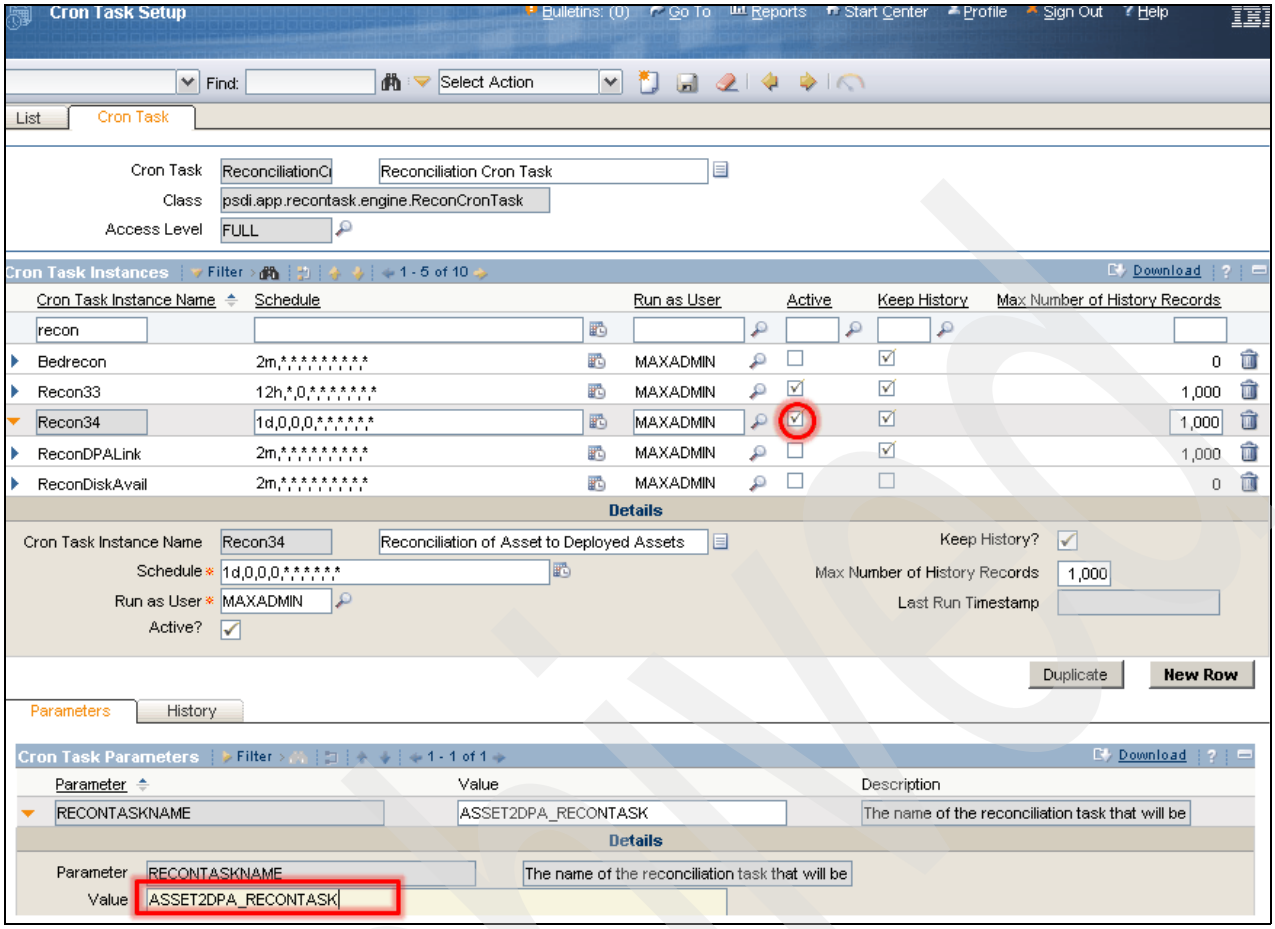


Figure 17-17 Checking the Cron Task Instance

c. Click **Select Action** → **Reload Request**. See Figure 17-18.

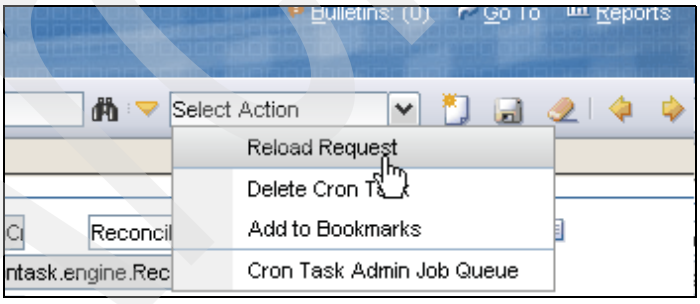


Figure 17-18 Reload Request window

d. Select the appropriate Cron Task Instance Name and click **OK**. See Figure 17-19 on page 492.

Archive

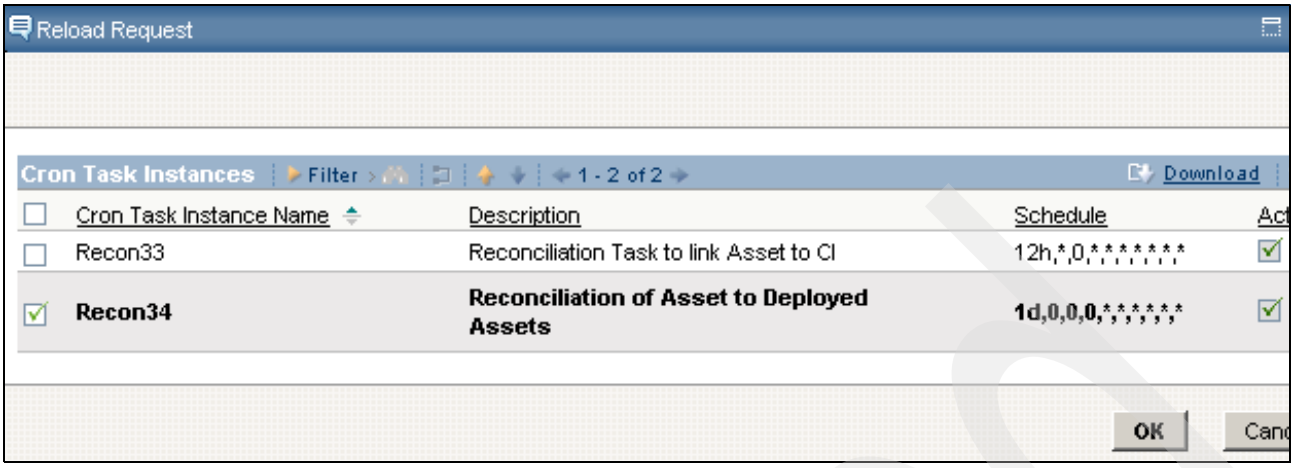


Figure 17-19 Click OK to Reload the Request

Specifying the actual CI to deployed asset relationship

This task creates a relationship between the actual CIs and deployed assets in the database. A relationship between actual CIs and deployed assets is required when you create the action to use in the escalation that links CIs to assets. This relationship is based on the GUID from both objects.

Perform the following steps:

1. On the navigation bar in Asset Management for IT, click **Go To** → **System Configuration** → **Platform Configuration** → **Database Configuration**. See Figure 17-20.

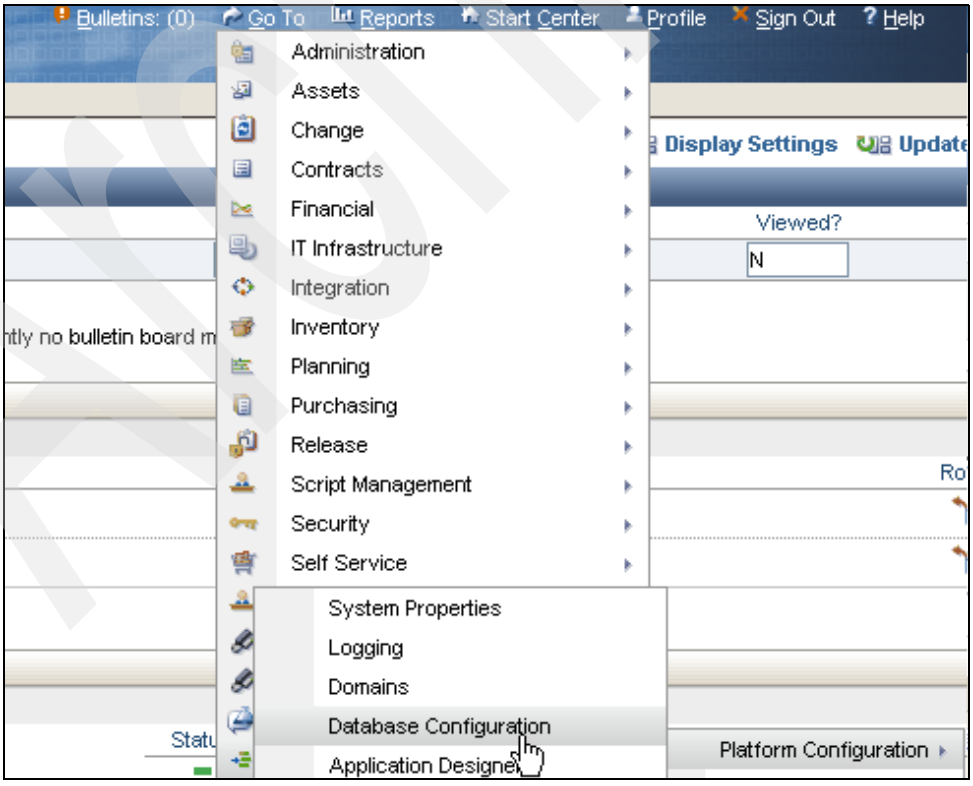


Figure 17-20 Acessing the Database Configuration application

2. In the Objects table window on the List tab, select **ACTCI** (Actual CI Table). See Figure 17-21.

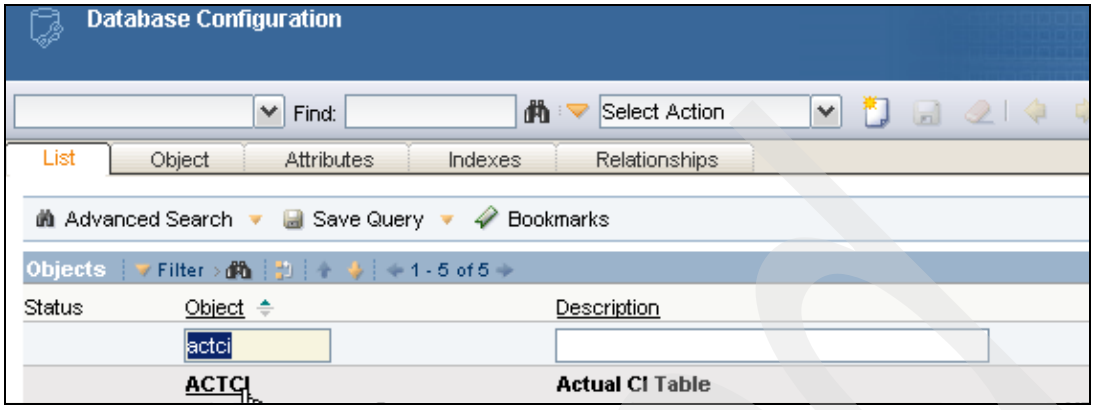


Figure 17-21 Object list

3. On the Relationships tab, click **New Row**. See Figure 17-22.

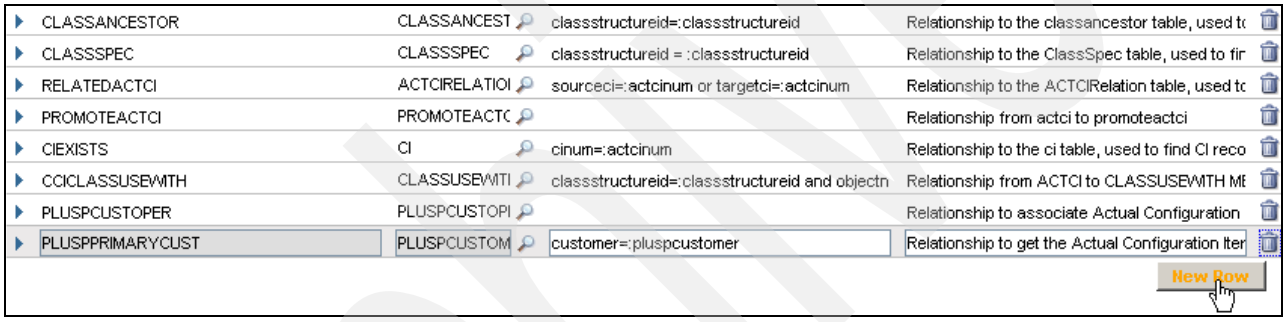


Figure 17-22 Database Configuration Relationships window

4. In the Relationship field, specify a relationship name, for example DPA_GUID.
5. In the Child Object field, click **Select Value** and specify DEPLOYEDASSET.
6. In the Where Clause field, specify the following clause (and see Figure 17-23):

```
guid = :guid
```

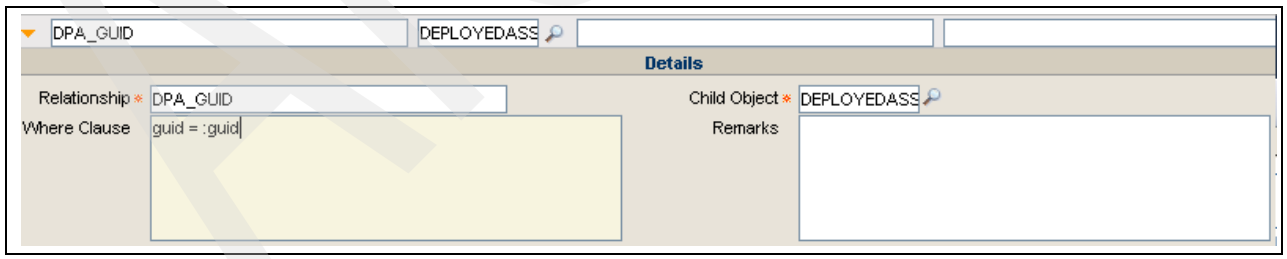


Figure 17-23 Where Clause in Database Configuration

7. Optional: In the Remarks field, enter additional remarks about the relationship.
8. Click **Save Object.**

Configuring the escalation to link CIs with IT assets

To configure the escalation for linking CIs to assets, create an action in the Actions application and an escalation in the Escalations application. You use the Escalations application to automatically configure a response to events.

For example, you can create an escalation to change the status of a record in the receiving application if the item received has passed inspection.

For CI-to-asset links, create an escalation that links CIs to assets if the appropriate conditions are met:

1. On the navigation bar in Asset Management for IT, click **Go To** → **System Configuration** → **Platform Configuration** → **Actions**. See Figure 17-24.

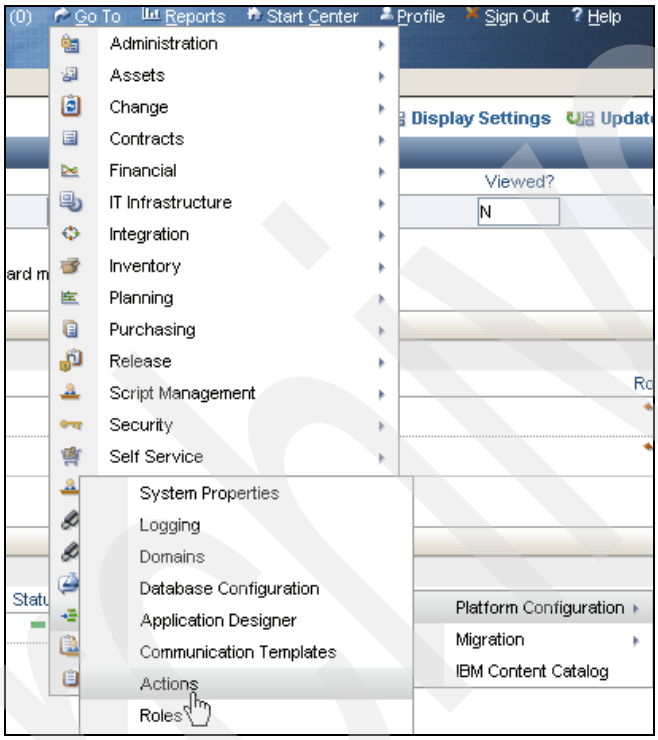


Figure 17-24 Actions application

2. On the toolbar in the Actions application, click **New Action**.
3. In the Action field, specify a name for the action. If your enterprise automatically numbers the actions, specify text that identifies the type of action in the Description field, for example `LINK_CI2ASSET_GUID`.
4. In the Object field, click **Select Value** and specify **CI** (Configuration Item table).

5. In the Type field, select **Set Value**. See Figure 17-25.

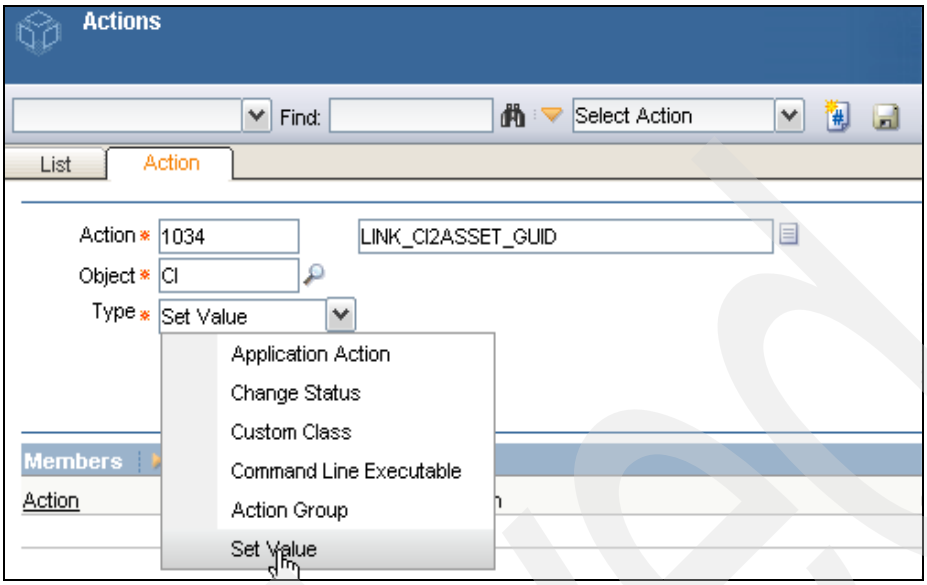


Figure 17-25 Actions application

6. In the Value field, specify the following value:

   `:actualci.<relationship_name>.reconlink.assetnum`

   The *<relationship_name>* is the name of the relationship that you defined in step 2 of “Specifying the actual CI to deployed asset relationship” on page 492.

Archived

7. In the Parameter/Attribute field, click **Detail Menu** and select **ASSETNUM** from the Select Fields window. See Figure 17-26.

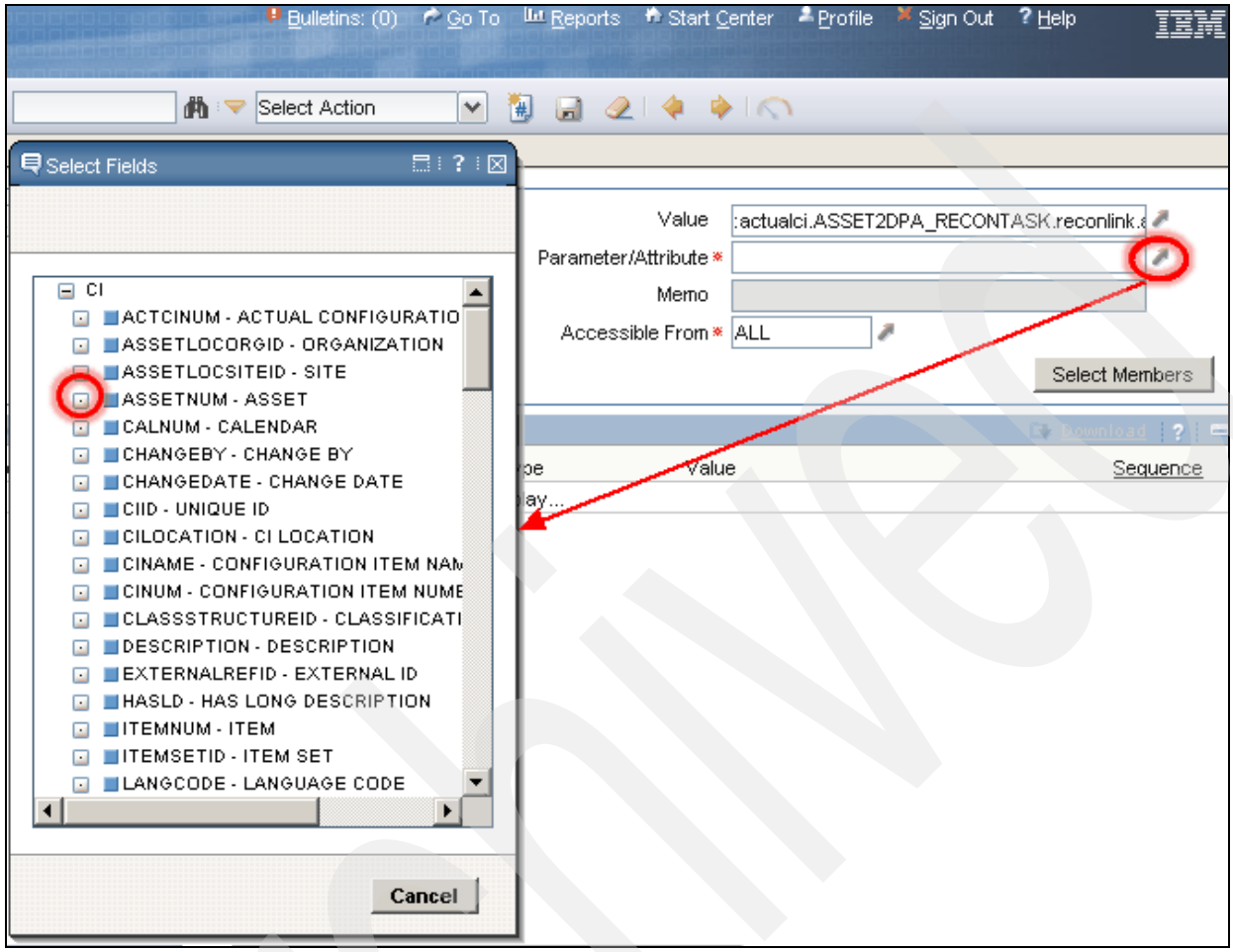


Figure 17-26 Selecting Parameter / Attribute

8. Click **Save Action**.

Archive

Configuring the escalation for the CI-to-Asset link

Complete the following steps to create an escalation for the CI-to-Asset link process:

1. On the navigation bar in Asset Management for IT, click **Go To** → **System Configuration** → **Platform Configuration** → **Escalations**. See Figure 17-27.

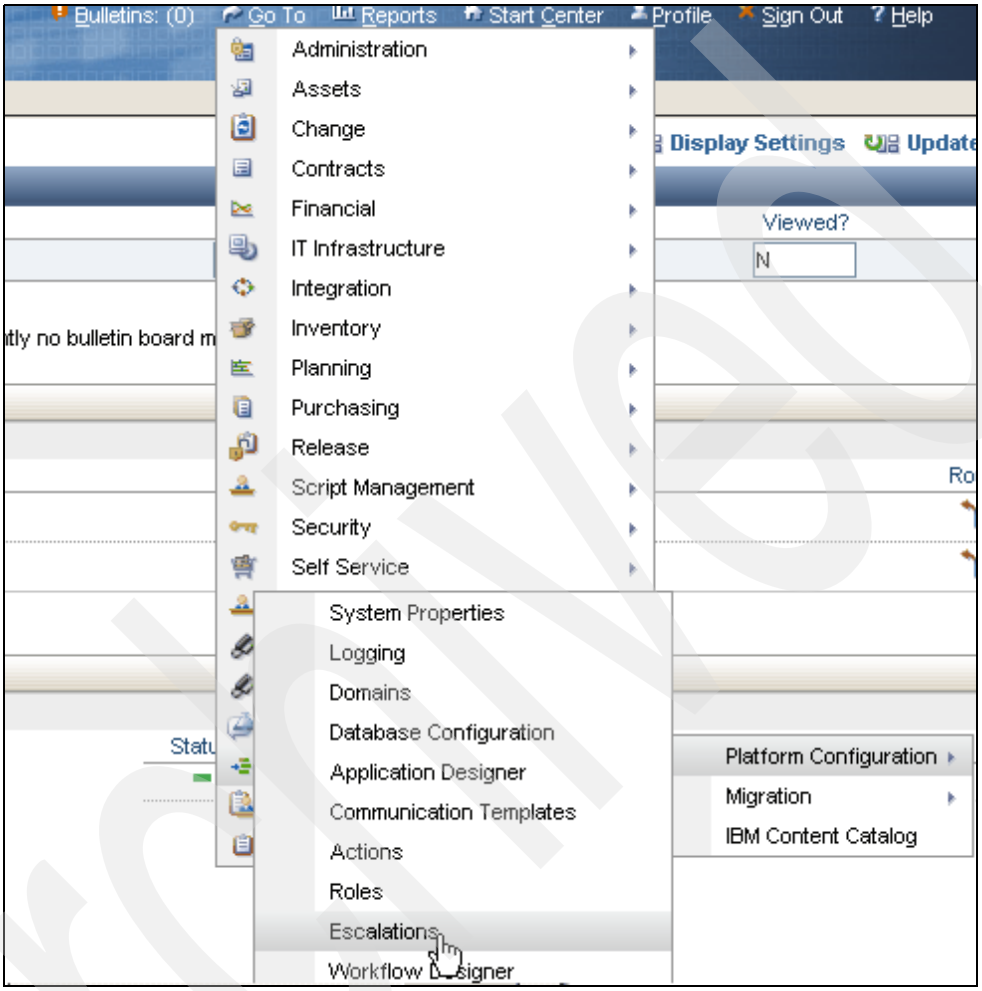


Figure 17-27 Escalation Application

2. On the toolbar in the Escalations application, click **New Escalation**.
3. In the Escalation field, specify a name for the escalation. If your enterprise automatically numbers escalations, specify text that identifies the escalation in the Description field, for example CI2ASSET_GUID.
4. In the Applies To field, click **Select Value** and specify CI (Configuration Item Table).
5. In the Condition field, specify `assetnum is null`.

Archive

6. In the Schedule field, click **Set Schedule** and specify a schedule for running the escalation. See Figure 17-28.

> **Note:** Set the escalation schedule to an interval greater than the reconciliation task schedule.

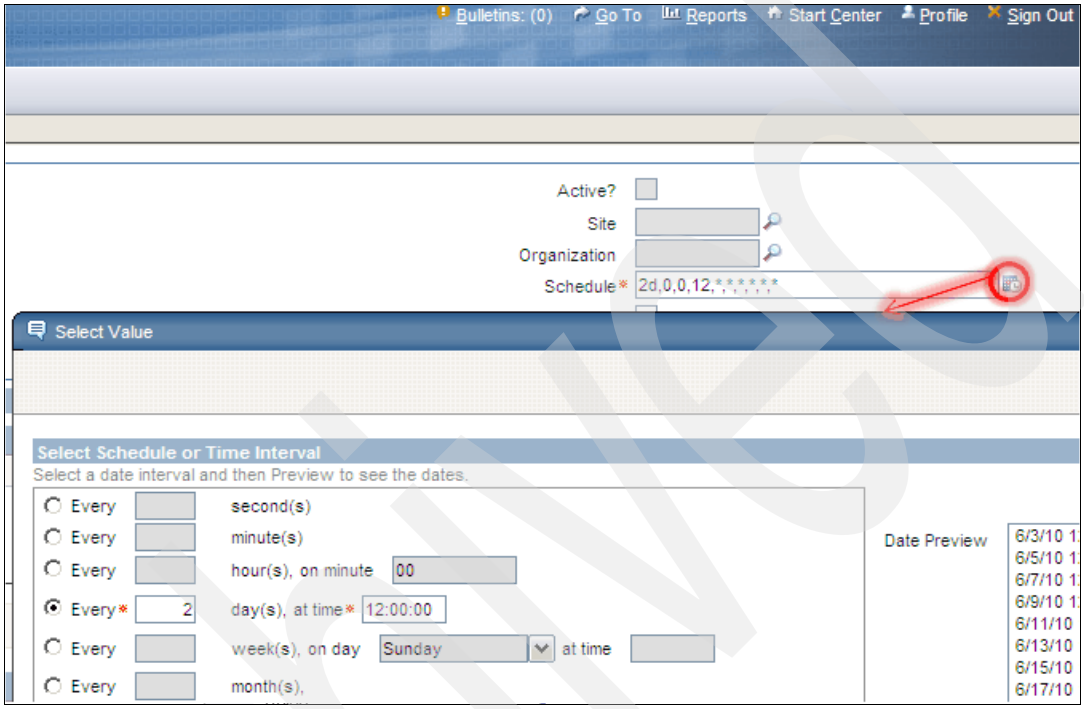


Figure 17-28 Configuring the Escalation

7. In the Escalation Points section, click **New Row**.
8. In the Escalation Point Condition field, specify an escalation point condition, for example:

   ```
   1 = 1
   ```

9. Select the **Repeat** check box. See Figure 17-29.

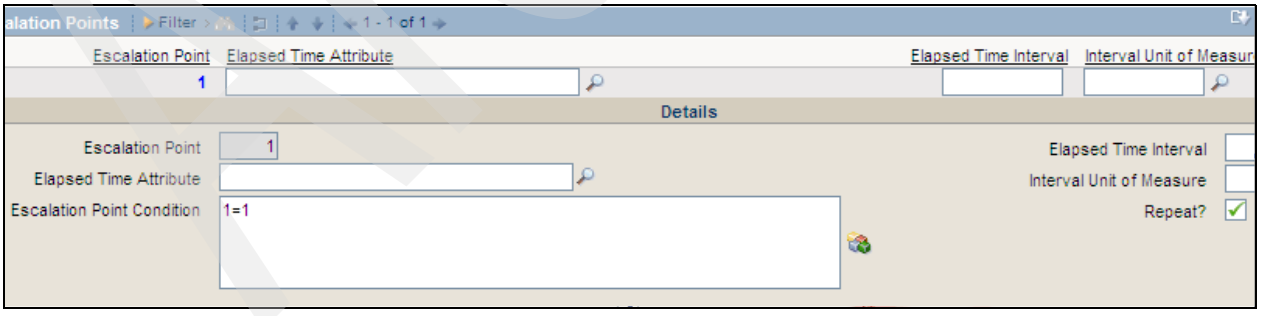


Figure 17-29 Escalation Points window

10. On the Actions subtab, click **New Row**.

11. In the Action field, click **Select Value** and select the name of the action that you previously created for this escalation. See Figure 17-30.

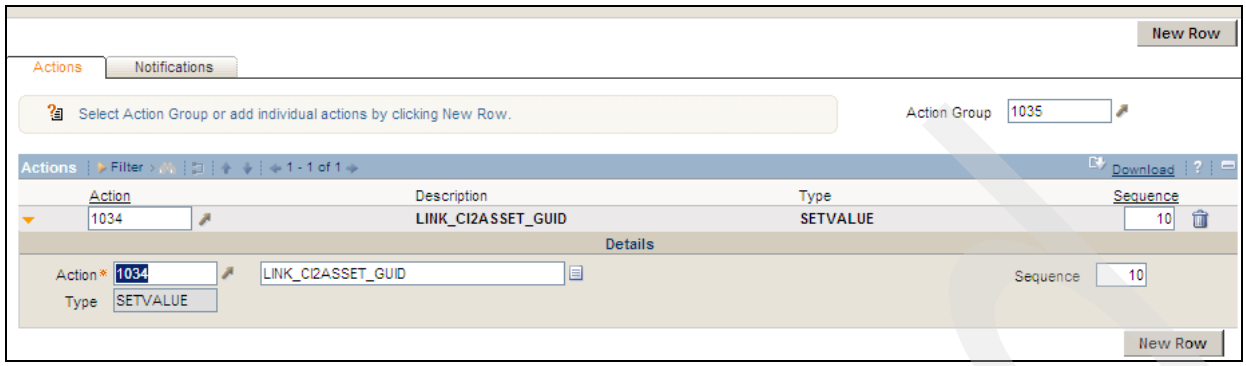


Figure 17-30 Actions tab

12. Click **Save Escalation**.
13. To activate the escalation, click **Select Action** → **Activate/Deactivate Escalation**. See Figure 17-31.

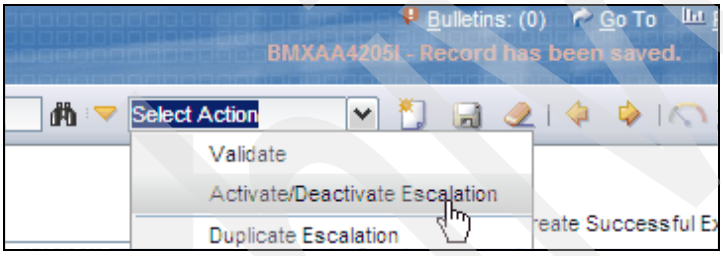


Figure 17-31 Activate the Escalation

17.4.2 Importing CIs to the CCMDB using Tivoli Integration Composer

Perform the following steps:

1. Create a CI type schema in the database:
   a. Start the Tivoli Integration Composer application. For example, in the following folder, run **`startFusion.bat`** (for Windows) or **`startFusion.sh`** (for UNIX-based systems):

      `<ITIC_root_dir>\bin`

   b. Log on to the Tivoli Integration Composer.
2. Create a TADDM data source (see Figure 17-32 on page 500):
   a. Start the Tivoli Integration Composer application. For example, in the following folder, run **`startFusion.bat`** (for Windows) or **`startFusion.sh`** (for UNIX-based systems):

      `<ITIC_root_dir>\bin`

Archive

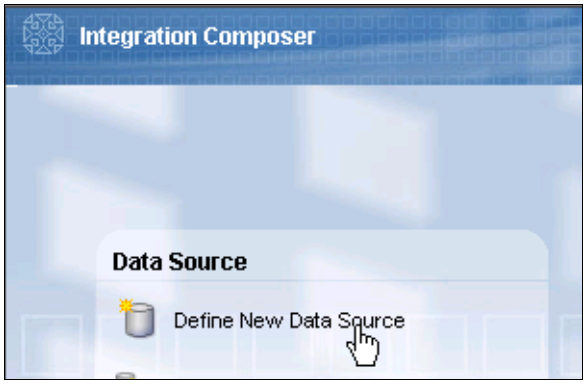


Figure 17-32 Tivoli Integration Composer menu

b. Click **Define New Data Source**.

c. Select **TADDM72CIType**. See Figure 17-33.

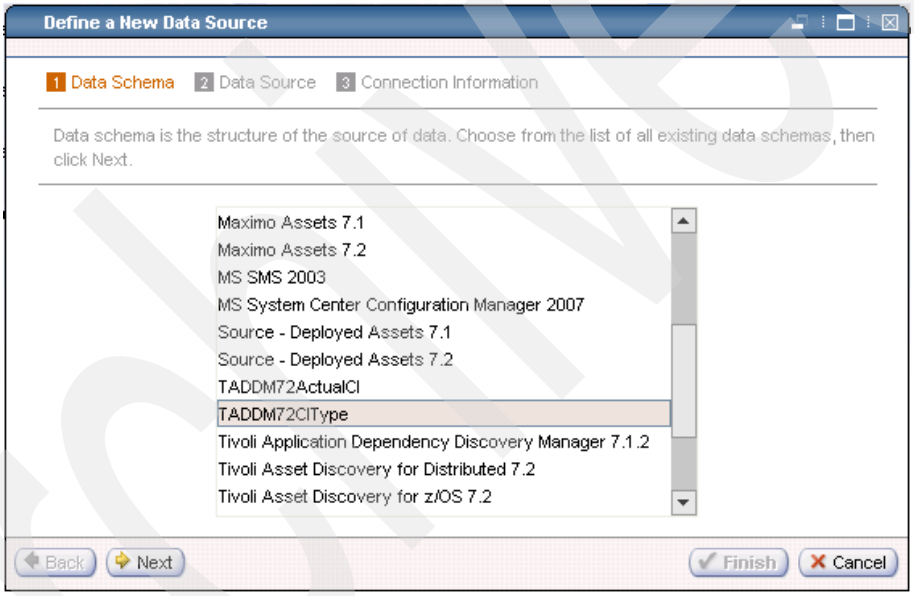


Figure 17-33 New data source

Archived

d. For the data source name, enter `TADDM72CIType`. See Figure 17-34.

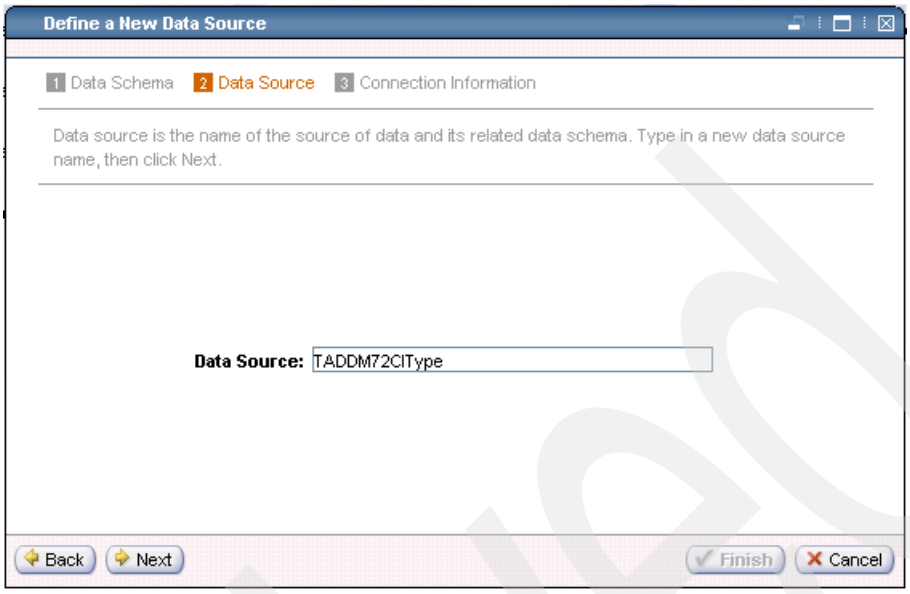


Figure 17-34 New data source

e. Make the following selections:

- Select Connection Method: IBM Configuration and Tracking API
- Host name: Type the host name for the TADDM server (not the database).
- Port: Type the RMI port for the TADDM server. The default value is 9530. If you are using SSL, the default value is 9531.
- Trusted Location: Fully-qualified path of the SSL certificate file to use to connect to a server such as the Configuration Discovery and Tracking Server, for example, `c:\Integration Composer\...\jssecacerts.cert` (Not a required field).
- User name: Type the user name for a TADDM account with administrator privileges.
- Password: Type the password for the TADDM use.

Archived

Figure 17-35 shows the completed fields.

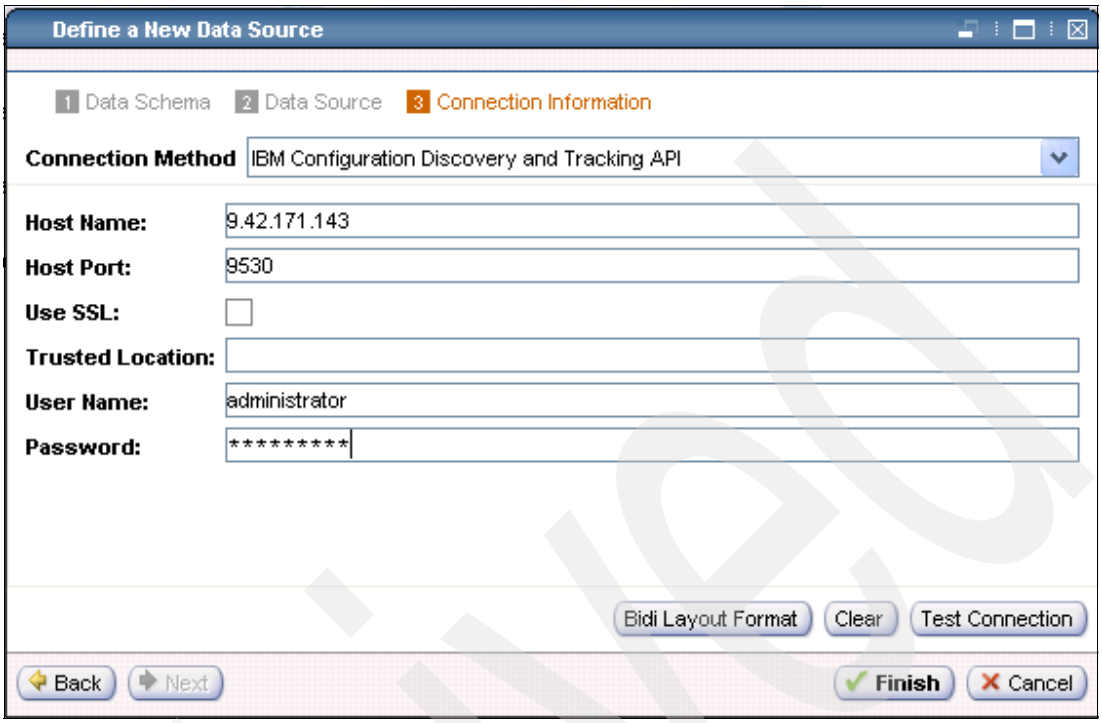


Figure 17-35 New data source

f. Click **Test Connection**. If an error occurs, check the values you entered.

g. Click **OK** and then click **Finish**.

3. Create CI type mappings:

a. Go to Tivoli Integration Composer home.

b. Select **Create New Mapping**.

c. On the next window, select the following values (Figure 17-36):

- Source: TADDM72CIType
- Target: CCMDB72citype
- Name: taddm-2-ccmdb-ci-types

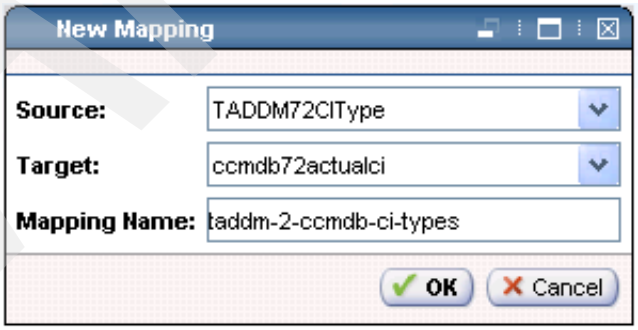


Figure 17-36 New mapping

d. Complete the password field and verify that the other required data from TADDM server is correct. It is similar to step “Make the following selections:” on page 501.

e. Click **Test Connection**. Upon success, click **OK** and click **Finish**. See Figure 17-37.

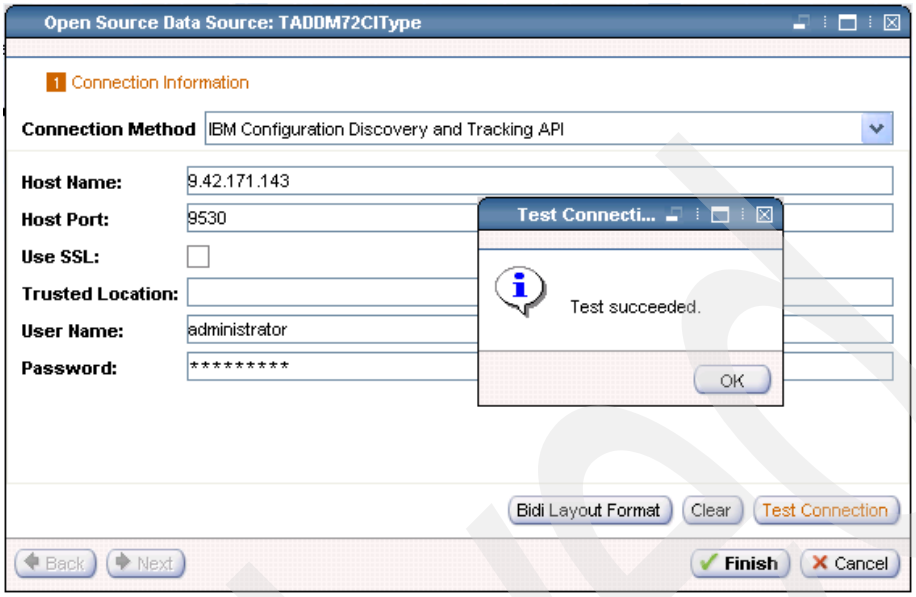


Figure 17-37 Test succeeded

f. Fill in the appropriate data for your environment:

i. Connection Method

- DB2: IBM DB2 JDBC Driver
- Oracle: Oracle JDBC Thin Driver
- SQL server: i-net OPTA JDBC Driver for Microsoft SQL Server 7/2000
- Host name: Type the host name for the CCMDB database server.
- Port: Type the port number for the machine. The default values are:
- DB2: 50005
- Oracle: 1521
- SQL server: 1433
- Database: Type the name of the database: maxdb71.
- User name: Type the user name for the user who created the Maximo tables in the database.
- Password: Type the user's password.
- Table owner: Type the user name for the user who created the Maximo tables in the database.

Archive

g. Click **Test Connection**. See Figure 17-38.

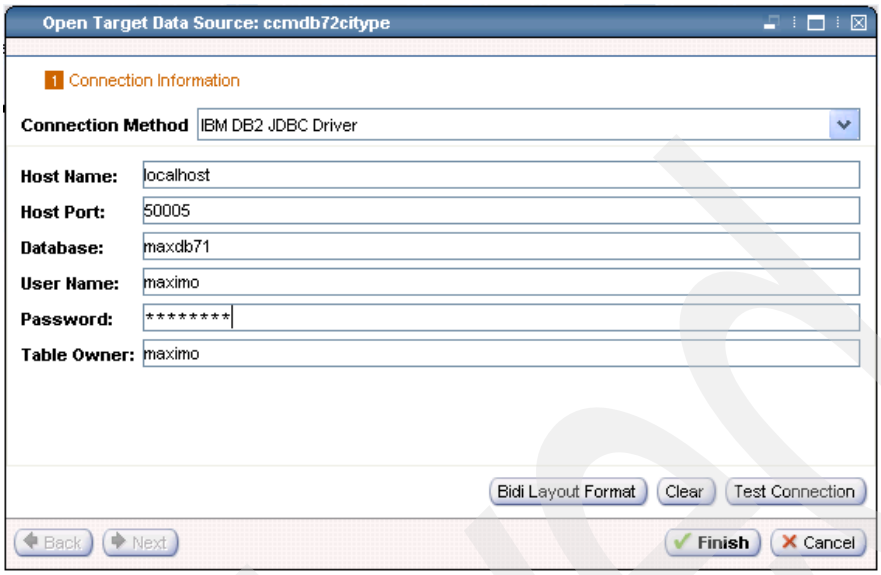


Figure 17-38 Test Connection

h. At the confirmation message, saying the information is OK, click **Finish**.

A window showing the relationships will be displayed.

i. Make any other changes you want and then click **Select Action** → **Save**. When you are done, click **Select Action** → **Close**.

4. Execute the CI type mapping:

a. Go to the command (cmd) prompt.

b. Change directory (**cd**) to the integration `composer\bin` directory.

c. Run the following command, where <password> is the password for the Maximo database user:

```
taddm-2-ccmdb-ci-types maximo password administrator collation maximo
password
```

An error message is generated because we did not define an expression yet (Example 17-1).

Note: The `commandline.bat/sh` parameters are case-sensitive.

Example 17-1 Error message

```
C:\Integration Composer>commandline.bat taddm-2-ccmdb-ci-types maximo PASSWORD
administrator collation maximo PASSWORD
Integration Composer will now start.
```

d. View the following log to verify your actions:

```
<installdir>\log\fusion.log
```

Archived

17.4.3 Activating the CI types

Next step is to activate the CI Types, because now, you choose what to import from TADDM:

1. Log in to the Base Services/CCMDB application (`http://host_name/maximo`).
2. Click **Go To** → **Administration** → **CI Types**. See Figure 17-39.

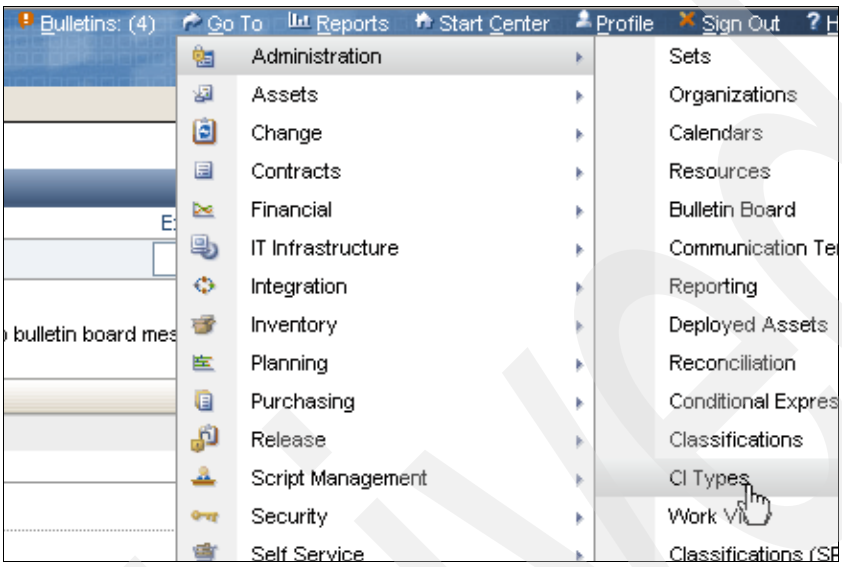


Figure 17-39 CI Types

3. Search for the following files, and others that you want. This step is when you choose what will be imported from TADDM:
   - SYS.COMPUTERSYSTEM
   - SYS.OPERATINGSYSTEM
   - APP. SOFTWAREIMAGE

Notes:

- For details about how to do this task, see the online help for the CI Types application in CCMDB.
- You must activate the first CI type before importing actual CIs by using the TADDM adapter. When activating the first CI type from the CI type application, if the initial JVM setting in WebSphere Application Server is set to a value less than 2 GB, out-of-memory exceptions can occur, which prevents the CI type from activating.

4. Using the Change Status icon, change the status to **Active**. See Figure 17-40 on page 506.

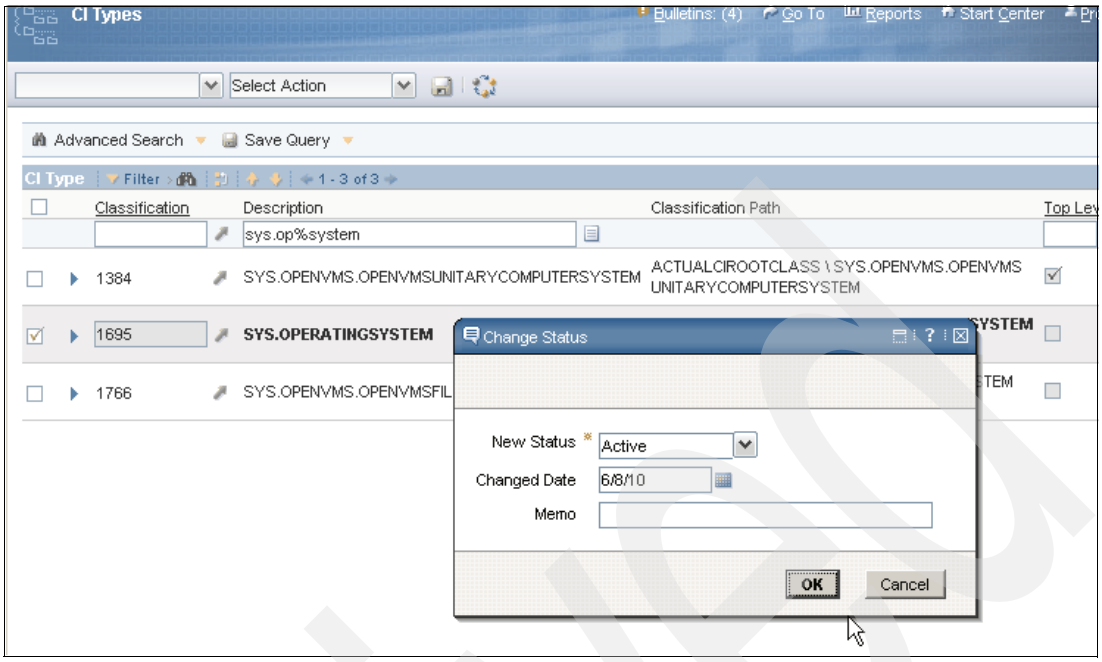


Figure 17-40 CI Types

5. Click **OK** and then click **Save.**

17.4.4 Setting up the actual CI adapter

Set up the actual CI adapter:

1. Create CI type mappings:
   a. Go to Tivoli Integration Composer home.
   b. Click **Create New Mapping**. See Figure 17-41.

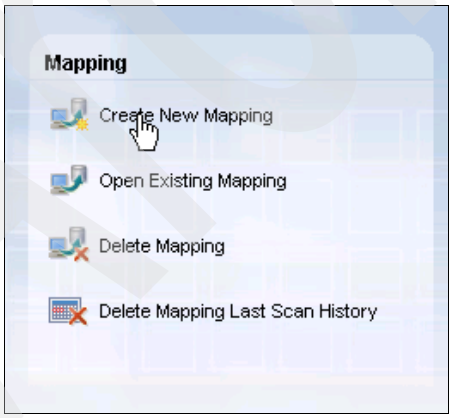


Figure 17-41 Mapping

   c. On the next window (Figure 17-42 on page 507), provide the following values:
      - Source: TADDM72CIType
      - Target: CCMDB72citype
      - Name: taddm-2-ccmdb-ci-types

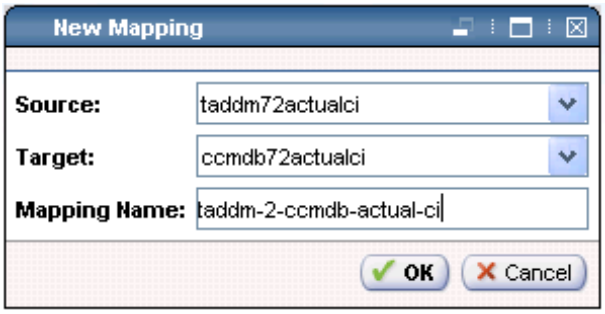


Figure 17-42 New Mapping

d. Complete the Password field (Figure 17-43) and verify whether the other required data from the TADDM server is correct. This step is similar to the step “Make the following selections:” on page 501.

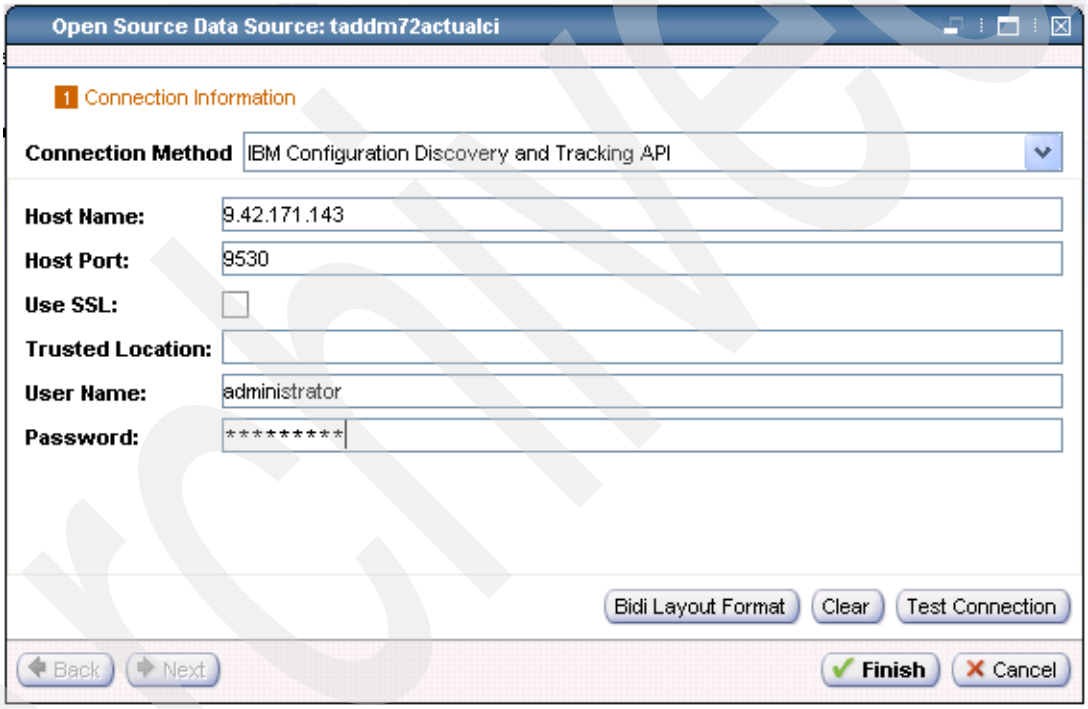


Figure 17-43 Source data source

e. Click **Test Connection**. Upon success, click **OK** and click **Finish.**

f. Specify the appropriate data for your environment:

- Connection Method:

  DB2: IBM DB2 JDBC Driver
  Oracle: Oracle JDBC Thin Driver
  SQL server: i-net OPTA JDBC Driver for Microsoft SQL Server 7/2000

- Host name: Type the host name for the CCMDB database server.
- Port: Type the port number for the machine. The default values are as follows:

  DB2: 50005
  Oracle: 1521
  SQL server: 1433

Archived

- Database: Type the name of the database: maxdb71.
- User name: Type the user name for the user who created the Maximo tables in the database.
- Password: Type the user's password.
- Table owner: Type the user name for the user who created the Maximo tables in the database. See Figure 17-44.

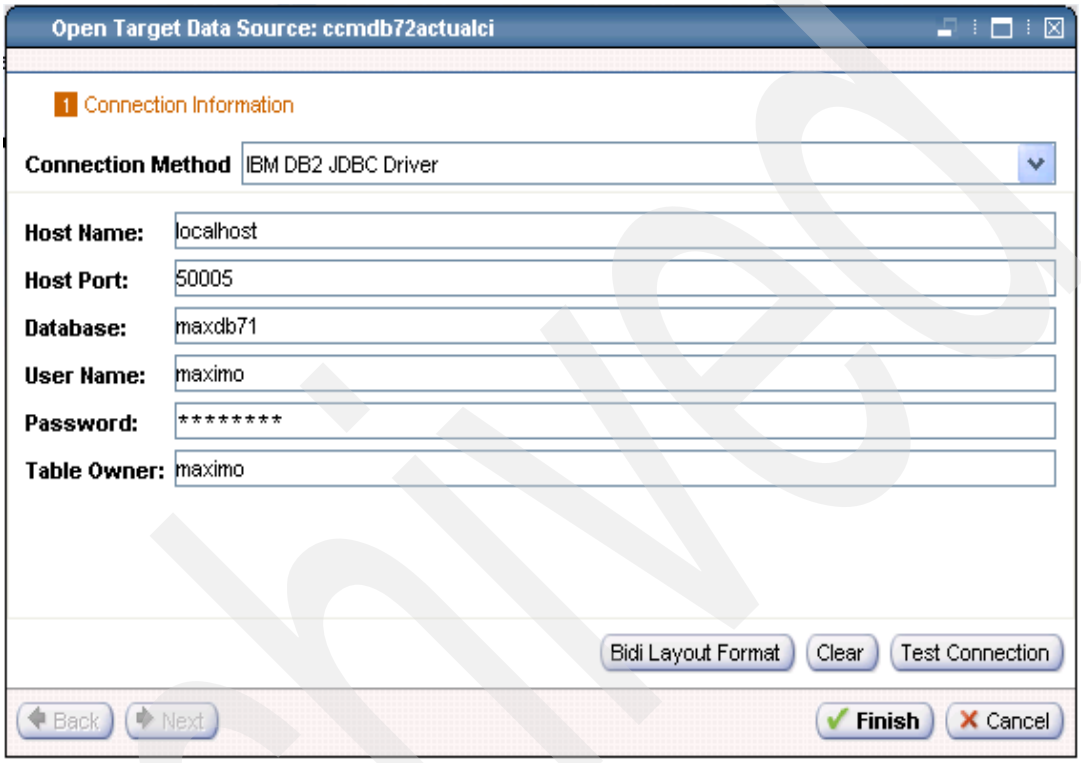


Figure 17-44 Target data source

g. Click **Test Connection**.

h. When a pop-up window indicates that the connection is ok, click **Finish**.

A window showing the relationships opens.

Archive

i. Verify that the column expressions have codes (Figure 17-45), and then click **Close**.

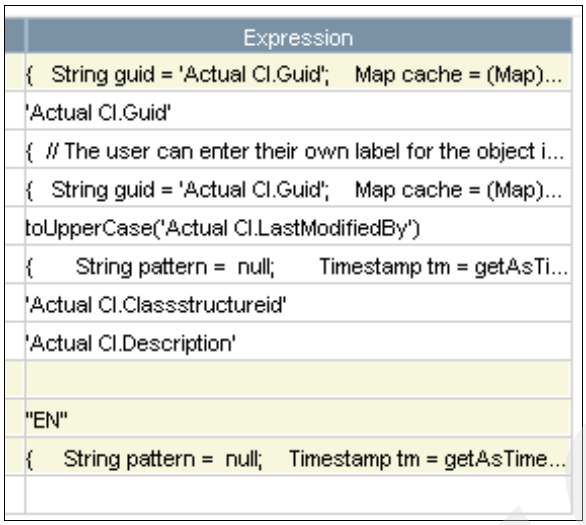


Figure 17-45 Column expressions

2. Run the CI type mapping:
   a. Go to a command prompt.
   b. Change directory (cd) to the integration composer\bin directory.
   c. Run **taddm-2-ccmdb-actual-ci maximo password administrator collation maximo password** where <password> is the password for the Maximo database user. See Example 17-2.

Example 17-2 Command to run

```
C:\Integration Composer>commandline.bat taddm-2-ccmdb-ci-types maximo PASSWORD
administrator collation maximo PASSWORD
Integration Composer will now start.
```

3. View the logs if necessary: `<installdir>\log\fusion.log`

Archived

17.4.5 Verifying that the data has been imported into CCMDB

Perform the following steps to verify that the data has been imported into CCMDB:

1. In CCMDB, click **Go To** → **IT Infrastructure** → **Actual Configuration Items** (Figure 17-46).

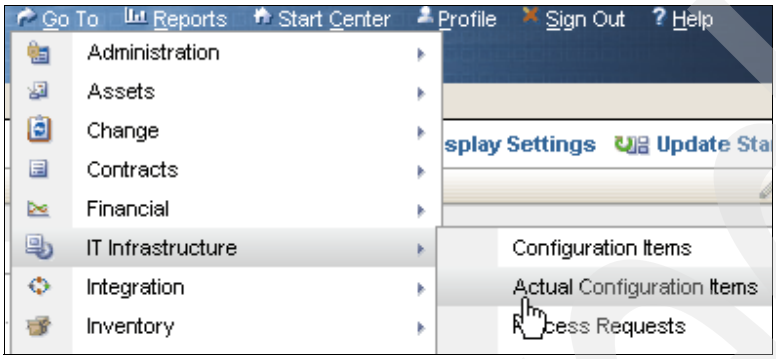


Figure 17-46 Actual Configuration Items

2. Search for items that should have been imported and confirm that they are present. If you find them, you have successfully imported the data.

Now, IBM RedBooks Company A is ready for the Asset and CI integration. This scenario, automated the link between CIs and assets based on the GUIDs of the associated actual CIs and deployed assets, respectively. Every time the condition that is created is met, the Tivoli Asset Management for IT integrates the assets and CIs.

The advantages of this is that IBM RedBooks Company A can save much time because the process is automatic now, even the Actual CIs will be included in this relationship. Promoting the CIs and comparing manually is unnecessary, this integration does it for you.

17.5 Adding a Configuration Item field in the Asset application

Because assets and CIs are implemented in various tables in the database, no association exists between these objects. However, certain applications use the same entity, for instance, a software in the production environment. In the CI application you can link it to an asset.

The default installation of IBM Tivoli Asset Management for IT does not display configuration item data in the Assets application. To view a configuration item that is associated with an asset in the Assets application, use the Application Designer application to add a Configuration Item field to the Assets application.

Archived

We first have to add a Configuration Item field to the Asset application:

1. On the navigation bar, click **Go To** → **System Configuration** → **Platform Configuration** → **Application Designer**. See Figure 17-47.

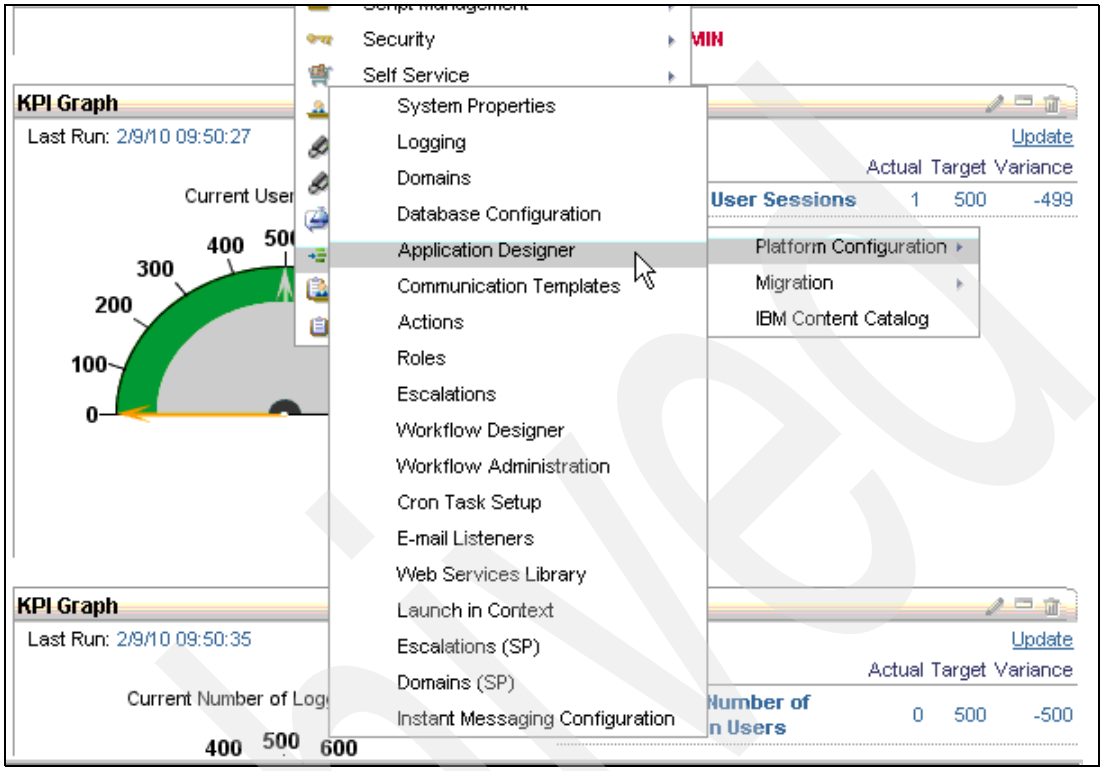


Figure 17-47 Acessing the Application Designer application

2. On the Applications tab, select **ASSET** for the Assets application. See Figure 17-48.

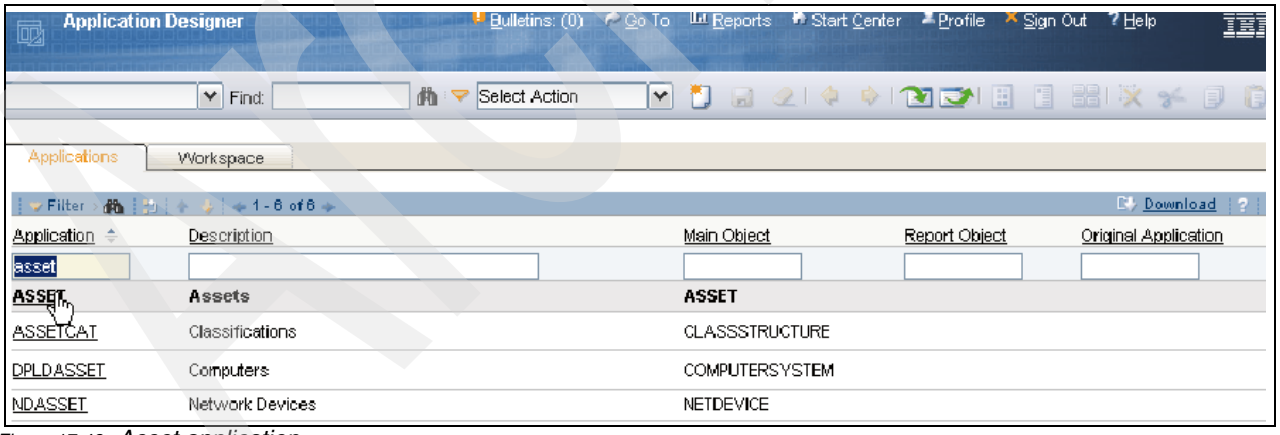


Figure 17-48 Asset application

3. On the Workspace tab, select the **Asset** subtab.

4. Select the **Details** section and click **Control Palette**. See Figure 17-49.

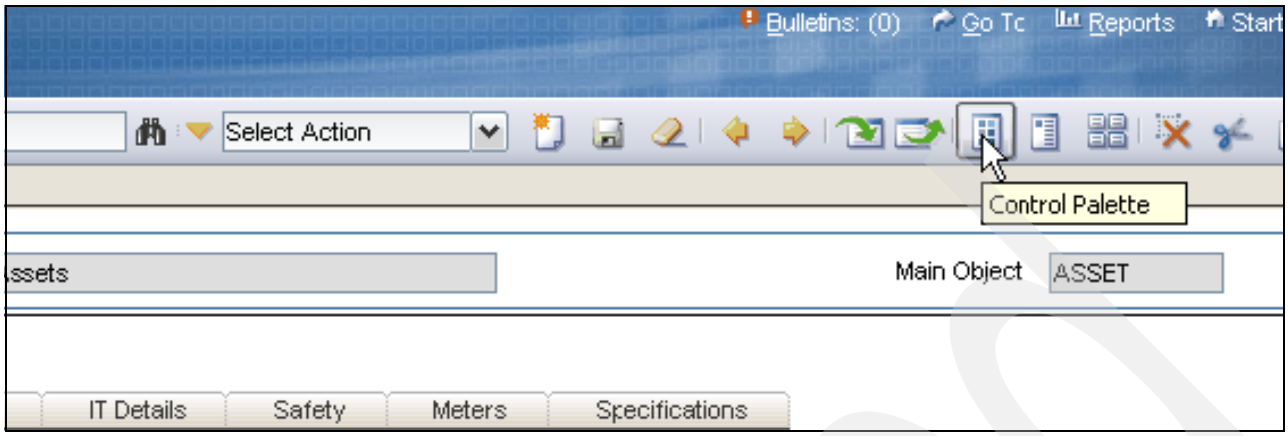


Figure 17-49 Acessing the Control Palette

5. Drag the **Multipart Textbox** icon from the Control Palette window to the location in the Details section where you want the new field. See Figure 17-50.

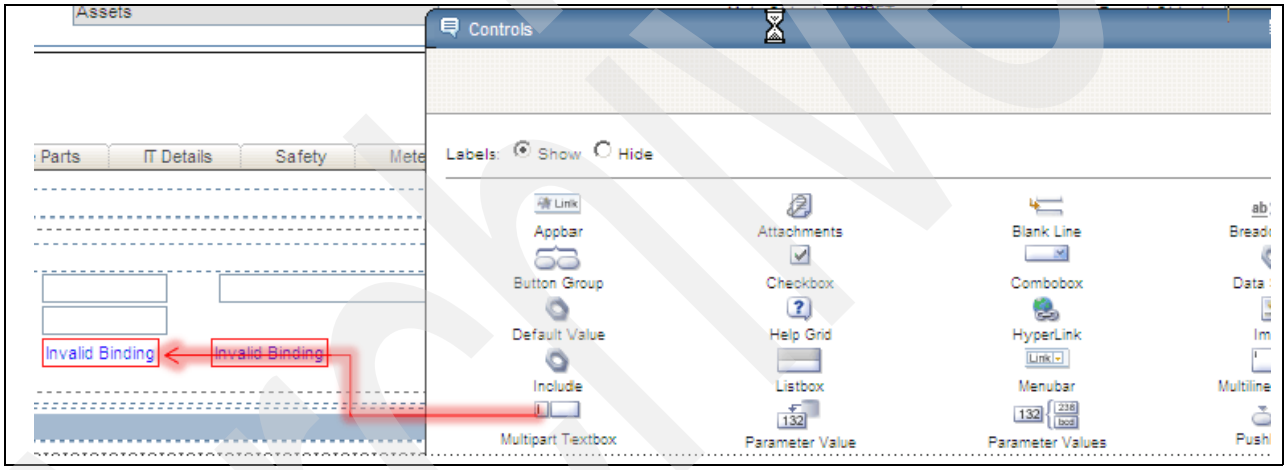


Figure 17-50 Drag Multipart Textbox

6. Close the Control Palette window.

Archive

7. Right-click the **Multipart Textbox** that you added and select **Properties** in the Multipart Textbox Properties window, specify values in the as indicated in Figure 17-51 and Figure 17-52.

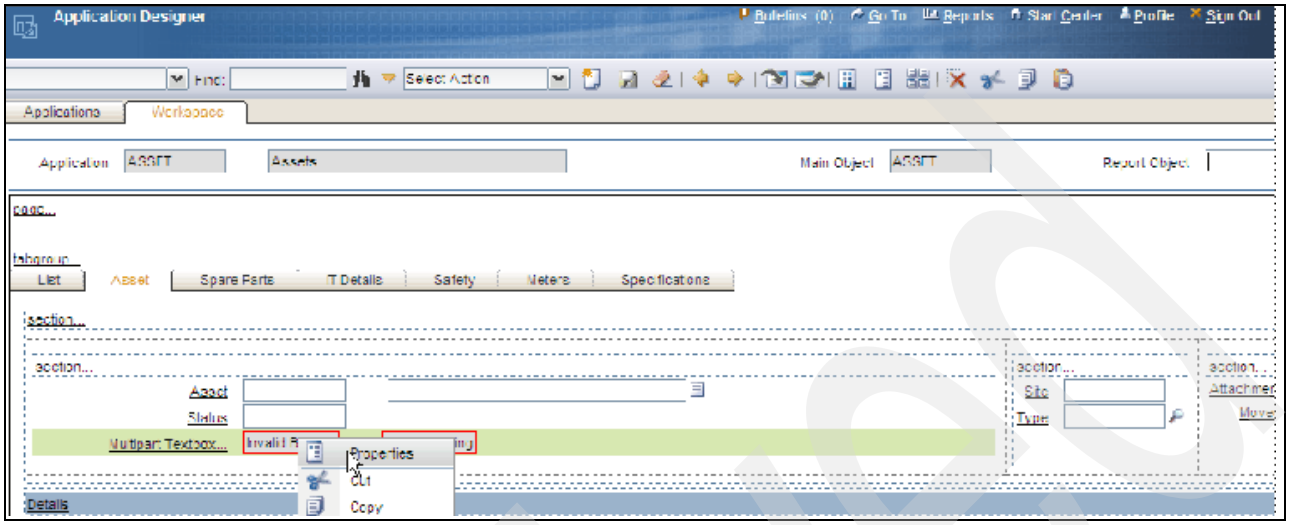


Figure 17-51 Check the properties from the Multipart Textbox

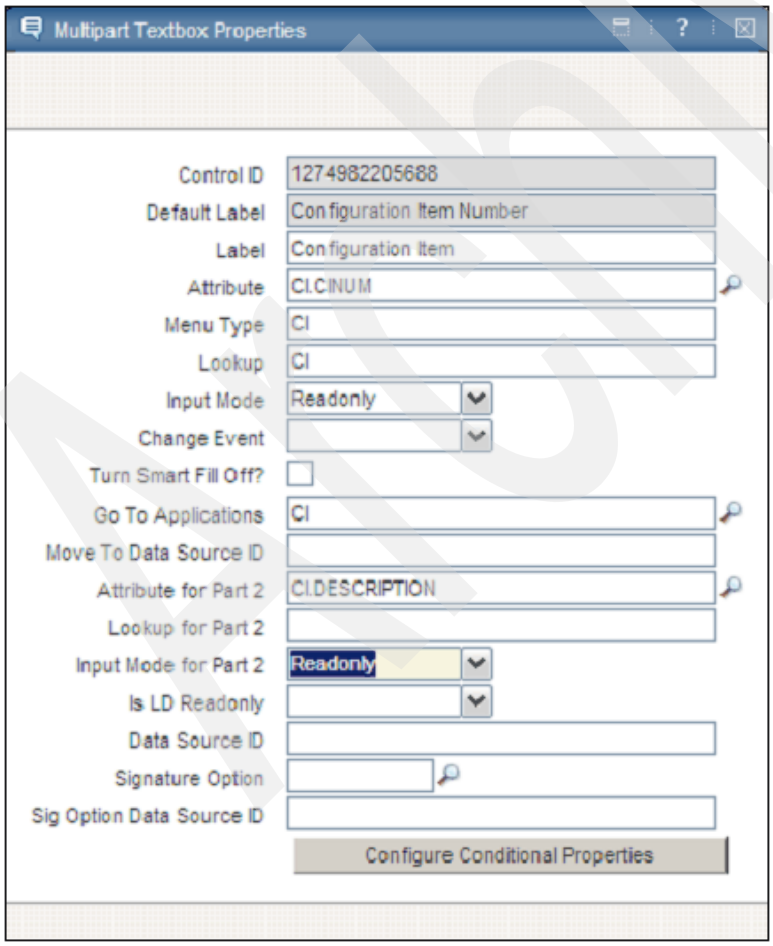


Figure 17-52 Multipart Textbox properties

8. Close the Multipart Textbox Properties window.
9. Click **Save Application Definition** icon. See Figure 17-53.

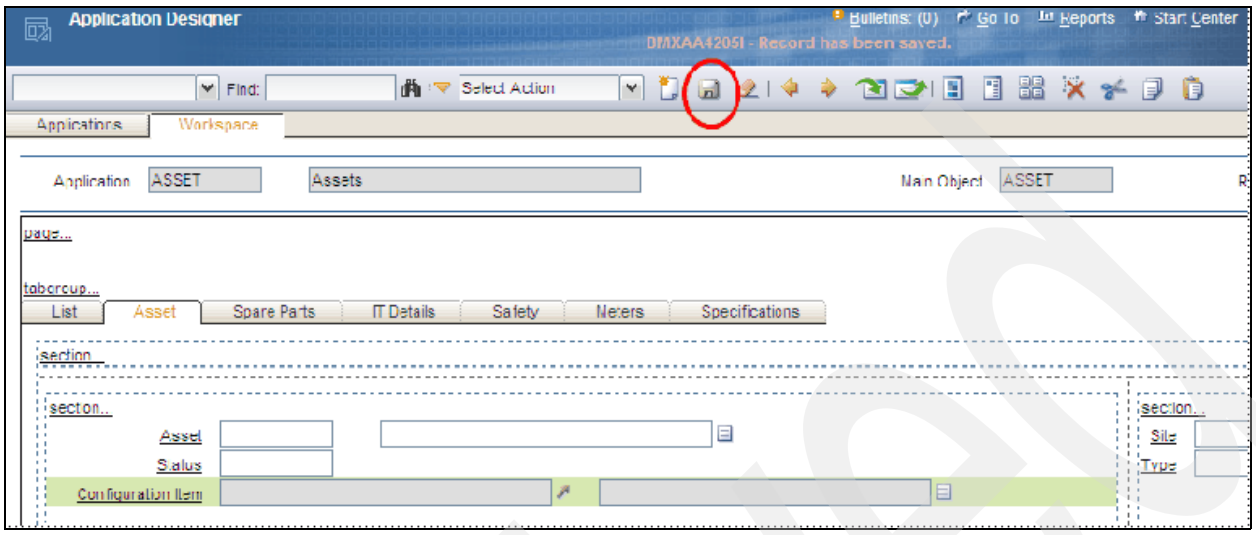


Figure 17-53 Save Application Definition

10. Verify whether the Assets application has the Configuration Item field you just created. See Figure 17-54.

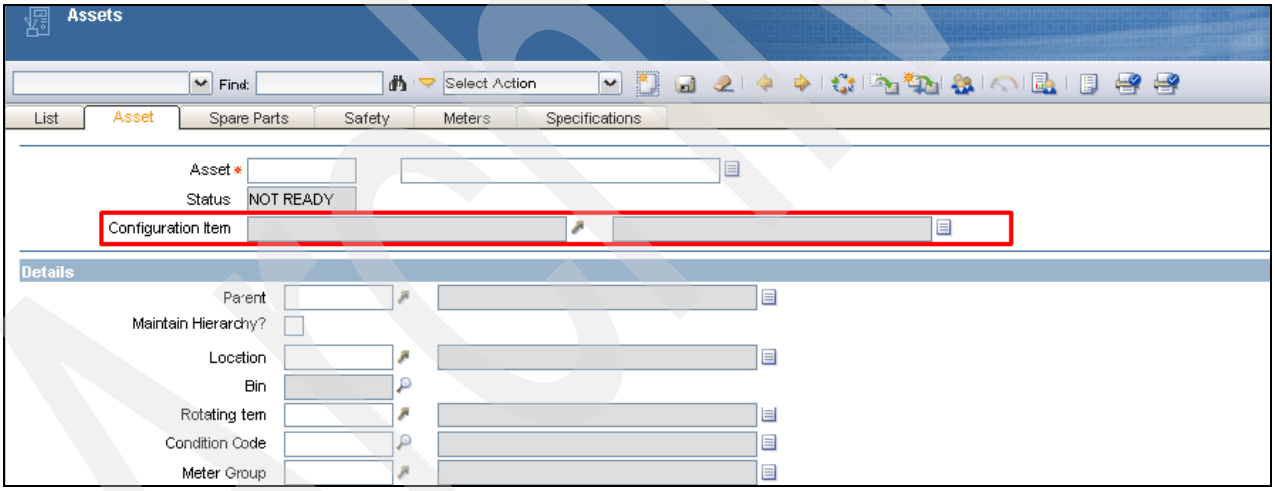


Figure 17-54 Configuration Item field created

Now, every time you want, you can relate the Asset with CIs and you will see it at the same window, making easier the work and management, also.

17.6 Using a link rule to link assets to CIs

IBM Tivoli Asset Management for IT maintains two kinds of data about information technology assets or items: authorized IT asset data and deployed asset data.

Using a link rule, linking authorized IT asset data to configuration items automatically is possible.

An authorized IT asset is data, which you recorded about IT assets, such as computers and network printers that you have acquired and installed. This data represents your authorized inventory, how things should be and what you have planned. IT asset records are a subset of asset records. It is the classification of an asset that is used to determine whether or not an asset is an IT asset. To distinguish IT assets from other types of assets, administrators define a top-level IT asset class in the Organizations application. Any asset that belongs to the hierarchy of the top-level IT classification is an IT asset.

Important: *Authorized IT asset data* is different from the data you discover when you use an asset discovery tool. That data is called as *deployed asset data*.

This scenario for linking CIs to assets uses a link rule to match a configuration item to an authorized IT asset.

Linking CIs to assets with a link rule is accomplished in the following steps:

1. Specify top-level classes for CIs and for IT assets.
2. Specify system properties for the reconciliation.
3. Configure the reconciliation, which includes the following tasks:
   - Creating a link rule.
   - Creating a reconciliation task.
4. Specify the relationship between the CI table in the database and the reconciliation link that was established during the reconciliation process.
5. Configure the escalation that adds the asset number of the linked asset to the CI.

Configuring the escalation includes the following tasks:

- Creating the action that specifies how to add the asset number
- Creating the escalation that schedules and performs the action

17.6.1 Specifying top-level classes for CIs and for IT assets

The first step to create an automated CI-to-asset linking process, is to verify that the top-level classifications have been defined and that they are accurate.

Before you begin: Before you can specify top-level classifications for IT assets and software, *a classification structure* for IT assets and configuration items *must exist* in the Classifications application. This step is generally performed by administrators as part of the post-installation configuration tasks.

To verify whether you have classifications for IT assets, access the Classifications application (**Go To** → **Administration** → **Classifications** and search in the list table.

Archive

Perform the following steps:

1. On the navigation bar in Asset Management for IT, click **Go To** → **Administration** → **Organizations**. See Figure 17-55.

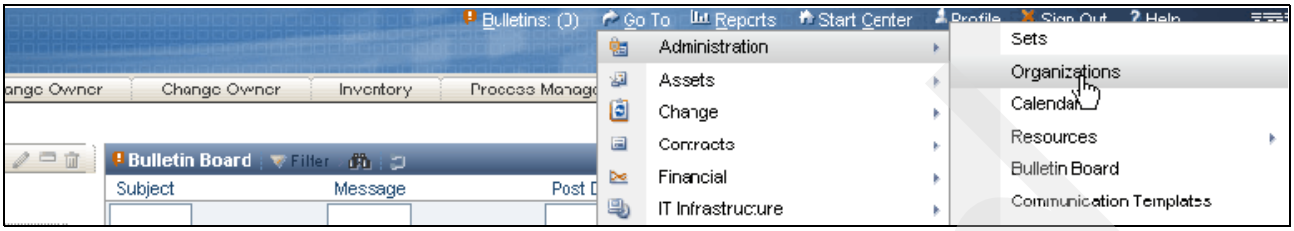


Figure 17-55 Organizations Application

2. In the Organizations application, click **Select Action** → **System Settings**. See Figure 17-56.

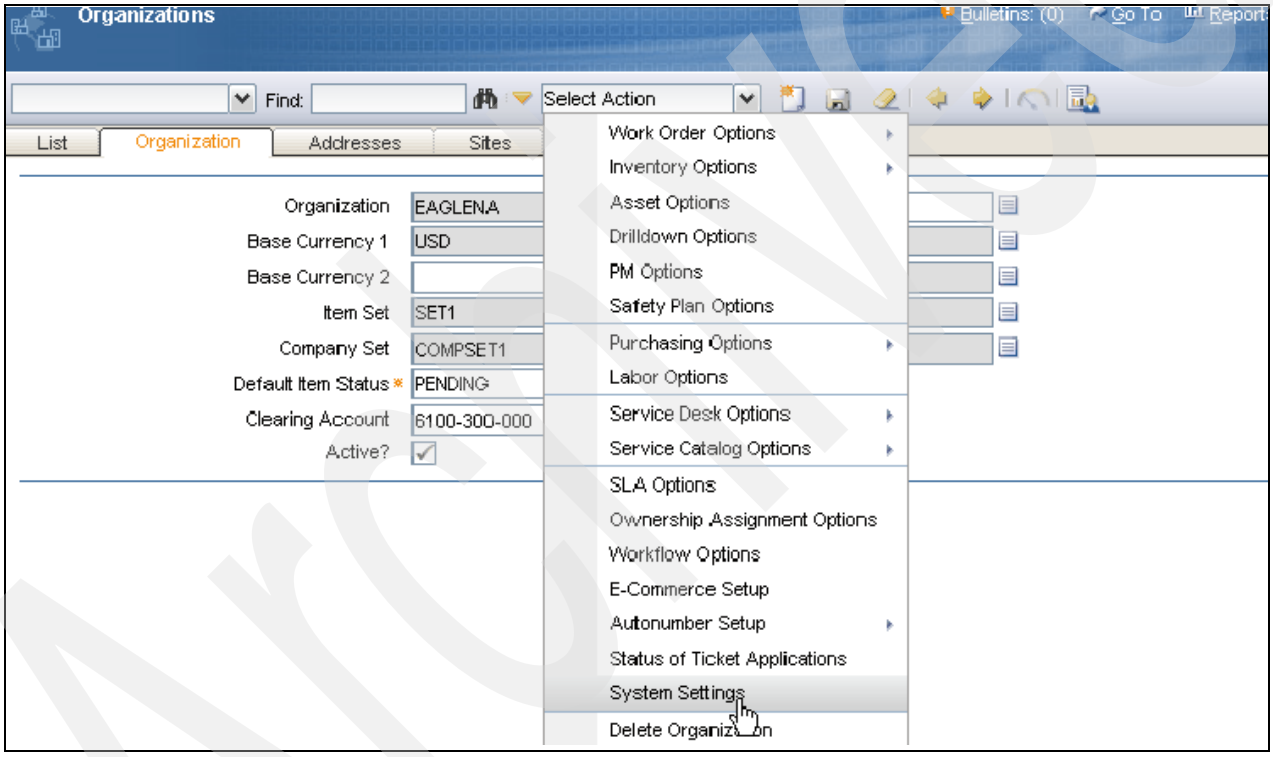


Figure 17-56 System Settings

3. Verify whether a top-level IT asset class is specified. If it is not defined, specify the top-level IT asset as follows:

   a. In the IT Options section in the Systems Settings window, click **IT Asset Top-Level Class** → **Detail Menu** → **Classify**. If necessary, you can select **Clear Classificatio**n to clear the value displayed and then click **Classify**. See Figure 17-57.

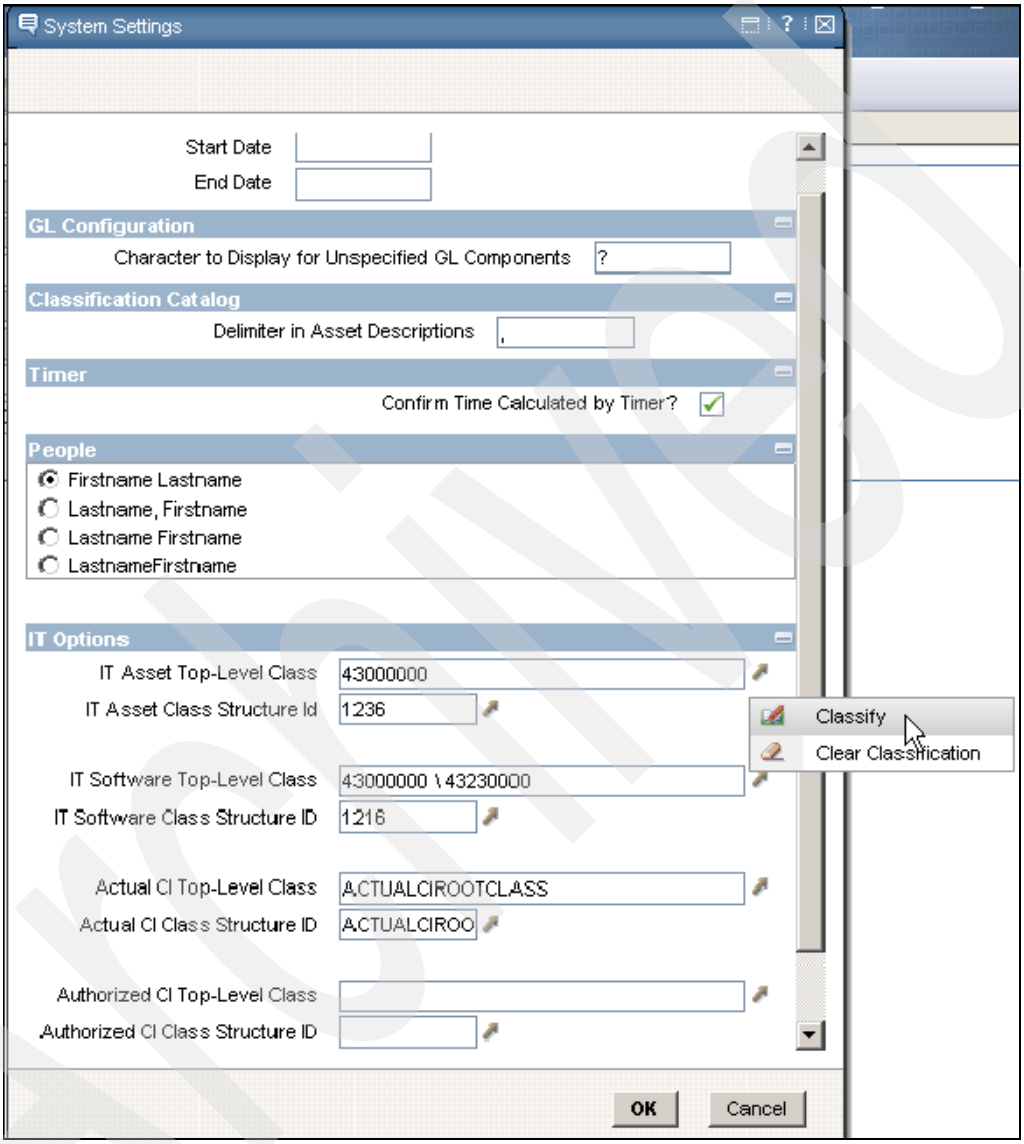


Figure 17-57 System Settings window

   b. In the Classify window, select the top-level asset class for IT assets by clicking the blue square to the left of the classification name. The application closes the Classify dialog box and populates the IT Asset Top-Level Class field. Let the System Settings window open to check the next settings. See Figure 17-58 on page 518.

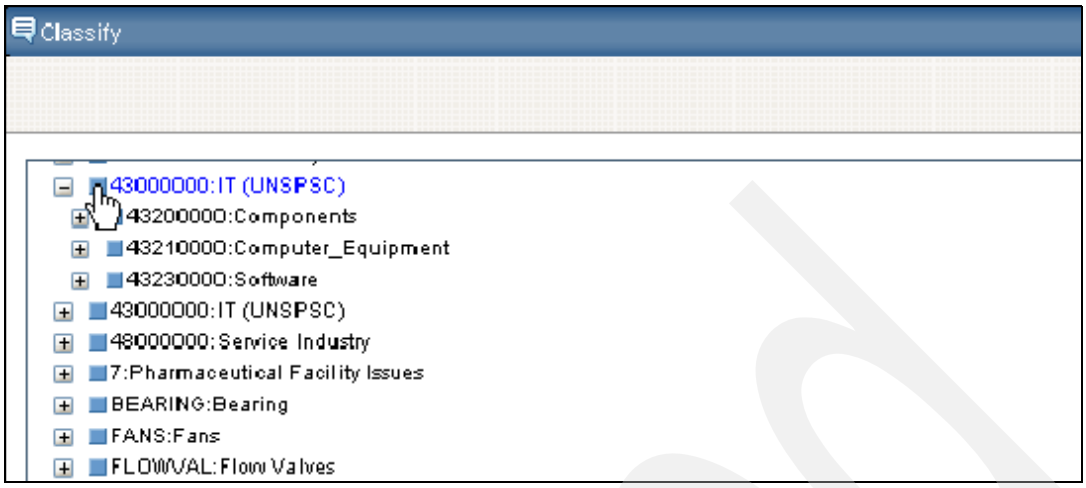


Figure 17-58 IT Asset Top-Level Classification

4. Verify whether a top-level authorized CI class is specified. If it is not, specify the top-level CI classification as follows:

   a. In the IT Options section in the Systems Settings window, in the Authorized CI Top-Level Class field, click the **Detail Menu** and select **Classify**. See Figure 17-59.

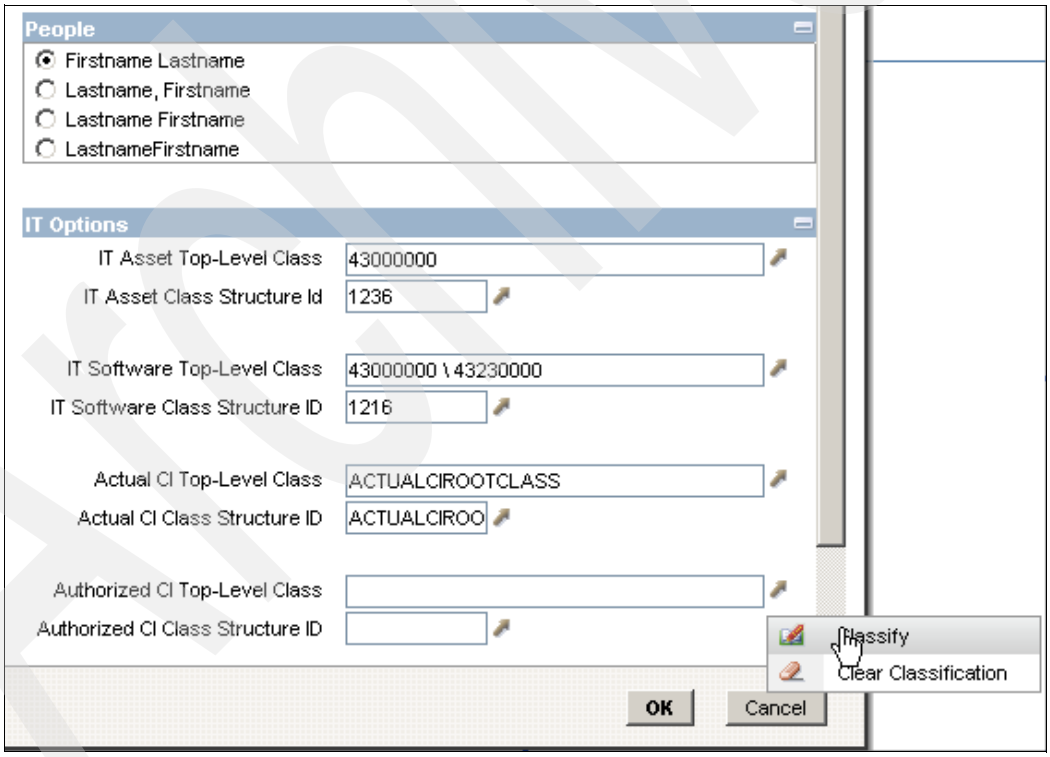


Figure 17-59 Authorized CI Top-Level Class

   b. In the Classify window, select the top-level class for configuration items by clicking the blue square to the left of the classification name. The application closes the Classify dialog box and populates the authorized CI Top-Level Class field. See Figure 17-60 on page 519.

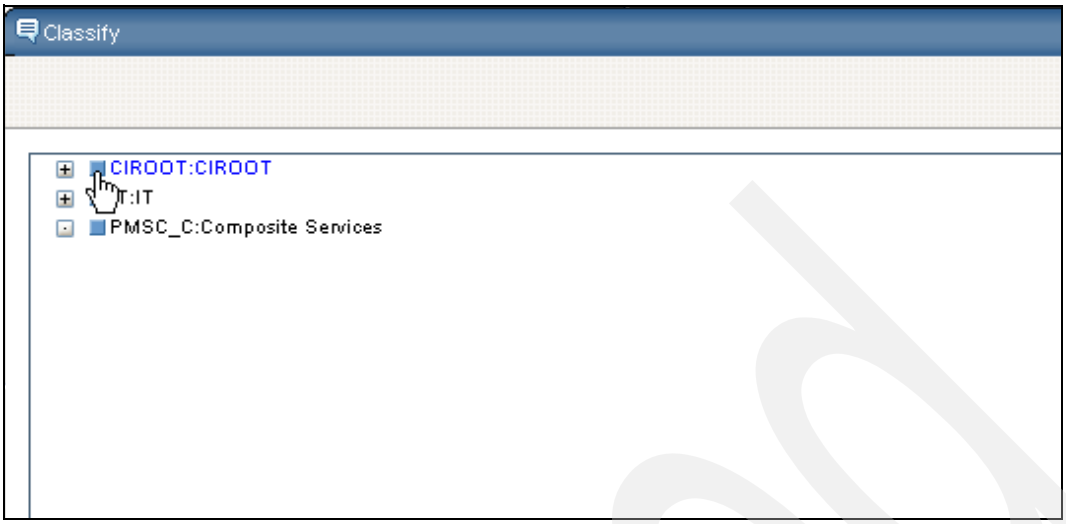


Figure 17-60 Authorized CI Top-Level Classification

5. Click **OK** to save the settings and close the dialog box.

17.6.2 Specifying reconciliation properties

Before you can create a link rule to link CIs to assets, configure reconciliation properties in the System Properties application:

1. On the navigation bar in Asset Management for IT, click **Go To** → **System Configuration** → **Platform Configuration** → **System Properties**. See Figure 17-61.

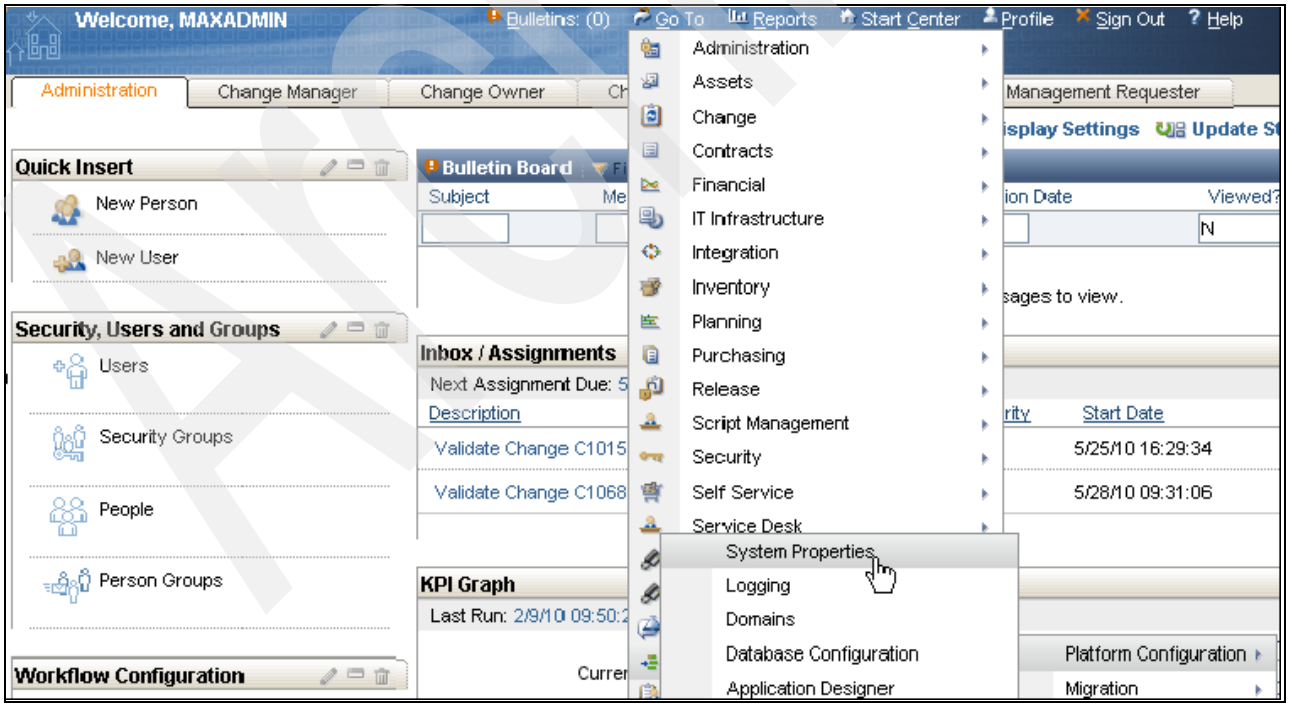


Figure 17-61 System Properties application

Archived

2. Find the `recon.engine.dataset.map.ASSET` system property, select the check box for the property, and expand the row. See Figure 17-62.

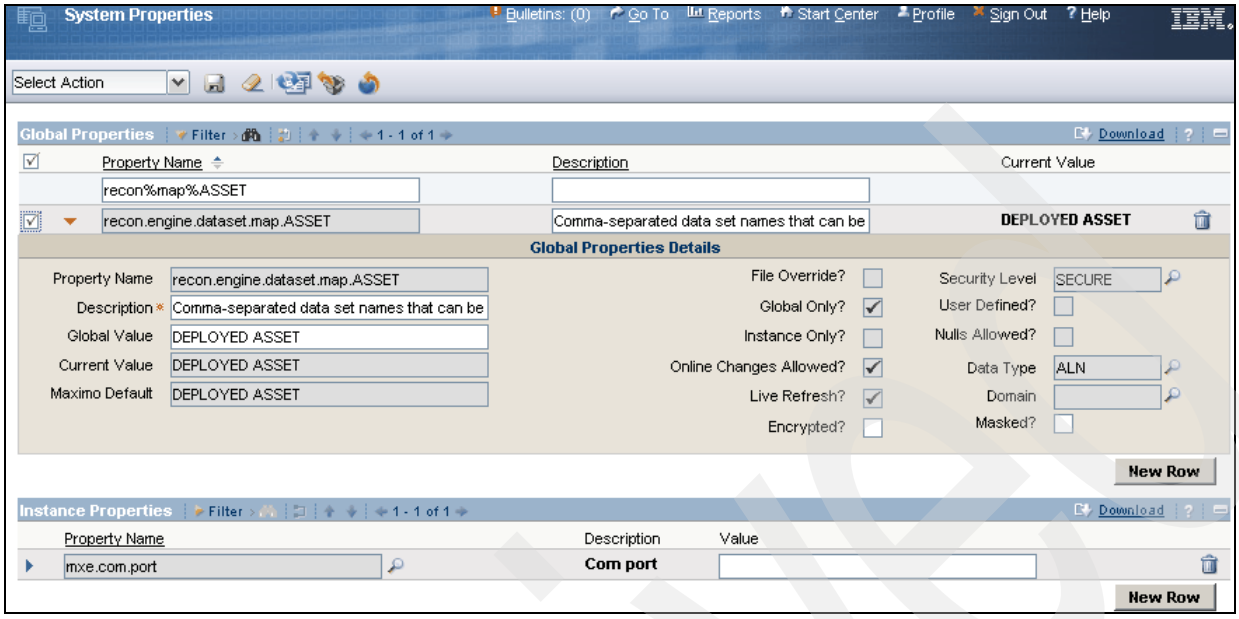


Figure 17-62 Global Properties

3. In the Global Value field, specify the value `DEPLOYED ASSET, CI`, and then click **Save property**. See Figure 17-63.

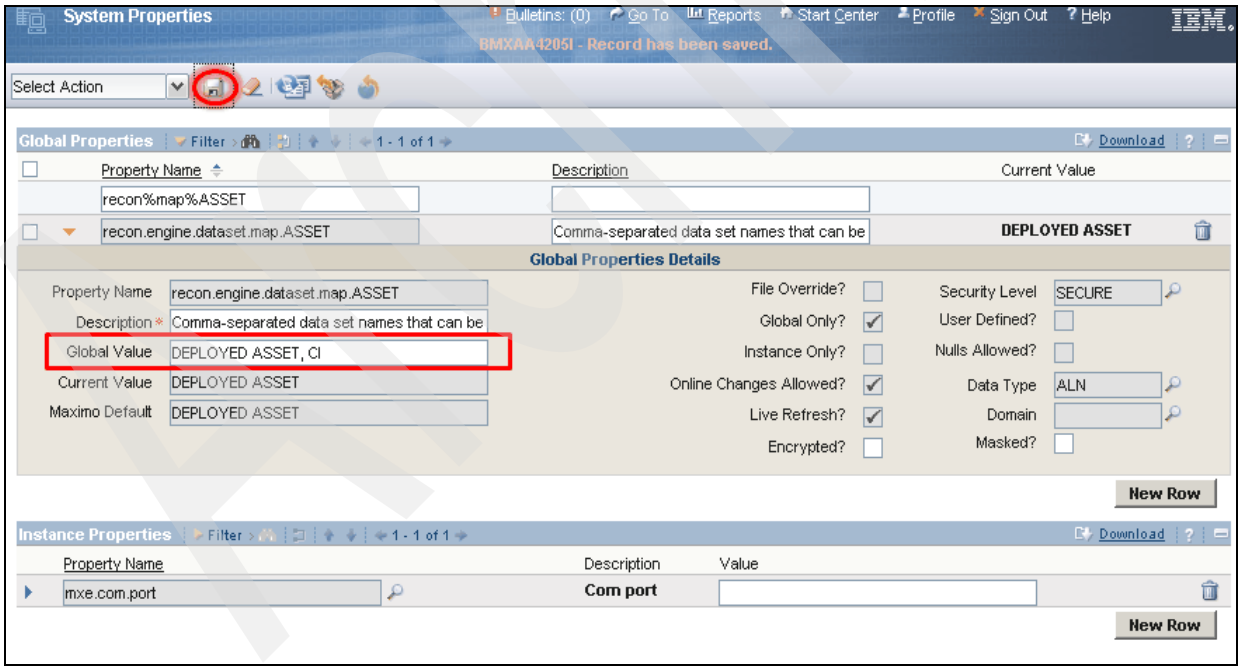


Figure 17-63 Global Value

4. Select the check box for the `recon.engine.dataset.map.ASSET` system property, and from Select Action menu, click **Live Refresh**. See Figure 17-64.

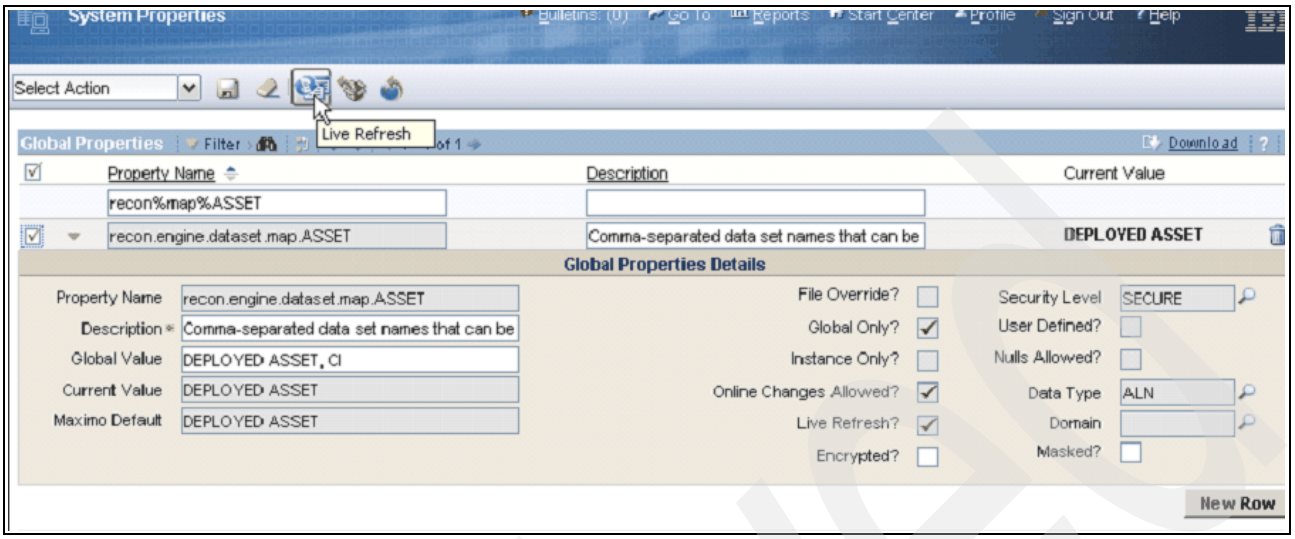


Figure 17-64 Live Refresh

5. In the Live Refresh window, click **OK**. See Figure 17-65.

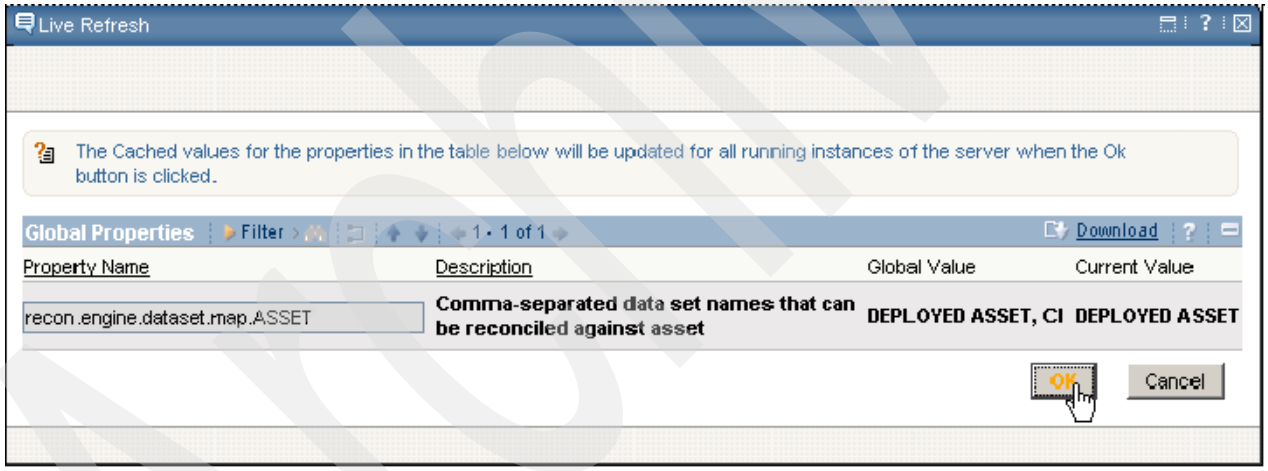


Figure 17-65 Live Refresh window

Now you have the current value `DEPLOYED ASSET, CI`. See Figure 17-66.

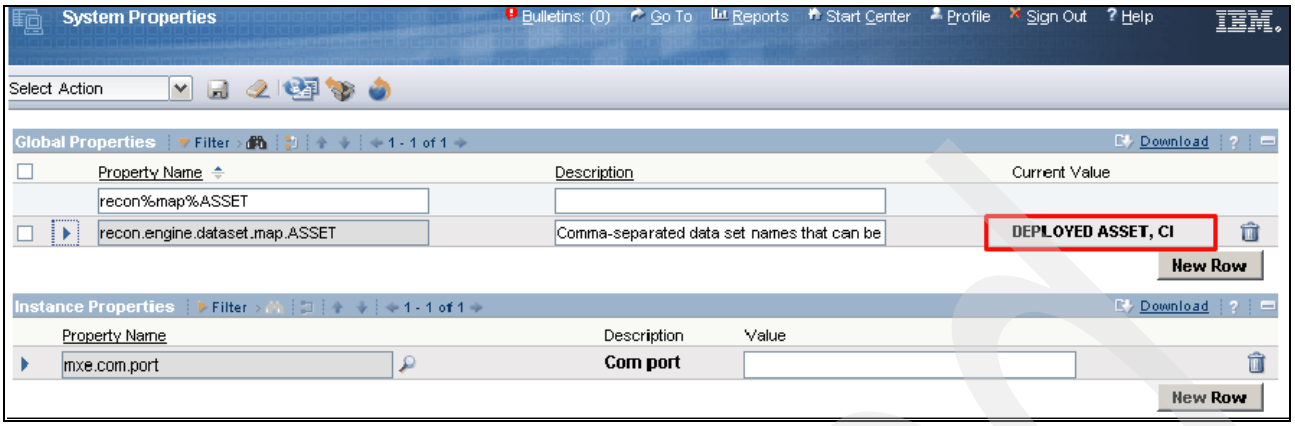


Figure 17-66 Check current value

17.6.3 Configuring the reconciliation for linking CIs to assets with a link rule

To configure the escalation for linking CIs to assets, create an action in the Actions application and an escalation in the Escalations application.

You use the Escalations application to automatically configure a response to events.

For example, you can create an escalation to change the status of a record in the receiving application, if the item received has passed inspection. For CI-to-asset links, you create an escalation that links CIs to assets if the appropriate conditions are met.

When you define an escalation, you must associate it with a specific record, in this case CI records. You can then define the conditions for specific attributes of a record and have the system trigger an escalation when those conditions are met. An *escalation point* represents a condition or threshold that must be met, and you can use one or more of these to define your escalation.

When you create an escalation point, you must associate at least one action or notification to it before you activate the escalation. An action is an event that you want the system to trigger when it finds records that meet the condition (or conditions) defined by the escalation point, such as changing a status or an owner.

For CI-to-asset linking, you define an escalation that adds an asset number to a CI record when a match is found between a CI and an IT asset.

Configuring the escalation for the CI-to-asset link includes the following tasks:

- Creating an action that specifies the action to occur when the escalation is processed
- Creating an escalation that specifies the action to take, the escalation point that triggers the action, and the schedule for processing the escalation

Archive

Perform the following steps:

1. On the navigation bar in Asset Management for IT, click **Go To** → **Administration** → **Reconciliation** → **Link Rules**. See Figure 17-67.

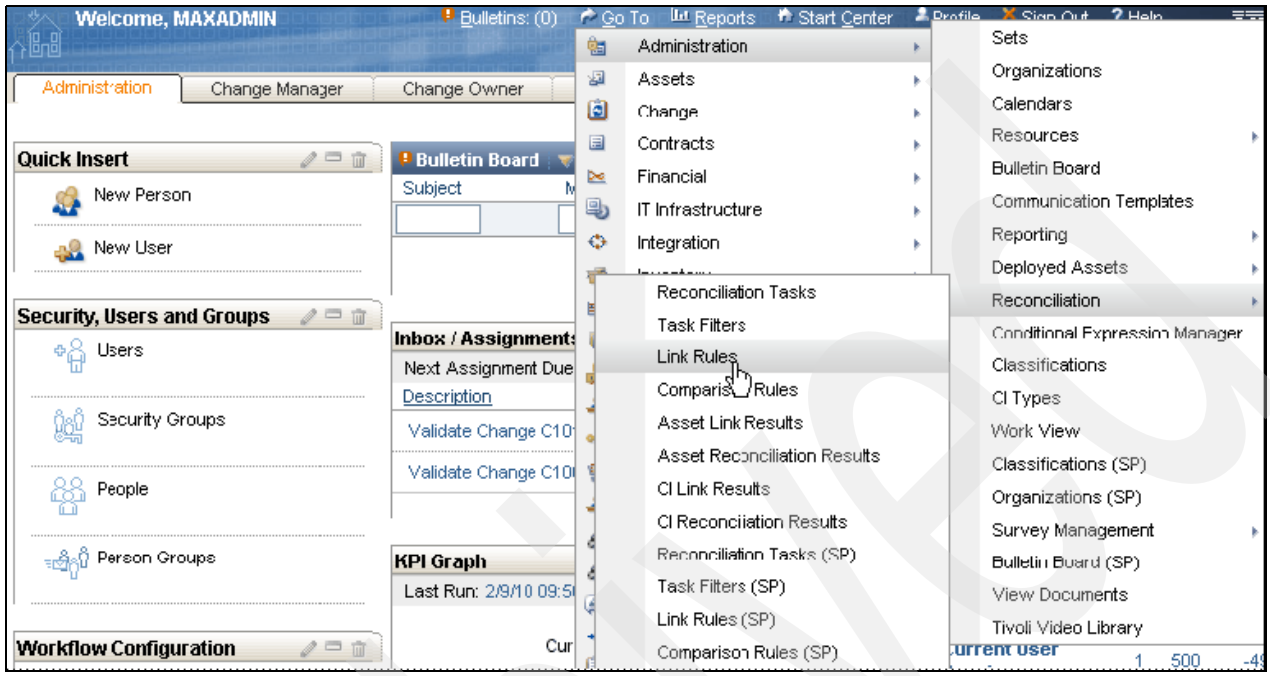


Figure 17-67 Link Rules application

2. On the toolbar in the Link Rules application, click **New Link Rule**. See Figure 17-68.

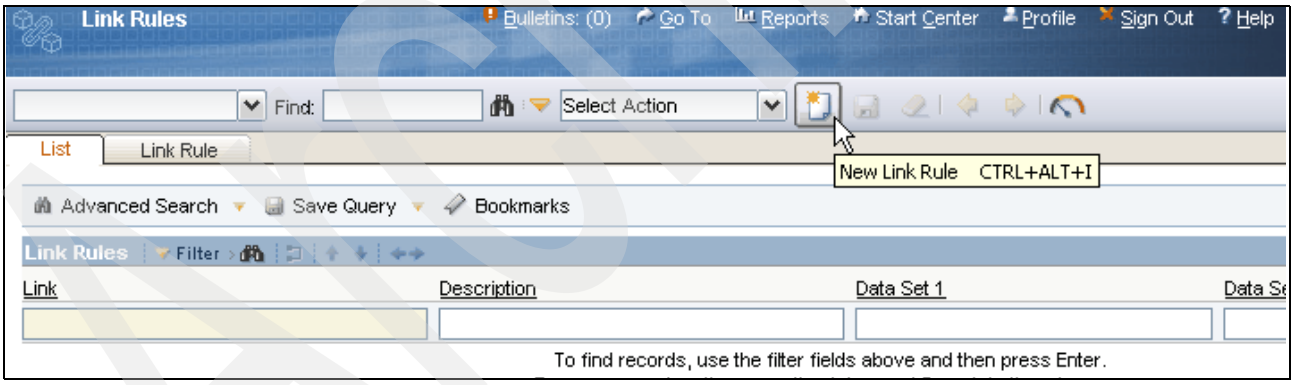


Figure 17-68 Click New Link Rule

3. In the Link field, type the name that you want to assign to the link rule, for example `ASSET2CI_LINKRULE`. This name must be unique.

4. In the Description field, type a brief description of the link rule. See Figure 17-69.

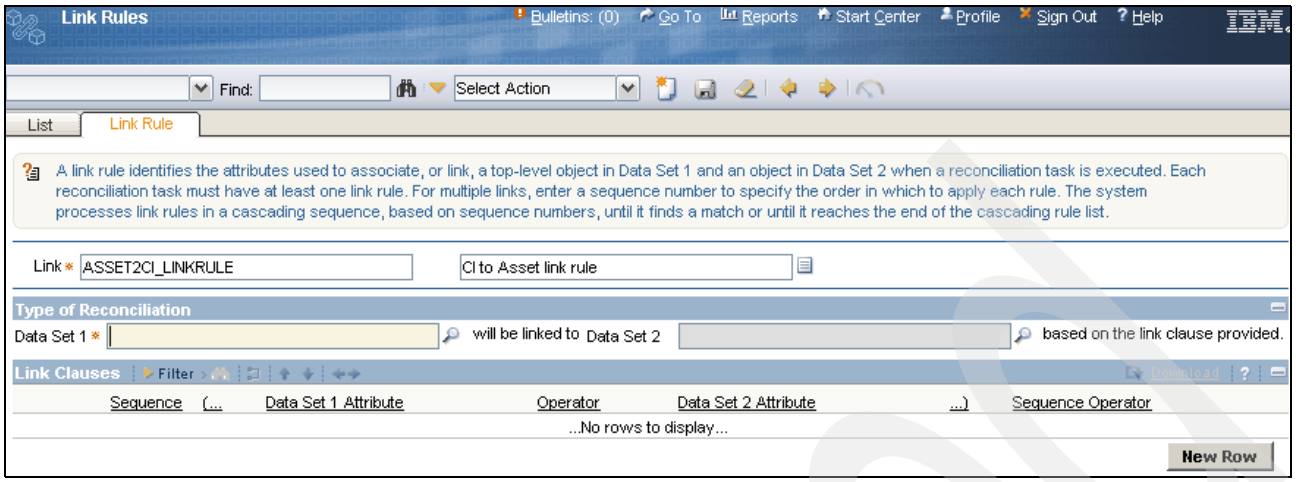


Figure 17-69 Link Rules

5. In the Type of Reconciliation table window, in the Data Set 1 field, click **Select Value** and select **ASSET**. See Figure 17-70.

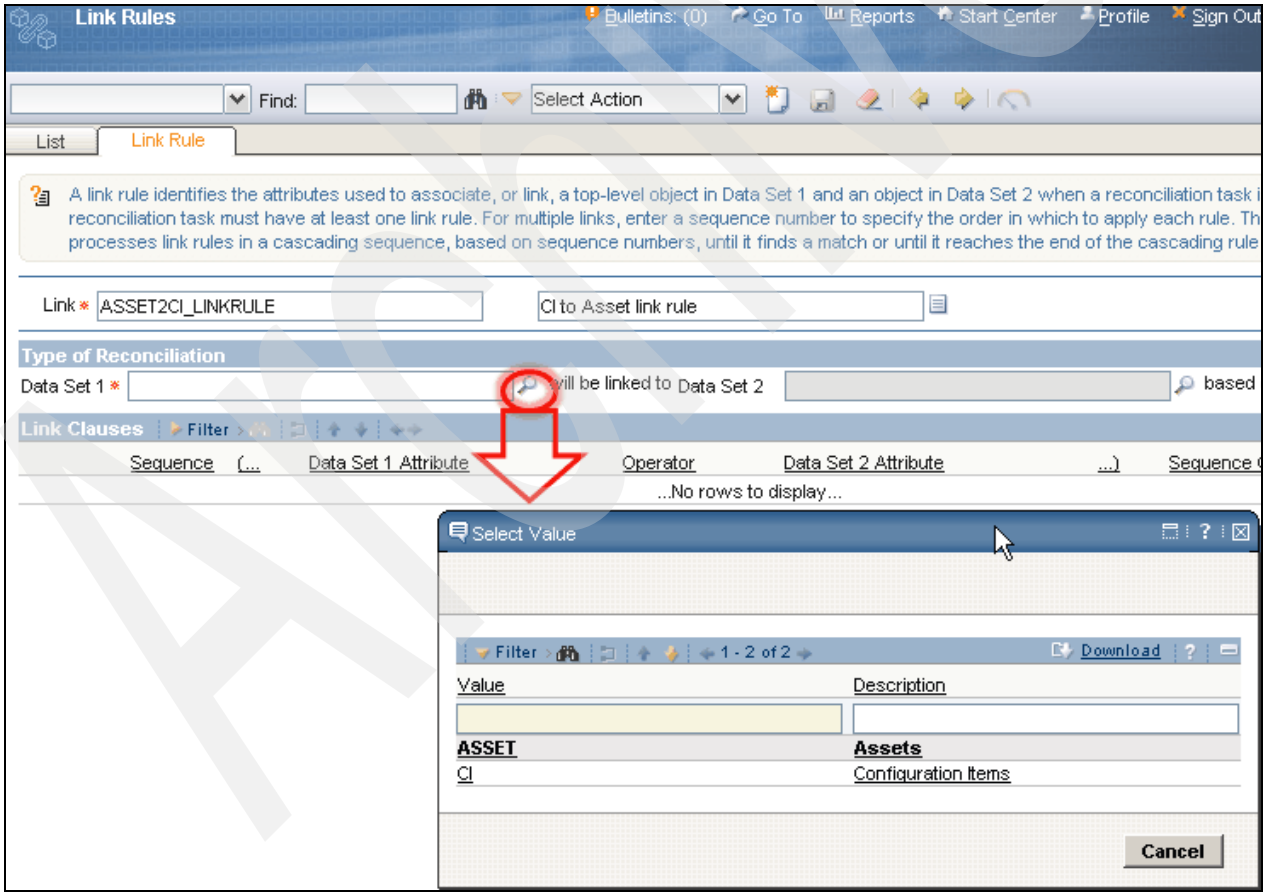


Figure 17-70 Setting the Type of Reconciliation

6. In the Data Set 2 field, click **Select Value** and select **Configuration Item**. See Figure 17-71.

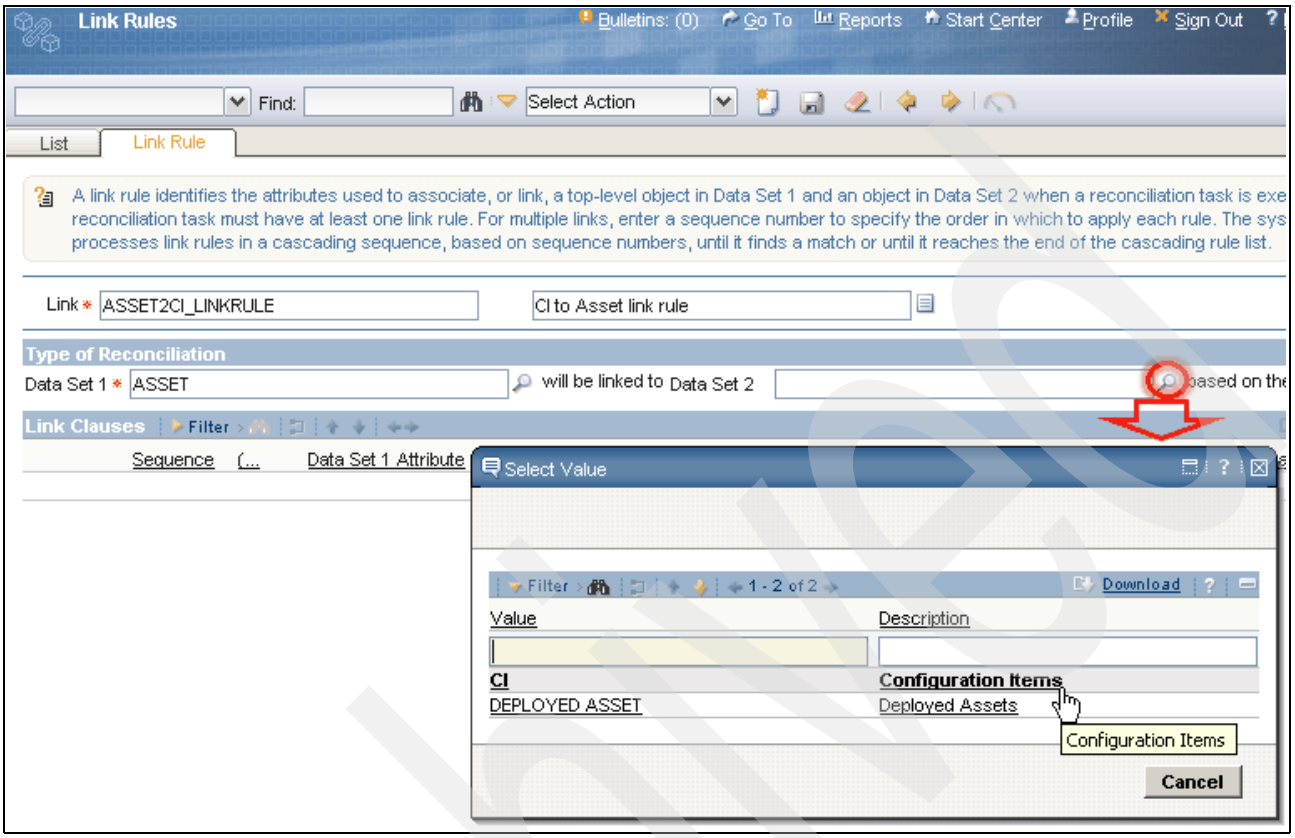


Figure 17-71 Select CI

7. To add a link rule clause, in the Link Clauses table window (Figure 17-72 on page 526), click **New Row**.

> **Tip:** The application generates a sequence number for each new row that is created. By default, the application creates sequence numbers in increments of ten in ascending order. You can modify the numbers as needed or add new clauses with sequence numbers that fall between already-existing numbers. When you save the record, the application sorts the clauses in ascending numerical order.

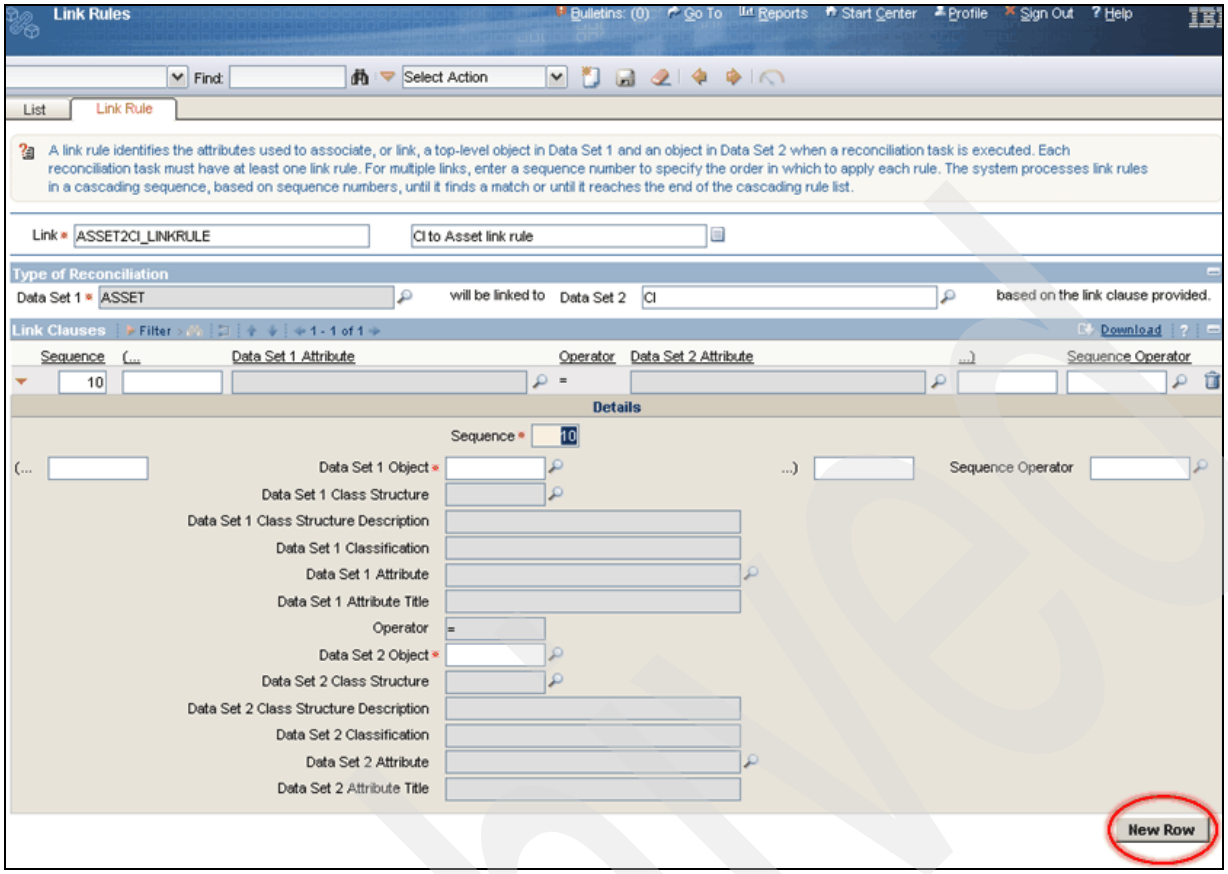


Figure 17-72 Click New Row in Link Rules application

8. If appropriate, in the **(...** field (open parenthesis), enter an open parenthesis character.
9. In the Data Set 1 Object field, click **Select Value** and select **ASSET**. The value ASSET populates the Data Set 1 Object field. See Figure 17-73 on page 527.

Archive

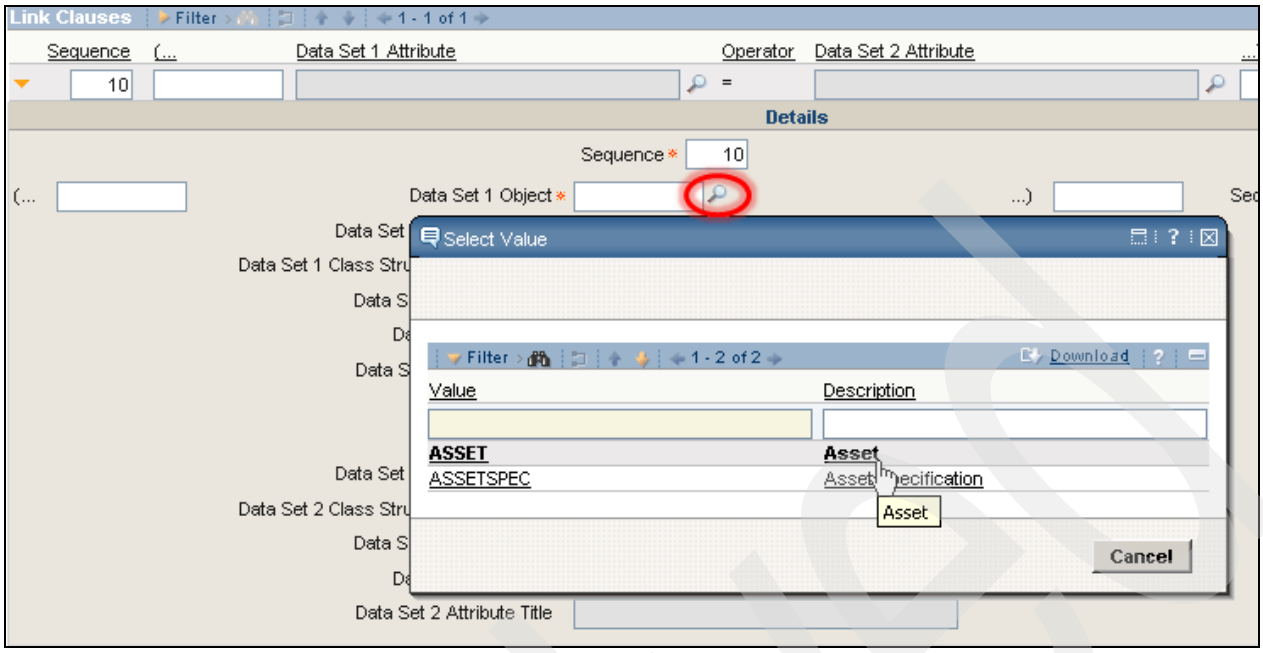


Figure 17-73 Link Clauses

10. In the Data Set 1 Attribute field, click **Select Value** and select the attribute appropriate for the asset that you want to link. The value selected populates the Data Set 1 Attribute field, and the application displays the title of the attribute in the Data Set 1 Attribute Title field. See Figure 17-74.

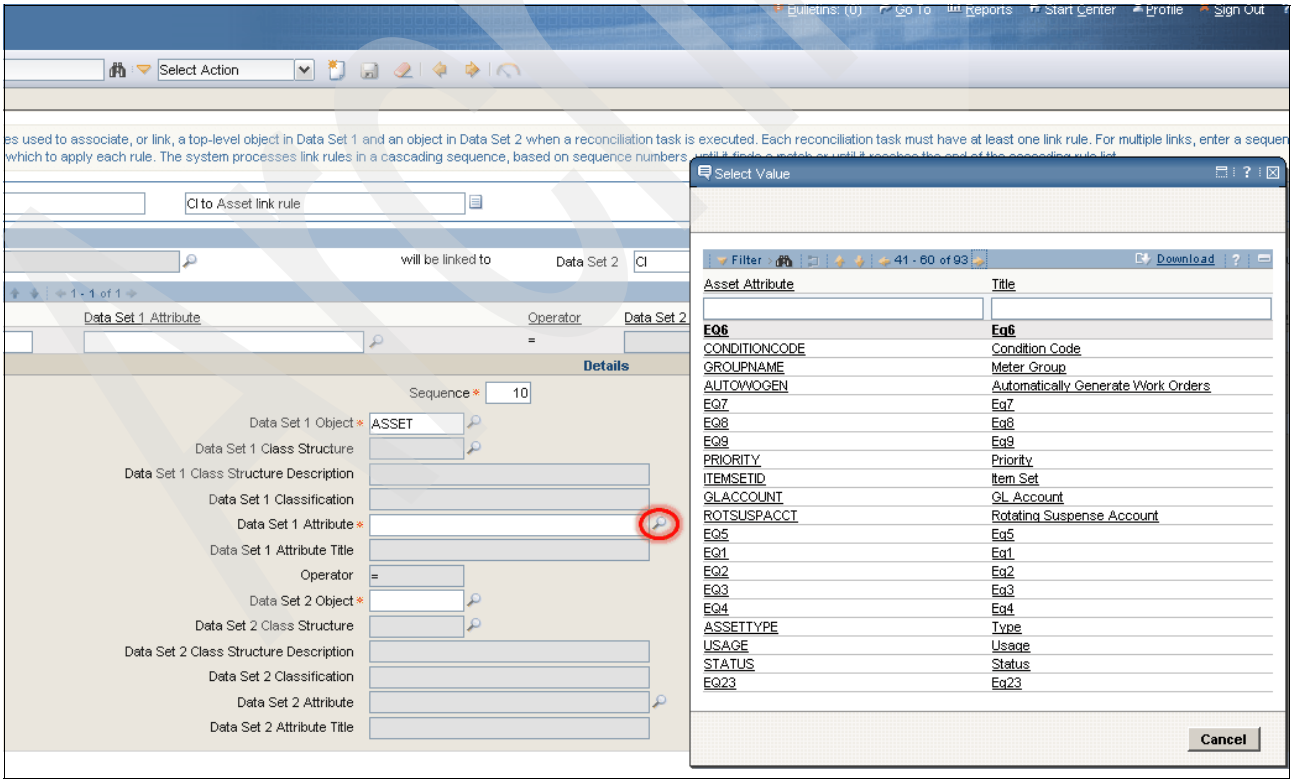


Figure 17-74 Click Select Value

11. In the Data Set 2 Object field, click **Select Value** → **CI**. The value CI populates the Data Set 2 Object field. See Figure 17-75.

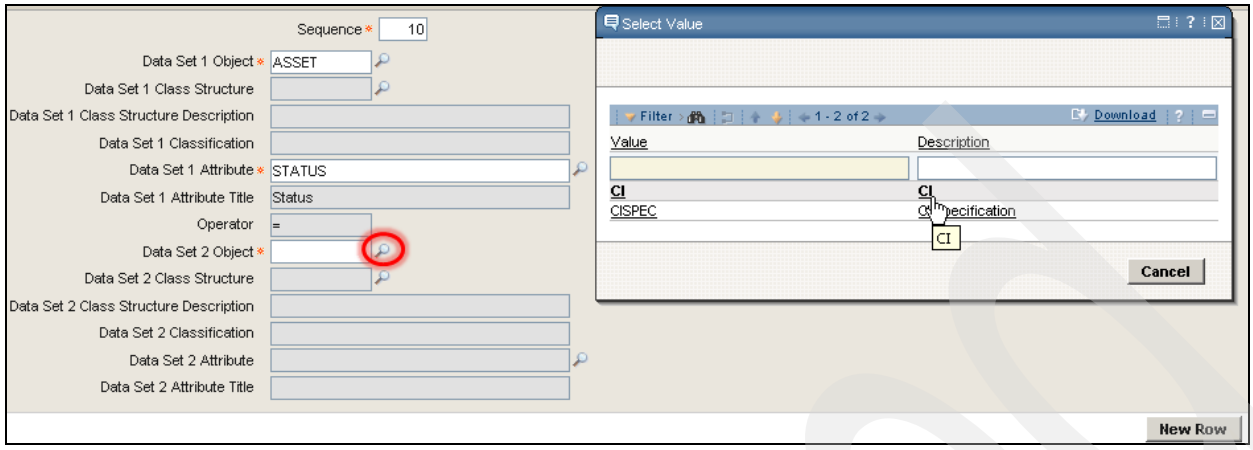


Figure 17-75 Select value for the Data Set 2 Class

12. In the Data Set 2 Attribute field, click **Select Value** and select the CI attribute that corresponds to the attribute you selected for the ASSET. The value selected populates the Data Set 2 Attribute field, and the application displays the title of the attribute in the Data Set 2 Attribute Title field. See Figure 17-76.

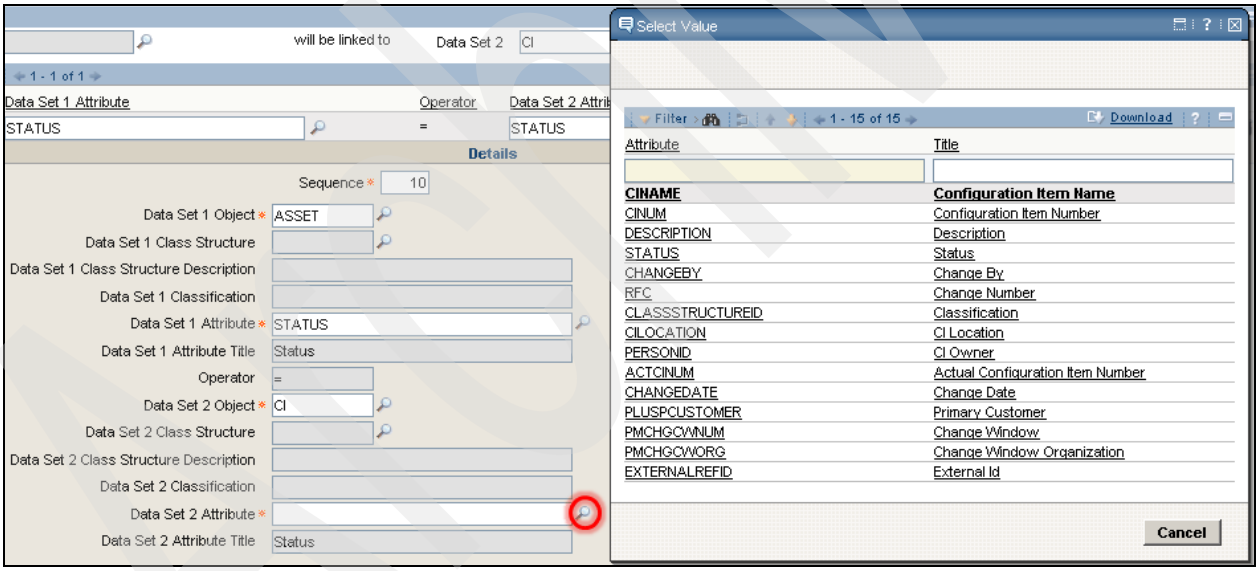


Figure 17-76 Select Value

13. If appropriate, in the **...)** field (close parenthesis), enter a close parenthesis.
14. If you want to add more link rule clauses to the link rule, perform the following steps:
    a. In the Sequence Operator field, click **Select Value**. A Select Value dialog box opens (Figure 17-77 on page 529).
    b. Click to select the appropriate sequence operator. The Select Value dialog box closes, and the selected operator populates the Sequence Operator field. When more than one link rule clause exists, the sequence operator is required. It specifies how the current clause relates to the next clause in the sequence.

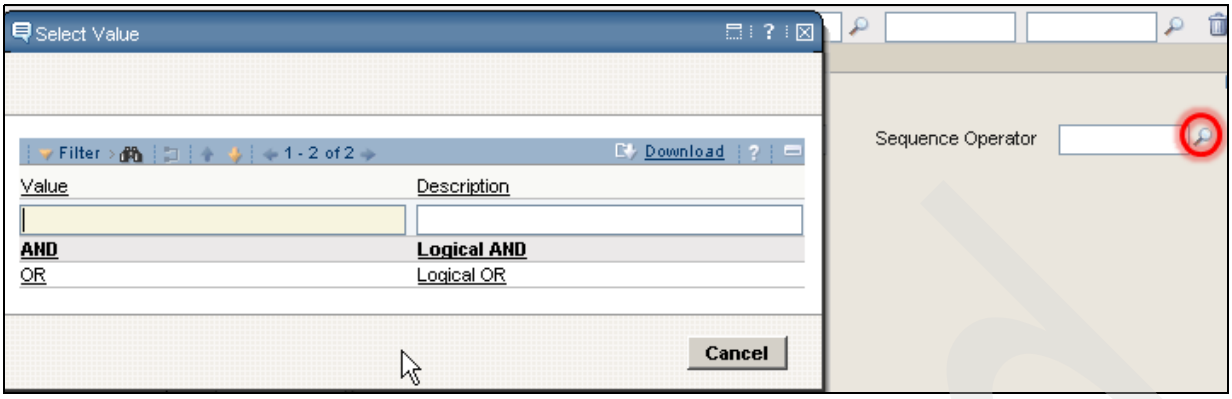


Figure 17-77 Select Value

15. If appropriate, create additional clauses to define the link rule for CI-to-asset linking by clicking **New Row**.

16. When you have entered all clauses, click **Save Link Rule**. See Figure 17-78.

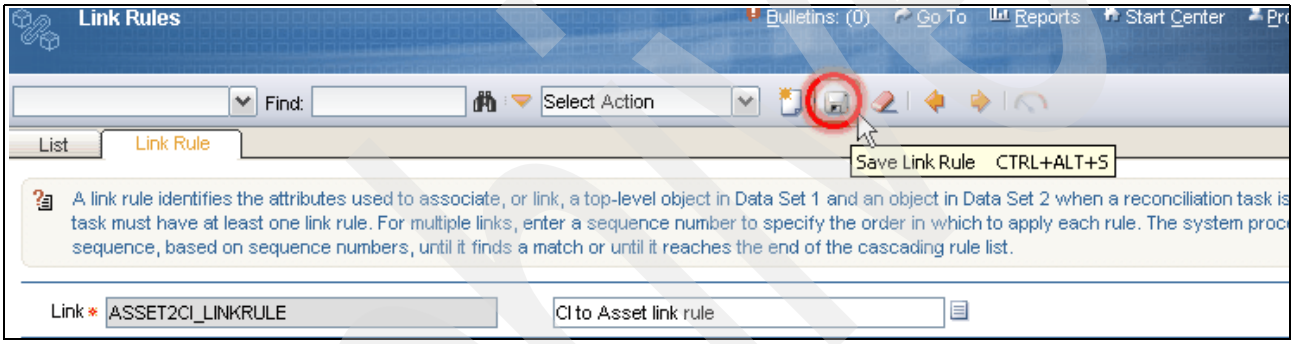


Figure 17-78 Saving the Link Rule created

Creating a reconciliation task

To create a reconciliation that links CIs with authorized IT assets, you must create a reconciliation task.

To set up a reconciliation, you define a task in the Reconciliation Tasks application.

The reconciliation task combines the elements required for the reconciliation into a specific task that you can execute. A task consists of three possible components: a task filter (optional), one or more link rules (required), and one or more comparison rules (optional). For the CI-to-asset linking, only a link rule is required.

Archive

In the reconciliation task, you also define the schedule for running the reconciliation, and you activate the schedule:

1. On the navigation bar in Asset Management for IT, click **Go To** → **Administration** → **Reconciliation** → **Reconciliation Tasks**. See Figure 17-79.

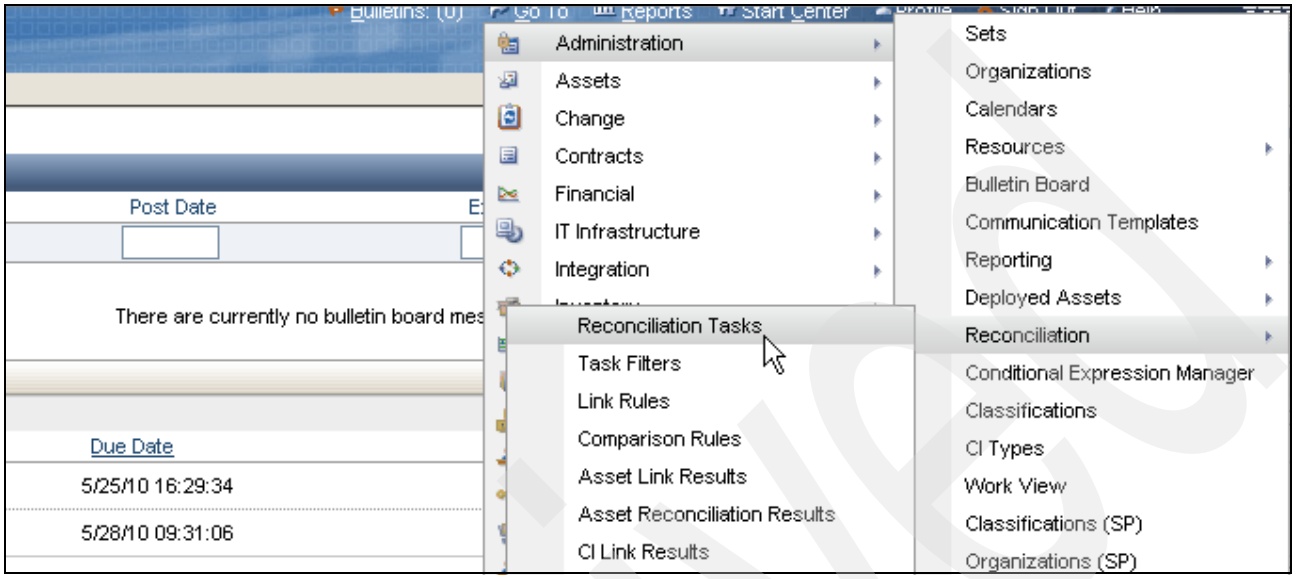


Figure 17-79 Reconciliation Tasks application

2. On the toolbar in the Reconciliation Tasks application, click **New Reconciliation Task**. See Figure 17-80.

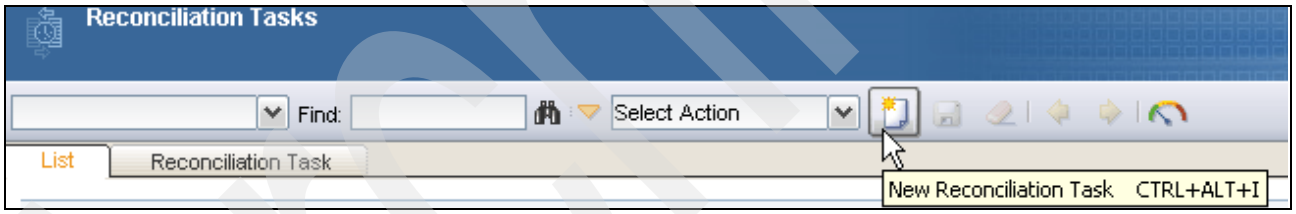


Figure 17-80 New Reconciliation Task

3. In the Reconciliation Task field, type the name that you want to assign to the reconciliation task, for example `ASSET2CI_RECONTASK`. This name must be unique.

4. In the Description field, type a brief description of the link rule. See Figure 17-81.

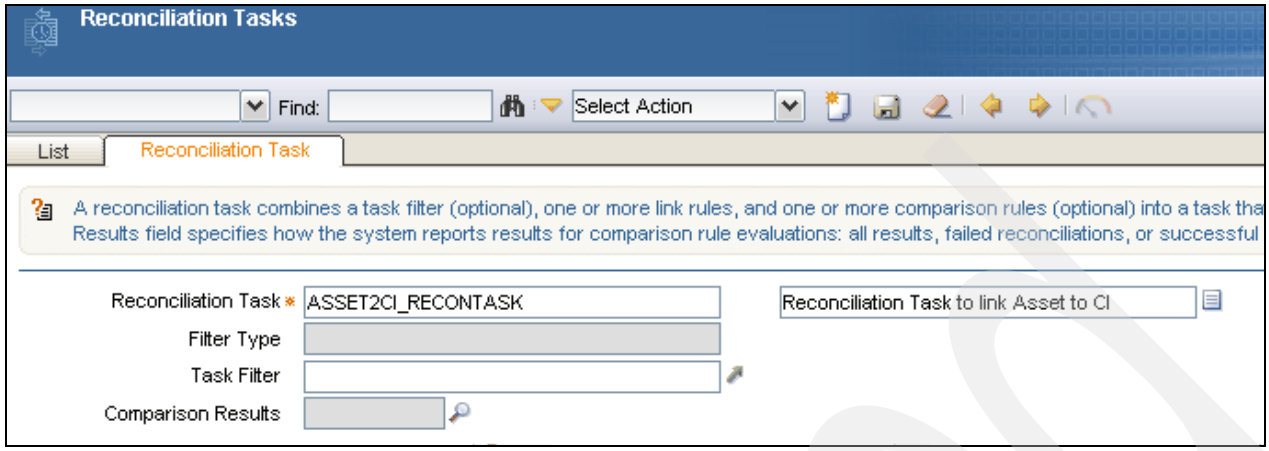


Figure 17-81 Reconciliation Task name

5. In the Type of Reconciliation table window, click **Select Value** and select **ASSET**.
6. In the Data Set 1 field of the Data Set 2 field, click **Select Value** and select **CI**. See Figure 17-82.

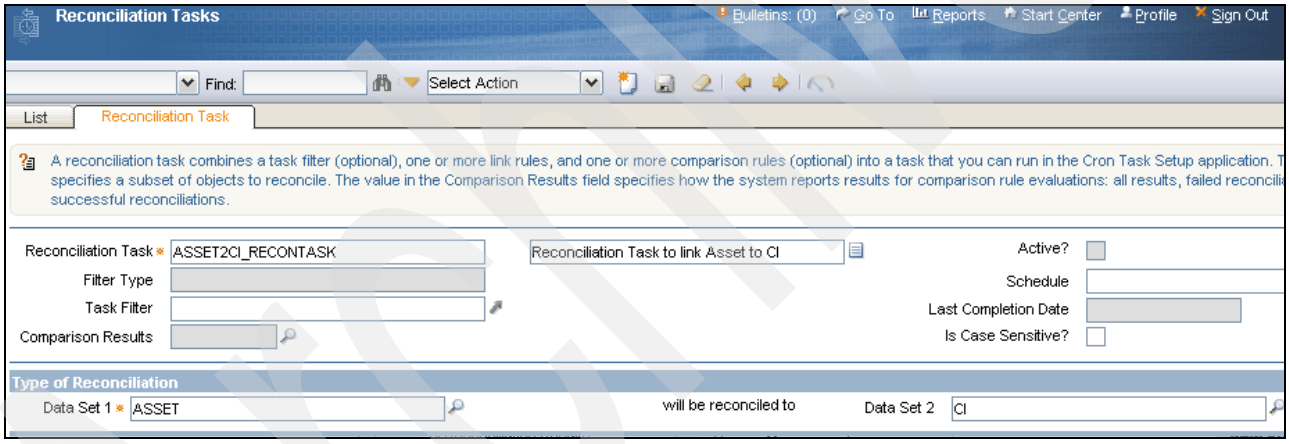


Figure 17-82 Check the Type of Reconciliation topic

7. In the Link Rules table window, click S**elect Link Rule** and, from the Select Link Rule window, select the link rule that you created for linking CIs to IT assets. Click **OK**. See Figure 17-83 on page 532.

Archive

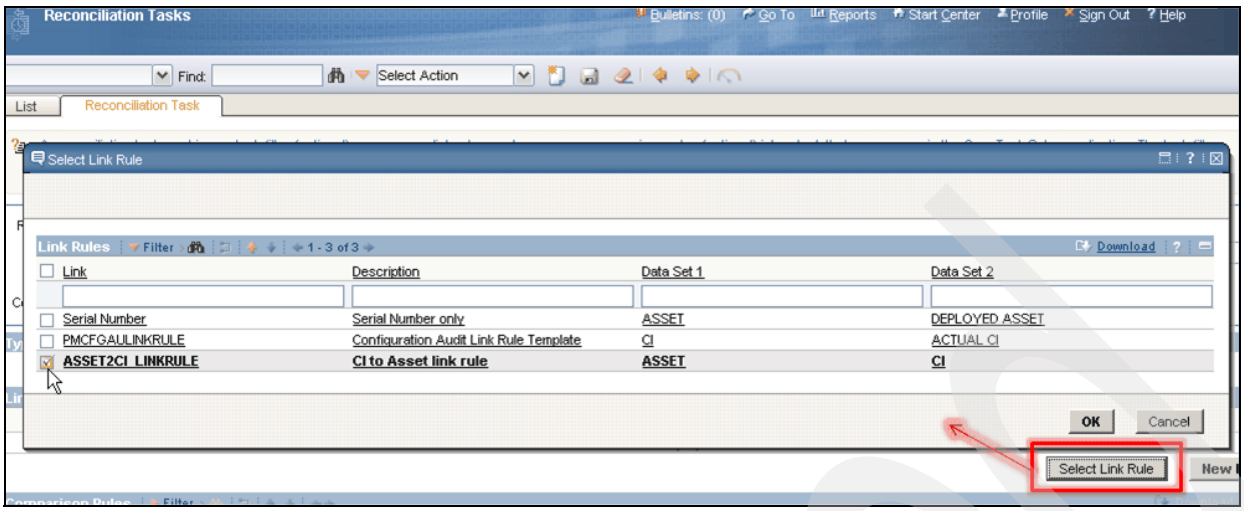


Figure 17-83 Select Link Rule

8. To schedule execution of the reconciliation task, complete the following steps:
   a. In the Schedule field, click **Set Schedule**.
   b. In the Select Value window, set the schedule for executing the reconciliation task, and click **OK**. See Figure 17-84.

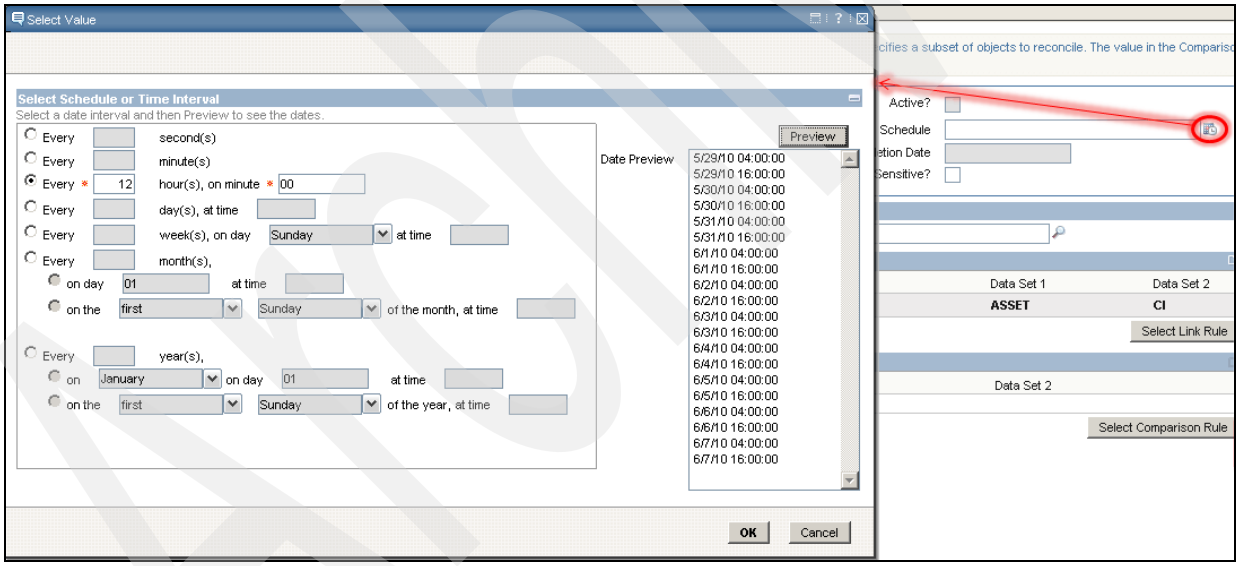


Figure 17-84 Set a Schedule

   c. Click **Save** to save the Reconciliation.
9. To activate this task, perform the following steps
   a. Click **Go To** → **System Configuration** → **Cron Task Setup**.
   b. Look for **ReconciliationCronTask**. See Figure 17-85 on page 533.

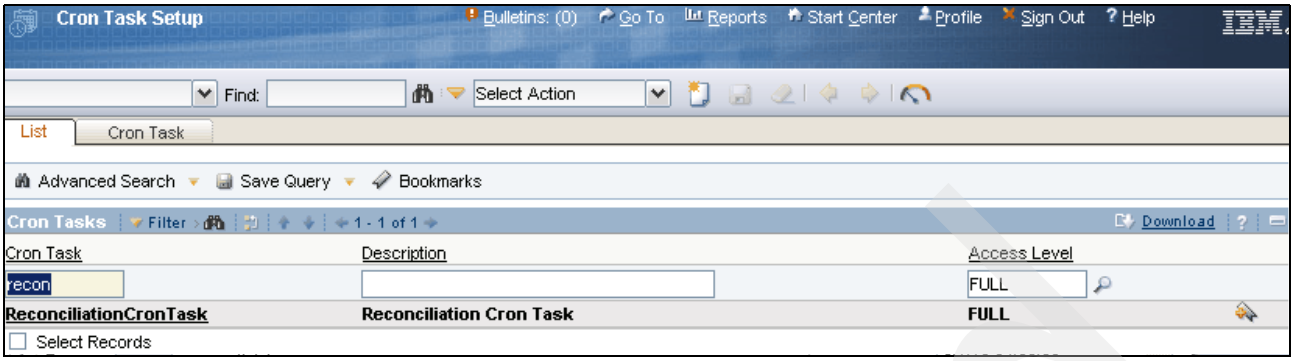


Figure 17-85 Reconciliation Task cron task

c. In the Cron Task Instance, find the appropriate name for the Reconciliation Task you have just created. If Active is not enabled, select the **Active** check box. See Figure 17-86.

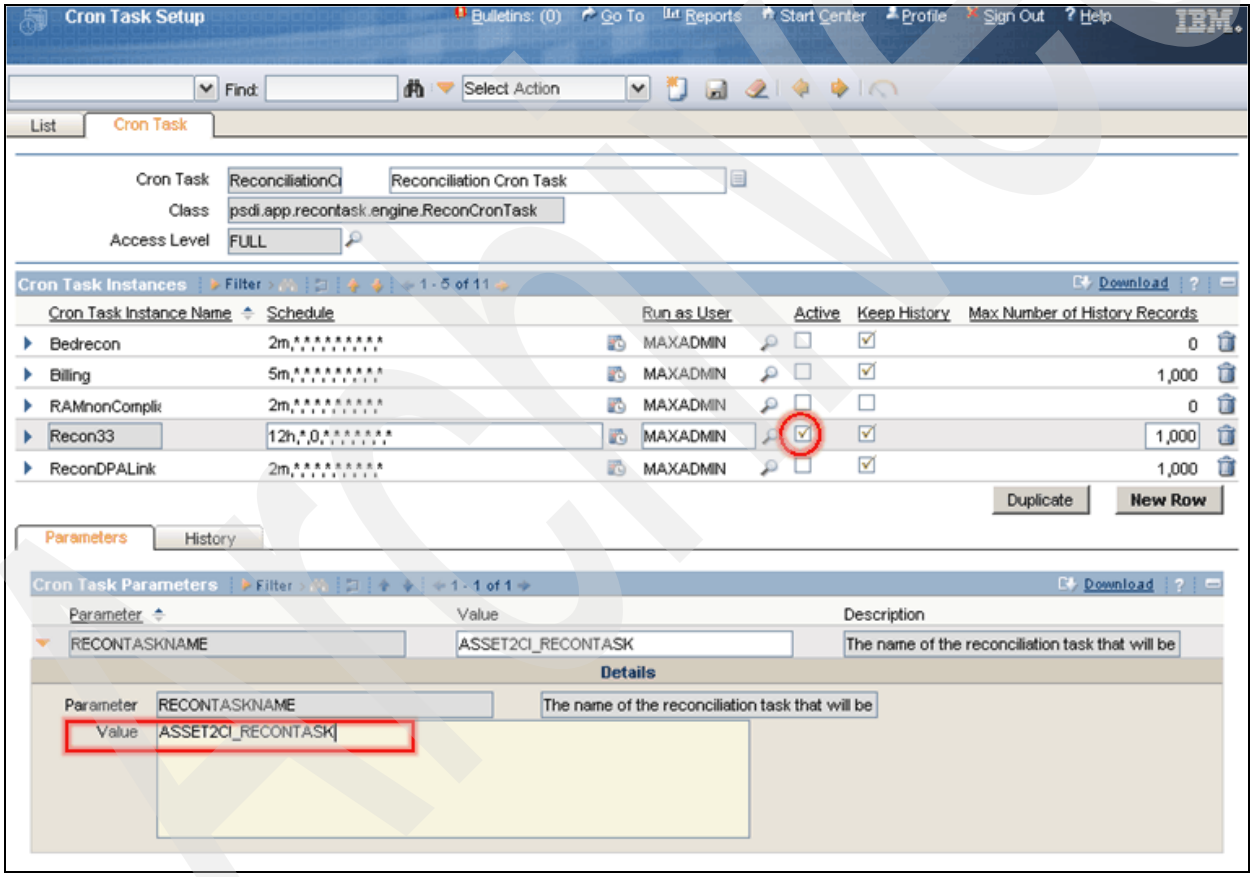


Figure 17-86 Cron Task Instance

d. Click **Save Cron Task Definition**. After it is saved, click **Select Action** → **Reload Request**. See Figure 17-87 on page 534.

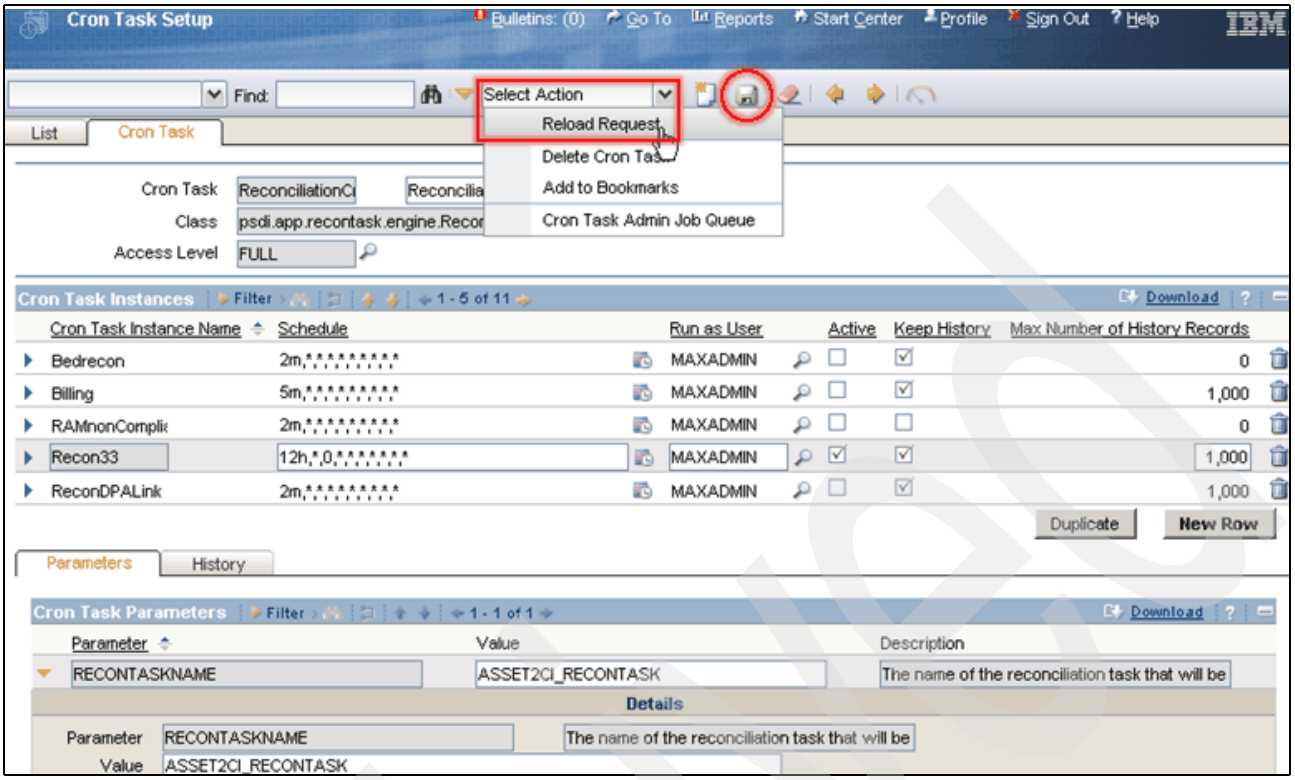


Figure 17-87 Reloading the request

e. Select the appropriate Cron Task Instance and click **OK**. See Figure 17-88.

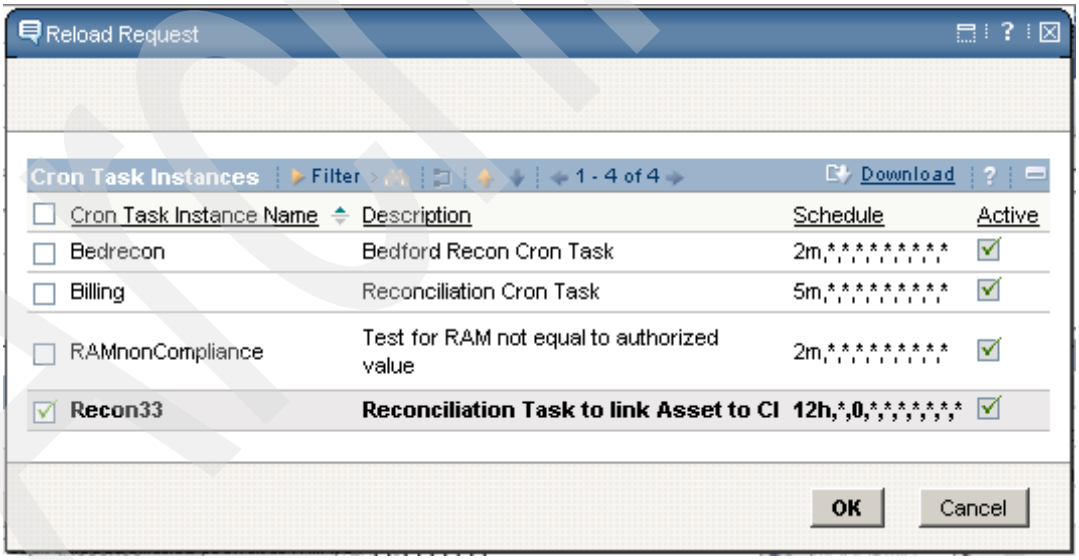


Figure 17-88 Click OK

17.6.4 Specifying the CI to reconciliation link relationship

Before you link CIs to assets, in the Database Configuration application, create a relationship between the CI table and the reconciliation link:

1. On the navigation bar in Asset Management for IT, click **Go To** → **System Configuration** → **Platform Configuration** → **Database Configuration**. See Figure 17-89.

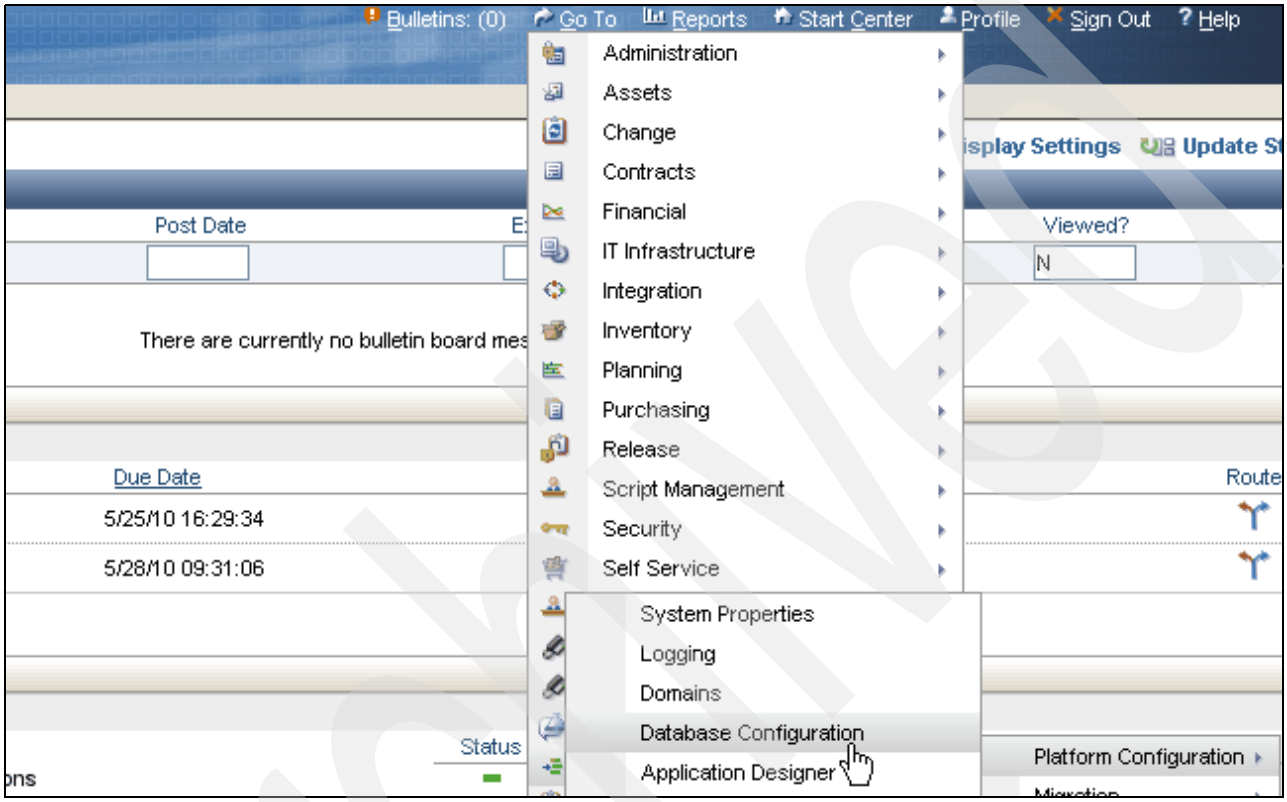


Figure 17-89 Database Configuration application

2. In the Objects table window on the List tab, select CI (Configuration Item table). See Figure 17-90

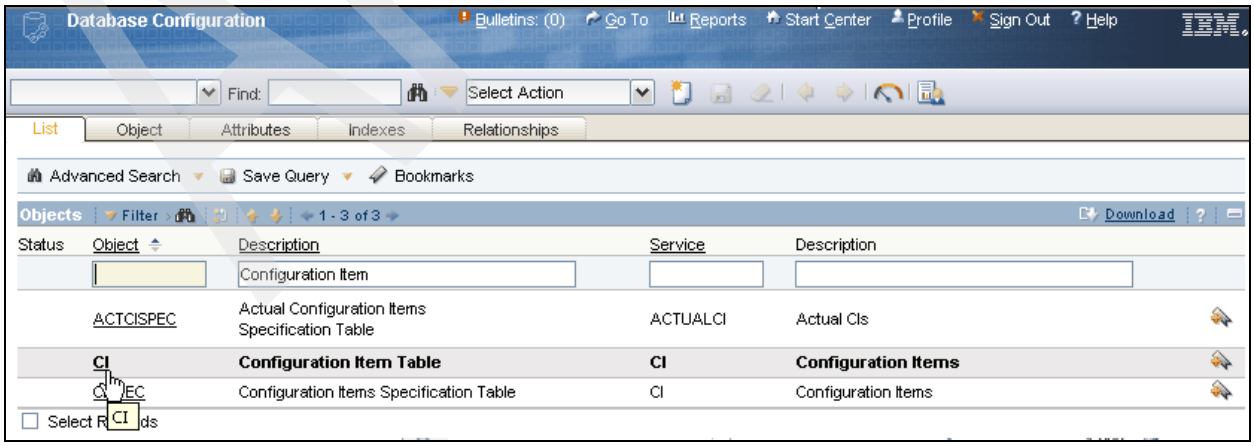


Figure 17-90 Select CI

3. On the Relationships tab, click **New Row**.
4. In the Relationship field, specify a relationship name, for example `CI_RECONLINK`.
5. In the Child Object field, click **Select Value** and specify `RECONLINK`
6. In the Where Clause field, specify the following clause:

   ```
   nodeid = :ciid and recontype = 'ASSET' and compset = 'CI'.
   ```
7. Optional: In the Remarks field, enter additional remarks about the relationship.
8. Verify that configuration is successful, as Figure 17-91 shows, and click **Save Object**.

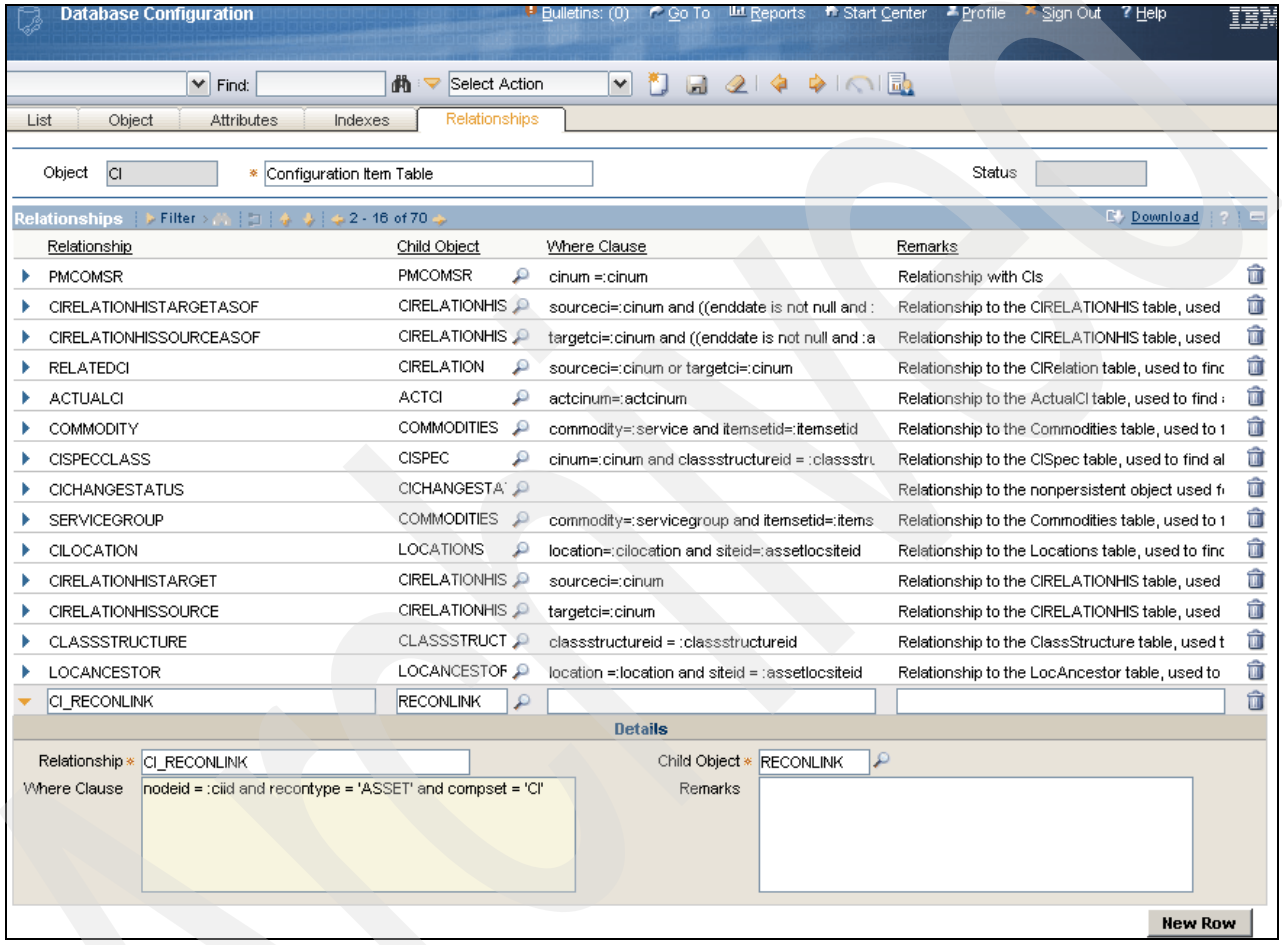


Figure 17-91 Save the Object

17.6.5 Creating the action for the escalation for CI-to-asset link

The next step to have the integration working is to create an action for the escalation that we subsequently create:

1. On the navigation bar in Asset Management for IT, click **Go To** → **SystemConfiguration** → **Platform Configuration** → **Actions**. See Figure 17-92.

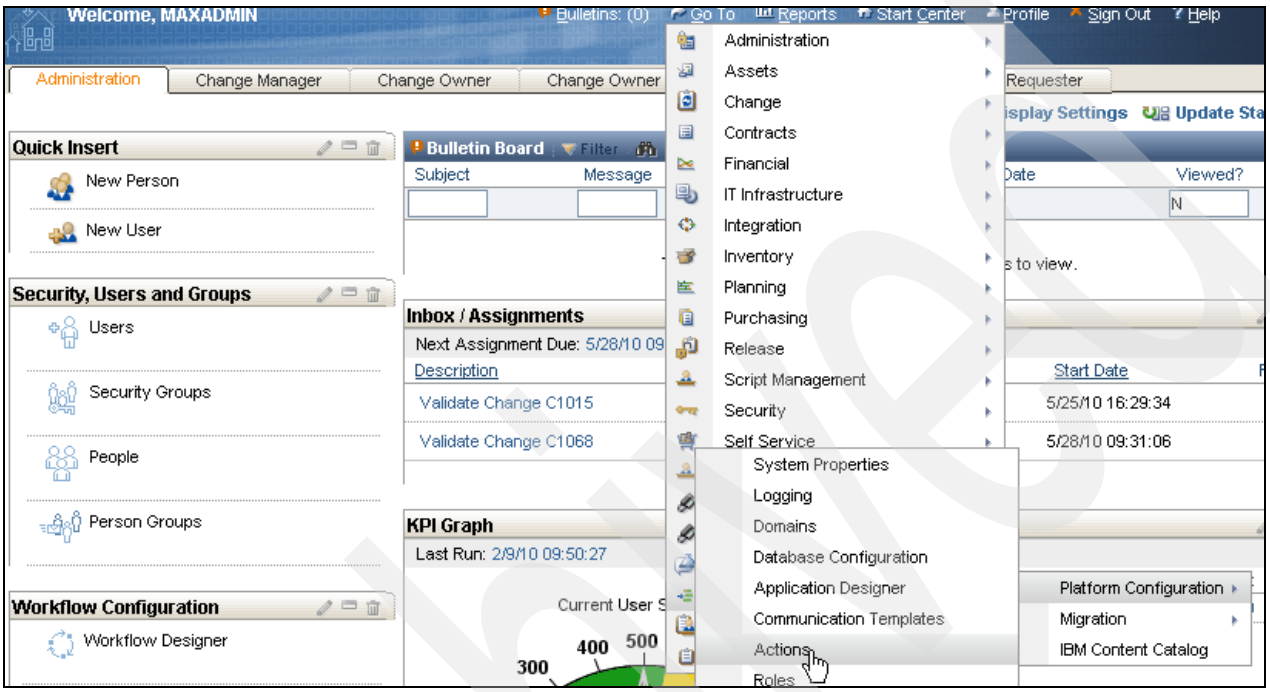


Figure 17-92 Actions application

2. On the toolbar in the Actions application, click **New Action**. See Figure 17-93.

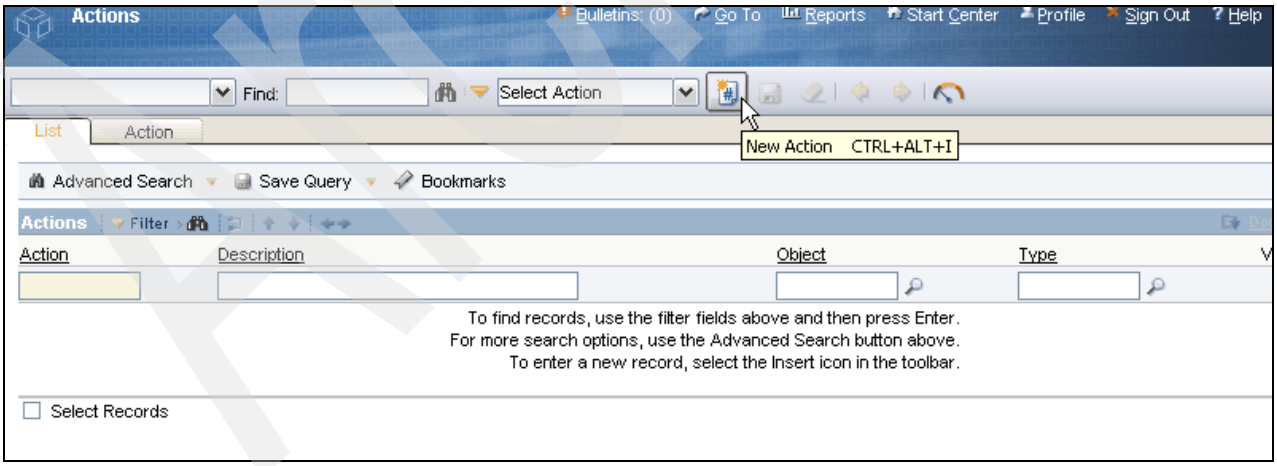


Figure 17-93 New action

3. In the Action field, specify a name for the action. If your enterprise automatically numbers the actions, specify text that identifies the type of action in the Description field, for example `LINK_CI2ASSET_RULES`.
4. In the Object field, click **Select Value** and specify CI (Configuration Item table).

5. In the Type field, select **Set Value**. See Figure 17-94.

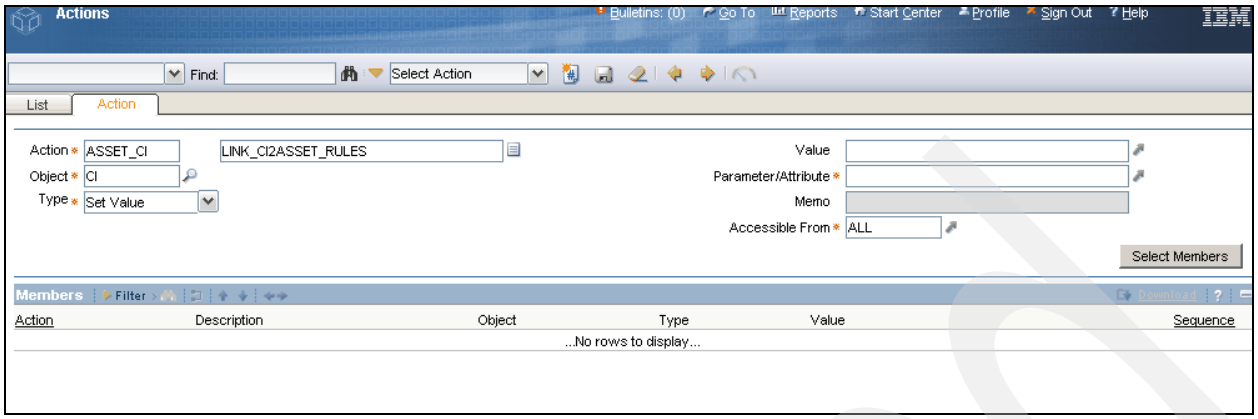


Figure 17-94 Actions application

6. In the Value field, specify the following value, where <*relationship_name*> is the name of the relationship that you defined:

   `:<relationship_name>.assetnum`

7. In the Parameter/Attribute field, click **Detail Menu** and select **assetnum** from the Select Fields window. See Figure 17-95.

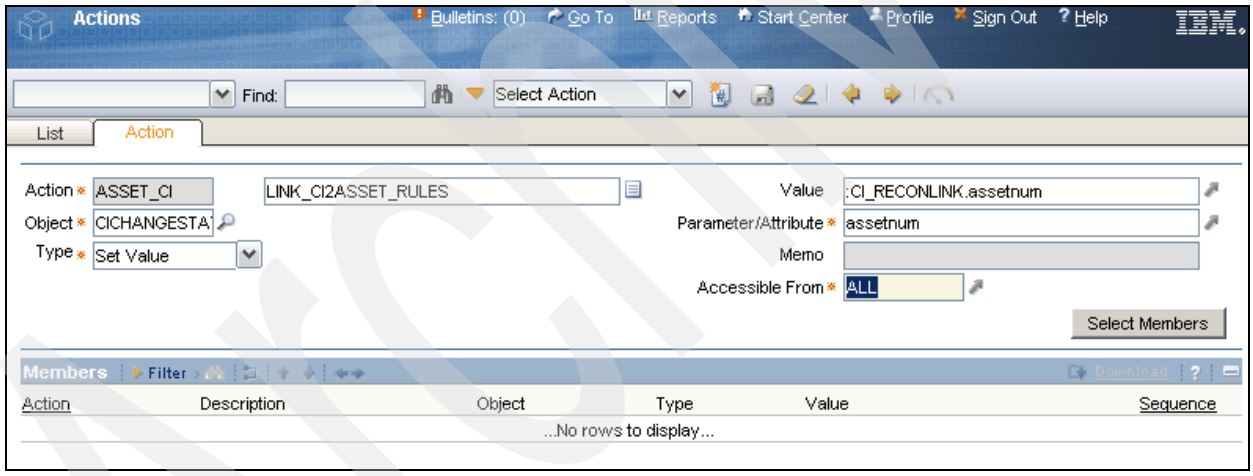


Figure 17-95 Parameter / Attribute field from the Action

8. Click **Save Action**.

17.6.6 Configuring the escalation to link CIs with IT assets

When you define an escalation, you must associate it with a specific record, in this case CI records. You can then define conditions for specific attributes of a record and have the system trigger an escalation when those conditions are met. An escalation point represents a condition or threshold that must be met, and you can use one or more of these to define your escalation.

When you create an escalation point, you must associate at least one action or notification to it before you can activate the escalation. An action is an event you want the system to trigger

when it finds records that meet the condition (or conditions) defined by the escalation point, such as changing a status or an owner.

For CI-to-asset linking, you define an escalation that adds an asset number to a CI record when a match is found between a CI and an IT asset:

1. On the navigation bar in Asset Management for IT, click **Go To** → **System Configuration** → **Platform Configuration** → **Escalations**. See Figure 17-96.

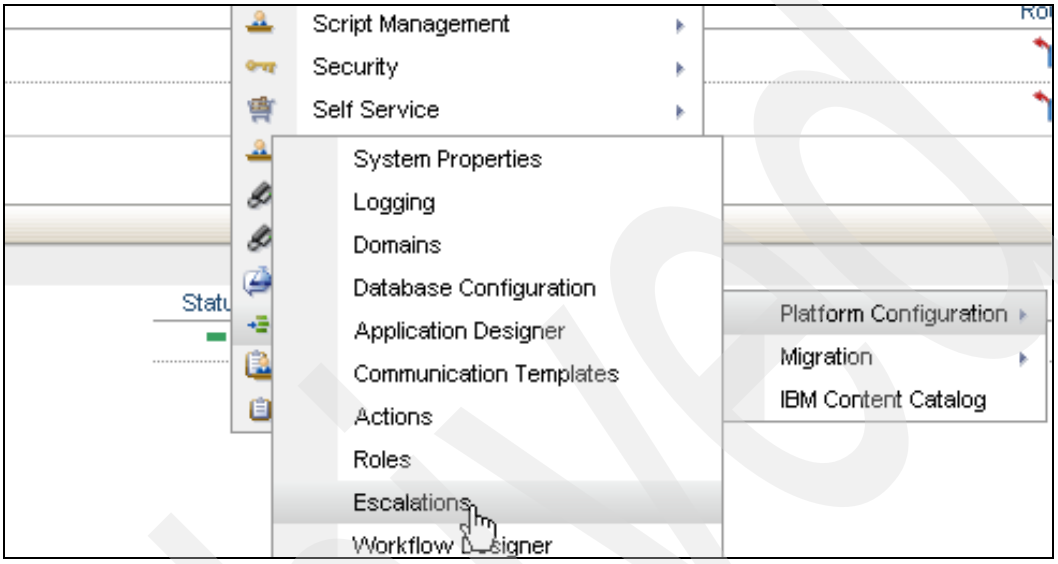


Figure 17-96 Acessing Escalations application

2. On the toolbar in the Escalations application, click **New Escalation**.
3. In the Escalation field, specify a name for the escalation. If your enterprise automatically numbers escalations, specify text that identifies the escalation in the Description field, for example CI2ASSET_RULES.

Archive

4. In the Applies To field, click **Select Value** and specify `CI` (Configuration Item table). See Figure 17-97.

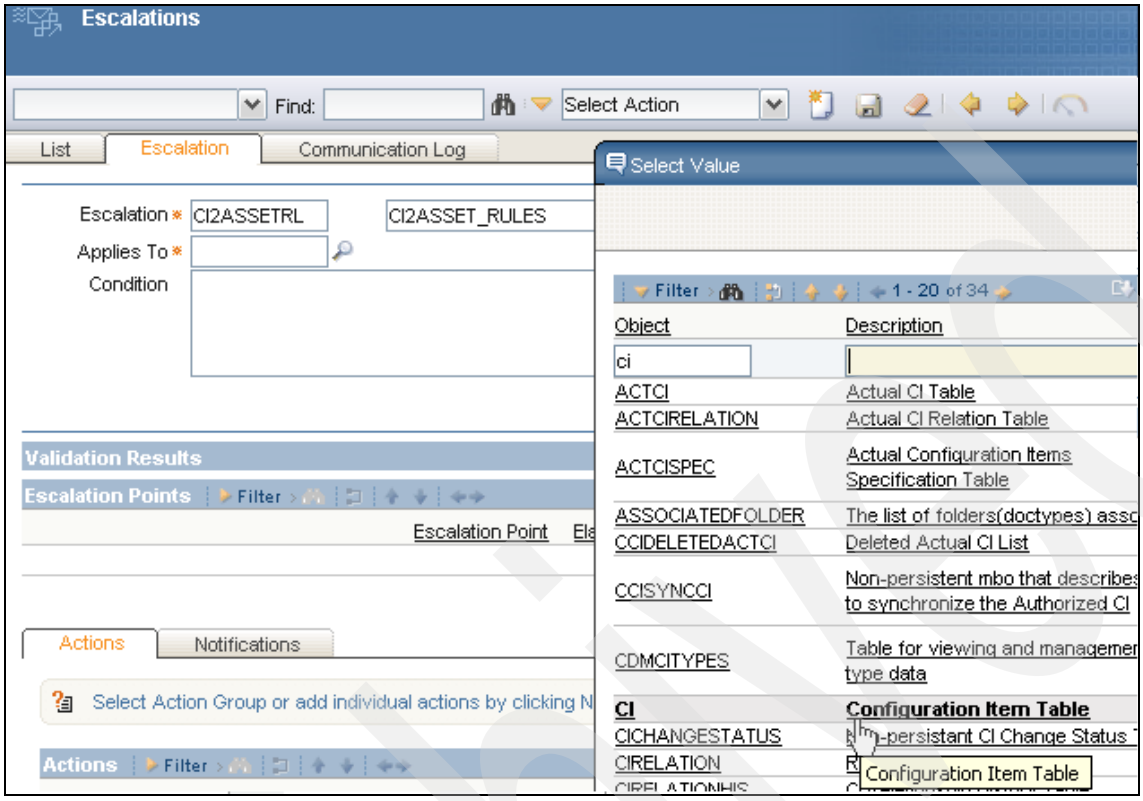


Figure 17-97 Select CI

5. In the Condition field, specify `assetnum is null`. See Figure 17-98.

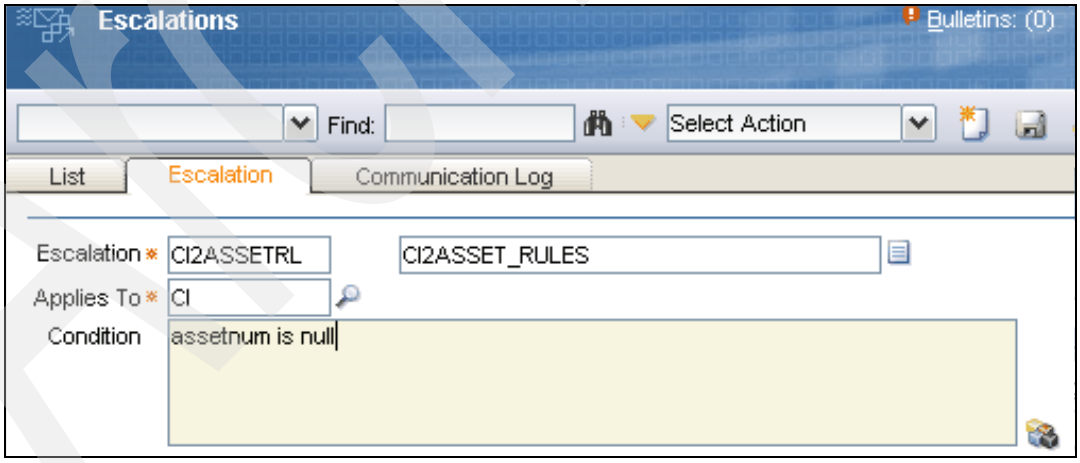


Figure 17-98 Condition of the Escalation

6. In the Schedule field, click **Set Schedule** and specify a schedule for running the escalation. Set the escalation schedule to an interval greater than the reconciliation task schedule. See Figure 17-99.

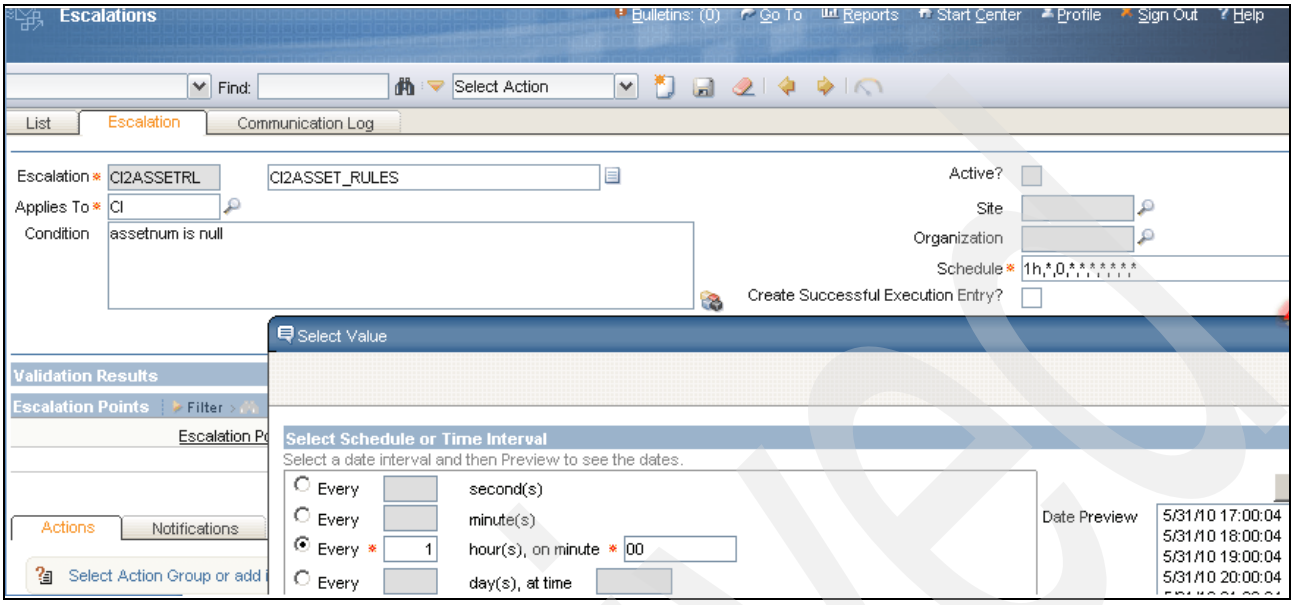


Figure 17-99 Setting the Schedule

7. In the Escalation Points section, click **New Row**. See Figure 17-100.
8. In the Escalation Point Condition field, specify an escalation point condition, for example:

   1 = 1
9. Select the **Repeat** check box.

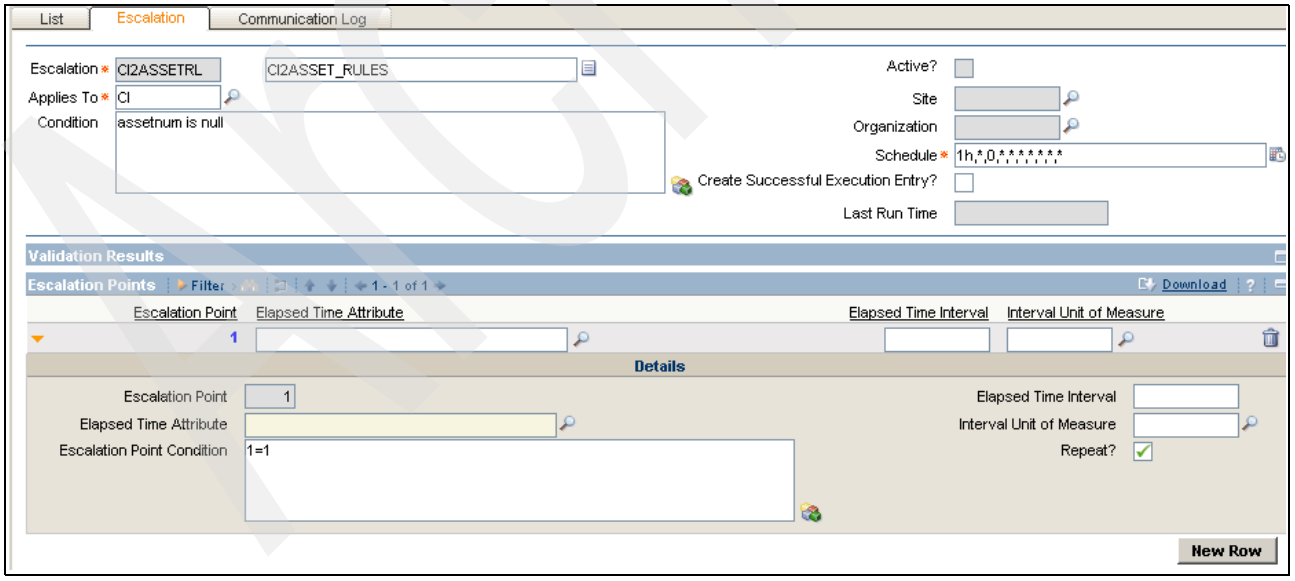


Figure 17-100 Check the Repeat

10. On the Actions subtab, in the Action field, click **Select Value** and select the name of the action that you created for this escalation.

11. Click **Save Escalation**. See Figure 17-101.

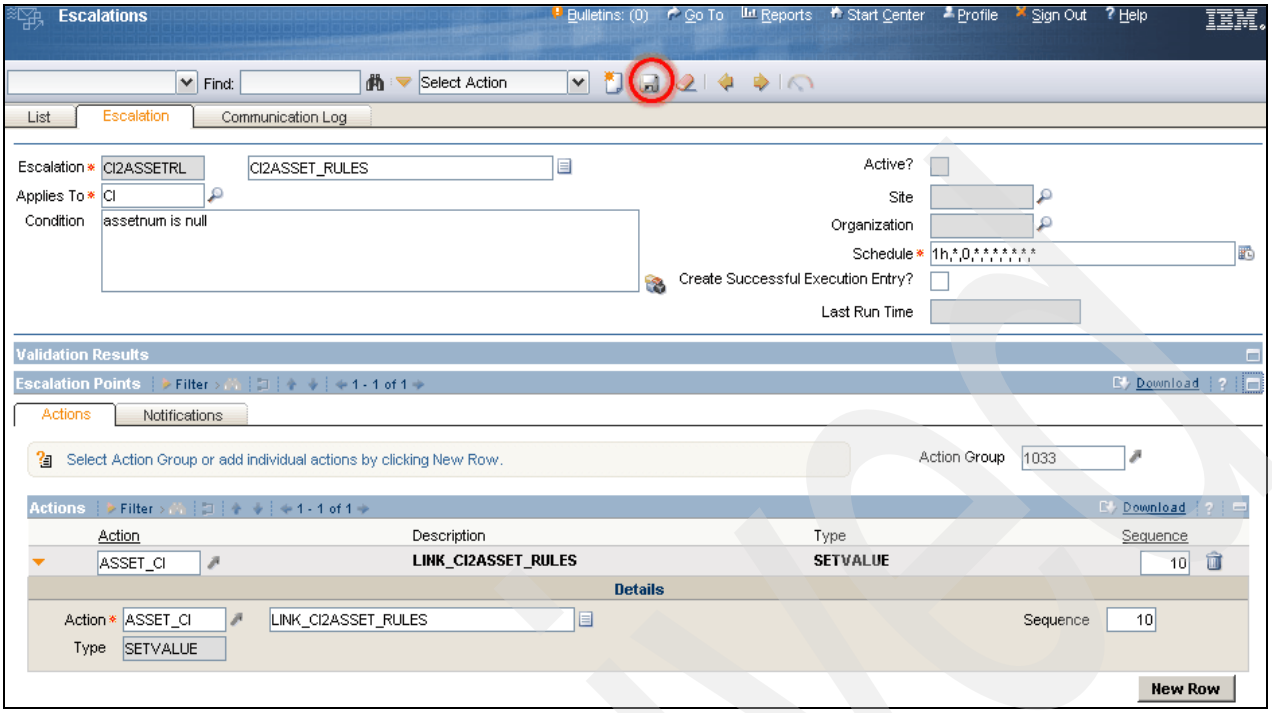


Figure 17-101 Click Save

12. To activate the escalation, click **Select Action** → **Activate/Deactivate Escalation**. See Figure 17-102.

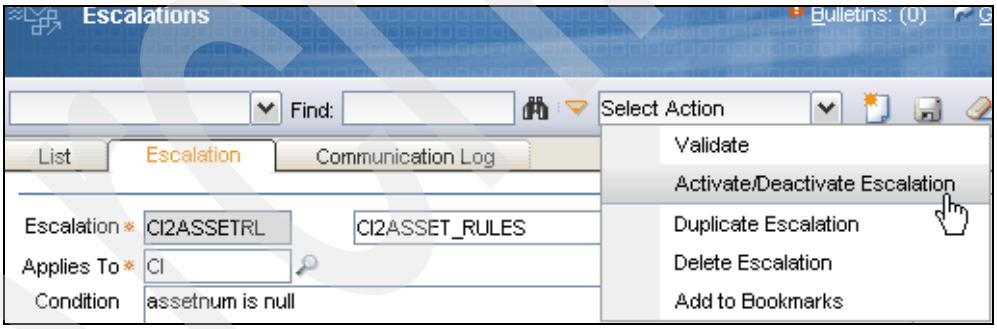


Figure 17-102 Activate the Escalation

Now you have automatically configured a response to events. For example, an escalation automatically links CIs to assets if the appropriate conditions are met.

17.7 Summary

See Figure 17-103 for the summary of this integration.

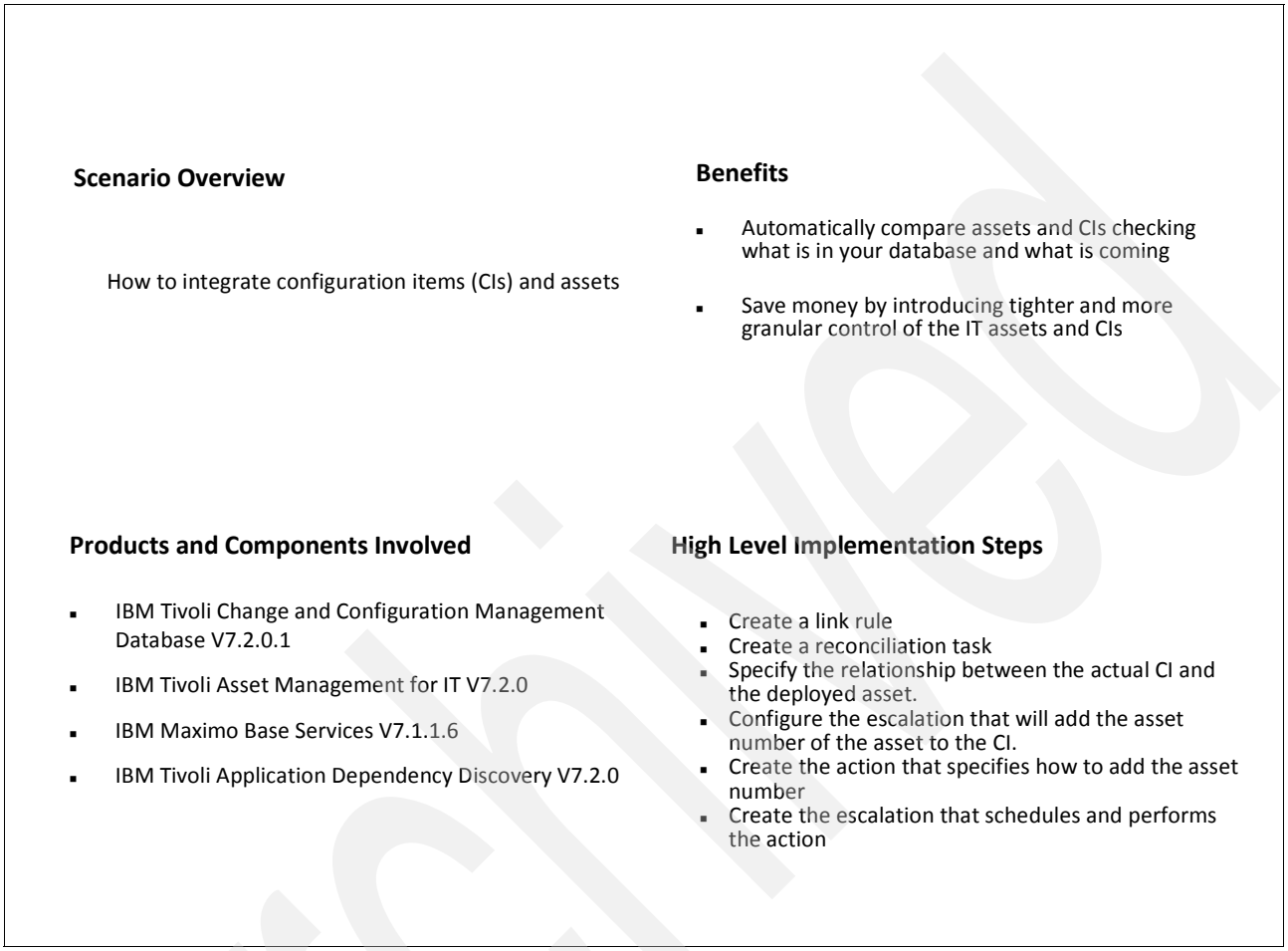


Figure 17-103 Quick summary

Archived

Archived

18

Rational Asset Manager and Tivoli Change and Configuration Management Database integration

This chapter covers the integration of IBM Rational Asset Manager and IBM Tivoli Change and Configuration Management Database (CCMDB). Using this integration, you can automatically create assets in Rational Asset Manager from existing configuration items (CIs) in the CCMDB. You can also use Rational Asset Manager as a definitive software library (DSL) for storing software images.

This chapter contains the following topics:

- 18.1, “Rational Asset Manager” on page 546
- 18.2, “Definitive Media Library” on page 546
- 18.3, “Scenario overview” on page 547
- 18.4, “Products involved” on page 548
- 18.5, “Benefits” on page 549
- 18.6, “Implementation” on page 549
- 18.7, “Summary” on page 593

© Copyright IBM Corp. 2011. All rights reserved.

18.1 Rational Asset Manager

Many software development organizations do not have a central place to store their reusable, high-value assets; therefore, they often store an asset as just another version of an artifact among the millions of other versioned artifacts in a software configuration management (SCM) system. Sometimes, they resort to storing the assets on a web server or on a wiki site. Even when companies provide a consistent place to store and search for these assets, the metadata that describes them is typically either inconsistent or nonexistent. Such makeshift arrangements can cause difficulty in finding, assessing, and reusing assets quickly. Rational Asset Manager ends this confusion by providing a central place to store reusable, high-value assets.

Rational Asset Manager also enables you to track versioned artifacts back to the assets that were used to create them. This traceability can help developers more easily maintain existing assets and create new versions, which saves them time, thus saving the company money.

18.2 Definitive Media Library

Definitive Media Library (DML), as defined by ITIL, is one or more locations in which the definitive and approved versions of all software configuration items (CIs) are securely stored. It may also contain associated CIs such as documentation, licences, and software keys

Note: In this scenario, we use the DML term within the context of CCMDB and IBM Tivoli Release Process Manager.

The DML is a single logical storage area, even if multiple locations exist. All CIs in the DML are under the control of CCMDB. See Figure 18-1 on page 547 to better understand the DML and interaction with CCMDB.

Archived

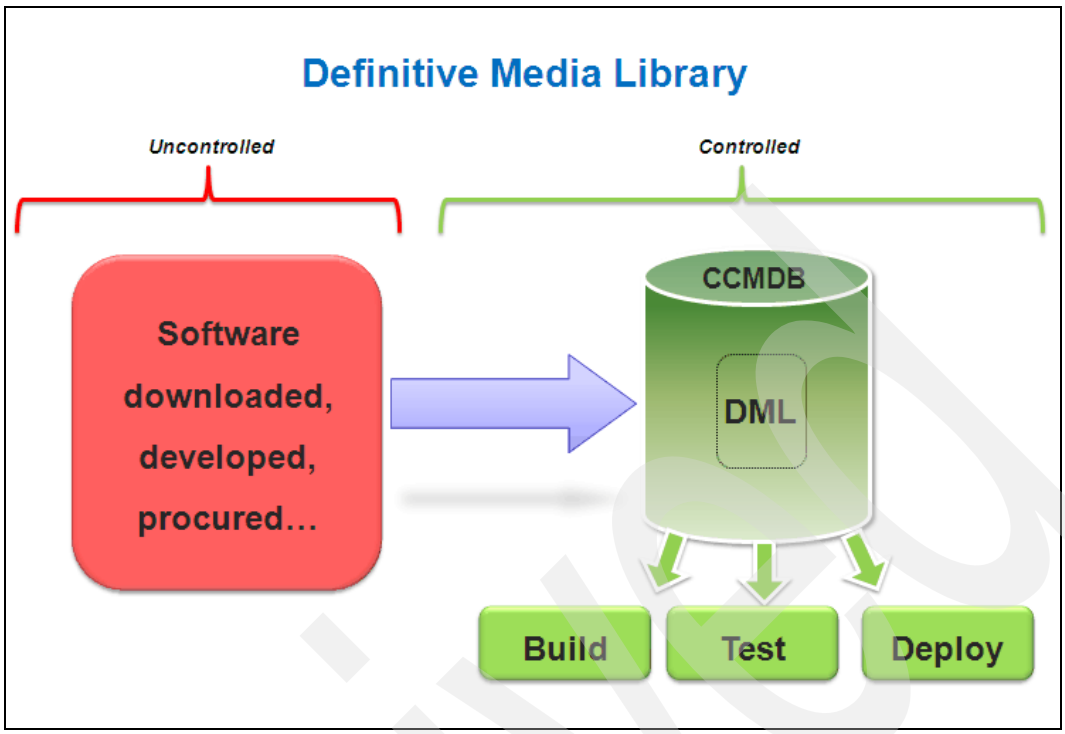


Figure 18-1 Definitive Media Library

18.3 Scenario overview

Figure 18-2 shows the architectural diagram of the scenario.

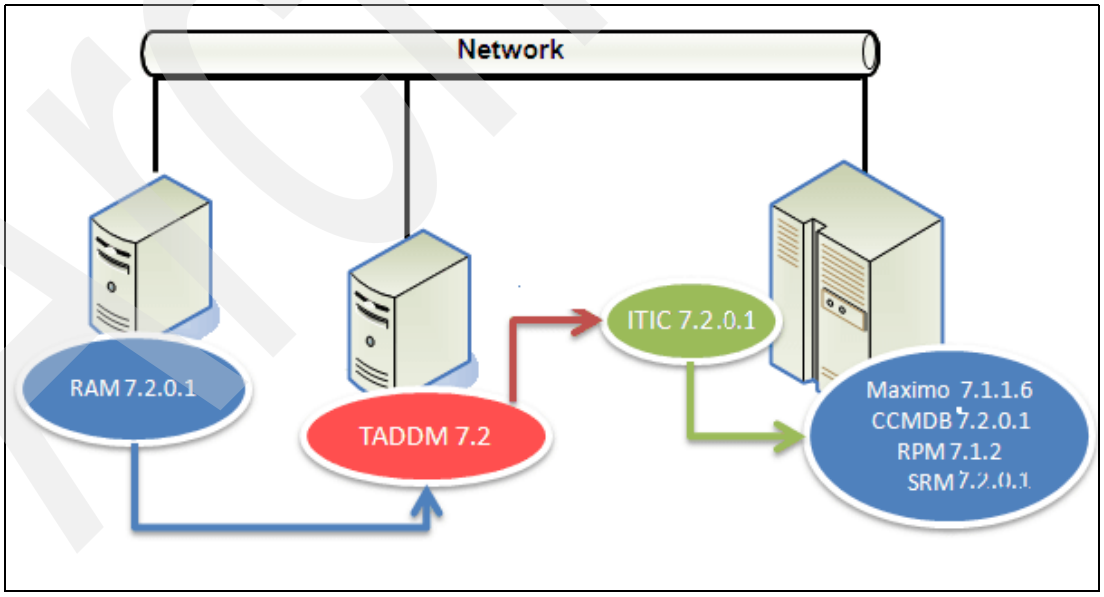


Figure 18-2 Scenario architecture

Note: For more information about the architecture of this integration, see *How to integrate IBM Rational Asset Manager with your IBM Tivoli database,* by Ramin Sanaie:

http://www.ibm.com/developerworks/rational/library/08/1223_sanaie/index.html

With this integration, you can create assets in Rational Asset Manager from existing configuration items (CIs) in the CCMDB. You can also use Rational Asset Manager as a Definitive Media Library (DML) for storing software images.

Note that this integration is bidirectional. Assets from Rational Asset Manager can be sent to the CCMDB and the CIs from CCDMB can be sent to the Rational Asset Manager. After the implementation, you will have a single repository for software images: the DML. Developers will be able to automatically create assets in Rational Asset Manager from existing CIs in CCMDB and use Rational Asset Manager as a DML for storing software images.

The assets created by Rational Asset Manager from a CCMDB CI, are referred to as remote assets. These remote assets contain a link to the CI in the CCMDB, and, similarly, you can navigate to the asset in Rational Asset Manager from the CI in the CCMDB. The remote asset in Rational Asset Manager is a reference to a remote resource, meaning that the asset is managed in a remote repository. In addition to storing the reference in the remote resource, the Rational Asset Manager repository stores metadata information for the asset, such as attributes and categories.

Similarly when the CCMDB creates new CIs from Rational Asset Manager assets, this information is stored in the CI, indicating that the asset originated in Rational Asset Manager. If the Rational Asset Manager repository is then synchronized with the CCMDB, repository and a remote asset is created from the CCMDB CI, the remote asset and the original asset become related assets: the original asset will have the “publish as” relationship to the remote asset, and the remote asset will have the “publish by” relationship to the original asset.

Remote assets in the Rational Asset Manager repository retain much of the metadata from the CCMDB CI. They are also given the asset type of Configuration Item and the category of CCMDB. These can be used as search filters to search for remote assets faster in the Rational Asset Manager repository.

After the asset is created in CCMDB as CI, it becomes available to the other applications running on Tivoli Process Automation Engine, such as Tivoli Service Request Manager and IBM Tivoli Release Process Manager

18.4 Products involved

To implement this scenario, your environment must have the following products installed:

- IBM Tivoli Change and Configuration Management Database 7.2.0.1
- IIBM Tivoli Release Process Manager 7.1.2
- IBM Tivoli Application Dependency Discovery Manager 7.2.0.1
- IBM Tivoli Integration Composer 7.2.0.1
- IBM Rational Asset Manager 7.2.0

18.5 Benefits

The benefits of this integration are as follows:

- Increased productivity by automated creation of assets in Rational Asset Manager from existing CIs in the CCMDB and automated creation of CIs in CCMDB from Rational Asset Manager assets.
- Access to the linked data (assets linked to the CIs and CIs linked to assets) by various groups in an organization for various perspectives.
- Secure and centralized control of the company software assets from a single place.

18.6 Implementation

In this section, we describe the implementation steps for this scenario.

18.6.1 Downloading and installing the integration module

To integrate CCMDB with Rational Asset Manager, the first thing is to install a module for the products talk to each other. This module is present in the Rational Asset Manager. In this part we will show how to install it in the CCMDB.

1. Log on the Rational Asset Manager (Figure 18-3) with your admin user account.

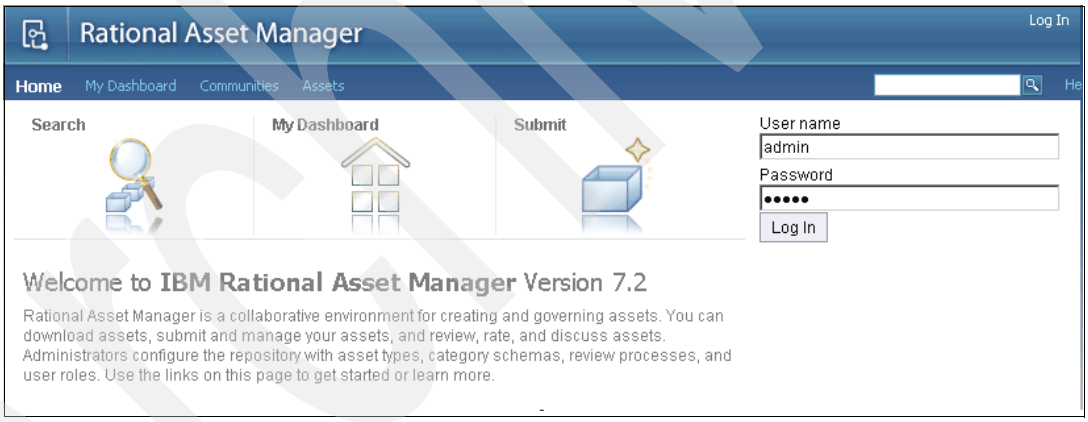


Figure 18-3 RAM Home window

2. Click **Help** → **Extensions**. See Figure 18-4 on page 550.

Archived

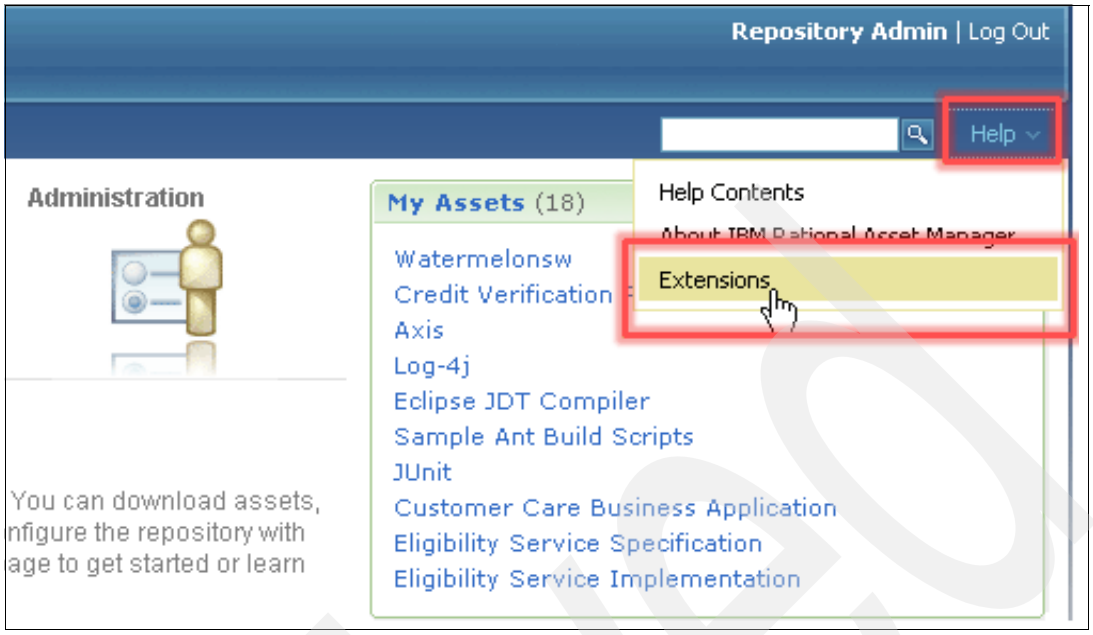


Figure 18-4 RAM Extensions

3. Select the `imram_x.x.zip` file (where `x.x` is the version of Rational Asset Manager that you are using) to download the integration module. See Figure 18-5.

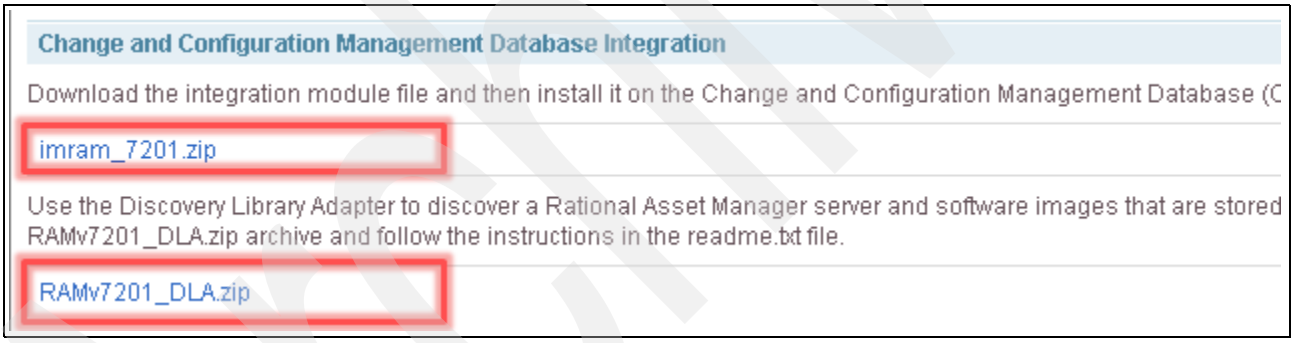


Figure 18-5 The .zip files for integrating with CCMDB

Note: Do not extract the `.zip` file; only download and save it.

4. Install the Integration module in the CCMDB. First, run a backup, in case something goes wrong.

   Back up the maxdb71 database and the registry on the CCMDB server as follows:

   a. To back up maxdb71 in IBM DB2, run the **backup** command in a DB2 command window, as shown in Example 18-1.

   Example 18-1 Back up maxdb71

```
CONNECT TO MAXDB71;
QUIESCE DATABASE IMMEDIATE FORCE CONNECTIONS;
CONNECT RESET;
BACKUP DATABASE MAXDB71 TO "C:\db2backupfolder" WITH 2 BUFFERS BUFFER 1024
PARALLELISM 1 WITHOUT PROMPTING;
CONNECT TO MAXDB71;
UNQUIESCE DATABASE;
CONNECT RESET;
```

   Note: If your Maximo database is Oracle or SQL Server, review the database documentation for the appropriate command line to use to back up your database

   b. To back up the registry, use the following path (see Example 18-2):

```
C:\Program Files\IBM\Common\acsi\bin>backupdb.cmd
```

   Example 18-2 Backup the registry

```
C:\Program Files\IBM\Common\acsi\bin>backupdb.cmd
ACUDB0004I A database backup command was issued, but directory C:\Program
Files\IBM\Common\acsi\backupDBs, the default directory where database backup
information is stored, does not exist. Deployment Engine will create the
directory.
ACUDB0005I The backupdb command completed successfully.
```

5. Use the Tivoli base services to install the integration module on the CCMDB repository. Open a command line and change to `CCMDBInstallation\bin` directory.
6. Run `solutionInstallerGU` file. See Example 18-3.

   Example 18-3 solutionInstallerGU

```
C:\>cd C:\ibm\SMP\bin
C:\ibm\SMP\bin>solutionInstallerGUI.bat
```

   The base services window opens (Figure 18-6 on page 552).

Perform the following steps to install the Rational Asset Manager IM (`imram_x.x.zip`) by using the Process Solution Installer (PSI):

a. Select a language and click **OK**.



Figure 18-6 Language window

The next window opens (Figure 18-7).

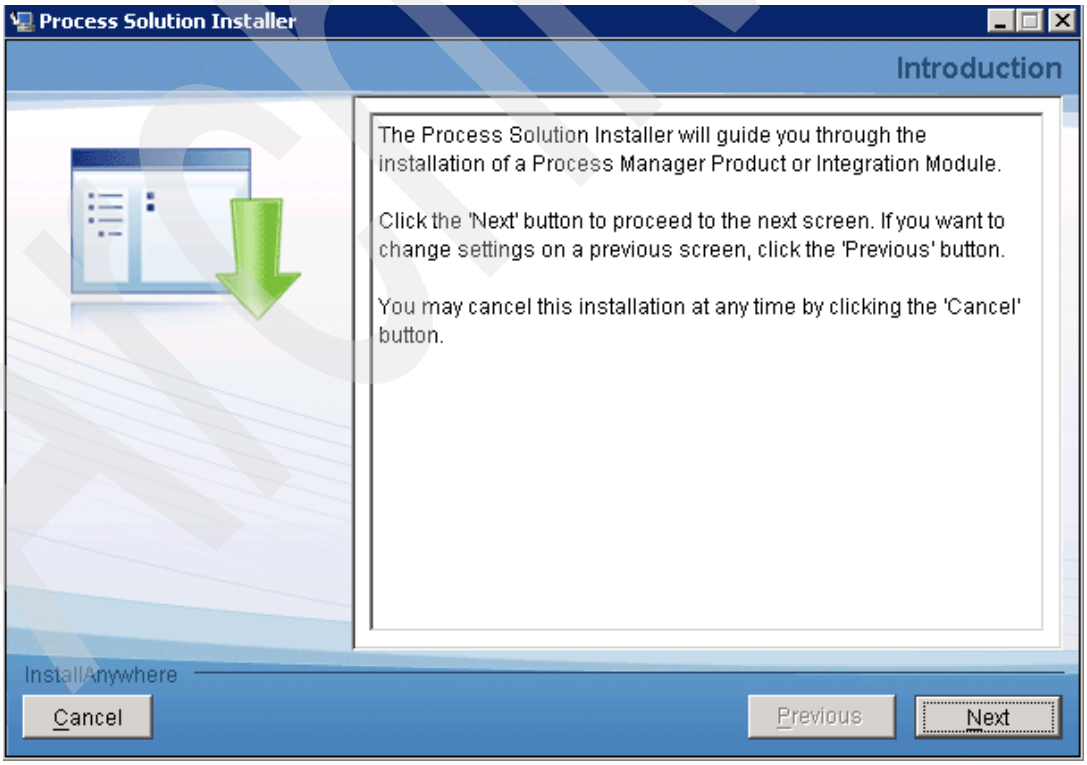


Figure 18-7 Introduction window

b. Click **Next**.

c. On the next window (Figure 18-8), select a PSI installable package by clicking **Choose**, selecting **imram_x.x.zip**, and clicking **Next.**

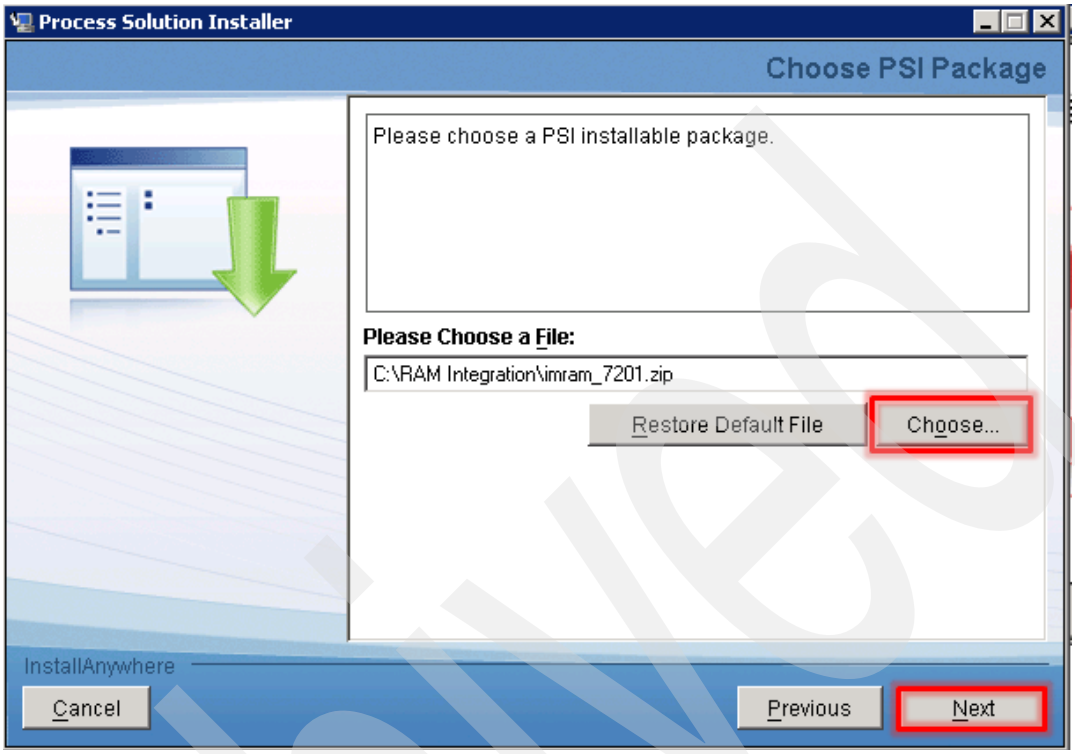


Figure 18-8 Choose PSI Package

The package is validated (Figure 18-9).



Figure 18-9 Validating Package window

- If an error occurs, return to step 3 on page 550 and download the file again.
- If the package is ok, the next window (Figure 18-10 on page 554) opens.

Archived

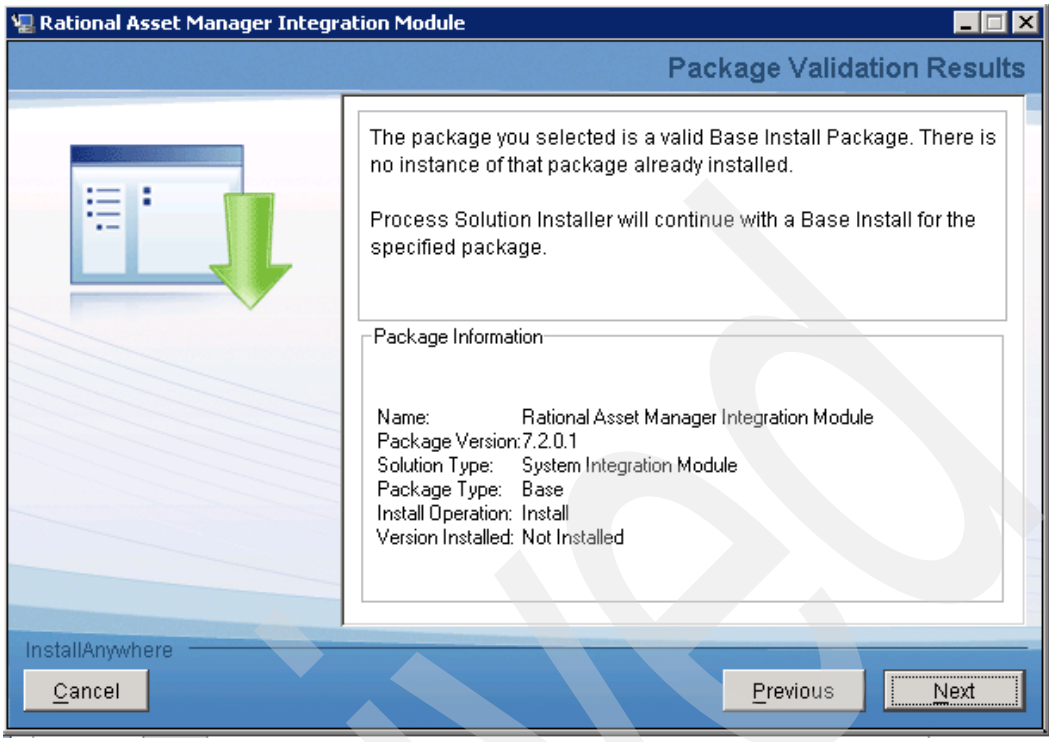


Figure 18-10 Package Validation Results

d. To proceed, accept the terms in the license agreement. See Figure 18-11.

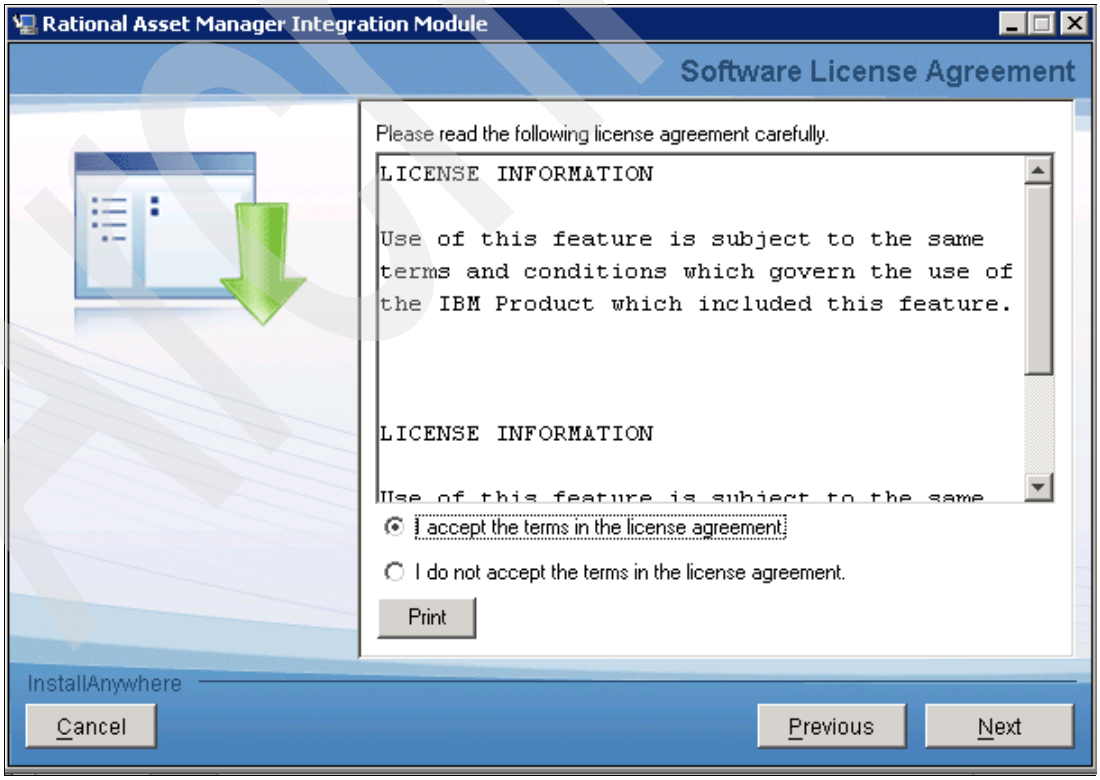


Figure 18-11 Software license agreement

e. Optional: If you want to install the Language Support Feature, besides the Rational Asset Manager Integration Module, select the check box in Figure 18-12.

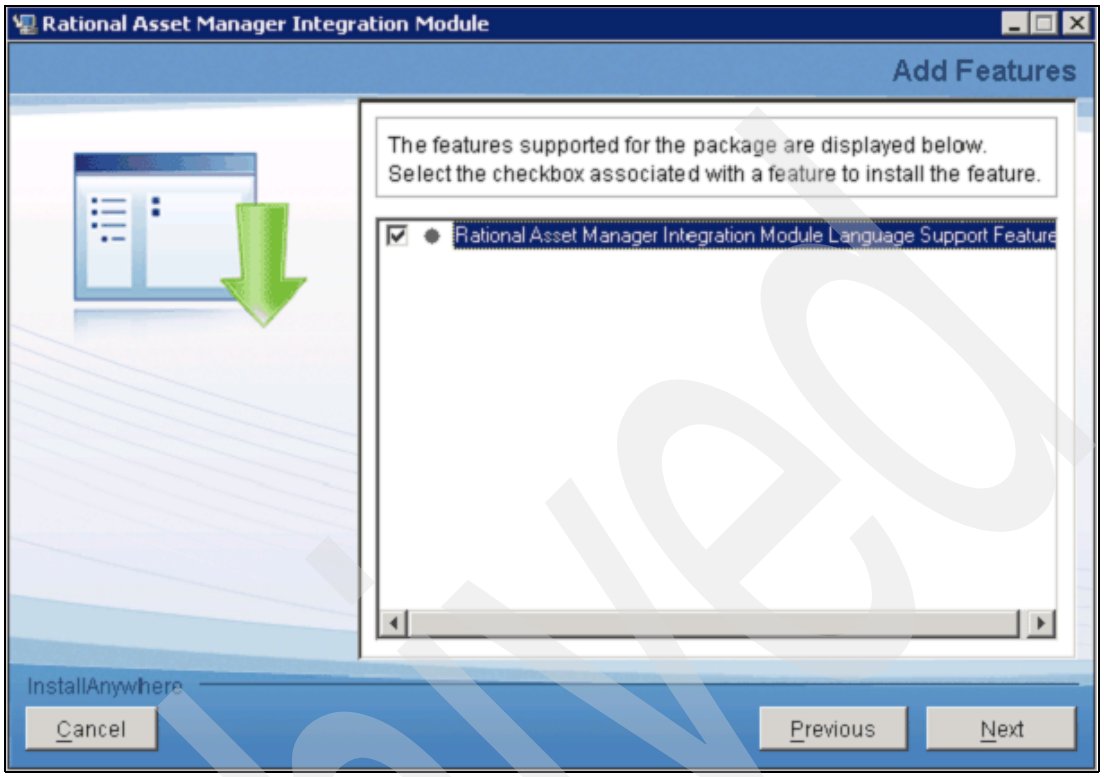


Figure 18-12 Adding features

f. Enter the required user IDs and passwords, as in Figure 18-13 on page 556.

Archive

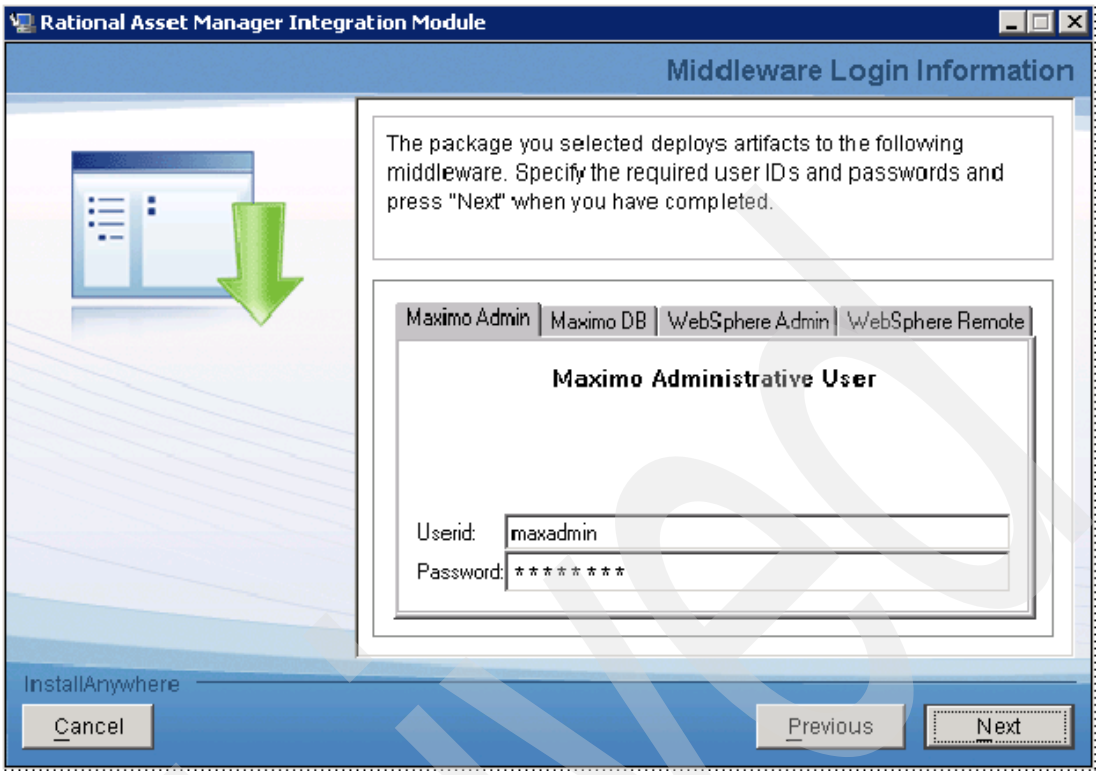


Figure 18-13 Login information

g. Click **Next**, and verify that all requirements have been satisfied. See Figure 18-14.

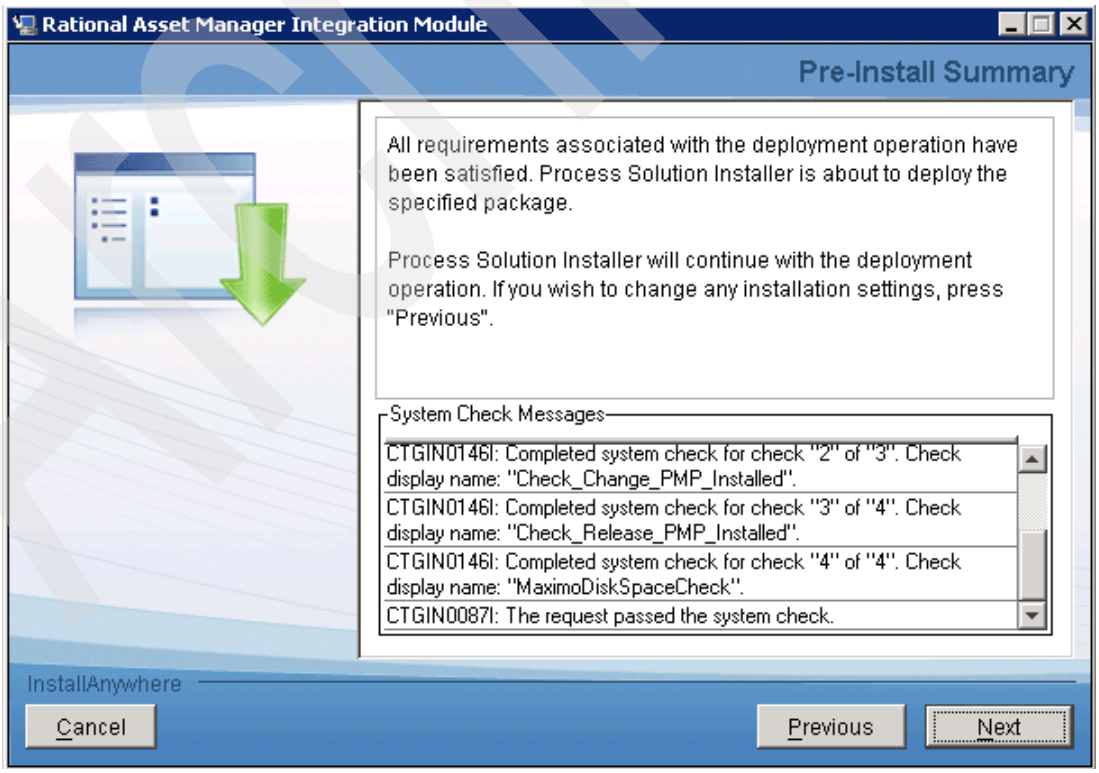


Figure 18-14 Pre-Install Summary

h. When the installation completes successfully, a message is displayed (Figure 18-15). Click **Next.** In the next window, click **Done**.

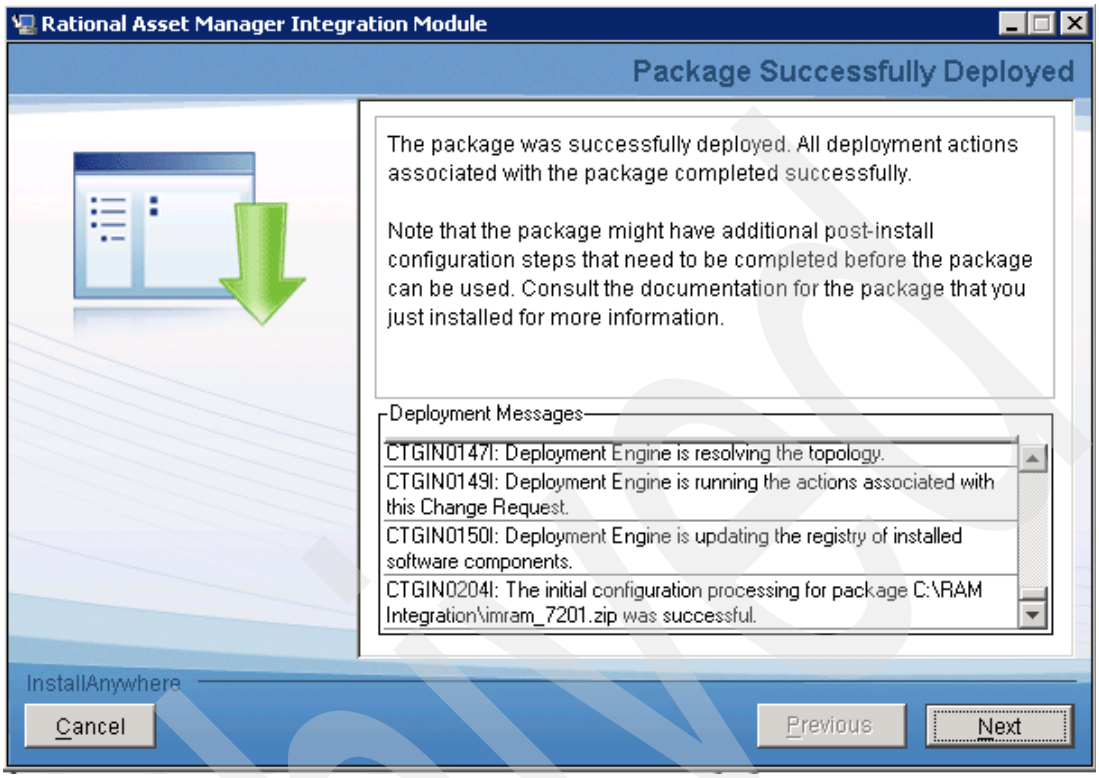


Figure 18-15 Package deployment results

7. Log in to Maximo (`http://host_name/maximo`) and click **Help** → **System Information**.
8. Verify that the Rational Asset Manager integration module is installed. See Figure 18-16.

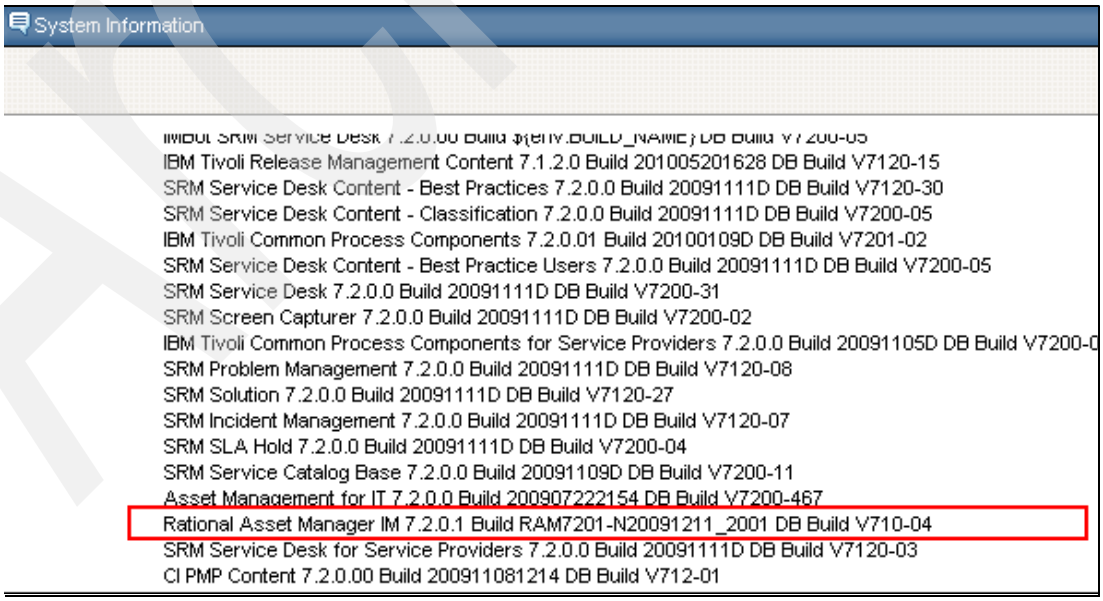


Figure 18-16 Maximo System Information

The integration package is now installed.

18.6.2 Creating a discovery library adapter (DLA) to import in the TADDM

This section has instructions for manually generating the DLA output file and installing and configuring a discovery library adapter that will create a DLA output file by searching the Rational Asset Manager repository.

Note: DLA is discovery library adapter. To propagate information contained in a Rational Asset Manager to a operational management product (OMP) we have to create a DLA output file.

Perform the following steps:

1. Create a DLA output file that uses a search filter to define a set of assets for the DSL:
   a. In the Rational Asset Manager web client, click **Administration**. See Figure 18-17.



Figure 18-17 Administration button

   b. Click the name of the community that you want to connect to the CCMDB, for example, RAMCCMDB, as shown in Figure 18-18 on page 559.

Archive

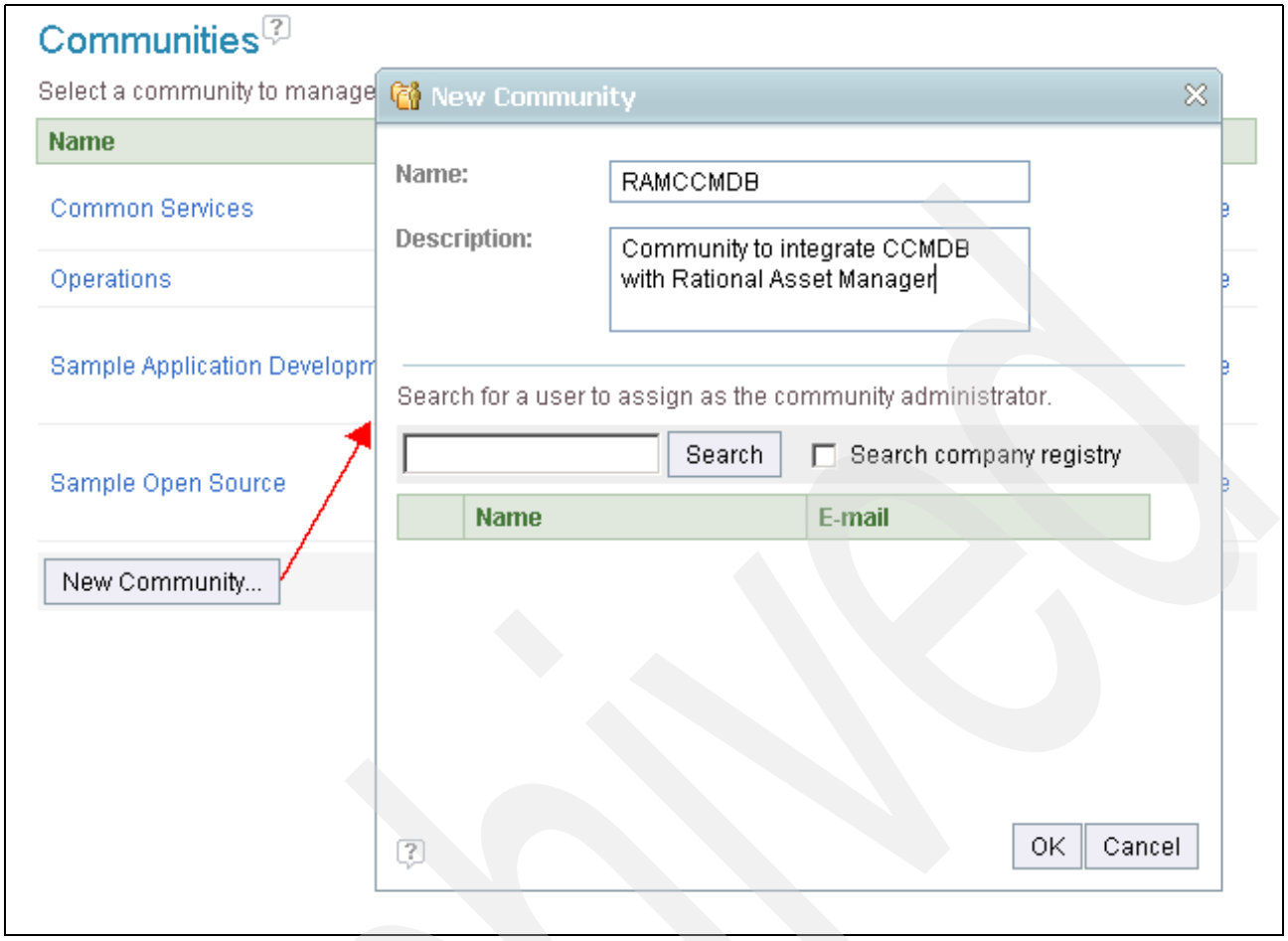


Figure 18-18 Rational community

c. Click the community name you have just created.

d. Click the **Connections** tab (Figure 18-19).



Figure 18-19 Connections tab

Archived

e. In the Change and Configuration Management Database section (Figure 18-20), click **New Connection**.

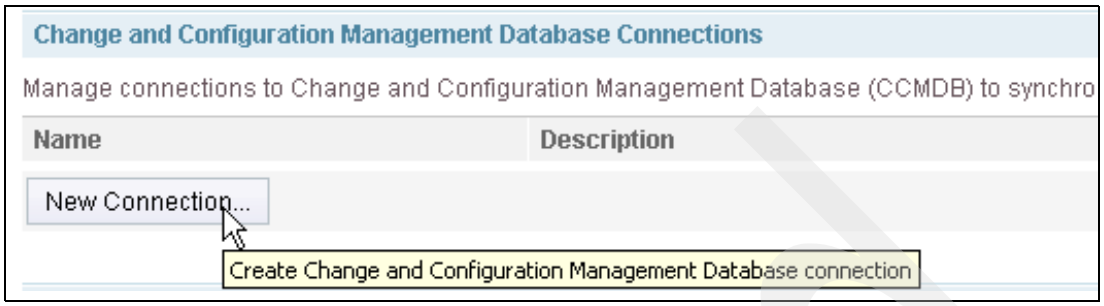


Figure 18-20 Create a CCMDB connection

f. On the connection properties page, complete the form with the following details of the CCMDB installation:

- Name: Type a name for the connection, for example `RAM-CCMDB`.
- Description: Optionally, type a detailed description of the connection. The description can include spaces and special characters.
- URL: Enter the fully qualified URL, including the port number, to use when accessing the CCMDB application, for example:

  `http://ccmdbserver/meaweb/services`
- Login: Type the user ID of a user who is authorized to access the CCMDB.
- Password: Type the user's password for accessing CCMDB.
- Default asset owner: Click **Select Owner** to choose the Rational Asset Manager user who will own the assets that are created in the Rational Asset Manager repository from the CCMDB CIs.
- Synchronization schedule: Click **Edit** to modify how frequently the Rational Asset Manager repository will synchronize with the CCMDB repository. You may choose to synchronize daily or weekly at a specific time or at an interval specified in minutes or hours.

g. Click the **Test Connection**. If the connection is successful, the following message is displayed (see Figure 18-21 on page 561):

`Connection to CCMDB server was successful`

Note: If an error message is indicated (in a red box), verify the URL you typed.

h. Click **OK**.

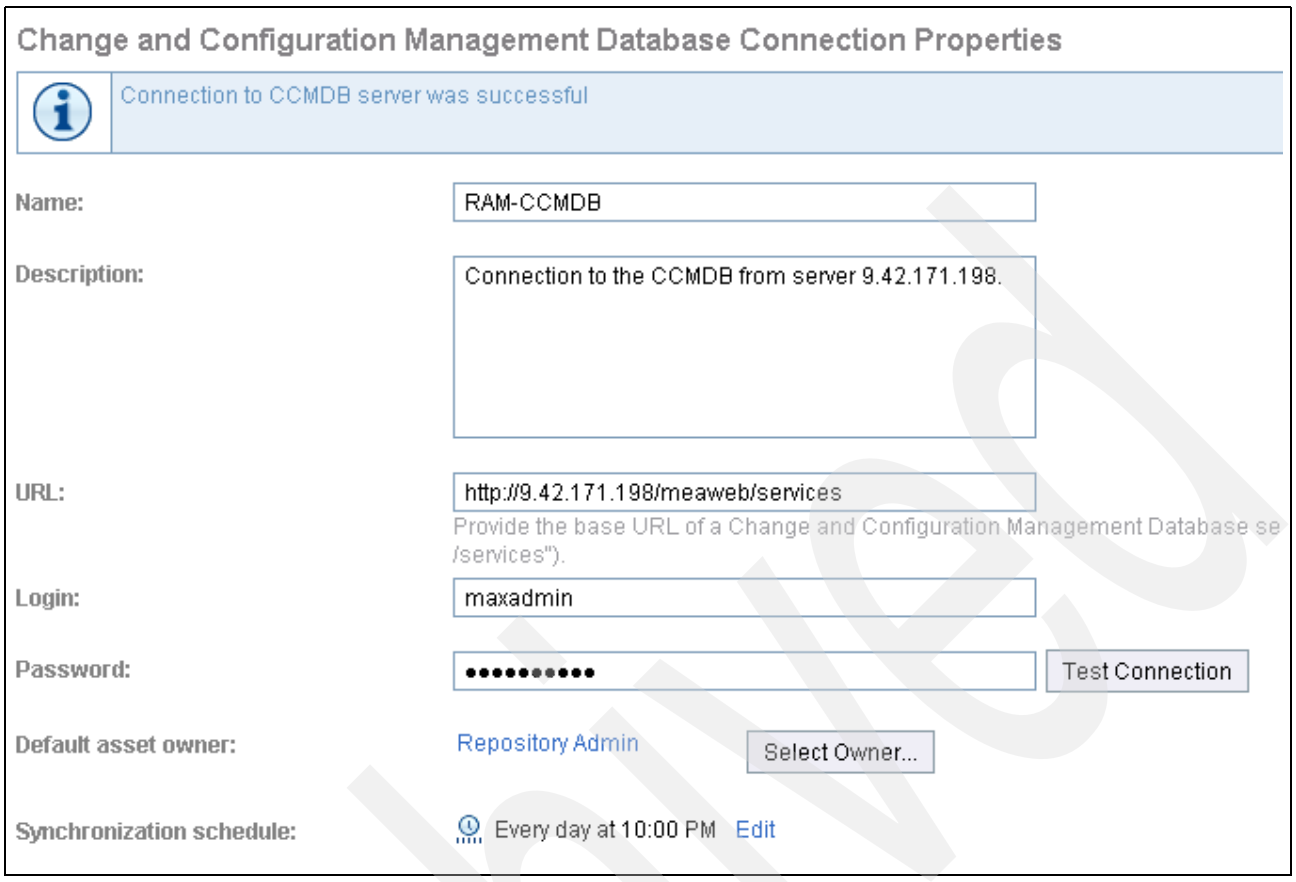


Figure 18-21 CCMDB Connection Properties

2. Click **Assets** to go to the Search for Assets page. See Figure 18-22.

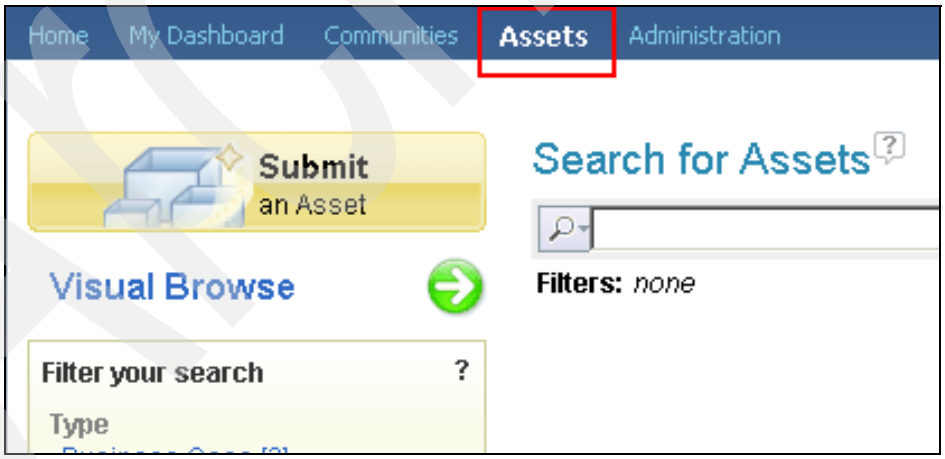


Figure 18-22 Assets tab

3. Search for the approved assets (Figure 18-23 on page 562). Use the filter at the left side, or type the following text in the search field:

```
ramSearch:(lfState,approved)
```



Figure 18-23 Asset State

4. Determine how many assets were displayed, and then copy the *entire search shortcut* from the text field. The search shortcut is an icon with a magnifying glass in front of a clipboard. See Figure 18-24.

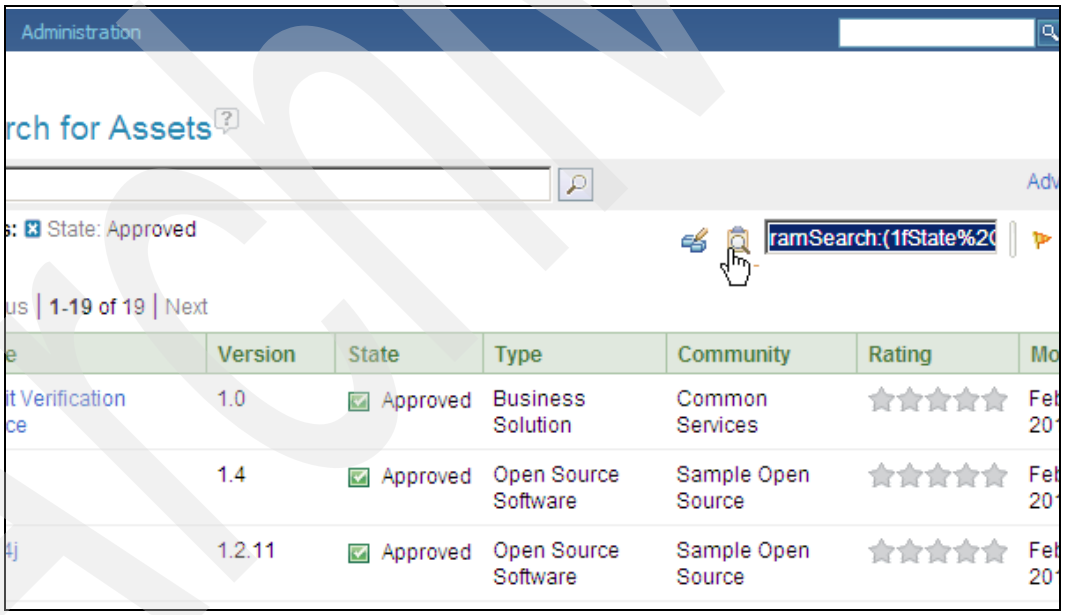


Figure 18-24 View Search Shortcut

5. Return to the Connection Properties page and *paste the search shortcut* into the Search Text field, under the topic Discovery Library Adapter Generation. See Figure 18-25 on page 563.

Archived

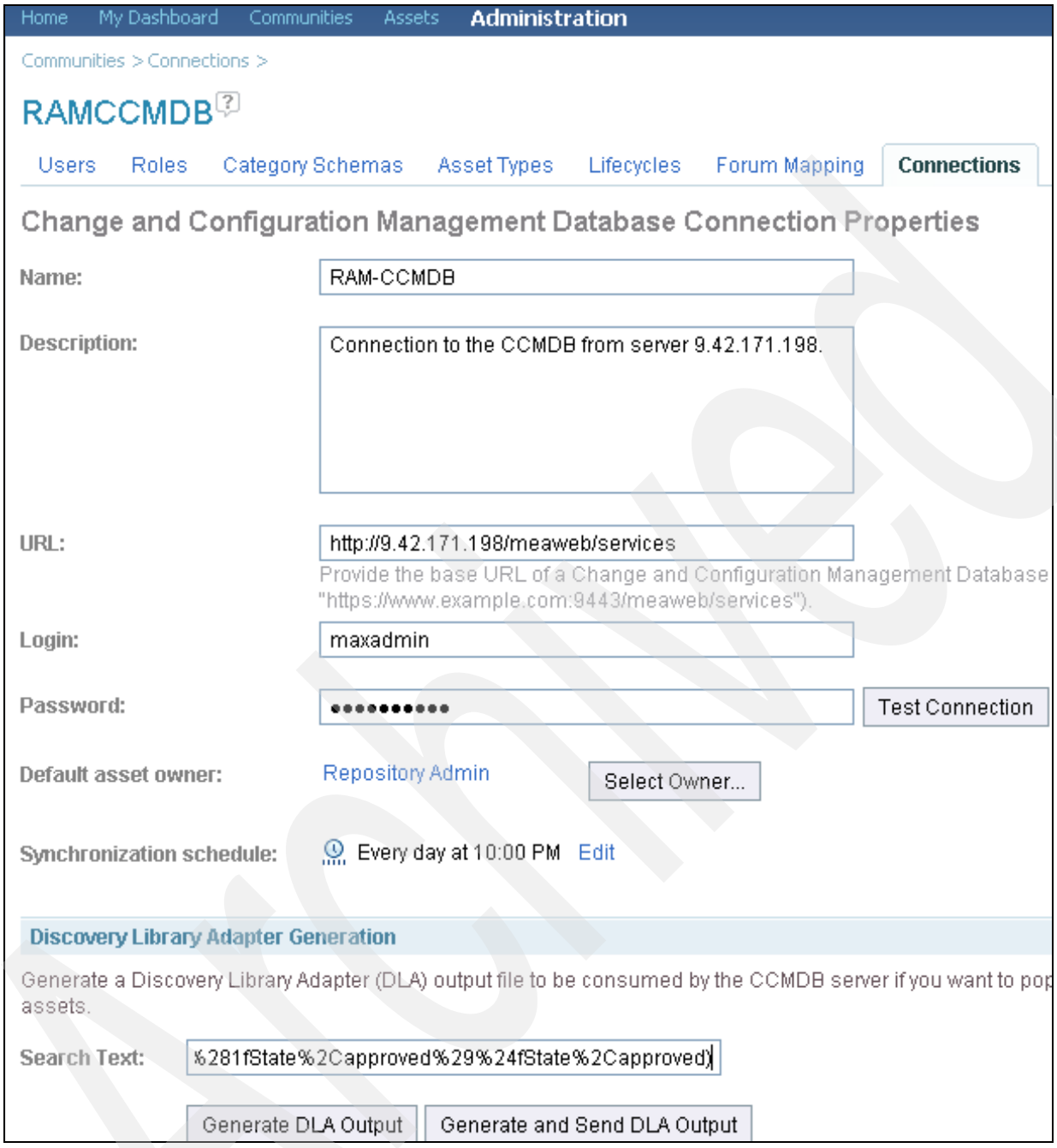


Figure 18-25 Synchronization schedule of CCMDB Connection

6. To create an output DLA file to be saved on the file system, click **Generate DLA Output**, which creates an `.xml` file.
7. Save the `.xml` file in the TADDM server.

18.6.3 Loading the DLA output file into the TADDM

A DML requires an operational management product (OMP). To create an OMP in the CCMDB, use the TADDM to propagate information contained in a Rational Asset Manager DLA output file to the CCMDB.

Perform the following steps:

1. Transfer the Rational Asset Manager DLA output file to TADDM server.
2. In the TADDM server, from a command line, navigate to the `cmdb\dist\bin` folder and run the following command (see Example 18-4):

   ```
   loadidml -f <dla_output_file> where <dla_output_file> is the full path to and
   name of the discovery library adapter file
   ```

   Note: To run the **`loadidml.sh`** or **`loadidml.bat`** command, you must be logged in as the taddm user.

Example 18-4 loadidml -f

```
taddm2:/opt/IBM/cmdb/dist/bin # su - cmdbuser
cmdbuser@taddm2:~> cd /opt/IBM/cmdb/dist/bin
cmdbuser@taddm2:/opt/IBM/cmdb/dist/bin> ./loadidml.sh -f
/opt/IBM/cmdb/ram-to-ccmdb/RAMv7201.ramlnx.2010-06-07T20.34.47.340Z.refresh.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.
```

3. Verify that the DLA output file was loaded. Launch the TADDM application and click **Start Product Console**. If prompted, enter your user ID and password.
4. Click **Edit** → **MSS**, as shown in Figure 18-26.

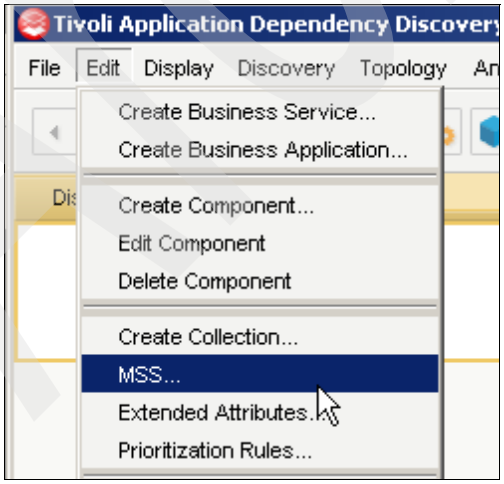


Figure 18-26 Accessing MSS from TADDM

5. The MSS list contains an entry that matches the DLA output file. The product name in the MSSName entry is `Rational Asset Manager`. See Figure 18-27 on page 565.

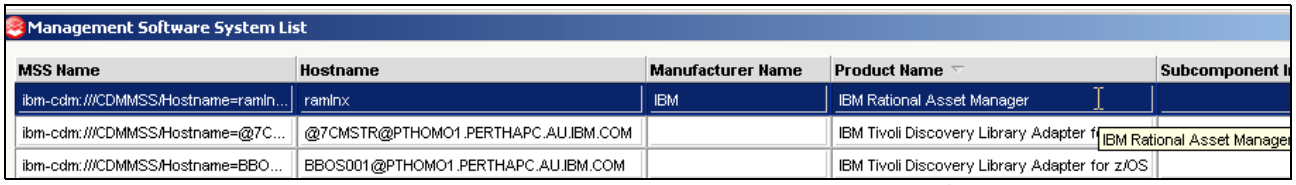


Figure 18-27 MSS list

6. Select the **MSS** and click **List CIs** (Configuration Items). See Figure 18-28.

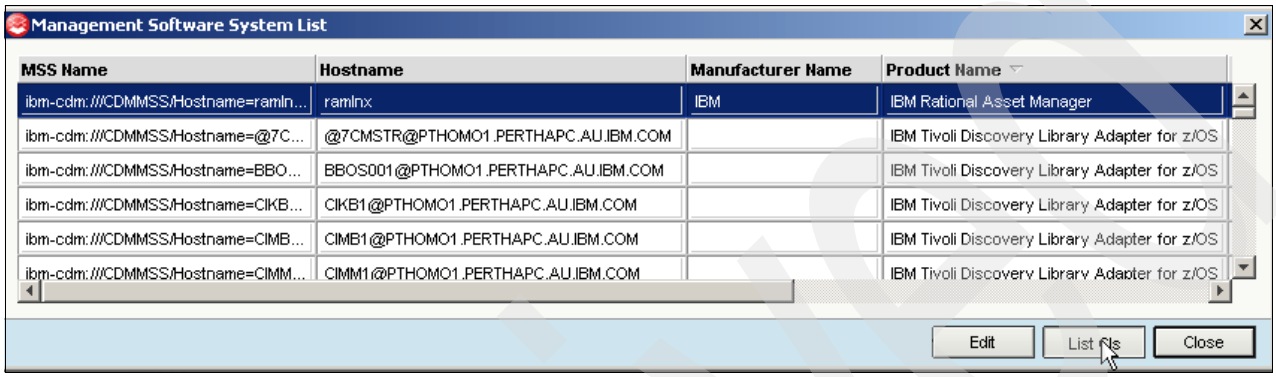


Figure 18-28 MSS list

7. Click **OK** when the following message is displayed:

   `This operation can take a long time. Do you want to continue?`

8. The Configuration Item List in Figure 18-29 shows the Rational Asset Manager assets.

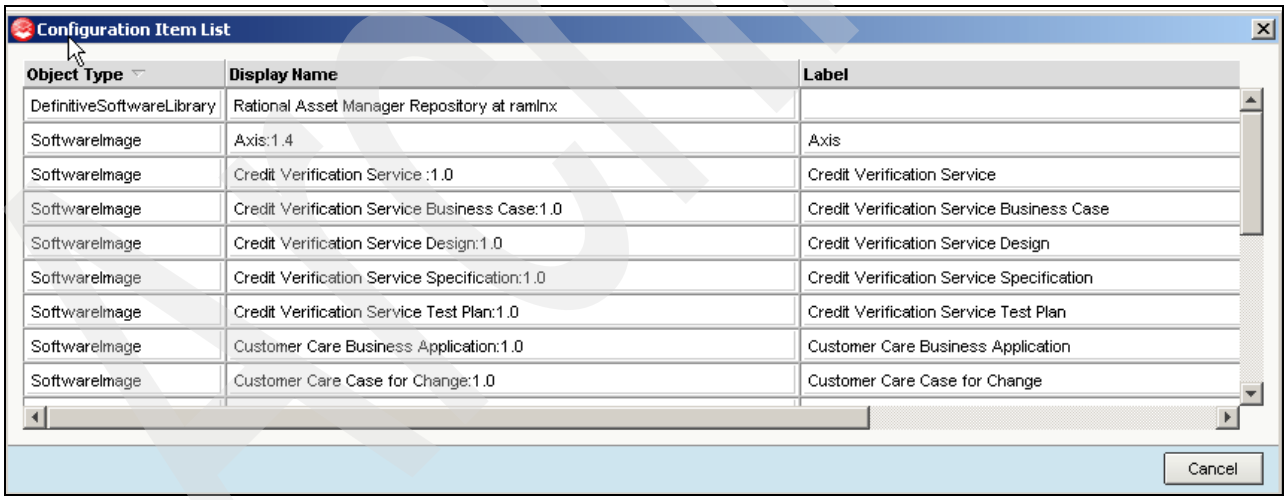


Figure 18-29 Configuration Item List

18.6.4 Mapping actual CIs from TADDM to CCMDB

Tivoli Integration Composer uses information from the Rational Asset Manager DLA output file to create the OMP in CCMDB. This section describes mapping actual CIs from TADDM to CCMDB by using Tivoli Integration Composer adapter.

To set up the CI type adapter, perform the following steps:

1. Create a TADDM data source:
   a. Start the Tivoli Integration Composer application (for example, in the `<ITIC_root_dir>\bin` folder, run `startFusion.bat` (for Windows) or `startFusion.sh` (for UNIX-based systems). Log on to the Tivoli Integration Composer.
   b. Click **Define New Data Source**. See Figure 18-30.

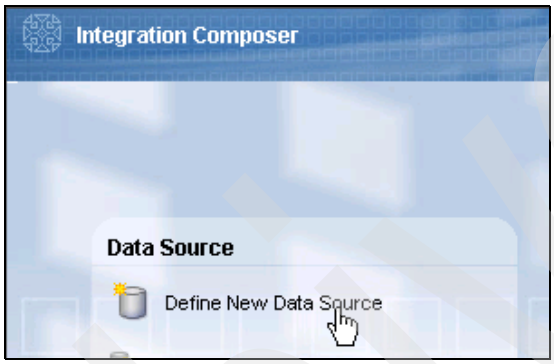


Figure 18-30 New Data Source on Tivoli Integration Composer

   c. Select **TADDM72CIType**. See Figure 18-31.

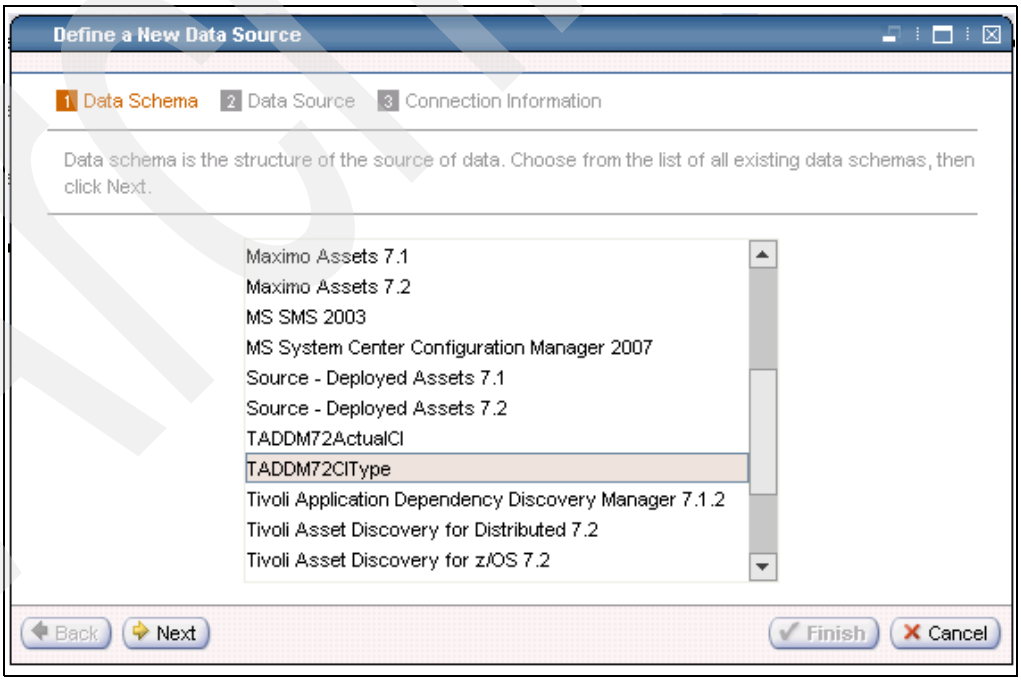


Figure 18-31 New Data Source

Archived

d. For the data source name, enter `TADDM72CIType`. See Figure 18-32.

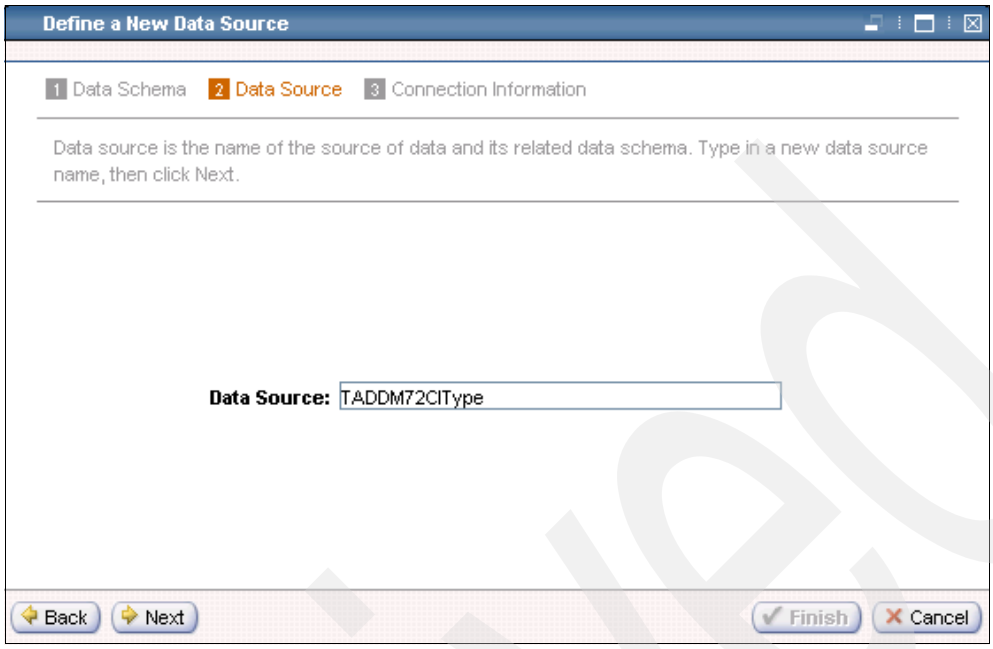


Figure 18-32 Define the Data Source name

e. Enter the following information (see Figure 18-33 on page 568):

- Select Connection Method: Select **IBM Configuration and Tracking API**.
- Host name: Type the host name for the TADDM server (not the database).
- Port: Type the RMI port for the TADDM server. The default value is 9530. If you are using SSL, the default value is 9531.
- Trusted Location: Type the fully-qualified path of the SSL certificate file to use to connect to a server, such as the Configuration Discovery (Not required field) and Tracking Server, for example:

  `c:\Integration Composer\...\jssecacerts.cert`
- User name: Type the user name for a TADDM account with administrator privileges.
- Password: Type the password for the TADDM use.

Archived

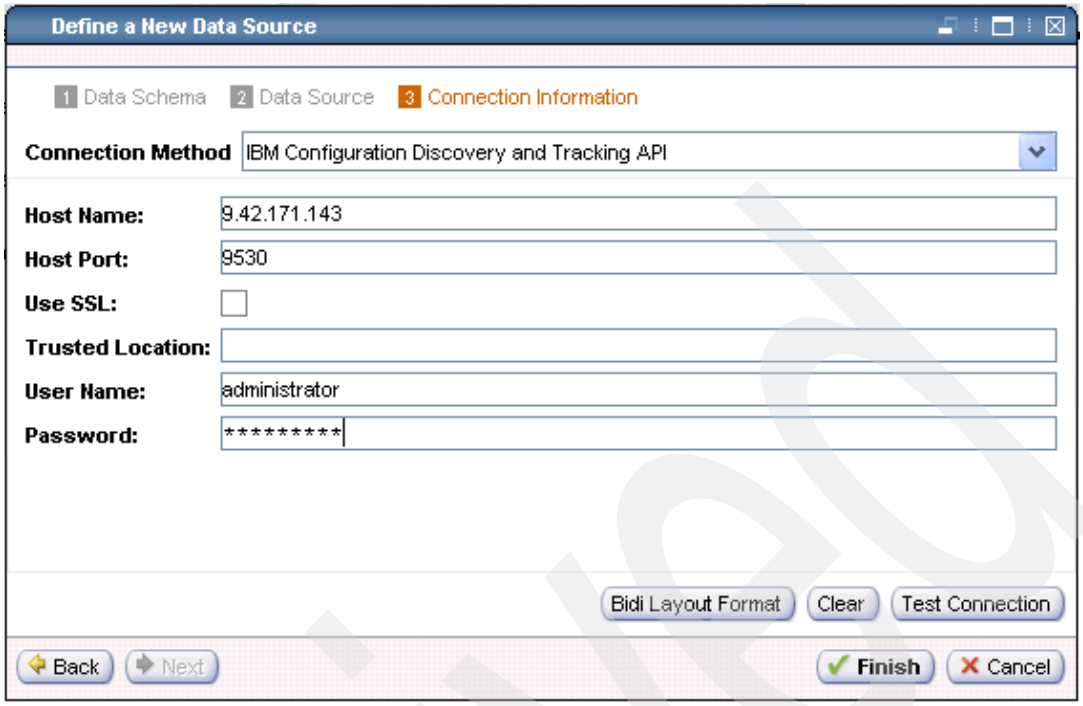


Figure 18-33 Data Source connection information

f. Click **Test Connection**. If the test is successful, the pop-up window shown in Figure 18-34 opens. If an error occurs, check the values you have entered.

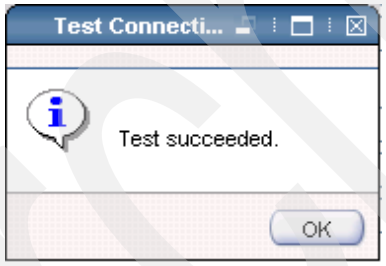


Figure 18-34 Test connection

ii. Click **OK** and then click **Finish**.

2. Create CI type mappings:

a. Go to Tivoli Integration Composer home.

b. Click **Create New Mapping**. See Figure 18-35 on page 569.

Archived

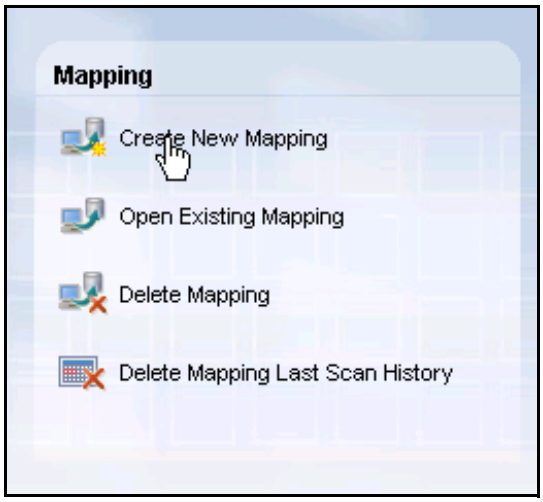


Figure 18-35 Creating a new Mapping

c. On the next window, select the following values (shown in Figure 18-36):

- Source: `TADDM72CIType`
- Target: `CCMDB72citype`
- Mapping Name: `taddm-2-ccmdb-ci-types`

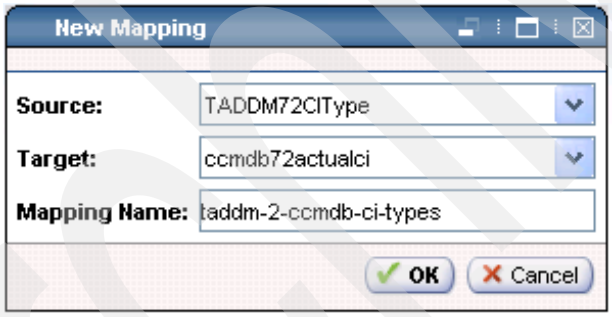


Figure 18-36 New mapping

d. Complete the Password field and verify that the other required data from TADDM server is correct.

e. Click **Test Connection**. If the test succeeded, click **OK** and click **Finish**. See Figure 18-37 on page 570.

Archived

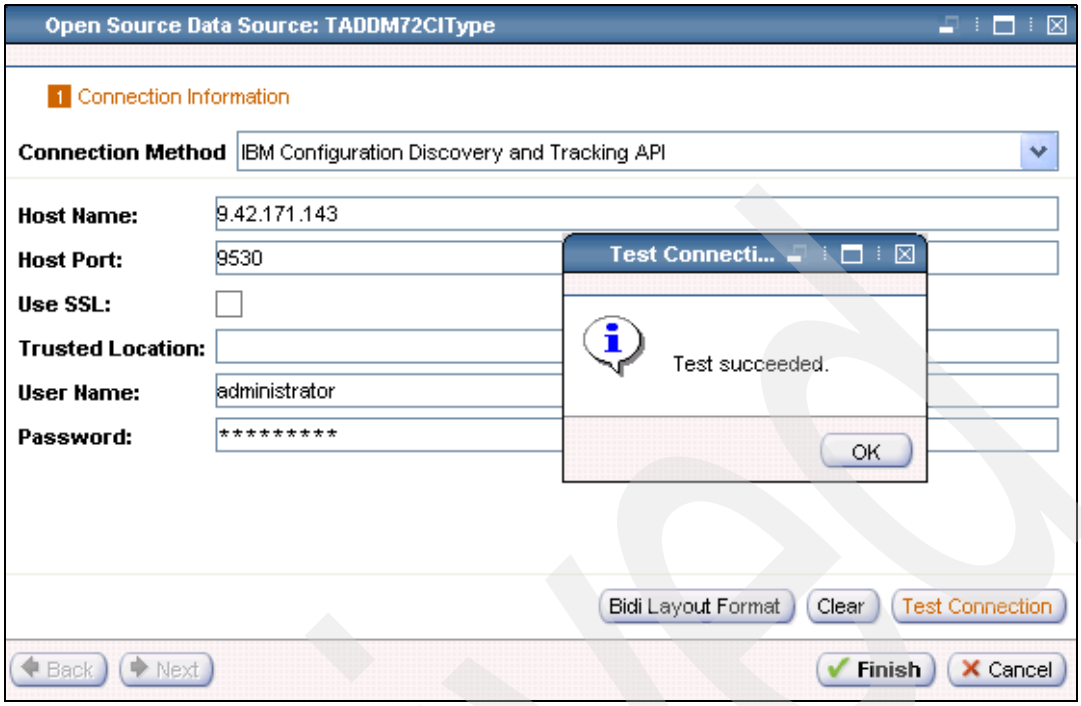


Figure 18-37 Test the connection

f. Fill in the appropriate data for your environment:

- Connection Method:

  DB2: IBM DB2 JDBC Driver

  Oracle: Oracle JDBC Thin Driver

  SQL server: i-net OPTA JDBC Driver for Microsoft SQL Server 7/2000
- Host name: Type the host name for the CCMDB database server.
- Port: Type the port number for the machine. The default values are as follows:

  DB2: 50005

  Oracle: 1521

  SQL server: 1433
- Database: Type the name of the database: maxdb71.
- User name: Type the user name for the user who created the Maximo tables in the database.
- Password: Type the user's password.
- Table owner: Type the user name for the user who created the Maximo tables in the database.

Archive

g. Click **Test Connection**. See Figure 18-38.

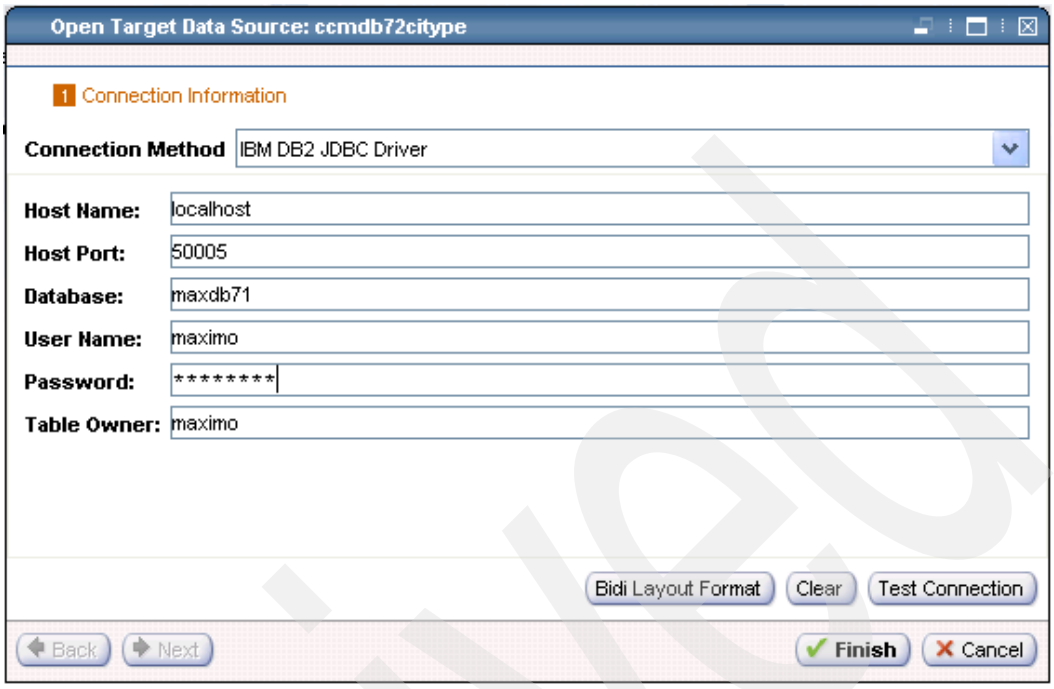


Figure 18-38 Connection Information

h. When you receive a pop-up window indicating that the information is OK, proceed and click **Finish**.

i. A window showing the relationships opens. If there is no relationship, you must import an `.fsn` file with the relationships by clicking **Select Action** → **Import** and then, selecting the `*.fsn` file.

j. Make any other changes you want and then click **Select Action** → **Save**. If you do not want to make further changes, click **Select Action** → **Close**.

3. Execute the CI type mapping:

a. Open a command (cmd) prompt.

b. Change directory (**cd**) to the integration `composer\bin` directory.

c. Run the following command, where *<password>* is the password for the Maximo database user:

```
commandline taddm-2-ccmdb-ci-types maximo password administrator collation maximo password
```

See Example 18-5.

> **Note:** The `commandline.bat/sh` parameters are case-sensitive.

Example 18-5 Run commandline.bat

```
C:\Integration Composer>commandline.bat taddm-2-ccmdb-ci-types maximo PASSWORD
administrator collation maximo PASSWORD
Integration Composer will now start.
IBM Tivoli Integration Composer 7.2.0.1 Build 200911031354
08 Jun 2010 11:06:32:953 [INFO] IBM Tivoli Integration Composer 7.2.0.1 Build 200911031354
Mapping: taddm-2-ccmdb-citypes
```

```
08 Jun 2010 11:06:32:969 [INFO] ObjectManager created. (This should only happen once.)
Creating data source Fusion...08 Jun 2010 11:06:32:985 [INFO] GlobalContext created. (This
should only happen once.)
08 Jun 2010 11:06:32:985 [INFO] ResourceManager created. (This should only happen once.) :
0 resource(s) loaded.
08 Jun 2010 11:06:33:453 [INFO] ResourceManager : Loading resource labels : 577 resource(s)
loaded.
08 Jun 2010 11:06:33:485 [INFO] ResourceManager : Loading resource messages : 279
resource(s) loaded .
08 Jun 2010 11:06:33:485 [INFO] IBM Tivoli Integration Composer database version: V7116-173
done
IBM Tivoli Integration Composer database version: V7116-173
Retrieving mapping definition taddm-2-ccmdb-citypes...  done
Creating source data source TADDM72CIType...    done
Connecting to source data source TADDM72CIType...
08 Jun 2010 11:06:33:953 [INFO] ResourceManager : Loading resource file C:\Integration
Composer\data
\properties\jdbcinfo.properties : 39 resource(s) loaded.
08 Jun 2010 11:06:34:016 [INFO] DataManager created. (This should only happen once.)
08 Jun 2010 11:06:36:875 [INFO] Collation version: 2.2, build: null, home: C:\Integration
Composer
Connecting to source data source TADDM72CIType...      done
Creating target data source ccmdb72citype...    done
Connecting to target data source ccmdb72citype...
Connecting to target data source ccmdb72citype...      done
.
.
.
Classification: RELATION.DONATEDTO
Classification: RELATION.ISACTUALLYCAUSEDBY
Classification: RELATION.VIRTUALIZES
Classification: RELATION.MIRRORS
*
Done
08 Jun 2010 18:46:44:750 [INFO]
=======================================================
               Mapping execution completed
               Mapping: taddm-2-ccmdb-ci-types
               Execution time: 00:19:26
               Classifications created: 0
               Records created: 971
               Records updated: 12
               Records deleted: 0
               Errors: 0
               NRS GUIDs created: 0
               NRS records updated on Alternate Keys: 0
               NRS records found as duplicate: 0
               NRS records deleted because of duplicate: 0
=======================================================

Done
Mapping completed successfully.
Integration Composer has finished.
```

d. Verify your action by viewing the logs: `<installdir>\log\fusion.log`

Archived

18.6.5 Activating the CI types

Perform the following steps:

1. Log in to the Base Services/CCMDB application (`http://host_name/maximo`).
2. Click **Go To** → **Administration** → **CI Types**. See Figure 18-39.

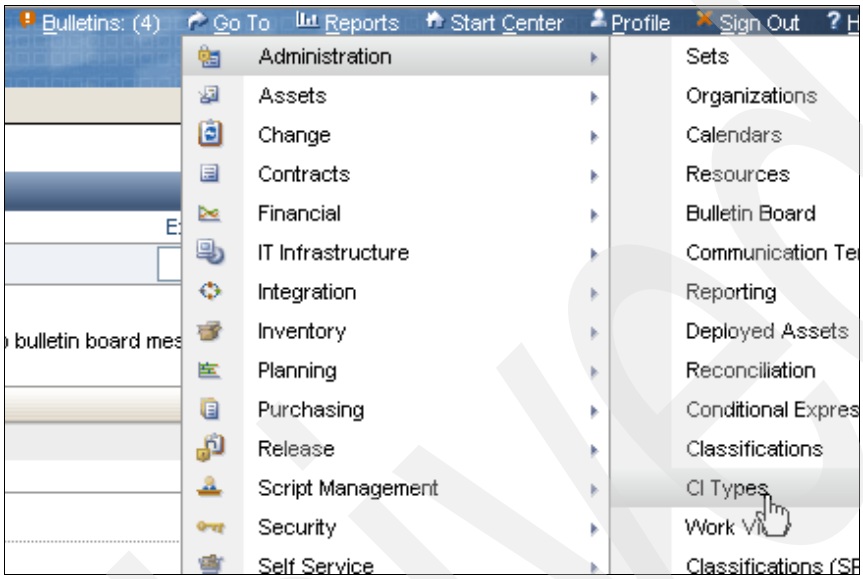


Figure 18-39 Accessing the CI Types application

3. Search for the following files:
   - `SYS.COMPUTERSYSTEM`
   - `SYS.OPERATINGSYSTEM`
   - `APP. DEFINITESOFTWARELIBRARY`
   - `APP. SOFTWAREIMAGE`
4. Click the **Change Status** icon and then change the status to **Active**. See Figure 18-40 on page 574.

Archived

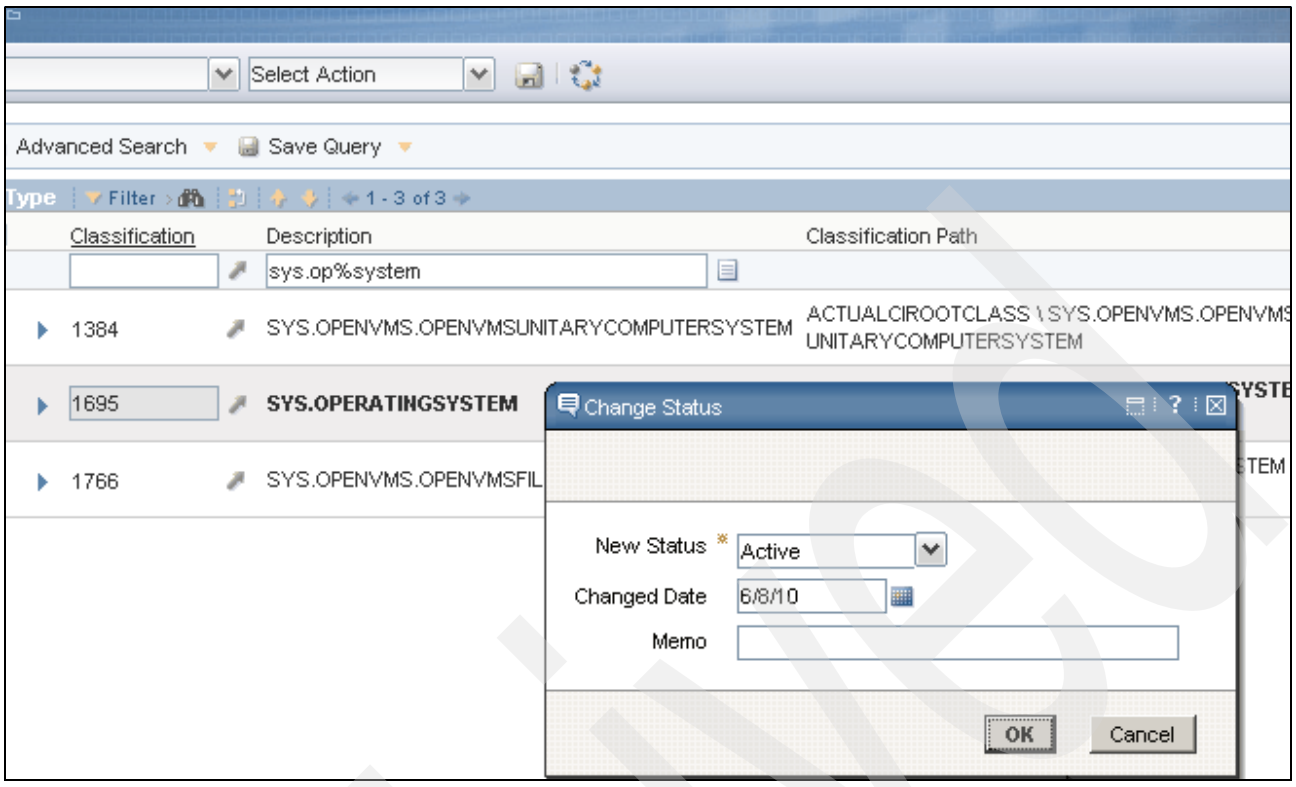


Figure 18-40 Changing status

5. Click **OK** and **Save**.

18.6.6 Setting up the actual CI adapter

Perform the following steps:

1. Create CI type mappings:
   a. Go to Tivoli Integration Composer home.
   b. Click **Create New Mapping**. See Figure 18-41.

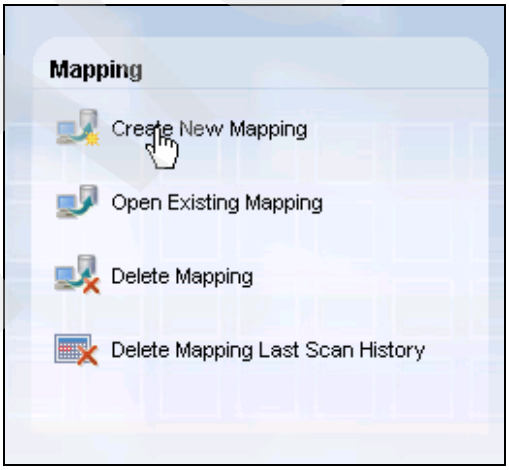


Figure 18-41 Creating a new mapping

c. In the next window, provide the following values (Figure 18-42):

- Source: TADDM72CIType
- Target: CCMDB72citype
- Name: taddm-2-ccmdb-ci-types

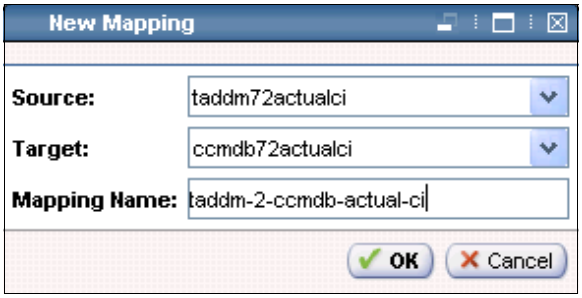


Figure 18-42 Mapping name

d. Complete the Password field and determine whether the other required data from TADDM server is correct. See Figure 18-43.

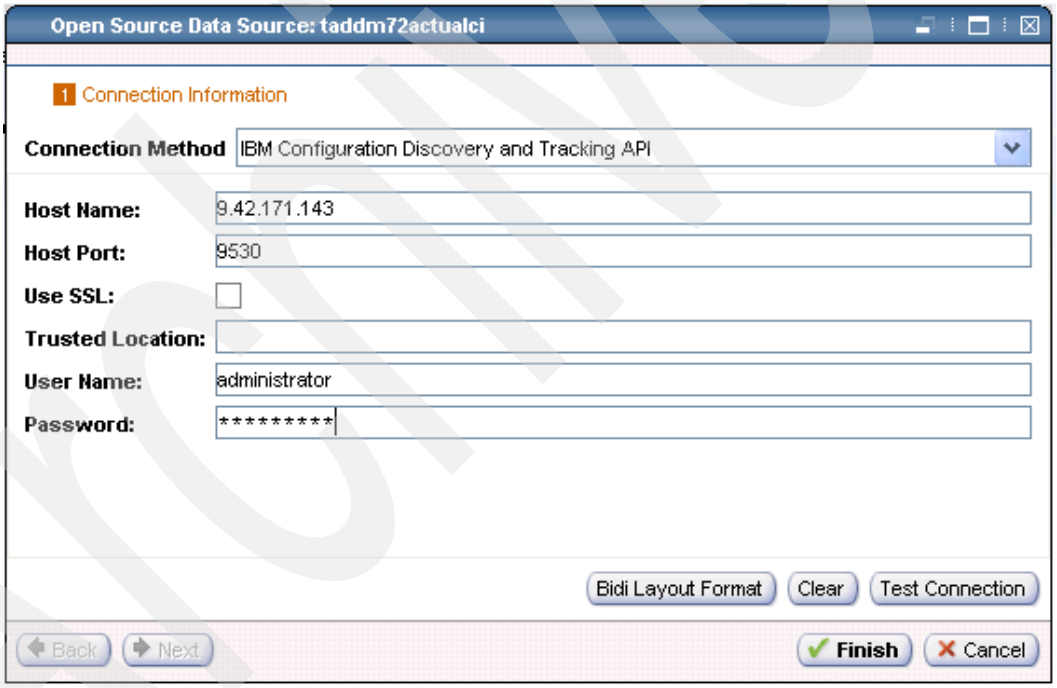


Figure 18-43 Connection information

e. Click **Test Connection**. If the test is successful, click **OK** and click **Finish**.

f. Enter the appropriate data for your environment:

- Connection Method:

  DB2: IBM DB2 JDBC Driver

  Oracle: Oracle JDBC Thin Driver

  SQL server: i-net OPTA JDBC Driver for Microsoft SQL Server 7/2000
- Host name: Type the host name for the CCMDB database server.

Archived

- Port: Type the port number for the machine. The default values are as follows:

  DB2: 50005

  Oracle: 1521

  SQL server: 1433
- Database: Type the name of the database: maxdb71.
- User name: Type the user name for the user who created the Maximo tables in the database.
- Password: Type the user's password.
- Table owner: Type the user name for the user who created the Maximo tables in the database. See Figure 18-44.

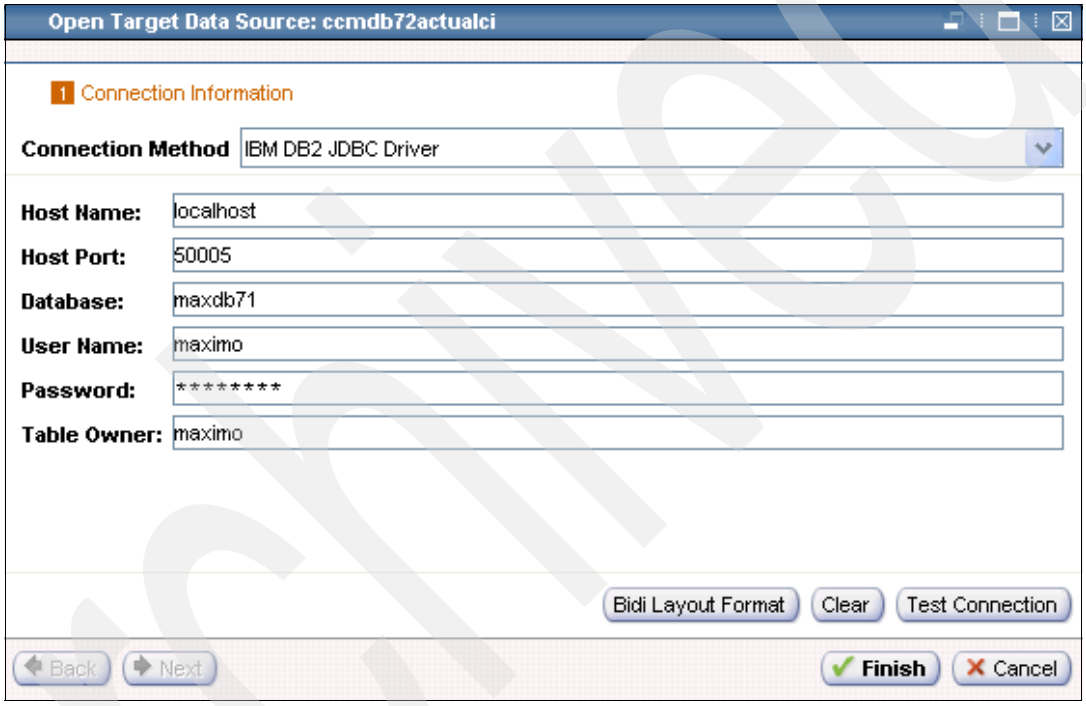


Figure 18-44 Connection information

g. Click **Test Connection**.

h. At the message indicating success, proceed and click **Finish**. A window showing the relationships opens.

i. If there is no relationship, import an `.fsn` file with the relationships by clicking **Select Action** → **Import,** and then selecting the *`.fsn` file.

j. Verify that the column expressions have codes and then click **Close**. See Figure 18-45 on page 577.

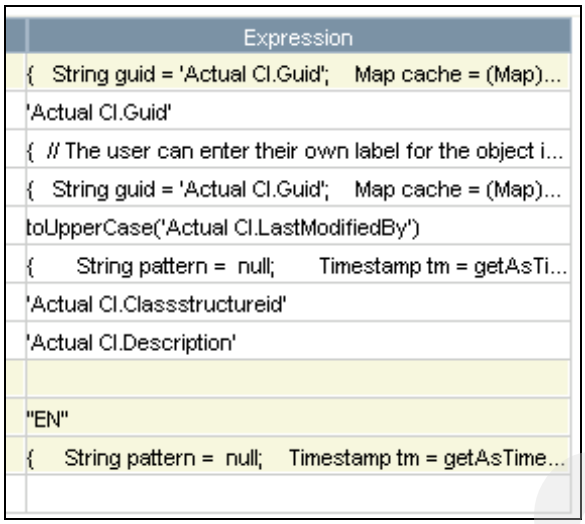


Figure 18-45 Expressions

2. Run the CI type mapping:
   a. Open a command (cmd) prompt.
   b. Change directory (**cd**) to the integration `composer\bin` directory.
   c. Run the following command, where <*password*> is the password for the Maximo database user (see Example 18-6):

      ```
      commandline taddm-2-ccmdb-actual-ci maximo password administrator collation
      maximo password
      ```

Example 18-6 Run commandline

```
C:\Integration Composer\bin>commandline.bat taddm-2-ccmdb-actual-ci maximo
smartway administrator collation maximo smartway
Integration Composer will now start.
IBM Tivoli Integration Composer 7.2.0.1 Build 200911031354
08 Jun 2010 19:30:55:078 [INFO] IBM Tivoli Integration Composer 7.2.0.1 Build
200911031354
Mapping: taddm-2-ccmdb-actual-ci
08 Jun 2010 19:30:55:109 [INFO] ObjectManager created. (This should only happen
once.)
Creating data source Fusion...08 Jun 2010 19:30:55:172 [INFO] GlobalContext
created. (This should only happen once.)
08 Jun 2010 19:30:55:172 [INFO] ResourceManager created. (This should only
happen once.) : 0 resource(s) loaded.
08 Jun 2010 19:31:24:297 [INFO] ResourceManager : Loading resource labels : 577
resource(s) loaded.
08 Jun 2010 19:31:25:016 [INFO] ResourceManager : Loading resource messages :
279 resource(s) loaded
.
08 Jun 2010 19:31:25:016 [INFO] IBM Tivoli Integration Composer database
version: V7116-173
        done
IBM Tivoli Integration Composer database version: V7116-173
Retrieving mapping definition taddm-2-ccmdb-actual-ci...        done
Creating source data source taddm72actualci...  done
Connecting to source data source taddm72actualci...
```

```
08 Jun 2010 19:31:27:313 [INFO] ResourceManager : Loading resource file
C:\Integration Composer\data
\properties\jdbcinfo.properties : 39 resource(s) loaded.
08 Jun 2010 19:31:28:953 [INFO] DataManager created. (This should only happen
once.)
08 Jun 2010 19:31:40:219 [INFO] Collation version: 2.2, build: null, home:
C:\Integration Composer
08 Jun 2010 19:31:42:828 [INFO] ResourceManager : Loading resource file
C:\Integration Composer\data
\properties\provider\ccmdb.properties : 853 resource(s) loaded.
Connecting to source data source taddm72actualci...     done
Creating target data source ccmdb72actualci...  done
Connecting to target data source ccmdb72actualci...
Connecting to target data source ccmdb72actualci...     done
Retrieving mapping expressions...       done
Retrieving mapping deciding classes...  done
retrieving mapping expressions 64
finished mapping expressions 1
finished mapping expressions 2
.
.
.
finished compiling 15
done
REMOVELISTENER
08 Jun 2010 19:32:29:500 [INFO]
=========================================================
               Start mapping execution
               Mapping: taddm-2-ccmdb-actual-ci
=========================================================
.
.
.
Actual CI: 71155   C0A01C1F7EBD3B02B979031AECA66E77
Actual CI: 71156   CAF8B47F3E353686936E008EC047FF4E
Actual CI: 71157   214725A156F9355AB0FF73B2A80083BF
Actual CI: 71158   D2702D161479375FBAF4CD5B079CC782
Actual CI: 71159   D946CC33E1613CFEB4BBA562697BAD5E
Actual CI: 71160   77973EC44C5E3AAB8C8194676416FF73
.
.
.
```

3. View the logs if necessary: `<installdir>\log\fusion.log`

Archived

18.6.7 Verifying that the data has been imported into CCMDB

Next, verify that the data has been imported into CCMDB:

1. In CCMDB, click **Go To** → **IT Infrastructure** → **Actual Configuration Items**. See Figure 18-46.

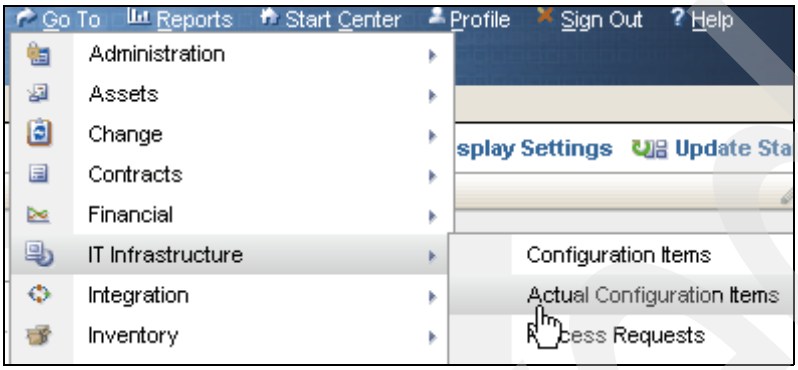


Figure 18-46 Actual Configuration Items

2. Search for items that should have been imported and confirm that they are present. If you find them, you have successfully imported the data. In the Classification field, type `app.softwareimage`. Figure 18-47 shows a list with the imported assets from Rational Asset Manager.

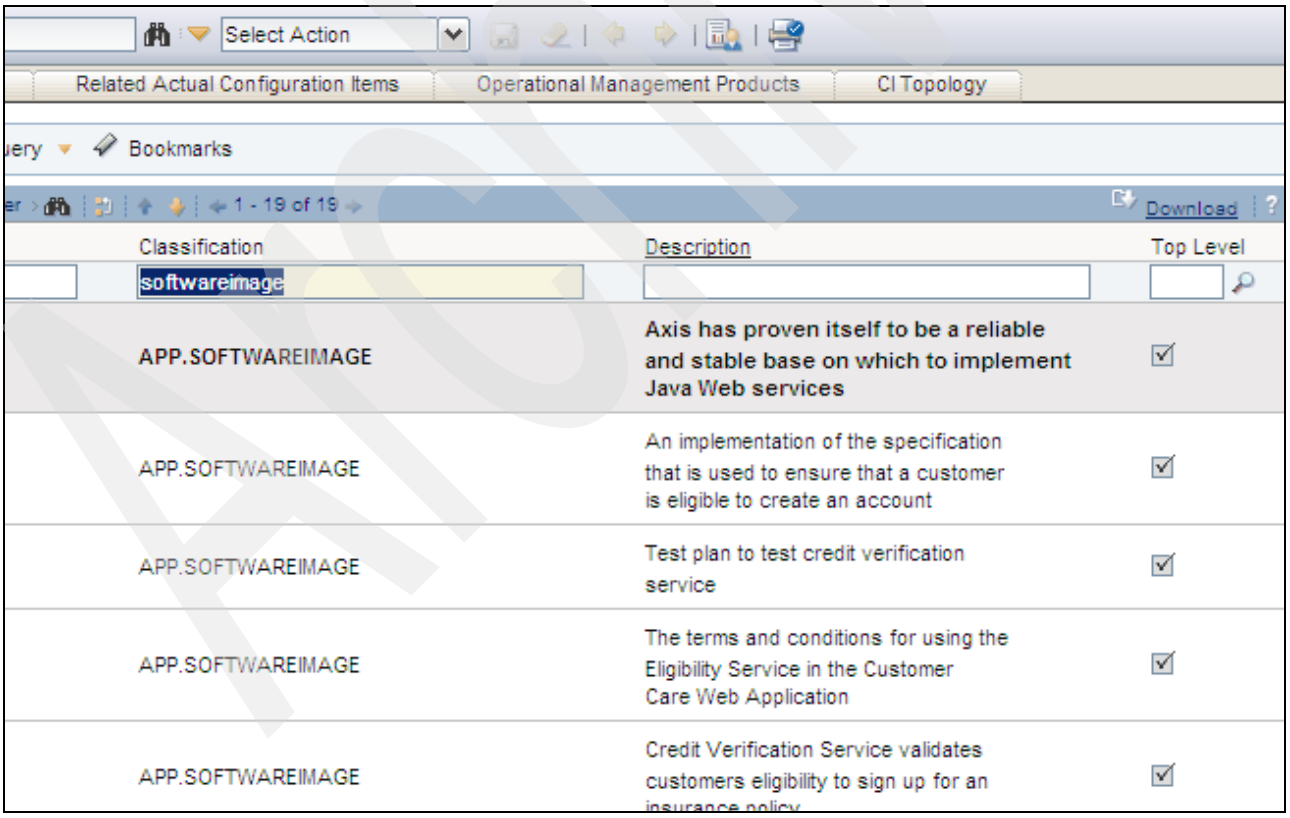


Figure 18-47 Imported assets

18.6.8 Creating and configuring the integration module in the CCMDB

At this time, the data from Rational Asset Manager is already in the CCMDB but it is not in the DML. Now, we configure to transfer the assets from the actual configuration items to the DML.

Set up web services in CCMDB

First you must set up web services in CCMDB:

1. Click **Go To** → **Integration** → **Web Services Library**. See Figure 18-48.

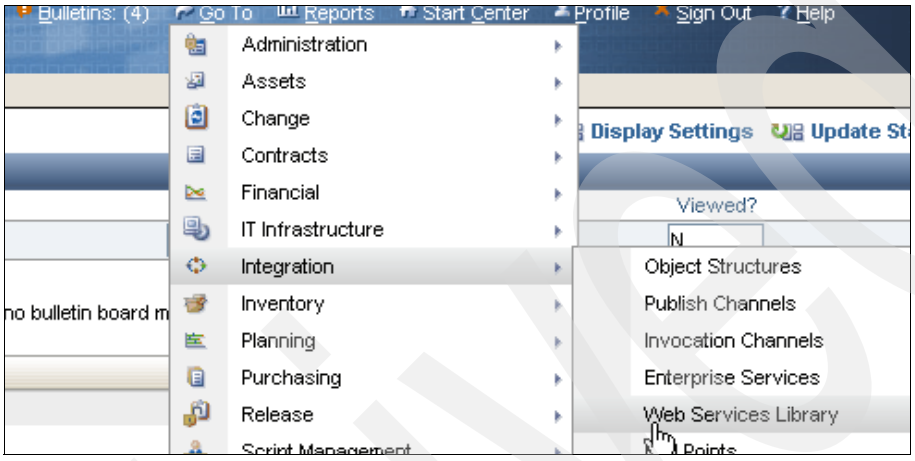


Figure 18-48 Web Services Library

2. In the Web Services Library, click **Select Action** → **Create Web Service** → **Create WS from Object Structure**. See Figure 18-49.

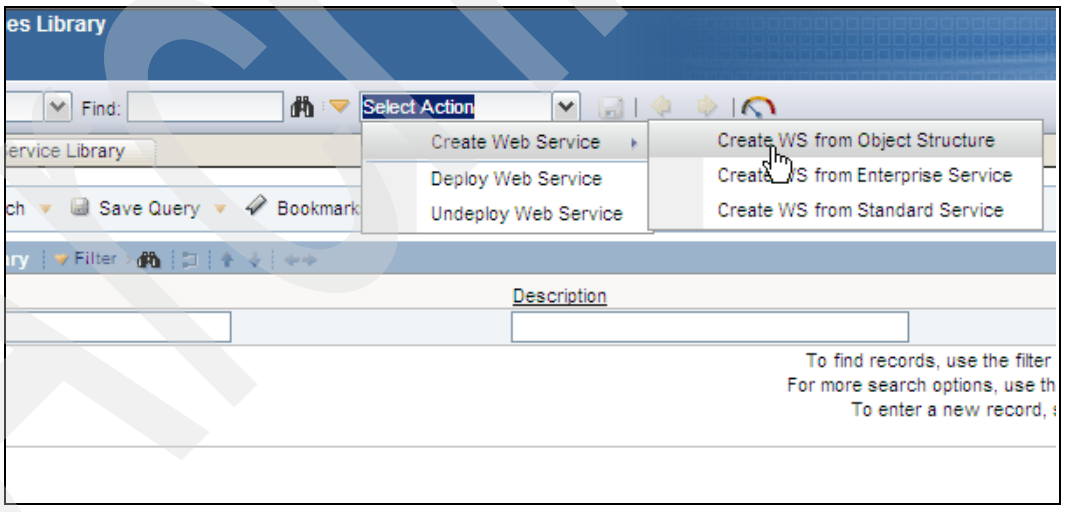


Figure 18-49 Create a Web Service from Object Structure

Archive

3. Create the following web services, as shown in Figure 18-50.
   - MXAUTHCI
   - MXCLASSIFICATION
   - MXOMP

Tip: Use the arrows to find those web services.

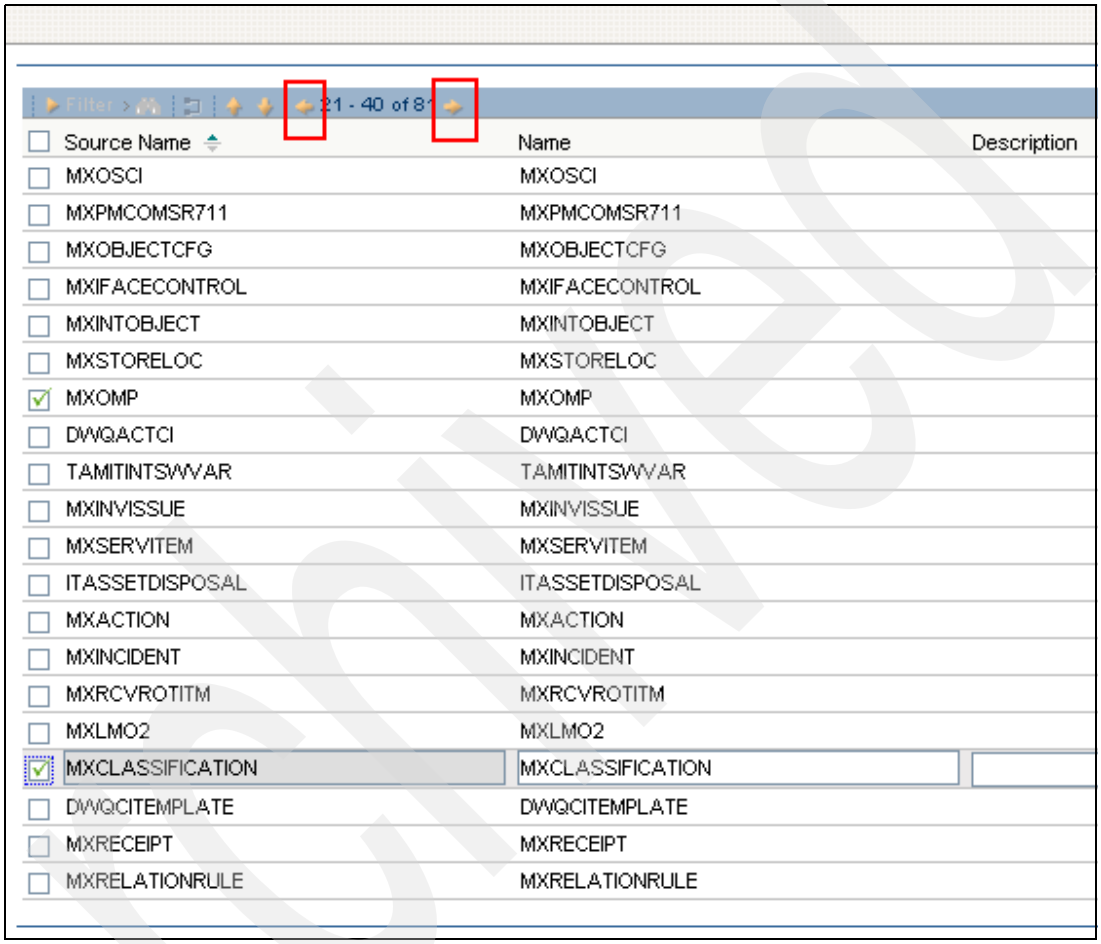


Figure 18-50 Web services

4. On the List page, verify that the Web Services were created. If they were, click **Select Records**, as shown in Figure 18-51.

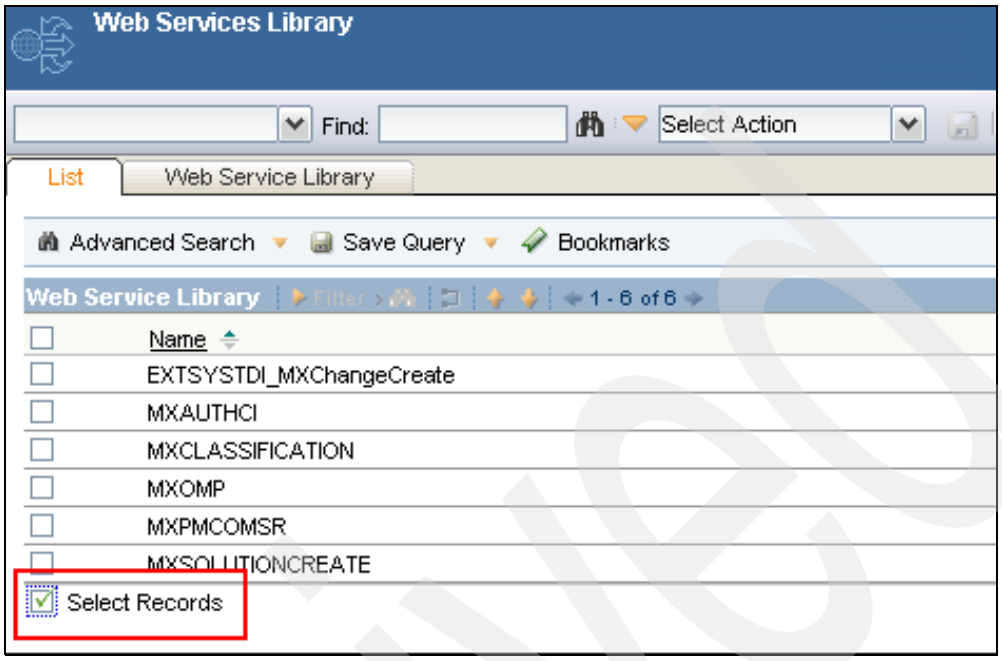


Figure 18-51 Select Records

5. Select the check box of each web service you created, and then click **Select Action menu → Deploy Web Service**. See Figure 18-52.

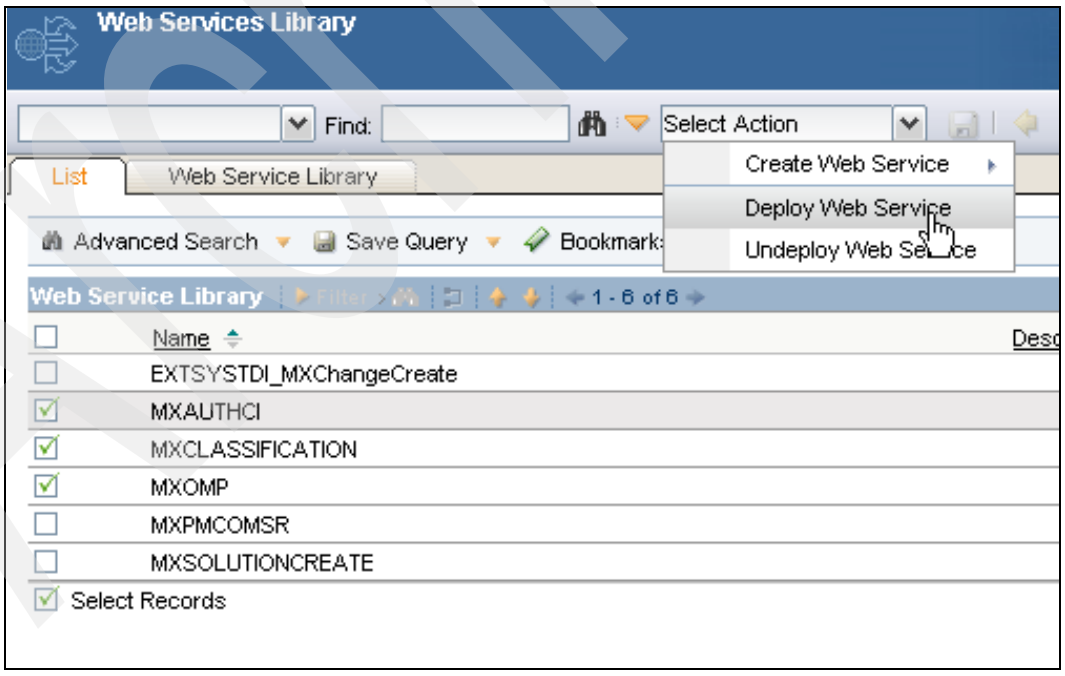


Figure 18-52 Deploy Web Service

Set up the Integration module

Next, set up an Integration module:

1. Click **Go To** → **Integration** → **Integration Modules**. See Figure 18-53.

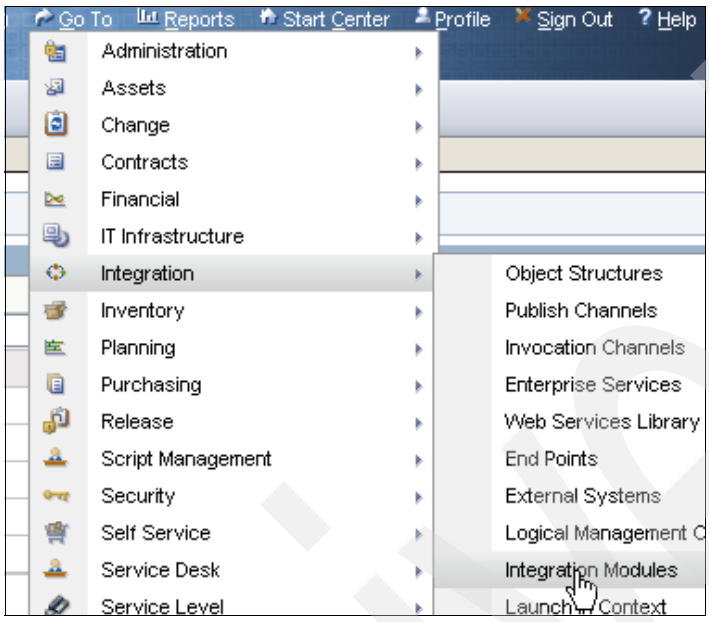


Figure 18-53 Opening Integration Modules application

2. Search for RAMIM by using the Integration Module page:

   a. Click the **Logical Management Operations** tab and verify that the IMPORTSOFTWAREIMAGE operation is added, as Figure 18-54 shows.

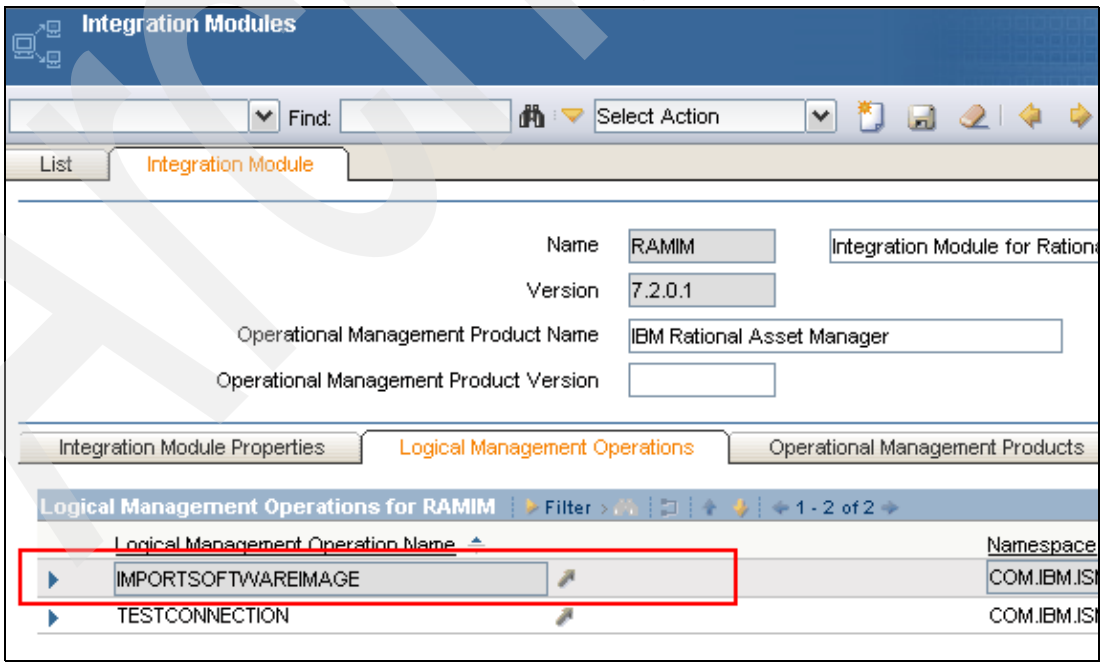


Figure 18-54 IMPORTSOFTWAREIMAGE operation is added

   b. Click the **Operational Management Products** tab.

c. Click the **New Row** under Operational Management Products for RAMIM to add the OMP.

d. Click the **Operational Management Product** and filter the product name by Rational. See Figure 18-55.

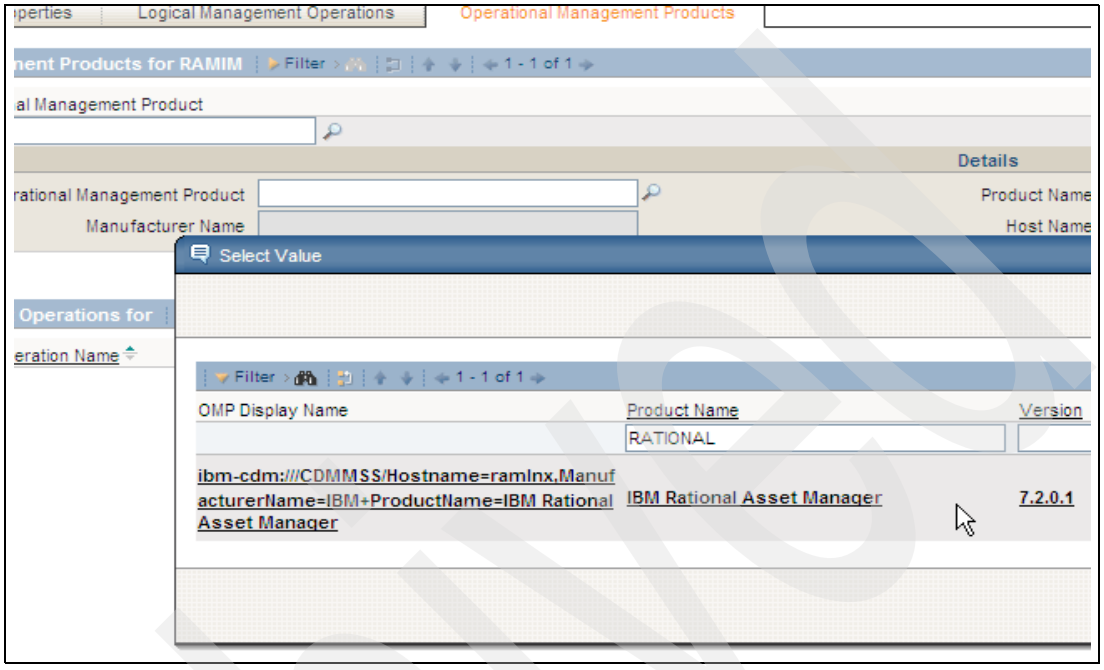


Figure 18-55 Select the OMP

Before saving, you have to define an endpoint.

Define an endpoint

Perform the following steps to define an endpoint:

1. Click the button next to the End Point field and click **Go To End Points**. See Figure 18-56.

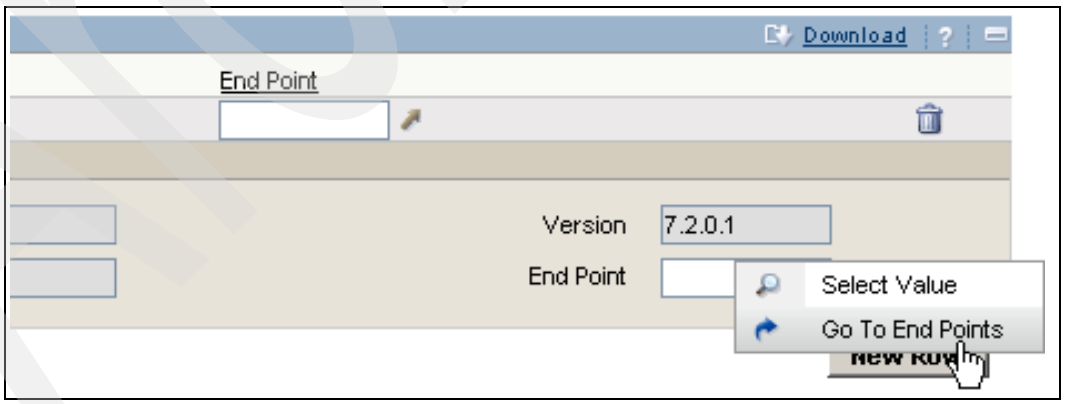


Figure 18-56 Go To End Points.

2. On the End Points page, click **New End Point** to create a new endpoint.
3. Enter `RAMENDPOINT` as name of the endpoint.
4. Click the **Handler** icon to select the handler.

5. Select **RAMHANDLER** (`com.ibm.ram.ccmdb.im.ram.RAMHandler` handler class name). See Figure 18-57.

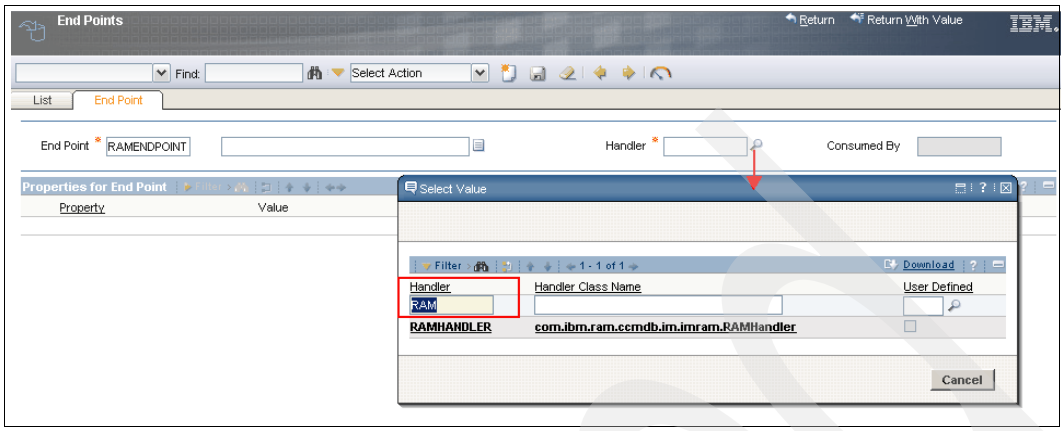


Figure 18-57 Select RAMHANDLER

6. Enter the URL of the Rational Asset Manager server application. First, locate the URL name as follows:
   a. Go to the Rational Asset Manager application.
   b. Click **Help** → **Extensions**. See Figure 18-58.

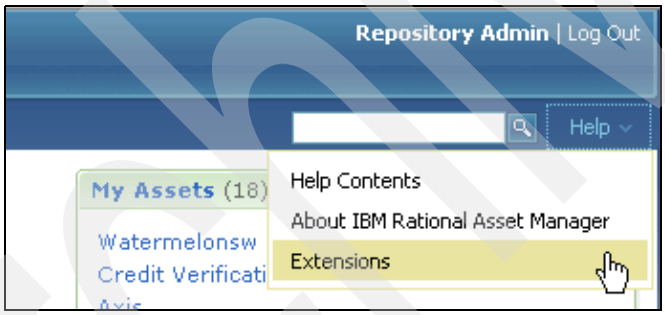


Figure 18-58 Extensions page from RAM

   c. Look for the repository location under the Eclipse Client Plug-in tab (Figure 18-59).

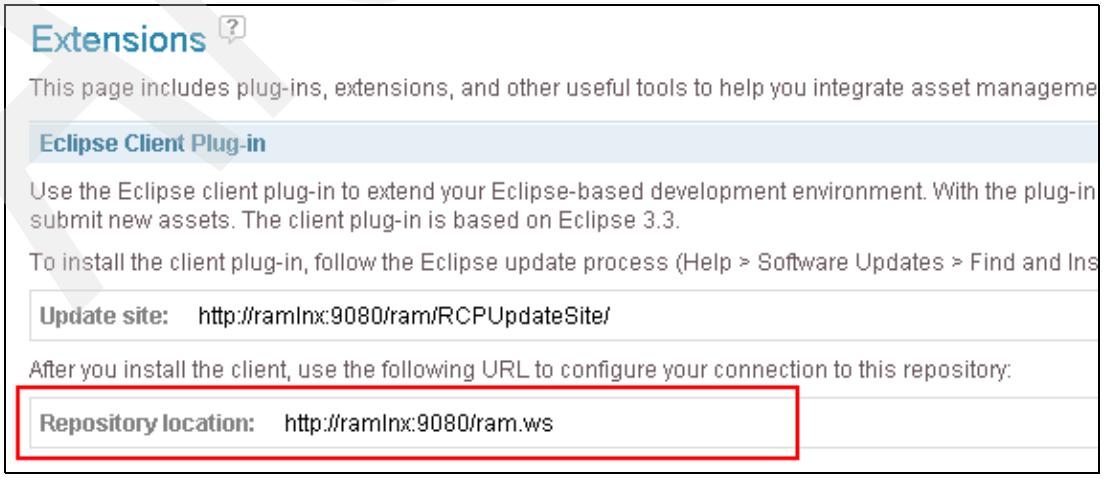


Figure 18-59 Extensions page

d. Return to the End Points application and type the name you have found. See Figure 18-60.

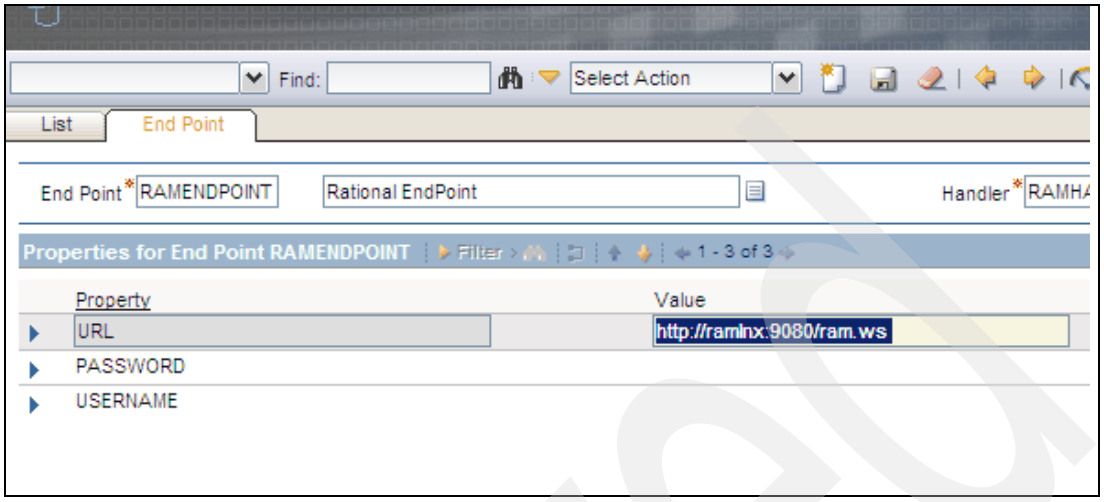


Figure 18-60 End Points page

7. Type the password for a Rational Asset Manager administrator.
8. Type the user ID for a Rational Asset Manager repository administrator. The window, shown in Figure 18-61 opens.

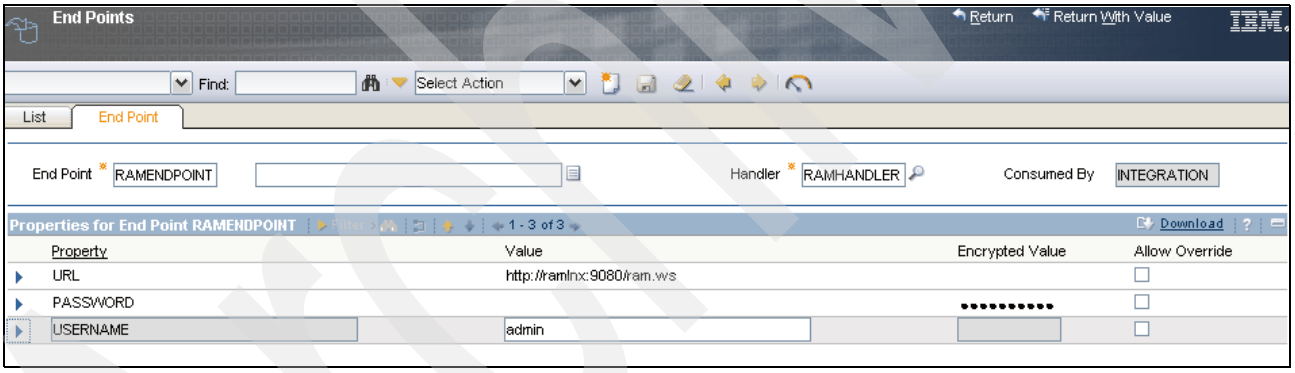


Figure 18-61 Rational Asset Manager repository administrator

9. Click **Save** to save the endpoint.
10. Click **Return With Value**.
11. On the Integration Module page, click **Save** to save the integration module configuration.

18.6.9 Creating the Definitive Media Library (DML)

Create the DML, as follows, because all the modules responsible for the integration are set:

1. In CCMDB, click **Go To** → **Release > Definitive Media Library**. See Figure 18-62.

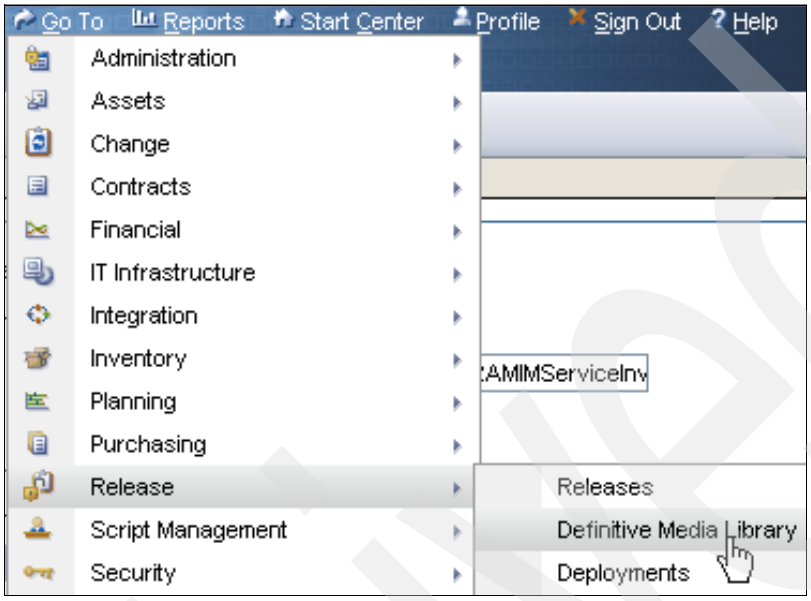


Figure 18-62 Create the DML

2. Click **Create DML**.
3. In the Name field, type `RAM 72` as the name of the DML configuration.
4. Optional: Type a description for the DML.
5. In the Owner field, click **Select Value** → **maxadmin** as owner for the DML. See Figure 18-63.

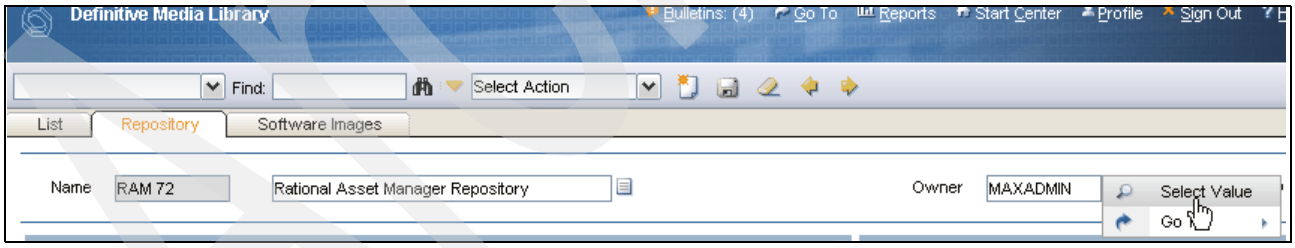


Figure 18-63 Owner field

6. Under the Details section, in the Classification Path field, click **Classify**. See Figure 18-64.

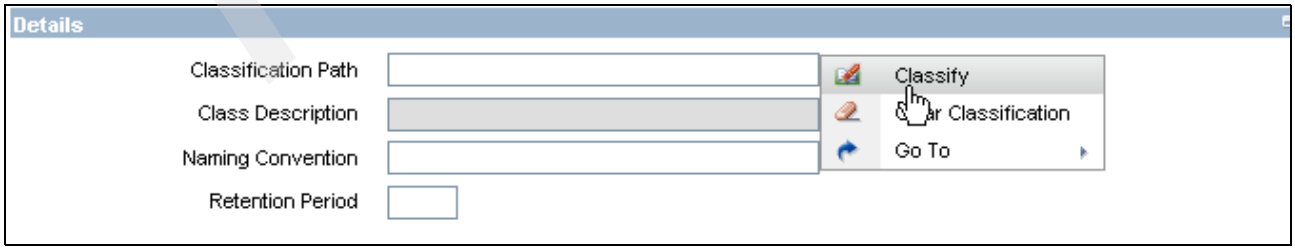


Figure 18-64 Details section

7. In the Classify window, select **Rational Asset Manager Repository**. The classification is from the Rational Asset Manager integration module that was installed in the CCMDB. See Figure 18-65.

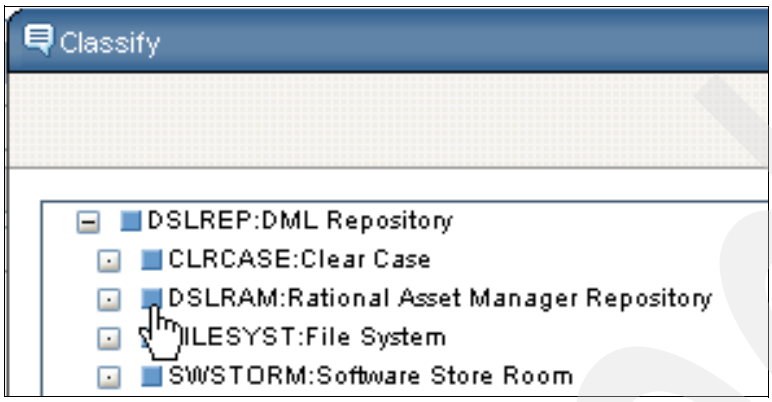


Figure 18-65 Classify window

8. In the Scope section, select the **Source Code** and **Binary** check boxes. See Figure 18-66.

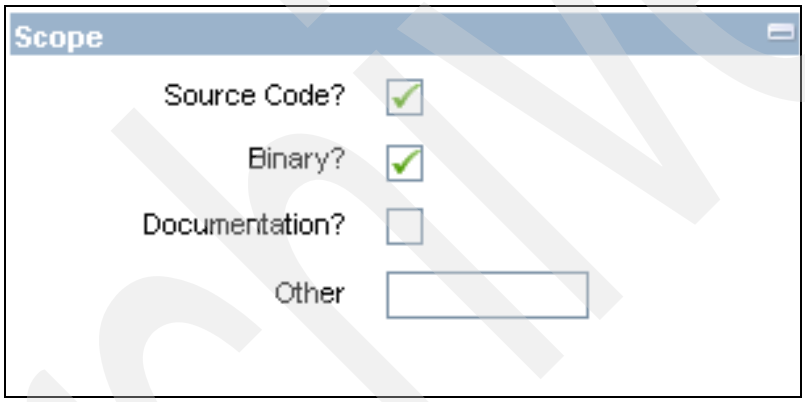


Figure 18-66 Scope section

9. In the Management Application section, click the arrow next to the name to select the OMP.

Archived

10. In the Select OMP window, click **Rational Asset Manager OMP** to use for this DML. See Figure 18-67.

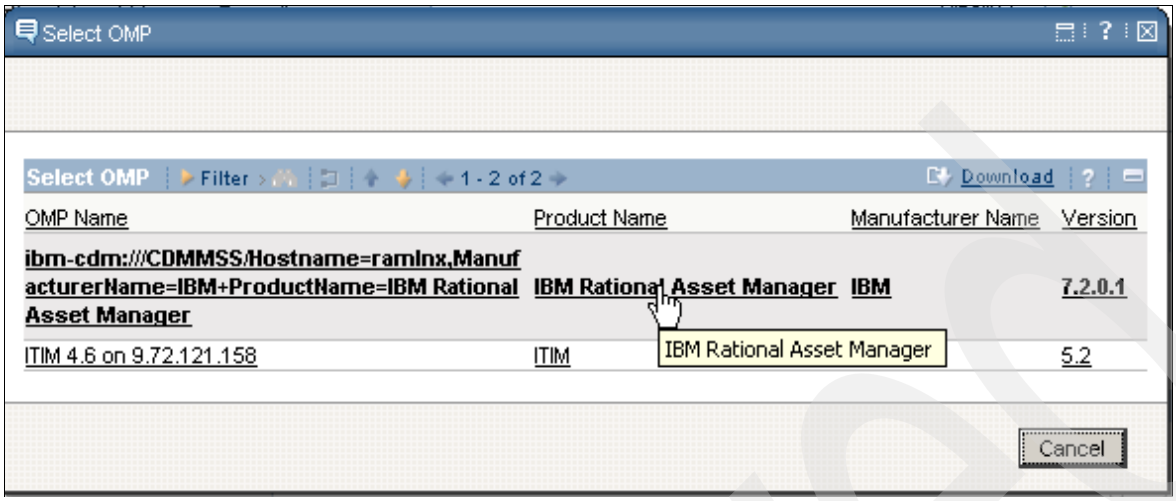


Figure 18-67 Rational Asset Manager OMP

11. Click **Save DML**.
12. Click the **Software Images** tab.
13. Click **Select from OMP**. See Figure 18-68.

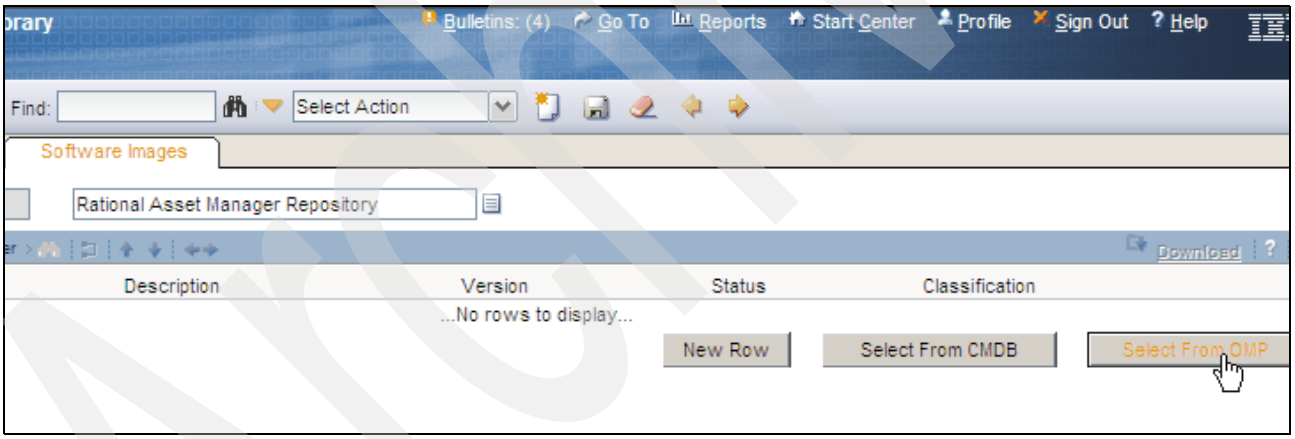


Figure 18-68 Select from OMP

Archived

14. Select the **Software Images** and click **OK**. See Figure 18-69.

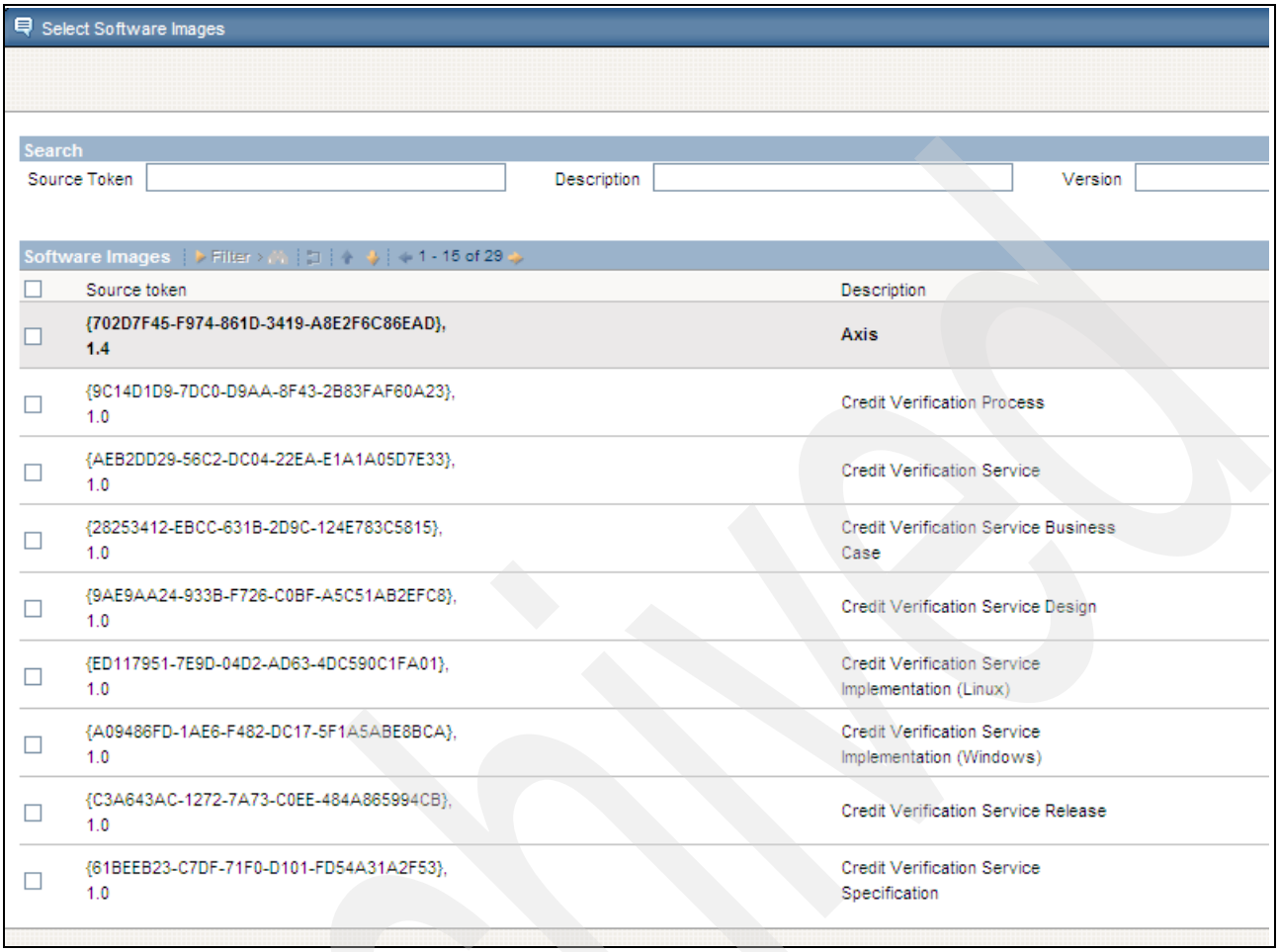


Figure 18-69 List of the imported software images

A list of all the imported software will be created and will be added to the CCMDB. Now the integration is done. All the software images from Rational Asset Manager are there.

15. Click **Save DML.**

The first part of integration is complete. All data from the Rational Asset Manager is in CCMDB.

Archive

18.6.10 Synchronizing the asset from CCMDB with Rational Asset Manager

As explained previously, synchronizing the data from CCMDB to the Rational Asset Manager is possible.

Perform the following steps:

1. Log in to **Rational Asset Manager** web client.
2. Go to the **RAMCCMDB** community (Figure 18-70).



Figure 18-70 RAMCCMDB community

3. Go to the Change and Configuration Management Database Connections section (Figure 18-71).

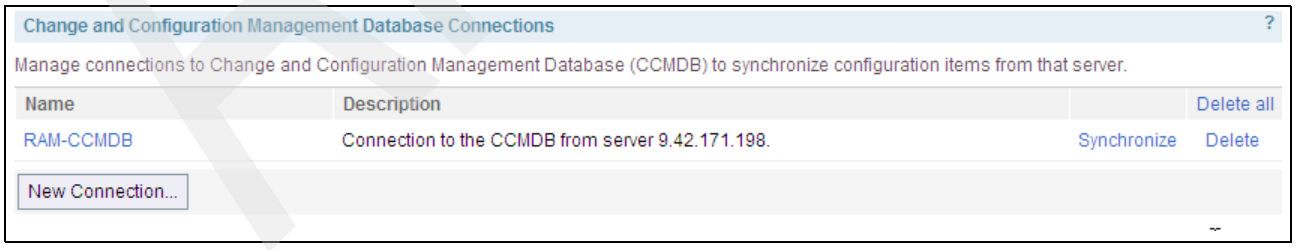


Figure 18-71 Change and Configuration Management Database Connections

4. Click **Synchronize**. The success message, shown in Figure 18-72, is displayed.

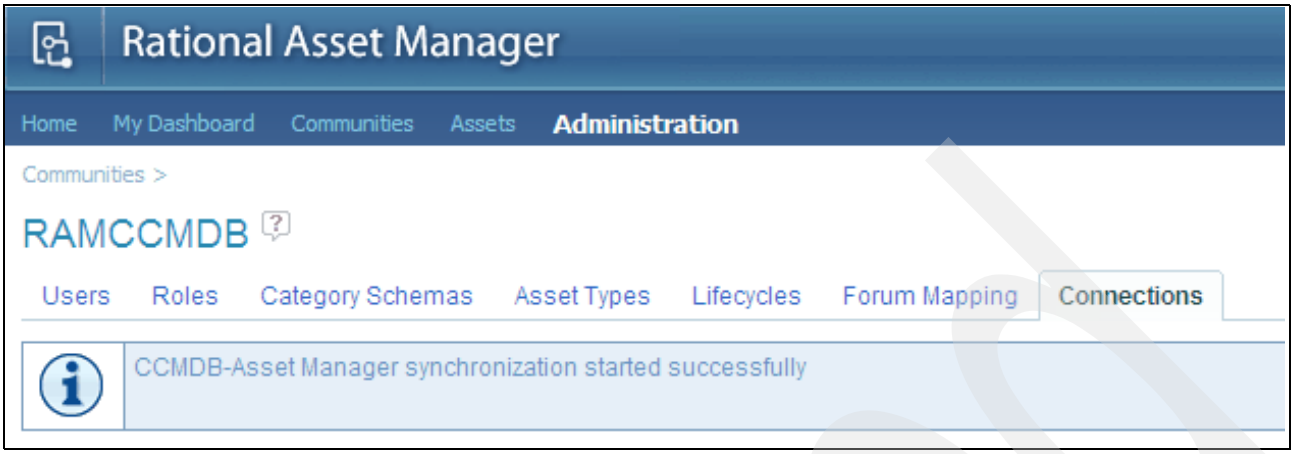


Figure 18-72 Synchronize

5. Click the **Assets** page.
6. Search for the remote asset. The search returns the CCMDB CIs. The CIs are synchronized from the Tivoli CCMDB to Rational Asset Manager.
7. Go to any remote asset. The remote assets in the Rational Asset Manager repository retain much of the information from the CCMDB CI.

Archived

18.7 Summary

Figure 18-73 shows the summary of this integration.

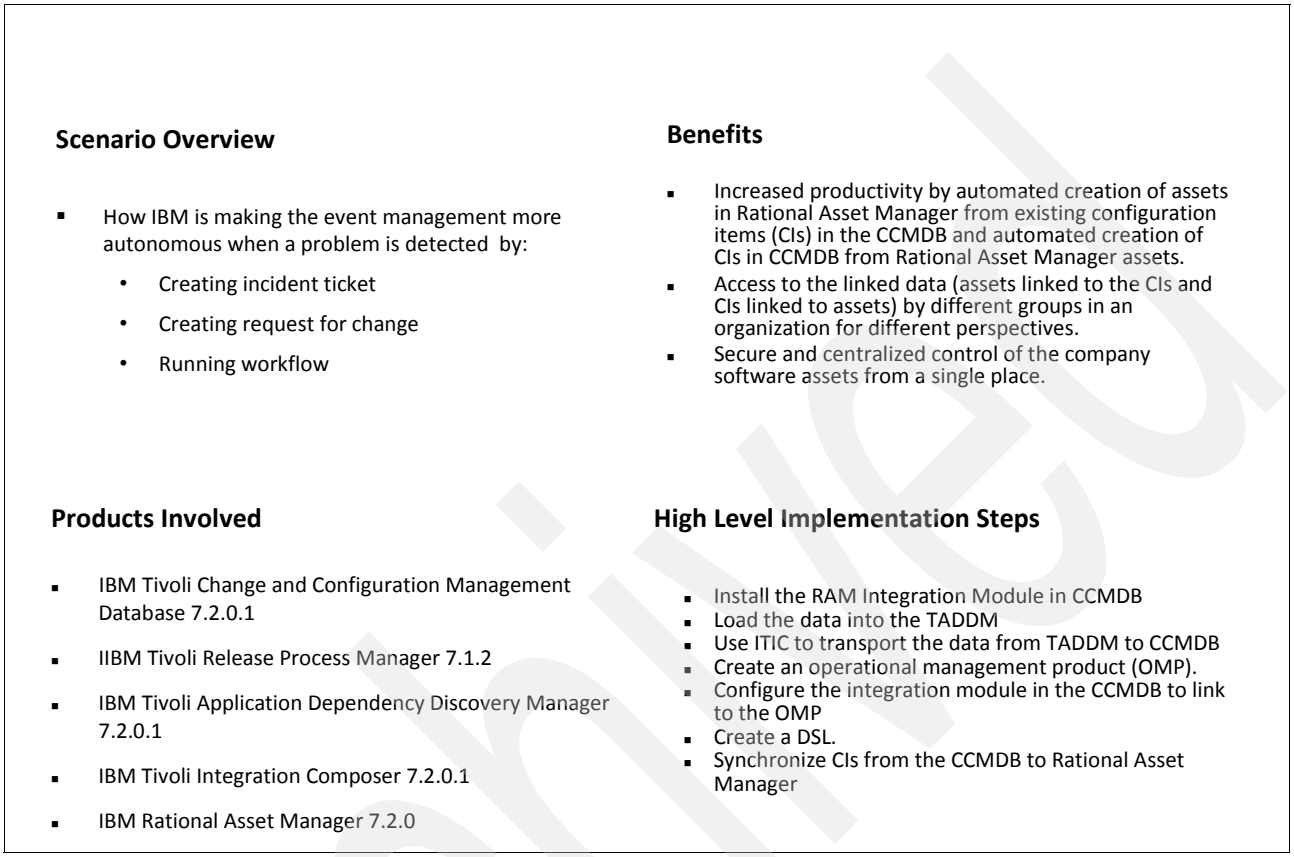


Figure 18-73 Quick summary

Archived

Archived

19

User and password management with Tivoli Identity Manager and Tivoli Service Request Manager

Integration of Tivoli Service Request Manager with Tivoli Identity Manager is described in this chapter.

This chapter contains the following topics:

- 19.1, “Scenario overview” on page 596
- 19.2, “Products involved” on page 596
- 19.3, “Benefits” on page 597
- 19.4, “Installation roadmap” on page 598
- 19.5, “Password change scenario” on page 615
- 19.6, “User management scenario” on page 618
- 19.7, “Summary” on page 622

© Copyright IBM Corp. 2011. All rights reserved.

19.1 Scenario overview

The integration between Tivoli Identity Manager and Tivoli Service Request Manager offers the following features:

- Management of Tivoli Service Request Manager users through Tivoli Identity Manager
- Ability to create Tivoli Service Request Manager service requests when any passwords are changed through Tivoli Identity Manager

Figure 19-1 shows the components of the integration between Tivoli Identity Manager and Tivoli Service Request Manager.

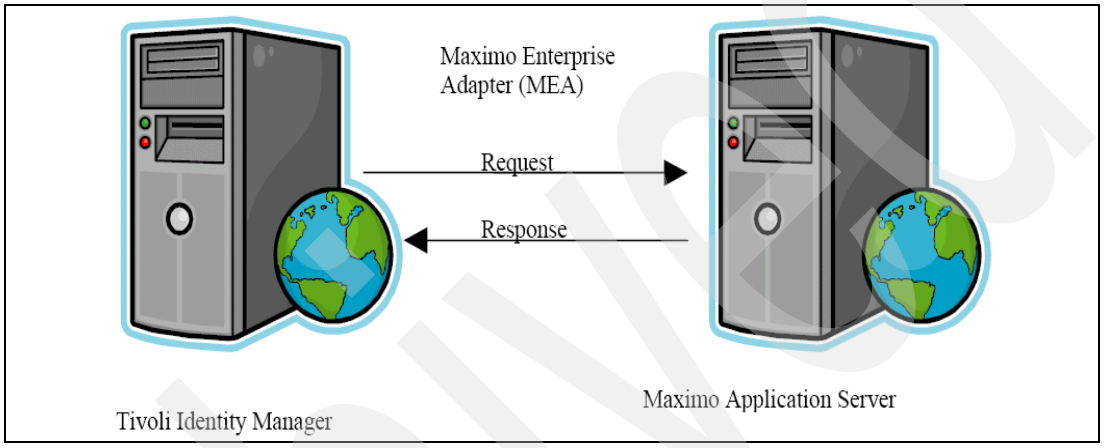


Figure 19-1 Architecture of the integration

Managing Lotus Notes® IDs: Tivoli Service Request Manager version 7.1 has an additional integration of managing Lotus Notes IDs. The integration provides Service Catalog offerings for user access to Lotus ID Account services (such as Reset Password and Create Account) using IBM Tivoli Identity Manager. These offerings allow automatic fulfillment of requests through integration.

Locate the Tivoli Identity Manager Integration for Tivoli Service Request Manager version 7.1 installation package on the IBM Tivoli Open Process Automation Library website:

http://www-01.ibm.com/software/brandcatalog/ismlibrary/details?catalog.label=1TW10SR06

19.2 Products involved

The following products and product components must be installed on one of the specified operating systems:

- IBM Tivoli Identity Manager Version 5.1 on Microsoft Windows, AIX, HP-UX, or Solaris
- IBM Tivoli Service Request Manager Version 7.2 on Windows, AIX, or Linux
- IBM Maximo Administration Machine with Base Services on Windows

The supported application and database servers are as follows for IBM WebSphere Application Server version 6.1.0.13 or later:

- Windows: IBM DB2, Oracle, or Microsoft SQL Server
- Linux or UNIX: IBM DB2 or Oracle

The Tivoli Service Request Manager product requires WebSphere Application Server and a database server. The supported database servers vary according to the operating system on which WebSphere Application Server is installed.

For information about installing prerequisite software, and specific versions of supported operating systems and database servers, see the information centers of the following products:

- IBM Tivoli Service Request Manager Version 7.2
- IBM Tivoli Identity Manager Version 5.1

To access the information centers, go to the following website:

https://www.ibm.com/developerworks/wikis/display/tivolidoccentral/Home

19.3 Benefits

The integration of Tivoli Identity Manager and Tivoli Service Request Manager provides the following benefits:

- Users can change their passwords in real time, which results in a dramatic reduction of tickets being raised formally through Service Desk agents.
- Lowers overall operational cost.
- Speeds up the process of user management in Tivoli Service Request Manager.

19.3.1 Manage Tivoli Service Request Manager users

By integrating Tivoli Identity Manager with Tivoli Service Request Manager, you can manage the Tivoli Service Request Manager user life cycle (add, modify, update and delete operations) through Tivoli Identity Manager. Use the Tivoli Service Request Manager native registry as the primary user repository to manage the users. When application server security is enabled, you can manage the Tivoli Service Request Manager users through Lightweight Directory Access Protocol (LDAP). You cannot use the Maximo Service Provider to manage the users. If Tivoli Service Request Manager is integrated with an LDAP system, such as Active Directory, you can use a Tivoli Identity Manager adapter, such as Active Directory adapter, to manage the users.

19.3.2 Create Tivoli Service Request Manager service requests

The majority of service requests in Tivoli Service Request Manager involves password changes. Automating the password change task by allowing the users to change their passwords in real time can help you to alleviate a process bottleneck. The Tivoli Identity Manager integration for Tivoli Service Request Manager has the ability to automate password change requests and create Service Request type tickets for all change Password operations that occur. The created Service Request is given the status of either Closed or New, depending on whether the change-password operation was successful in Tivoli Identity Manager. These tickets can be used to track all the change password operations. When a failure of this operation occurs, the tickets remain in the New state. These tickets can be routed automatically for fulfillment, thus reducing the time taken for resolution.

19.4 Installation roadmap

This section provides an overview of the required tasks to set up communication between Tivoli Identity Manager and Tivoli Service Request Manager.

After the prerequisite software is installed, perform the following tasks to set up Tivoli Identity Manager integration with Tivoli Service Request Manager:

- Install the integration package for Tivoli Identity Manager Communication with Tivoli Service Request Manager.
- Configure the Tivoli Service Request Manager server.
- Configure the Tivoli Identity Manager server.

19.4.1 Installing the integration package

Perform the following steps to install the integration package:

1. The integration package for Tivoli Identity Manager and Tivoli Service Request Manager integration is available in the `$ITIM_HOME\extensions\5.1\maximo` directory. The file that contains the package is `tim_sd_integration.zip`.The integration package consists of three folders: `applications`, `tim_51`, and `tools`. Extract the applications and tools folders to the Tivoli Service Request Manager.
2. Copy the `tim_sd_integration.zip` file to the Tivoli Service Request Manager base services installation directory. In Windows, the default installation directory is as follows:

   `C:\ibm\smp\maximo`
3. Extract the file to this directory. When the `tim_sd_integration.zip` file is extracted into the base services installation directory, it automatically adds the required files to the correct directories.

After extracting, verify that the files have been copied correctly as listed in Table 19-1.

Table 19-1 File and directory information.

File name	Directory
MaxUserProcess.class	$TSRM_HOME\applications\maximo\businessobjects\classes\psdi\iface\app\user
V710_02.db2 V710_02.ora V710_02.sqs	$TSRM_HOME\tools\maximo\en\itim_pmp

On the Tivoli Identity Manager server extract the `maximo.jar` and `maximoserviceprofile.jar` files to a directory. Copy the `maximo.jar` file to the Tivoli Identity Manager Shared Library, `$ITIM_HOME\lib` directory.

19.4.2 Configuring the Tivoli Service Request Manager server

The first step in the configuration of Tivoli Service Request Manager Server is the installation the Maximo integration interfaces that are required for communication between the Tivoli Service Request Manager application server and the IBM Tivoli Identity Manager Server. This step is done by running the **`updatedb`** script provided with the base services.

Running updatedb.bat

To run the script complete the following steps:

> **Note:** In these instructions, `TSRM_Install` refers to the directory where Base Services is installed.

1. Open a Windows command prompt.
2. Change to the `TSRM_Install\tools\maximo` directory.
3. Run the **updatedb.bat** script.

After the script finishes processing, complete the procedure to activate the Tivoli Service Request Manager interfaces.

Activating the Tivoli Service Request Manager interfaces

Complete the following procedure to deploy web services that activate the interfaces that are required for communication between Tivoli Service Request Manager and Tivoli Identity Manager:

1. Log on to the Tivoli SRM Server with Administrative permission.
2. In the Start Center window, click **Go To** → **System Configuration** → **Platform Configuration** → **System Properties**, as shown in Figure 19-2.

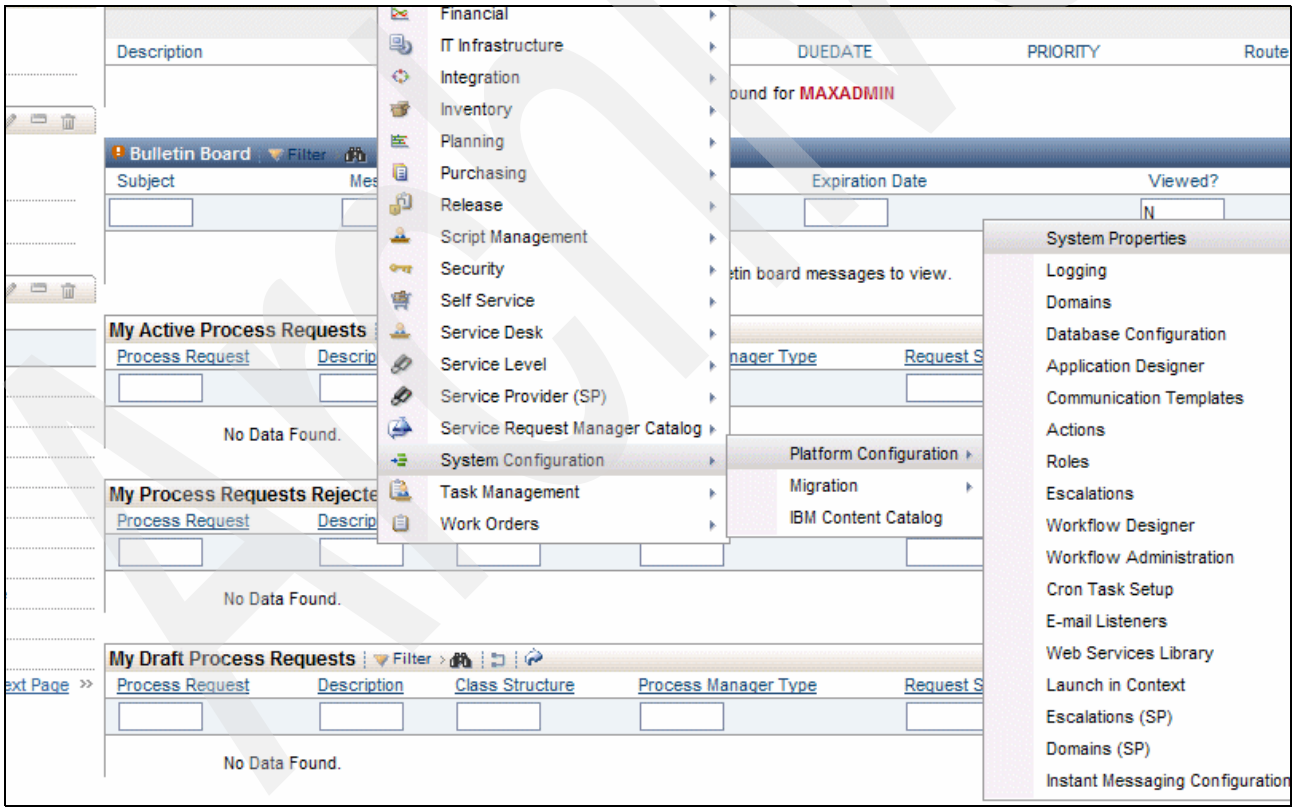


Figure 19-2 System Properties

3. Configure the Web Application URL to match the address of the Tivoli Service Request Manager application server:
   a. In the Property Name filter box, type `webappurl`.
   b. Press Enter.
   c. Expand `mxe.int.webappurl`.
   d. In the Global Value field, type the address of the Tivoli SRM server, for example:
      `http://tsrmserver.ibm.com:9080/meaweb`
   e. Save the property.
   f. Select the value by checking the box next to `mxe.int.webappurl`.
   g. Click **Live Refresh** to update the property value. See Figure 19-3.

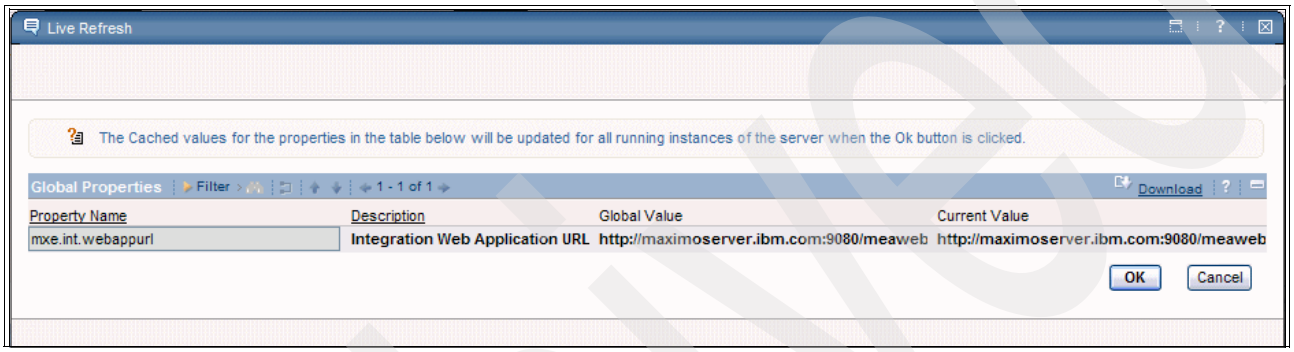


Figure 19-3 Live Refresh button

Click **OK** to update all running instances of the server with the cached values for the property.

4. Click **Go To** → **Integration** → **Web Services Library**.
5. Click **Select Action** → **Create Web Service** → **Create WS from Object Structure**. See Figure 19-4.

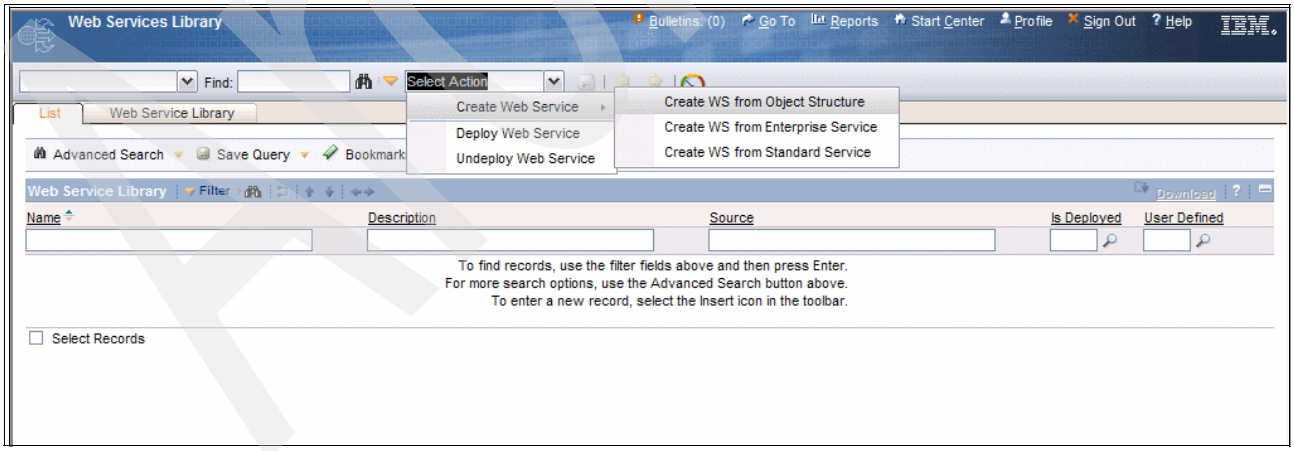


Figure 19-4 Create WS from Object Structure

6. In the Create Web Service from an Object Structure Service Definition window, select the following interfaces and click **Create**, as shown in Figure 19-5 on page 601.'
   – GRPASSIGN
   – MAXATTRIBUTE

- PERSON
- SITE
- SR
- SYNONYMDOMAIN
- USER
- USERQ

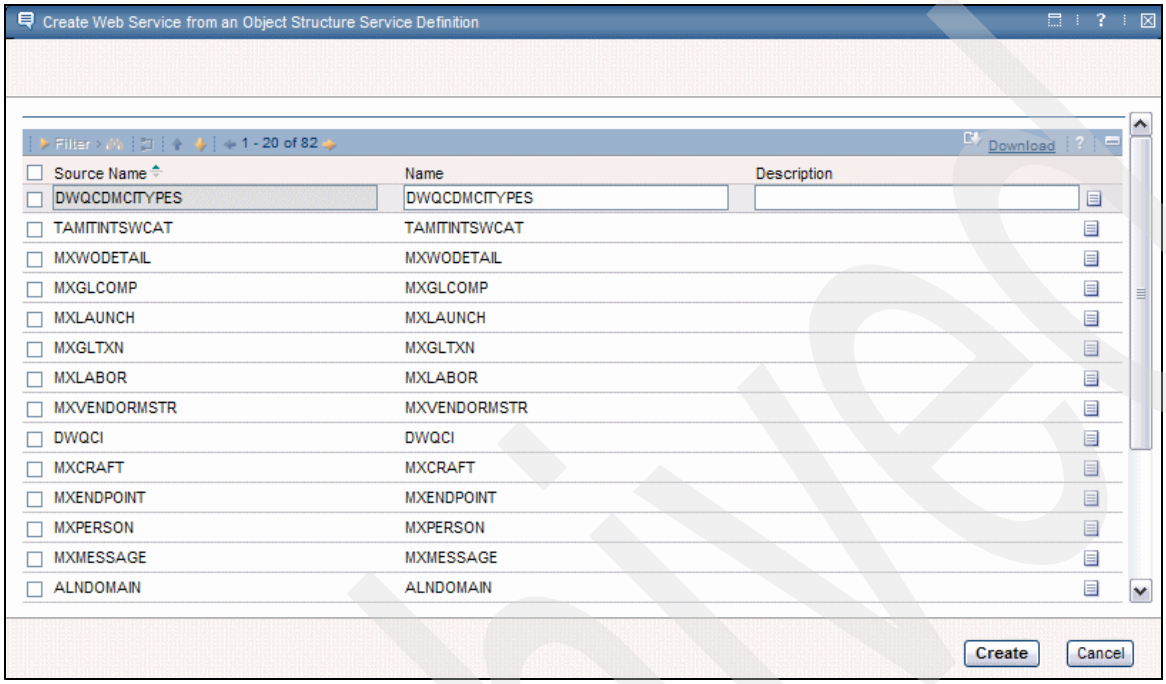


Figure 19-5 Create Web Service from an Object Structure Service Definition window

7. Activate the created interfaces by selecting the Object Structure Service Library interfaces and click **Select Action** → **Deploy Web Service**. See Figure 19-6.

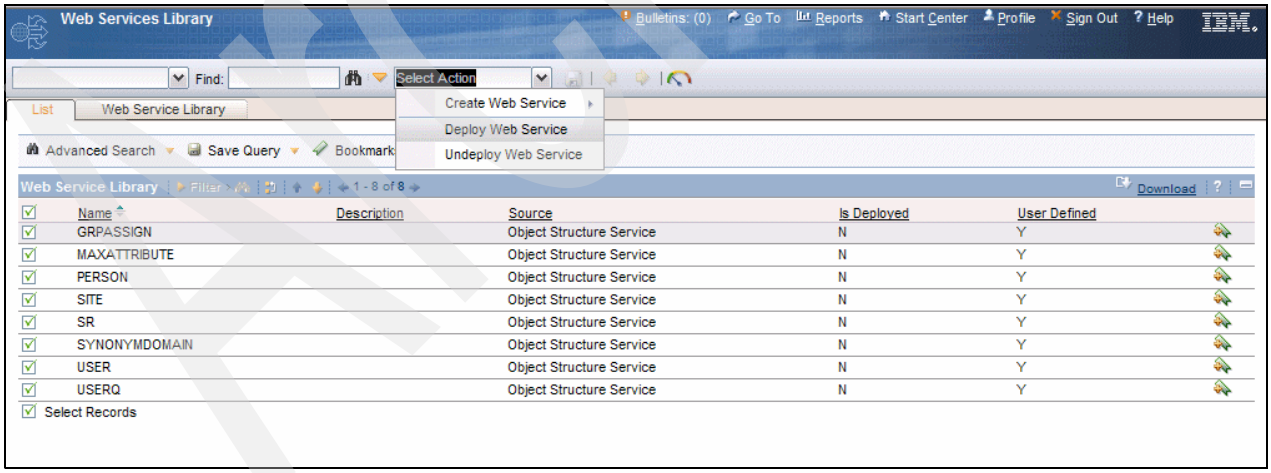


Figure 19-6 Web Services Library

Enabling Tivoli Service Request Manager user deletion

Tivoli Service Request Manager does not allow users to be deleted when the `LOGINTRACKING` variable is enabled. If you would like to be able to delete Tivoli Service Request Manager users then this variable needs to be disabled.

This can be performed using the following steps:

1. Log on to the Tivoli Service Request Manager server with administrative permissions.
2. Click **Go To** → **Security** → **Users**.
3. Click **Select Action** → **Security Controls** (Figure 19-7).

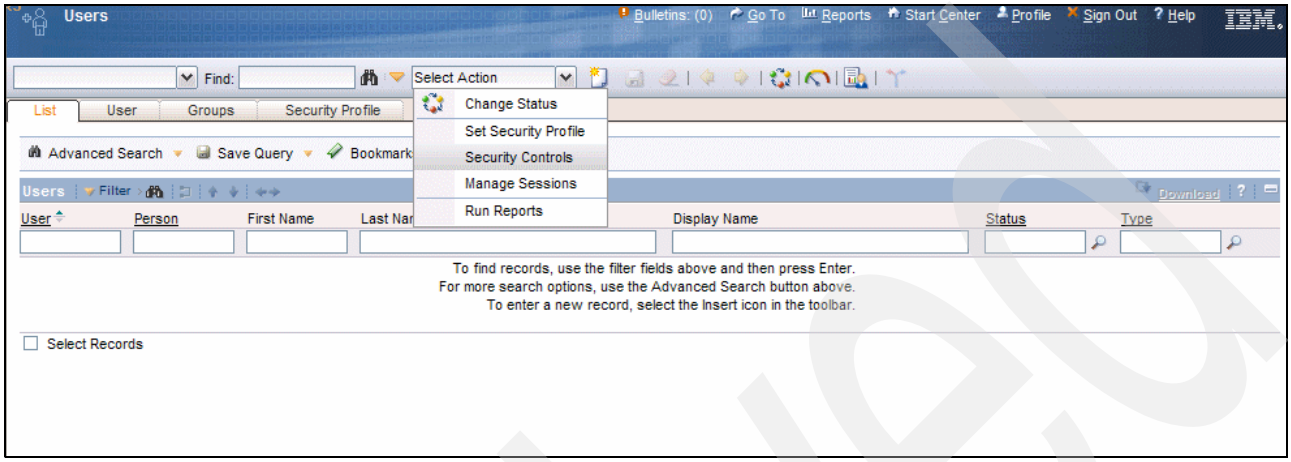


Figure 19-7 Users menu

4. Clear the **Enable Login Tracking?** check box, as shown in Figure 19-8.

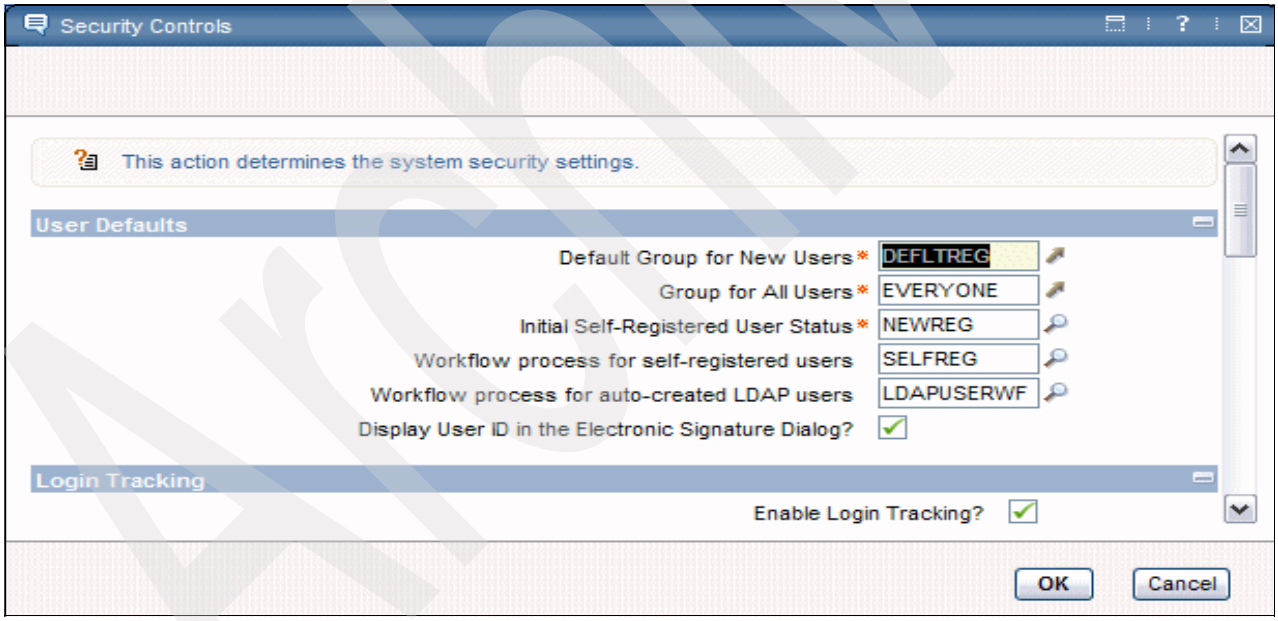


Figure 19-8 Enable Login Tracking

Important: If the Enable Login Tracking check box is not selected, you still must select the Maximo User Deletion Enabled check box on the Tivoli Service Request Manager service form. This check box in Tivoli Identity Manager is required to delete Tivoli Service Request Manager users. More information about configuring a Tivoli Service Request Manager service is in “Configure Tivoli Identity Manager Server” on page 610.

Adding password link to Tivoli Service Request Manager

This step is optional in the configuration. Tivoli Identity Manager can manage the Tivoli Service Request Manager users when either the native registry is used or when LDAP is used to store the user information. LDAP is used to store the user information when J2EE application server security is enabled. When the Tivoli Service Request Manager native registry is used, the Tivoli Service Request Manager service provider can be used to manage the users. However when LDAP is used, only the LDAP adapter can be used to manage the Tivoli Service Request Manager users through Tivoli Identity Manager.

The “Forgot Your Password?” link is not enabled when J2EE application server security is enabled. However, it is enabled when the native registry is being used. By optionally pointing the “Forgot Your Password?” link to the Tivoli Identity Manager self-service user interface, the password can be reset when either LDAP or the native registry is being used to store Tivoli Service Request Manager users providing that a service has been configured in Tivoli Identity Manager to manage the Tivoli Service Request Manager server. The Tivoli Service Request Manager interface can be optionally modified to point the “Forgot Your Password?” link to the Tivoli Identity Manager self-service interface. This way enable the Tivoli Service Request Manager user to manage the password through Tivoli Identity Manager and even reset the Tivoli Service Request Manager password if the user forgets it and is unable to log in.

Add the link to the Tivoli Service Request Manager login page as follows:

1. Navigate to the following directory:

   `Base_Service_Install\applications\maximo\maximouiweb\webmodule\webclient\login`

2. Modify the **`login.jsp`** file by adding the following text that is highlighted in bold as shown in Example 19-1.

   Example 19-1 Modify login.jsp

```
function forgotPwd()
{
showWait();
<!-- document.location = '<%=url + (hasQuery?"&":"?")
+"event=loadapp&value=forgotpswd&login=selfreg"
+ (langcode == null ? "" : "&langcode=" + langcode)%>'; -->
document.location = 'http://hostname:port/itim/self';
```

3. Modify the forgotPwd() function in **`login.jsp`** by commenting out the first `document.location` line and adding the second `document.location` line that points to the Tivoli Identity Manager Self Service interface.

4. Be sure to replace hostname and port with the appropriate values for the specific Tivoli Identity Manager deployment.

Configure Tivoli Service Request Manager WebSphere Server

The `MaxUserProcess.class` file that is provided with the Tivoli Identity Manager Integration must be built into the `maximo.ear` file for the integration to function properly. To build these into the `maximo.ear` file, complete the following steps:

1. Extract the `tim_sd_integration.zip` file to the `Base_Service_Install` directory. This step is explained in 19.4.1, “Installing the integration package” on page 598.

2. Open a Windows command prompt on the machine where the Base Services is installed.

3. Change to the following directory: `Base_Service_Install\deployment`

4. Run the **buildmaximoear.cmd** command to rebuild the `maximo.ear` file.

   The **buildmaximoear.cmd** command rebuilds the `maximo.ear` file while automatically pulling in the modified class files to replace the class files that were originally included in the `maximo.ear` file. See Figure 19-9.

```
C:\ibm\SMP\maximo\deployment>buildmaximoear.cmd
BuildMaximoEarTask process starts
Log File: buildmaximoear20100604031329.log
Unable to locate tools.jar. Expected to find it in C:\ibm\SMP\maximo\tools\java\lib\tools.jar
Buildfile: .\buildmaximoear.xml

init:
     [echo] maximo.deploydir=C:\ibm\SMP\maximo\deployment\default
     [echo] maximo.basedir=.\..\applications\maximo

propertiesBuild:
     [echo] properties.jar file=C:\ibm\SMP\maximo\deployment\default\temp/properties.jar
     [copy] Copying 1 file to C:\ibm\SMP\maximo\deployment\default\temp\properties
      [jar] Building jar: C:\ibm\SMP\maximo\deployment\default\temp\properties.jar

businessObjectsBuild:
     [echo] businessobjects.jar file=C:\ibm\SMP\maximo\deployment\default\temp/businessobjects.jar
   [delete] Deleting 1 files from C:\ibm\SMP\maximo\deployment\default\temp\importreports
      [zip] Building zip: C:\ibm\SMP\maximo\deployment\default\temp\importreports\reports.zip
_
```

Figure 19-9 buildmaximoear.cmd

5. Allow this process to complete successfully, as shown in Figure .

```
mboEJBBuild:
     [echo] mboEJBBuild jar file=C:\ibm\SMP\maximo\deployment\default\temp/mboejb.jar
      [jar] Building jar: C:\ibm\SMP\maximo\deployment\default\temp\mboejb.jar

meaWarBuild:
     [echo] meaWarBuild war file=C:\ibm\SMP\maximo\deployment\default\temp/meaweb.war
      [war] Building war: C:\ibm\SMP\maximo\deployment\default\temp\meaweb.war

mboJavaBuild:
     [echo] mboJavaBuild jar file=C:\ibm\SMP\maximo\deployment\default\temp/mbojava.jar
      [jar] Building jar: C:\ibm\SMP\maximo\deployment\default\temp\mbojava.jar

tipchartwsWarBuild:
     [echo] tipchartws.war file=C:\ibm\SMP\maximo\deployment\default\temp/tipchartws.war
      [war] Building war: C:\ibm\SMP\maximo\deployment\default\temp\tipchartws.war

earBuild:
      [ear] Building ear: C:\ibm\SMP\maximo\deployment\default\maximo.ear
   [delete] Deleting 11 files from C:\ibm\SMP\maximo\deployment\default\temp
   [delete] Deleting 52 files from C:\ibm\SMP\maximo\deployment\default\temp\properties
   [delete] Deleting directory C:\ibm\SMP\maximo\deployment\default\temp

BUILD SUCCESSFUL
Total time: 18 minutes 5 seconds
C:\ibm\SMP\maximo\deployment>_
```

Figure 19-10 buildmaximoear.cmd

6. Copy the new `maximo.ear` file from the `Base_Service_Install\deployment\default` folder to any location on the WebSphere server.

Deploy updated Maximo application on the WebSphere Server

To deploy the updated `maximo.ear` file on WebSphere, complete the following steps:

1. Log on to the WebSphere Administrative Console.
2. Click **Applications** in the navigation pane.

Archived

3. Select **Enterprise Applications**. The Enterprise Applications window is displayed, as shown in Figure 19-11.

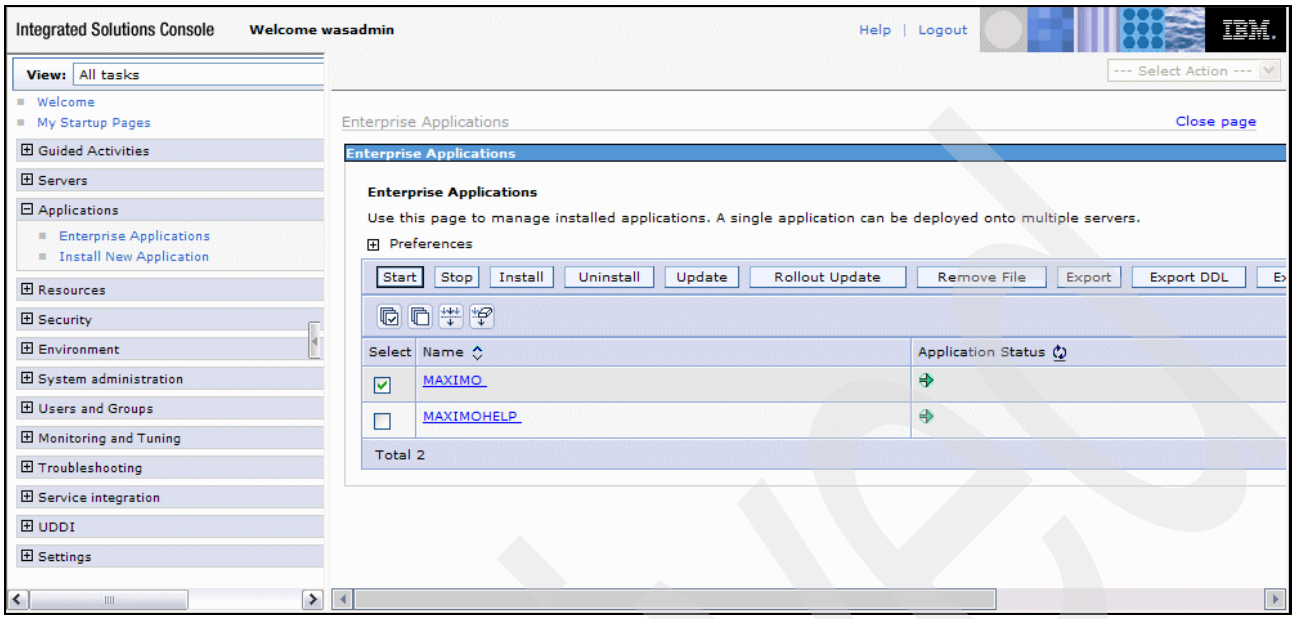


Figure 19-11 Enterprise Applications window

4. Select the check box next to **MAXIMO**.
5. Click **Update**.
6. Click **Replace the entire application**.
7. Click **Remote File System**. See Figure 19-12.

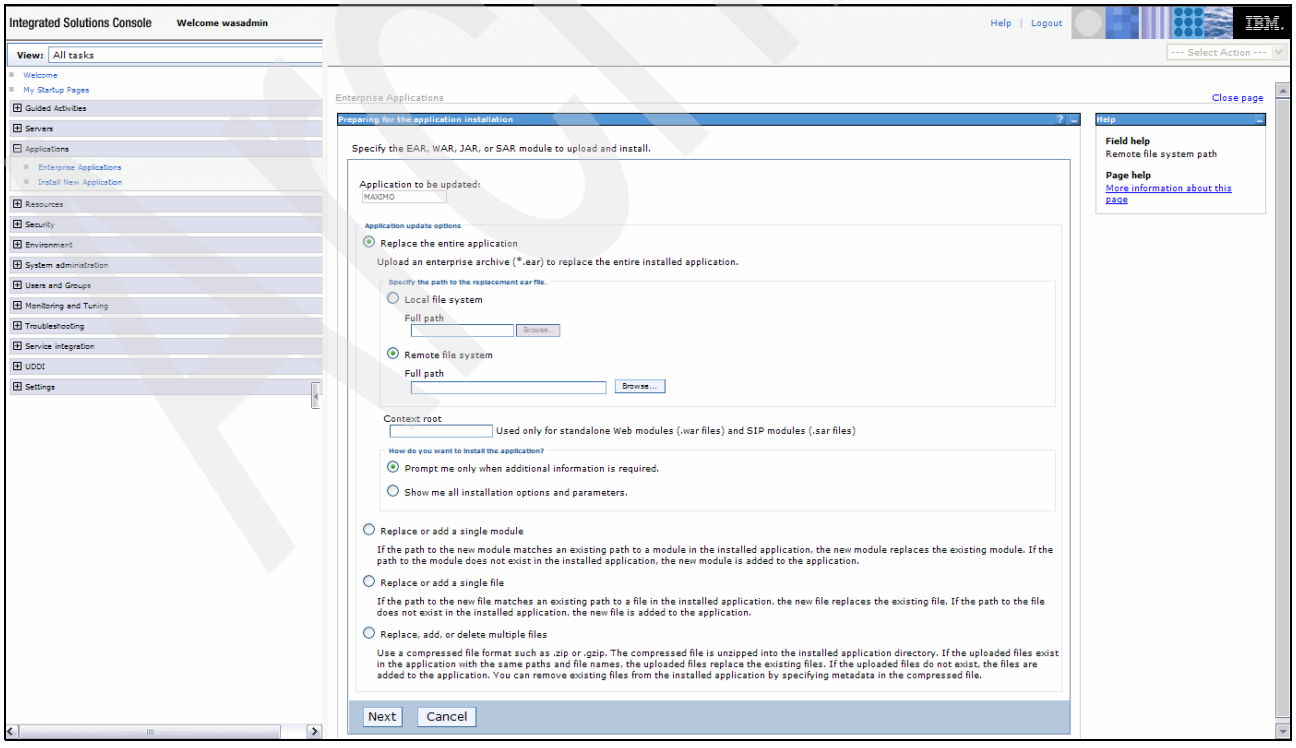


Figure 19-12 Remote File System.

8. Click **Browse**.
9. Select the node of your WebSphere server.
10. Browse to the location of the `maximo.ear` file that you copied, as shown in Figure 19-13.

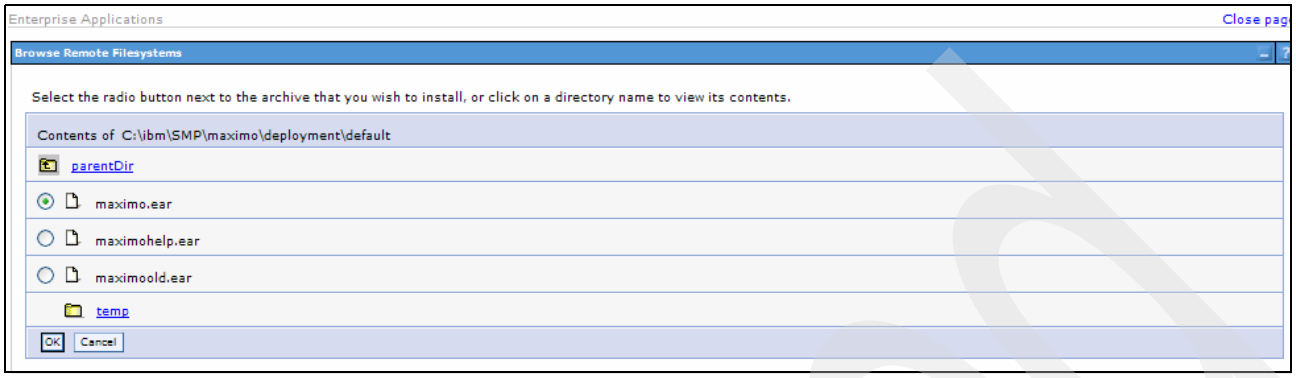


Figure 19-13 Browse to the location of the maximo.ear

11. Select the file.
12. Click **OK**.
13. Click **Next**.

Archived

14. Click **Next** on the Select installation options panel. See Figure 19-14.

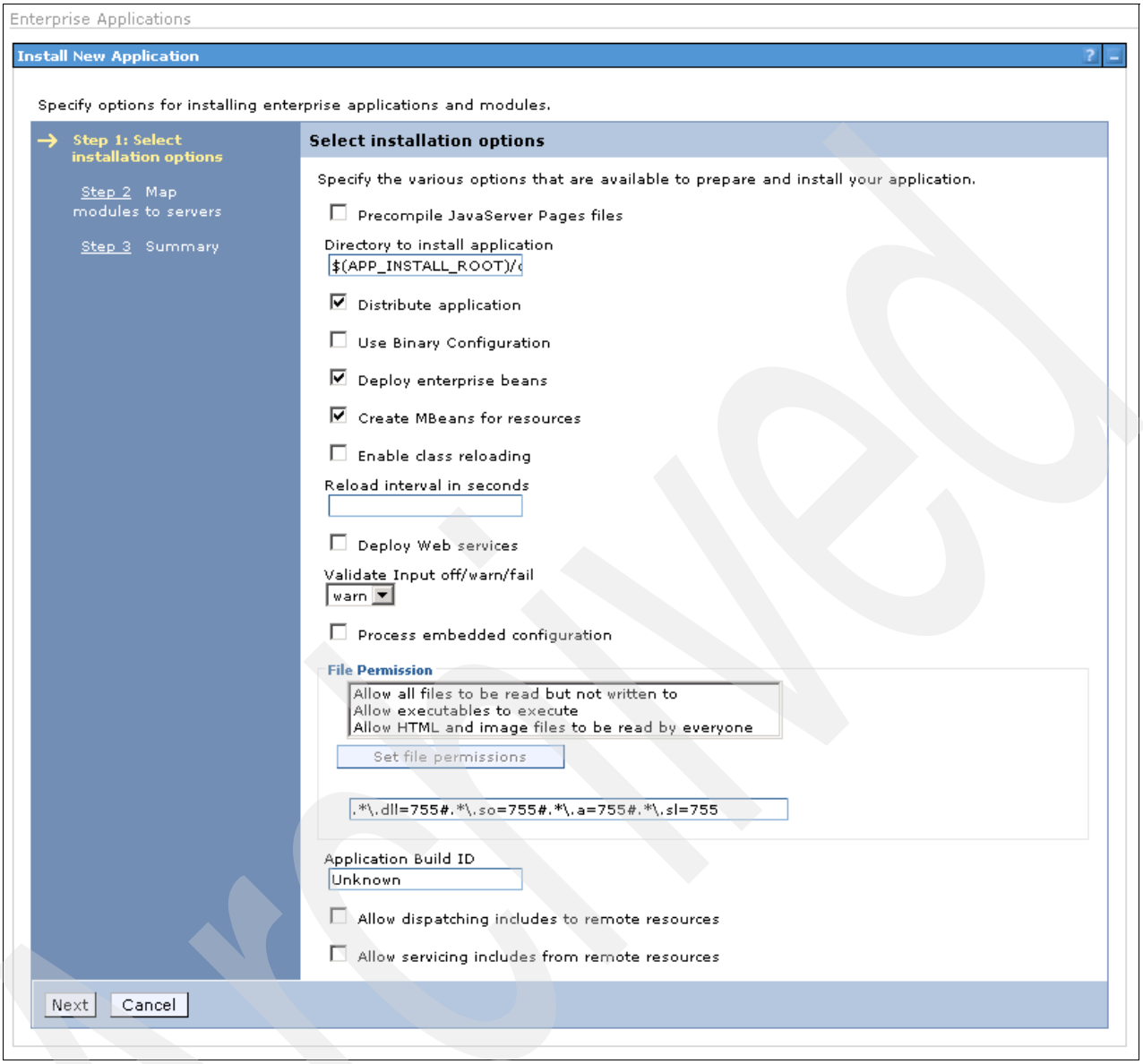


Figure 19-14 elect installation options window

Archived

15. Click **Next** on the Map modules to servers panel (Figure 19-15).

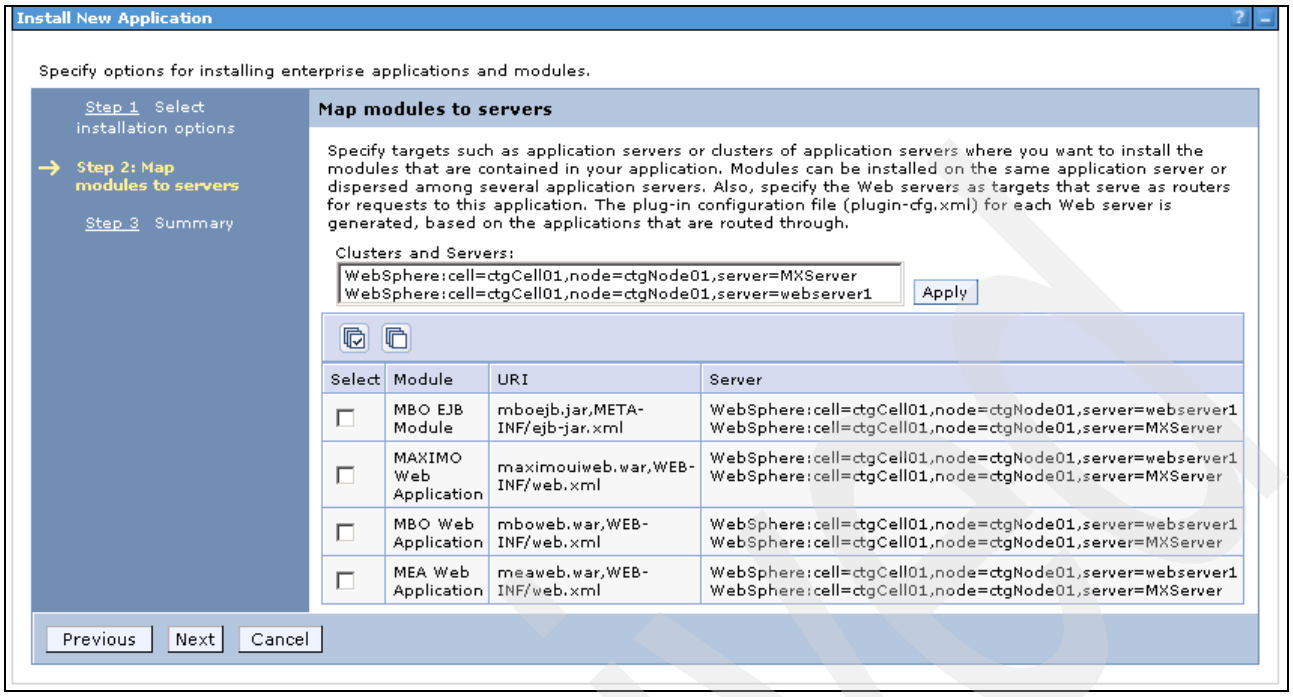


Figure 19-15 Map modules to servers window

Archive

16. Click **Finish** on the Summary page to start the redeployment of `maximo.ear` file. This process can take several minutes.

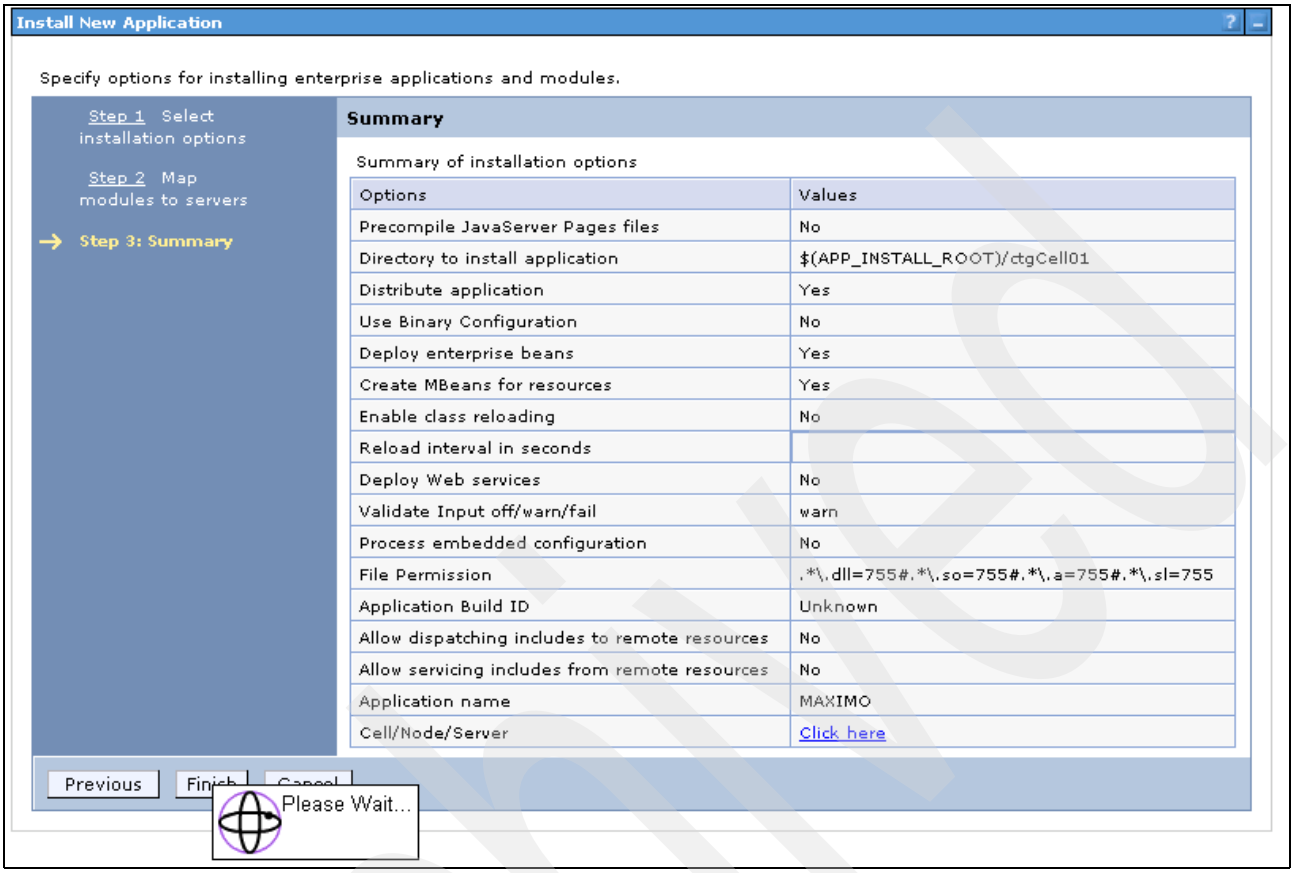


Figure 19-16 Summary page

Archived

17. Click **Save to Master Configuration**. See Figure 19-17.

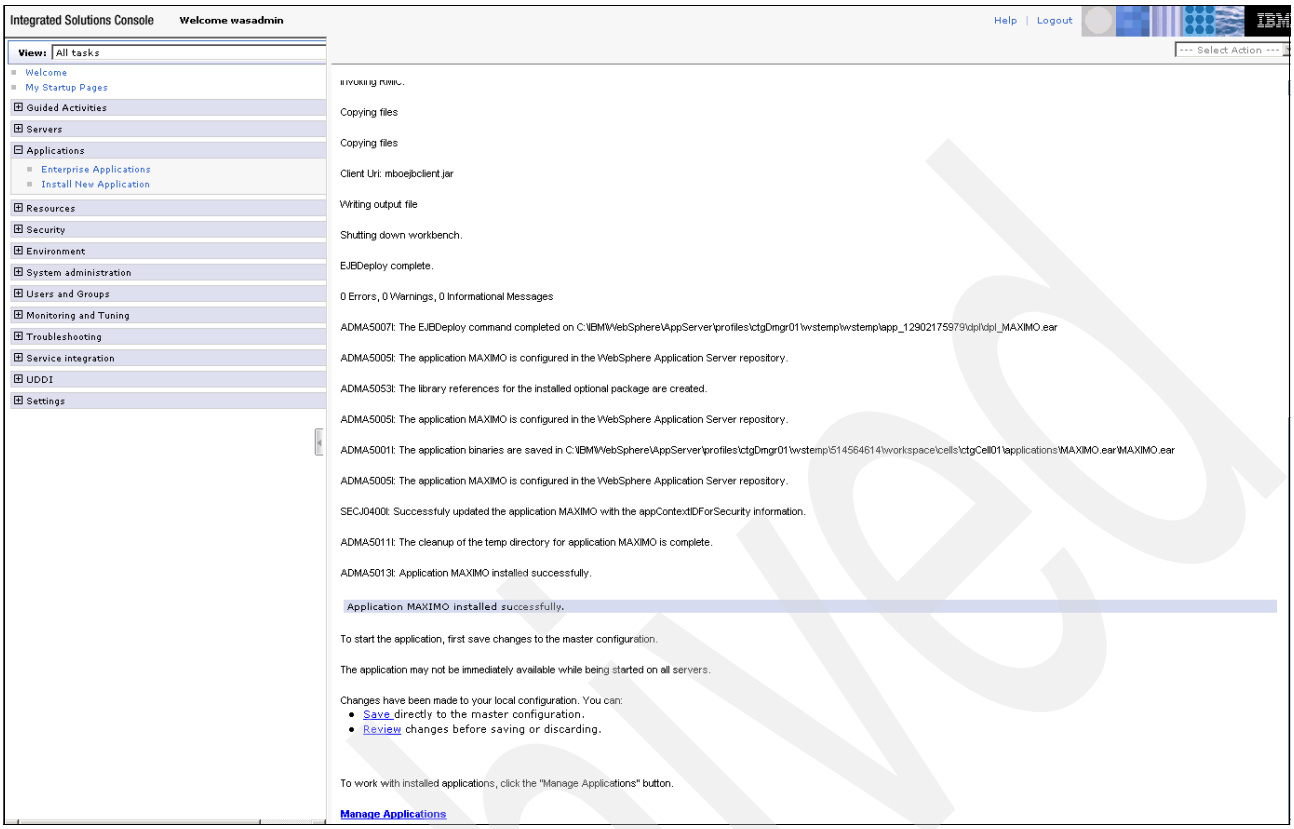


Figure 19-17 Save to Master Configuration

18. Click **Applications** in the navigation pane.
19. Select **Enterprise Applications**.
20. Select the check box next to **MAXIMO**.
21. Click **Start**. Allow this process to complete.
22. Log off of the WebSphere Administrative Console.

Configure Tivoli Identity Manager Server

Complete the following steps to configure the WebSphere server on Tivoli Identity Manager Server:

1. Copy the `maximo.jar` file from '`tim_51`' directory where the `tim_sd_integration.zip` file was extracted.
2. Go to the `$ITIM_HOME\lib` directory on the Tivoli Identity Manager server.
3. Paste the `maximo.jar` file to the `$ITIM_HOME\lib` directory.
4. For cluster environments, paste the file to the `$ITIM_HOME\lib` directory on each cluster member.
5. Log in to the WebSphere Administrative Console for the Tivoli Identity Manager installation.
6. Click **Environment** → **Shared Libraries** → **ITIM_LIB**.

Archived

7. Modify the Classpath by adding the following file:

   `${ITIM_HOME}/lib/maximo.jar`

8. Click **OK**.

The Classpath entry is shown in Figure 19-18.

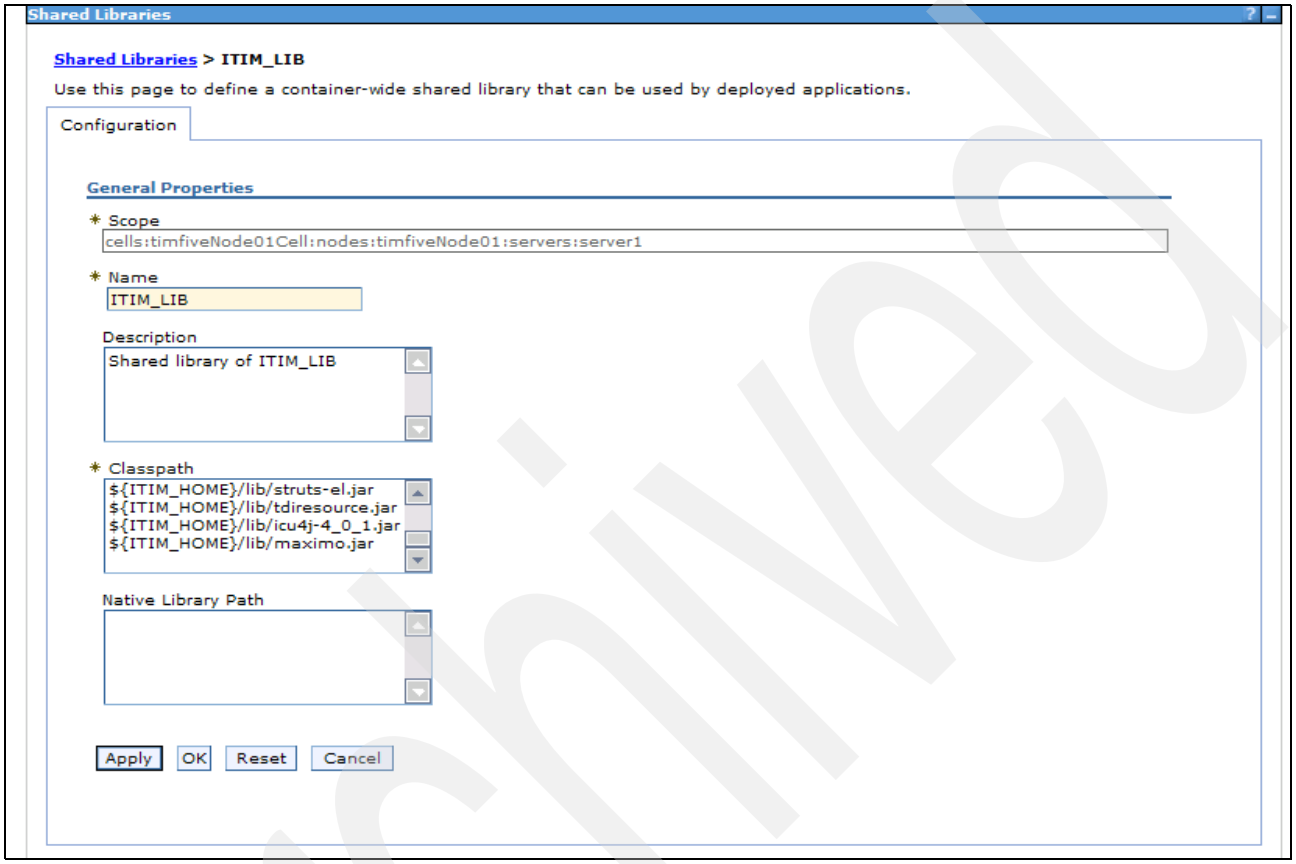


Figure 19-18 Classpath entry

Configuring Tivoli Identity Manager 5.1

This section has details for configuring Tivoli Identity Manager 5.1.

Modify enRole.properties file

Modify the `enRole.properties` file as follows:

1. Navigate to `ITIM_HOME\data` directory.
2. Edit the `enRole.properties` file as follows (see Example 19-2 on page 612):
   a. Replace hostname and port with the values that correspond to the Tivoli Service Request Manager environment.
   b. Set the `maximo.security` value to either true or false depending on whether the Application Server Security is enabled.
   c. If the `maximo.security` value is set to true, the `maximo.user` and `maximo.password` fields are required to enable Service Request creation for Tivoli Service Request Manager when Application Server Security is enabled.
   d. For cluster environments, modify the file on each cluster member.

Archived

Example 19-2 The enRole.properties file

```
#####################################
## Maximo Workflow Extension Properties
#####################################
maximo.url=http://hostname:port
maximo.security=true
maximo.user=maxadmin
maximo.password=maxadmin
#####################################
```

e. Save `enRole.properties` and close it.

Modify scriptframework.properties file

Modify the `scriptframework.properties` file as follows:

1. Navigate to the ITIM_HOME \data directory.
2. Edit `scriptframework.properties` file by adding the line as shown in Example 19-3 under the workflow extensions section.

Example 19-3 Add this line to the scriptframework.properties file

```
#########################################################
ITIM.extension.Workflow.Maximo=com.ibm.itim.maximo.MaximoExtension
#########################################################
```

3. Save the `scriptframework.properties` file and close it.

Restart WebSphere Application Server

After modifying the properties file, WebSphere Application Server must be restarted for the settings to take effect. For cluster environments, restart all the application cluster members.

Configure changePasssword Workflow Extension

The following steps are required in order for the changePassword extension to function.

1. Log in to Tivoli Identity Manager as an Administrator.
2. Click **Configure System**.
3. Click **Manage Operations** under Configure System.
4. Click the **Entity type level**.
5. Click the **changePassword** link.
6. Double-click the **CHANGEPASSWORD** extension box.
7. Click the **Postscript** tab and add the text that is shown in Example 19-4.

Example 19-4 Text to be added

```
#################################
 Maximo.addTicket(Entity.get(), activity);
#################################
```

See Figure 19-19 on page 613.

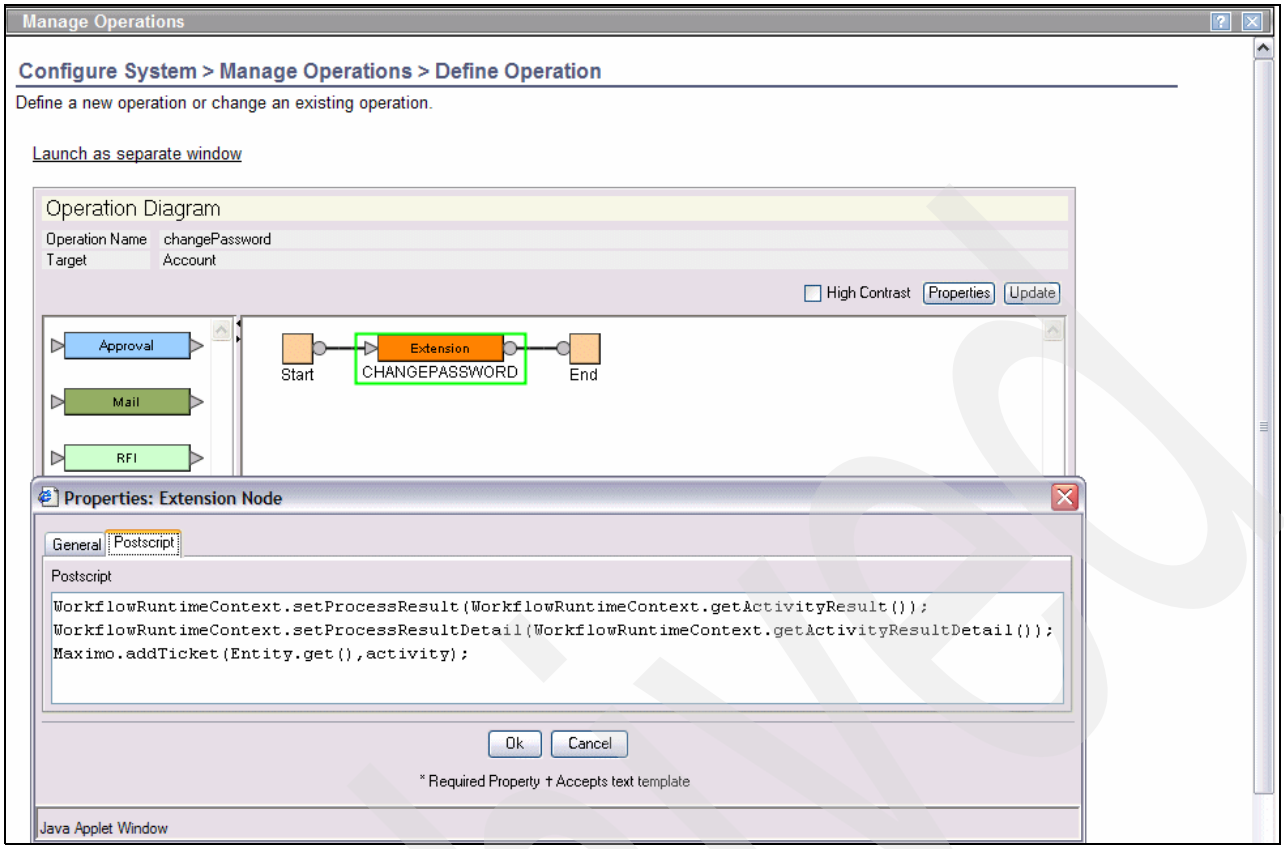


Figure 19-19 Define Operation

8. Click **OK**.
9. Click **Apply**.
10. Click **OK** to verify the changes.

Configure the Tivoli Service Request Manager Service Provider

Perform the following steps to enable support for Tivoli Service Request Manager user management:

1. Copy the `maximoserviceprofile.jar` file from the 'tim_51' directory where the `tim_sd_integration.zip` file was extracted.
2. Paste this `.jar` file to a directory on a machine with a web browser that can log in to Tivoli Identity Manager. Perform the remaining steps from that machine:
   a. Log in to Tivoli Identity Manager using the Console interface.
   b. Click **Manage Service Types**.
   c. Click **Import**.
   d. Click **Browse** and navigate to the directory containing `maximoserviceprofile.jar`.
   e. Select `maximoserviceprofile.jar`. See Figure 19-20 on page 614.

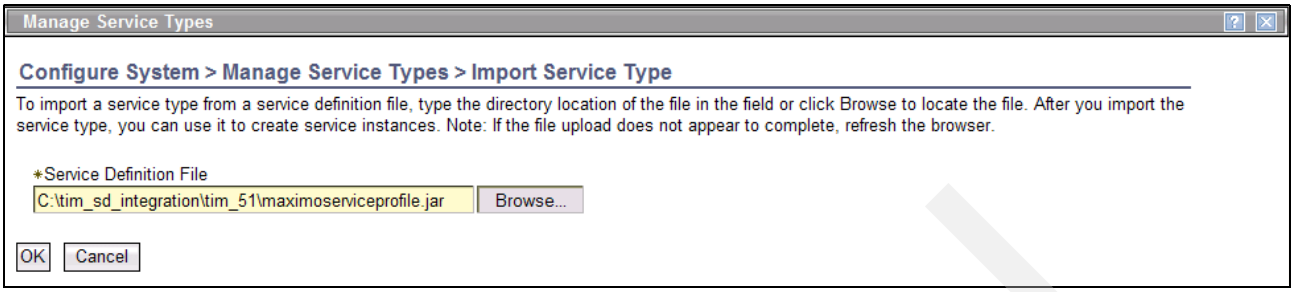


Figure 19-20 maximoserviceprofile.jar

f. Click **OK** and allow a few minutes for the operation to complete.

g. Click **Manage Services**.

h. Click **Create**, and select **Maximo Service** from the menu, and then click **Next**.

i. Type a unique service name and supply the Tivoli Service Request Manager URL in either of the following forms, depending on whether SSL is being used on the Tivoli Service Request Manager server:

```
http://hostname:port
https://hostname:port
```

j. Enter a user ID and password if you want to execute the operations as a specific user other than the default user MAXADMIN. Leaving these fields blank results in the operations being executed as MAXADMIN. See Figure 19-21.

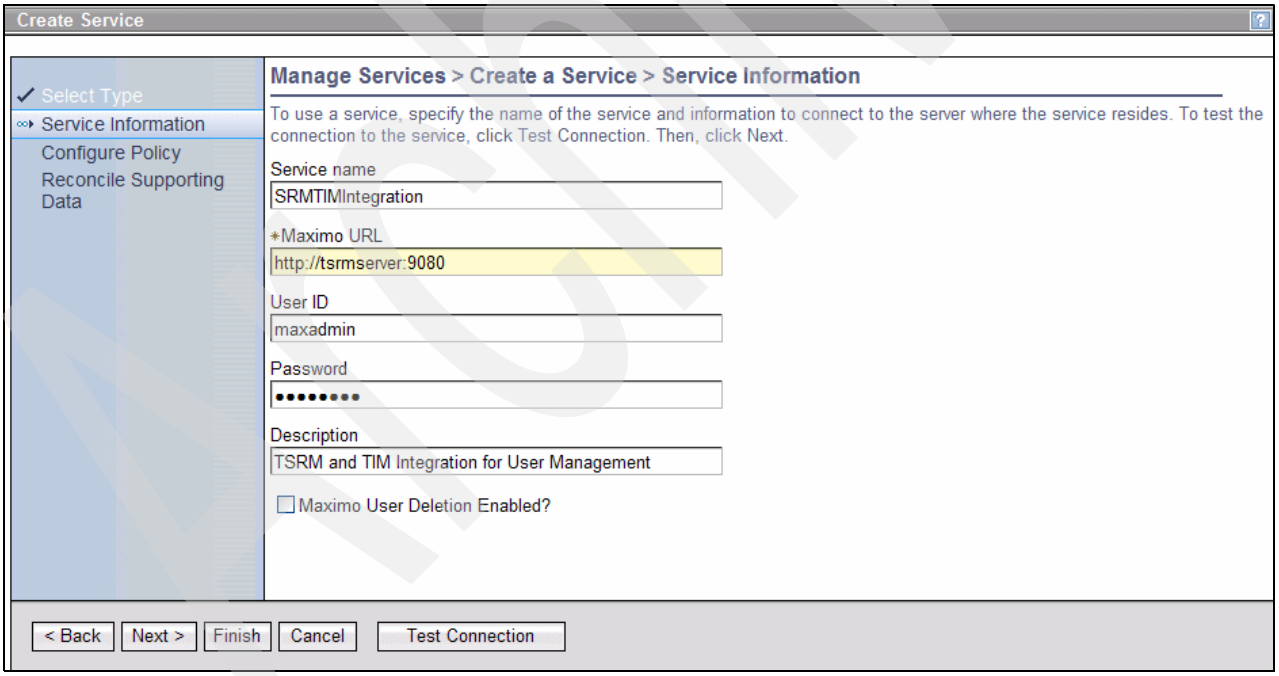


Figure 19-21 Create Service window

k. Select the Maximo User Deletion Enabled check box if the Enable Login Tracking was not selected (as shown in Figure 19-18 on page 611) and you want to delete Tivoli Service Request Manager users. The LOGINTRACKING parameter value is set by selecting the Enable Login Tracking check box.

l. Click **Test Connection**, verify that the test was successful, and then click **Finish**.

19.5 Password change scenario

Whenever a user changes the password in Tivoli Identity Manager, a ticket of service request type is created in Tivoli Service Request Manager for each password change action. The ticket is closed automatically if the password change operation is successful. If the password change operation fails, the ticket remains in a “NEW” state. The steps in this scenario are as follows:

1. Consider a user named Chuck Reigle. Log in as Chuck Reigle, or as an authorized user, to the Tivoli Identity Manager Self Service Console in the following URL format (see Figure 19-22):

   ```
   http://itimserver:9080/itim/self url
   ```

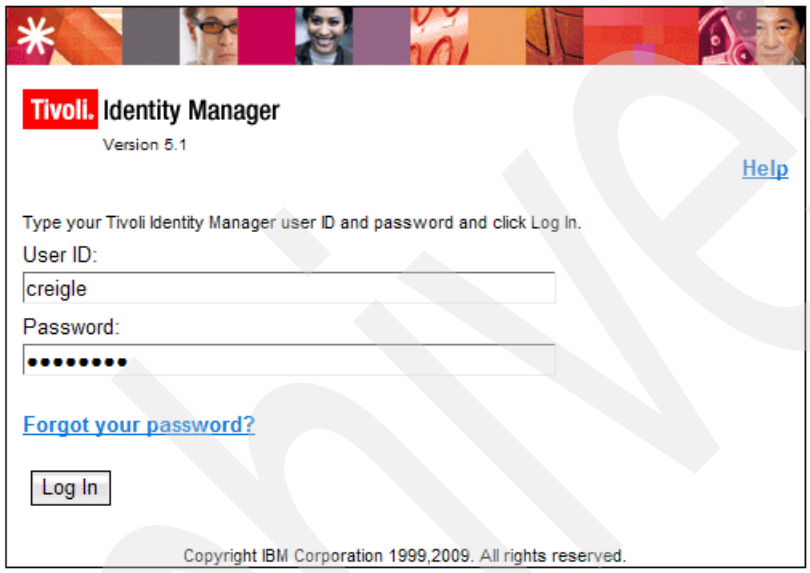


Figure 19-22 Tivoli Identity Manager Self Service Console

Important: Another approach in many organizations is to use the Tivoli Service Request Manager Self Service Portal (instead of the Tivoli Identity Manager Self Service Console) for these types of self-service requests, such as a password change. If that is the case in your organization, use Tivoli Service Request Manager Self Service Portal for this step

2. After logging in to the Self Service Console, click the **Change Password** option, as shown in Figure 19-23 on page 616.

Archived

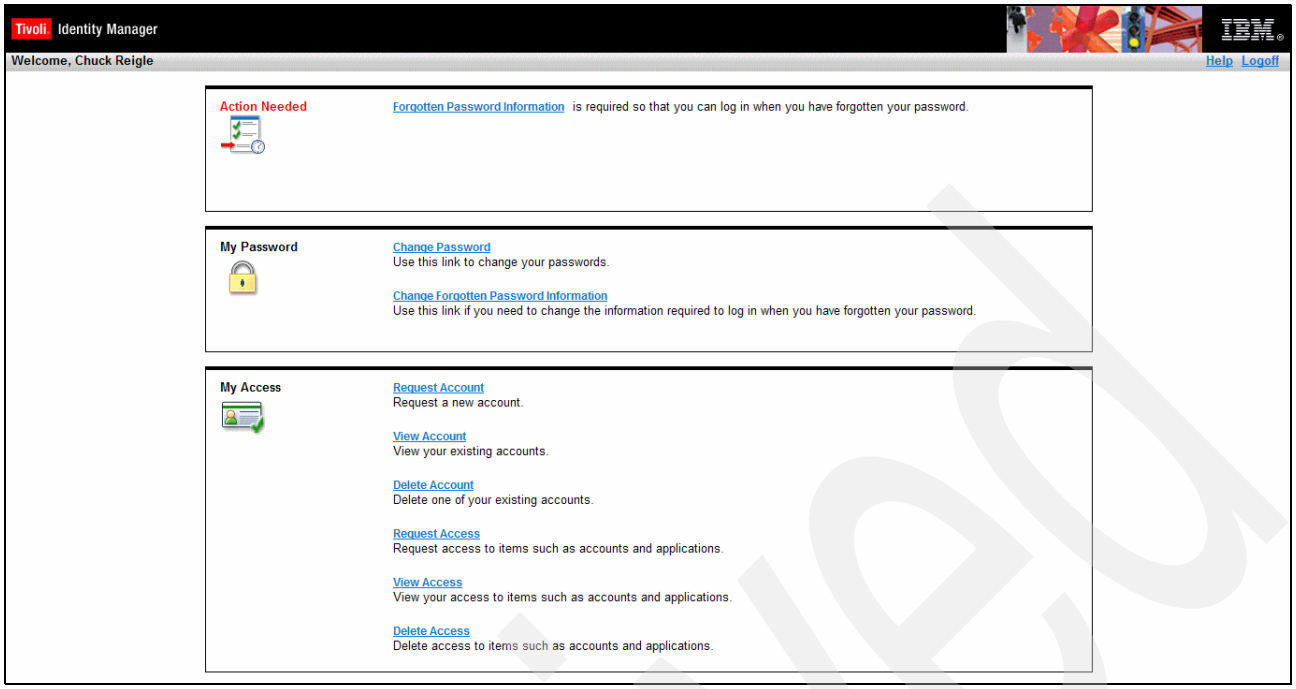


Figure 19-23 Change Password option

3. In the Change Password window, information about the accounts that would be affected by this change and the entries for existing and new password are presented. Enter the information about the existing password and the New password as shown in Figure 19-24 on page 617 and clicks **OK**.

Archived

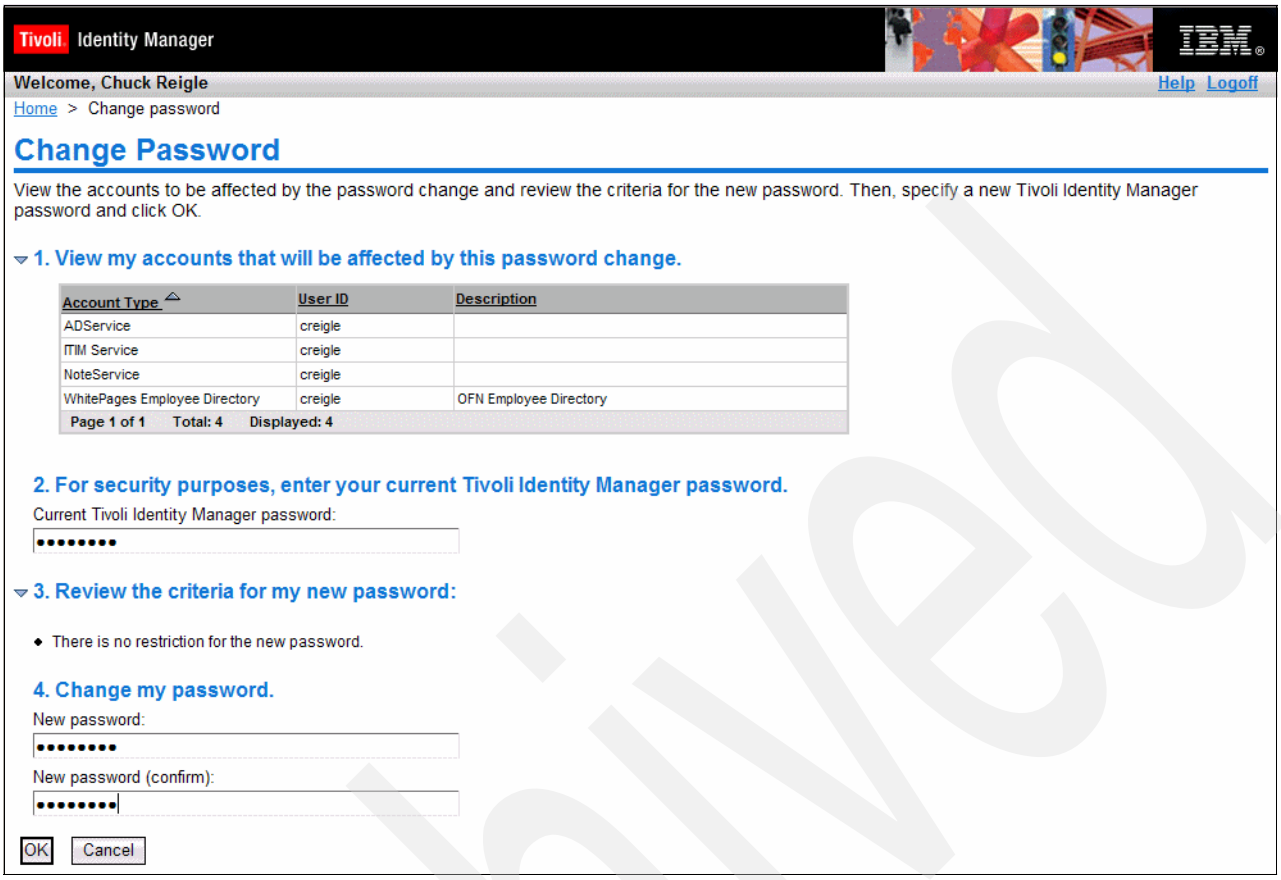


Figure 19-24 Change Password window

On clicking **OK**, the Request is submitted to the system. See Figure 19-25.

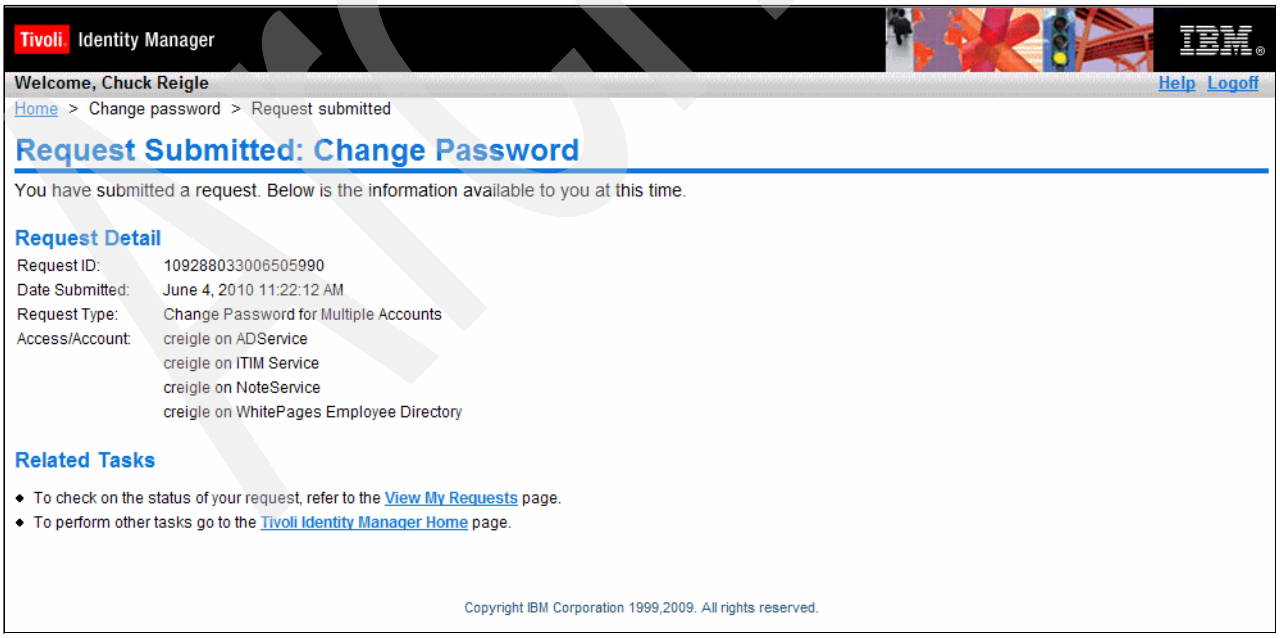


Figure 19-25 Request Submitted

4. To view the status of the request, click **View My Requests**, which opens the View My Requests page as shown in Figure 19-26.

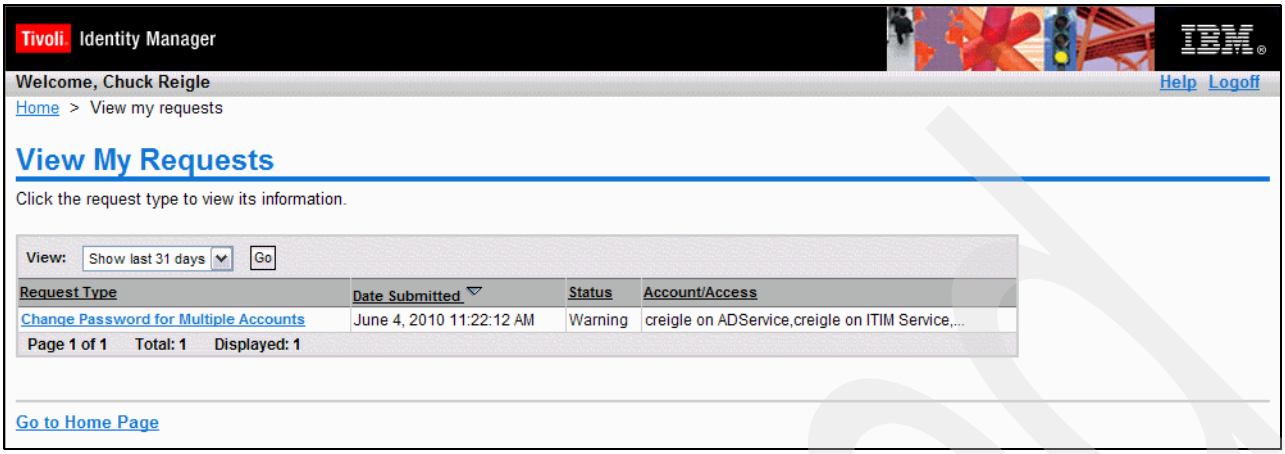


Figure 19-26 View My Requests

The Change Password operation is triggered in the background when the request was originally submitted, creating a ticket of service request type in Tivoli Service Request Manager.

5. By logging in to the Tivoli Service Request Manager and clicking **Go To** → **Service Desk** → **Service Requests**, you can view this ticket. Filter the request based on password as Summary and CREIGLE as Reported By. See Figure 19-27.

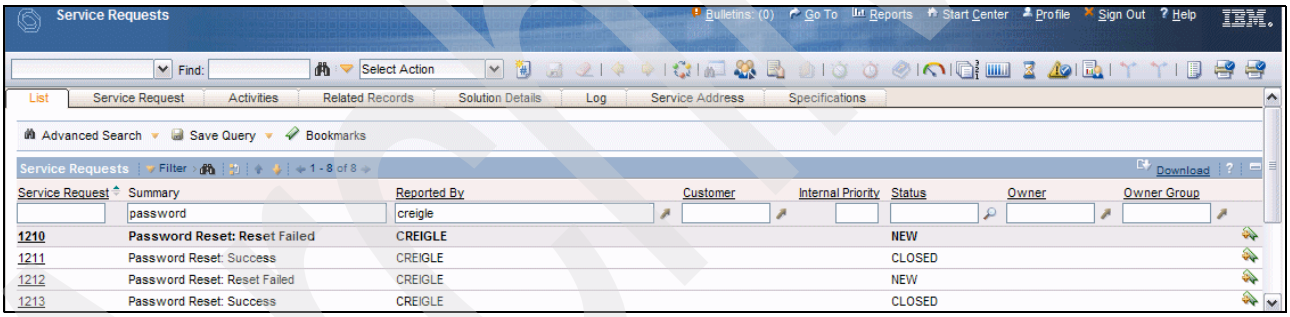


Figure 19-27 Service Requests window

As you can see, of the four requests (because user Chuck owns four accounts, four requests are created), two were successful. The service requests for successful changes were created and closed automatically. Two requests for password change had failed and the service request corresponding to this remains in NEW status. This service request can now be assigned to the right people who will troubleshoot the issue and find resolution to close it.

19.6 User management scenario

Tivoli Identity Manager can be used to manage the Tivoli Service Request Manager users. The users can be managed through Tivoli Service Request Manager Service Provider if the Tivoli Service Request Manager native registry is being used as the primary user repository. If Tivoli Service Request Manager is integrated with an LDAP system, such as Active Directory, you can use a Tivoli Identity Manager adapter, such as Active Directory adapter, to manage the users.

We again consider user Chuck who wants to log in to Tivoli Service Request Manager to raise a service request. For this step, Chuck must have an account created in Tivoli Service Request Manager. To create an account, Chuck uses the Tivoli Identity Manager system as follows:

1. Log in to the Tivoli Identity Manager Self Service console as Chuck Reigle (or any authorized user) and click Request Account. See Figure 19-28.

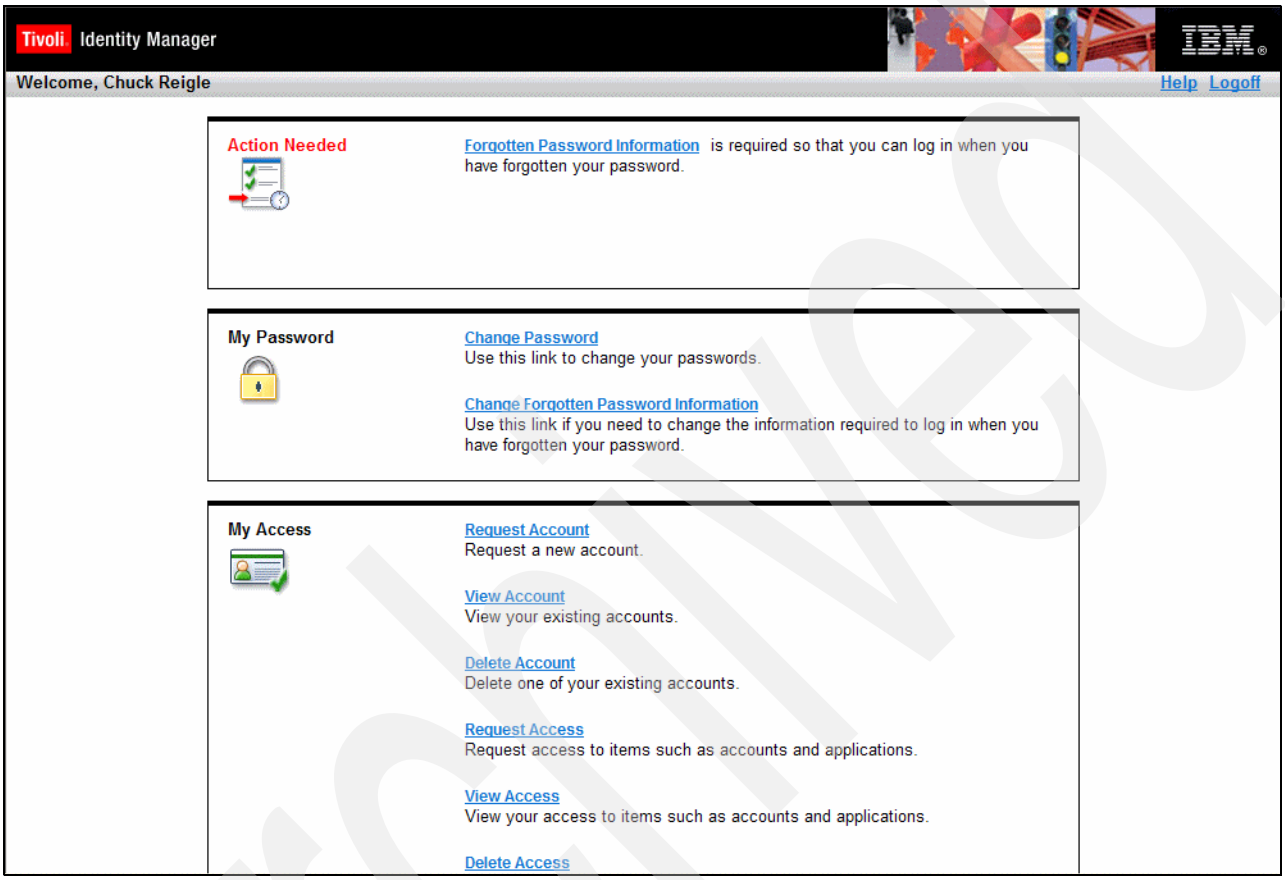


Figure 19-28 Tivoli Identity Manager Self Service console

This step opens the Request Account page as shown in Figure 19-29 on page 620.

Archive

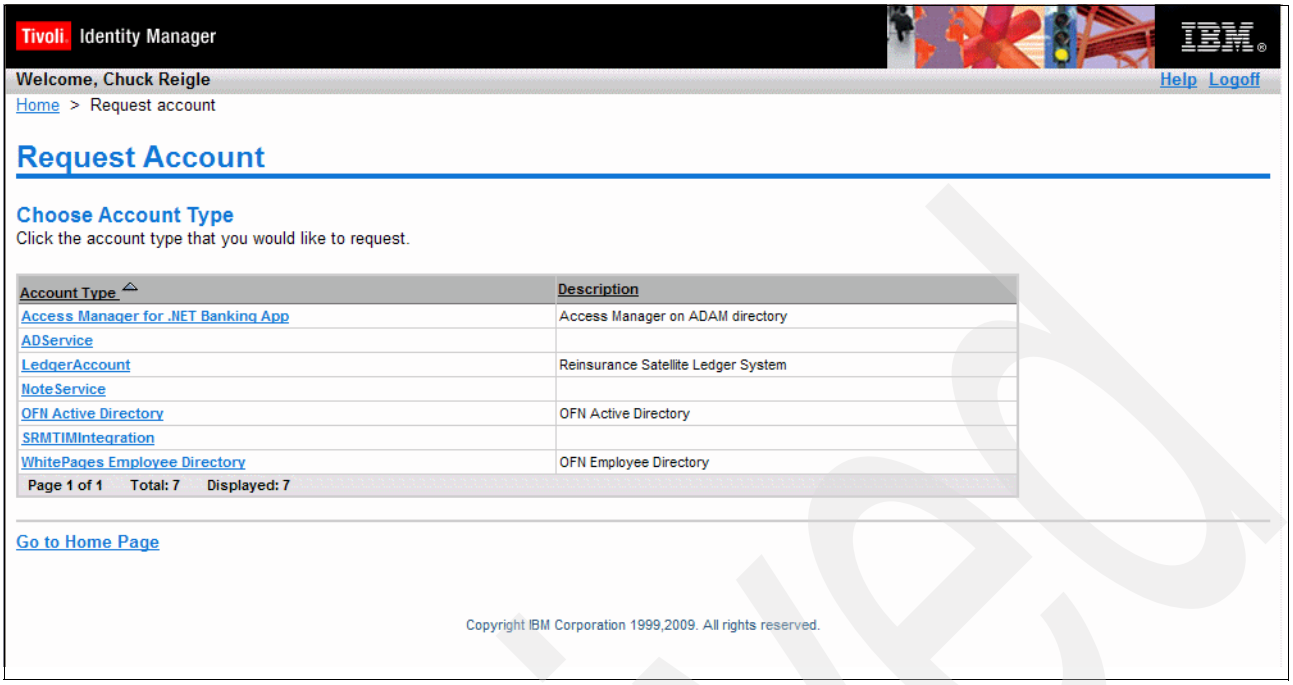


Figure 19-29 Request Account page

2. Click **SRMTIMIntegration** service that was created in the earlier step. The Account Information page opens, as shown in Figure 19-30.



Figure 19-30 SRMTIMIntegration service t

3. Click **Next**, The confirmation page opens. Click **Request Account** to confirm the request. See Figure 19-31 on page 621.

Archived

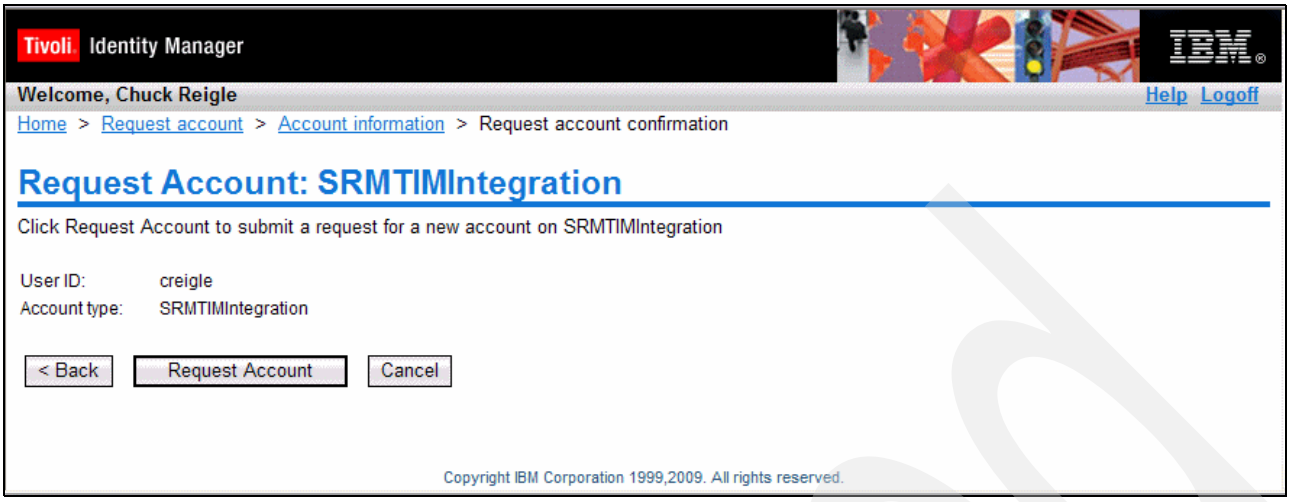


Figure 19-31 Request Account window

4. In the Request Submitted page, click **View My Requests** to check the status of your request. See Figure 19-32.

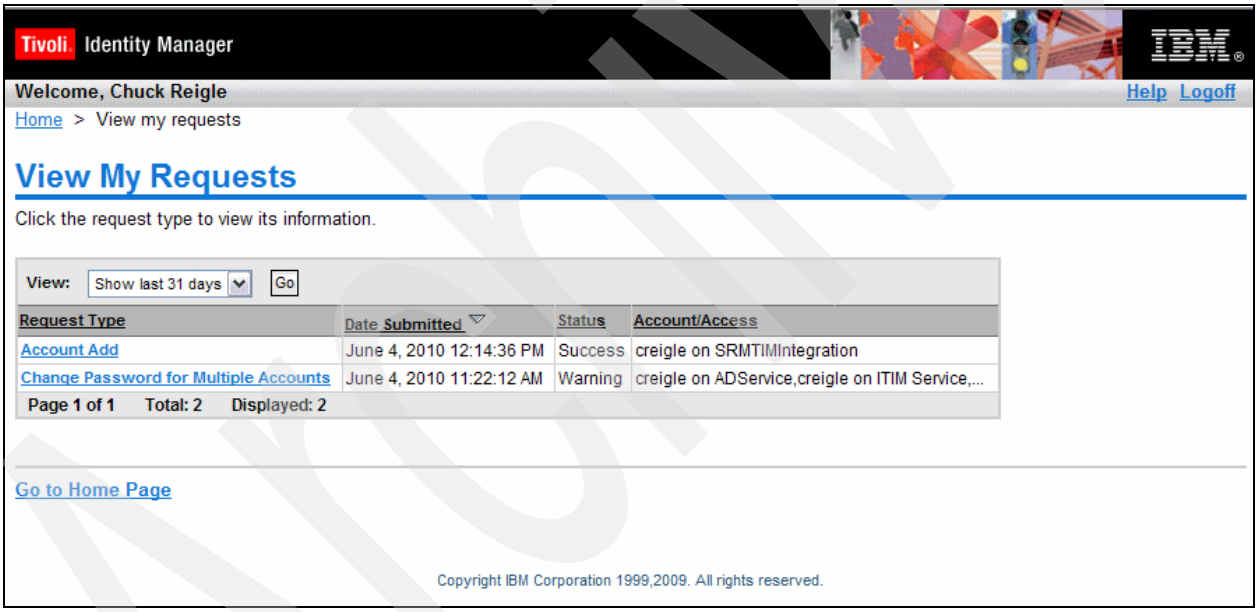


Figure 19-32 View My Requests

The Account Add operation has completed successfully. To login to the Tivoli Service Request Manager, go to the console at the following URL:

`http://tsrmserver:9080/maximo`

To reset the password click **Forgot your password?** link on the console. The Tivoli Identity Manager Self Service Console opens. (We had previously configured **Adding Password Link to Tivoli Service Request Manager**, which enables you to go to the self service console directly.) Complete the password change operation. See 19.5, “Password change scenario” on page 615 for more information.

After the password is reset to the Tivoli Service Request Manager console, log in by using the new password.

19.7 Summary

Figure 19-33 shows the summary for this integration.

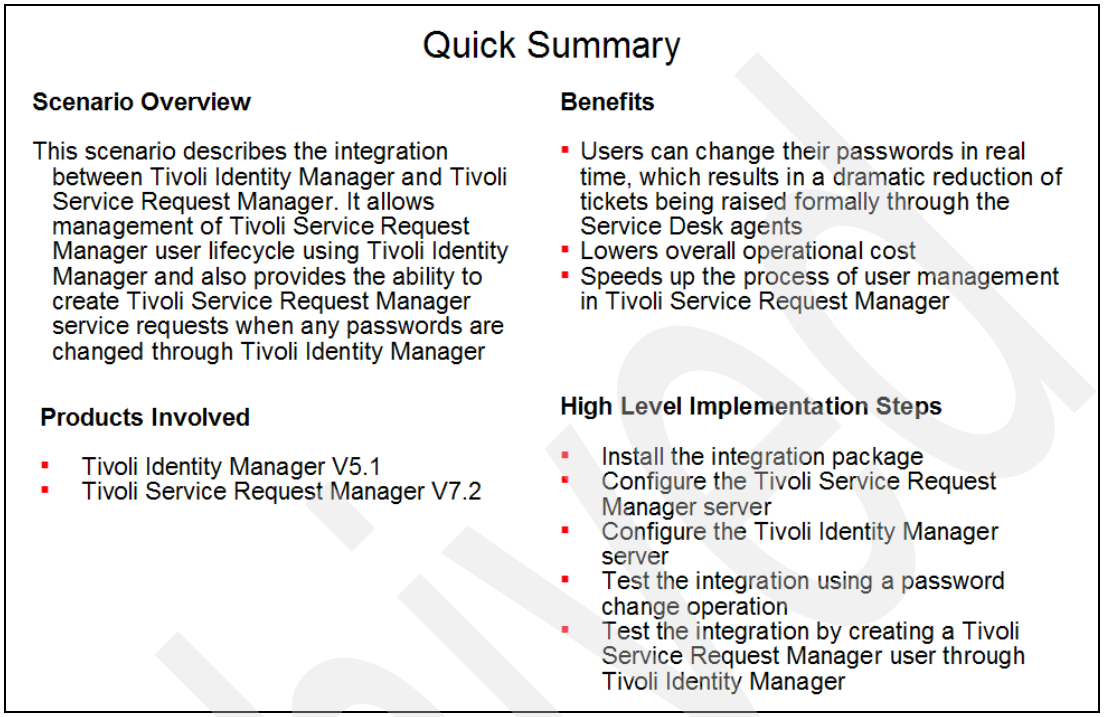


Figure 19-33 Quick summary

Archived

20

Employee onboarding

This chapter describes how Tivoli products can help during the process of receiving a new employee in a company, and separation of an employee from a company. It demonstrates how main functionalities of each product can fulfill services such as security, provisioning, asset management, and self-service in a smooth way.

A view of roles and tasks is also presented and can be used as a model for real implementations in the field.

This chapter does not cover configuration of the products involved. All integrations are already validated and are going to be used as references when necessary.

This chapter contains the following topics:

- 20.1, “Scenario overview” on page 624
- 20.2, “Products involved” on page 624
- 20.3, “Benefits” on page 624
- 20.4, “Roles” on page 625
- 20.5, “Architectural diagram of integration” on page 626
- 20.6, “Implementation steps” on page 627
- 20.7, “Scenario walk-through” on page 639
- 20.8, “Summary” on page 640

© Copyright IBM Corp. 2011. All rights reserved.

20.1 Scenario overview

The Employee Onboarding scenario is a look at a possible process to improve the experience of many companies around the world. In this chapter, we present an integration scenario, where we use several integrated Tivoli products to accelerate the process of receiving a new employee in a company and minimize the errors by automating many of the task in the Employee Onboarding process, reducing any appearance of disorganization in the onboarding experience of the employee.

20.2 Products involved

The following products and product components are involved in this scenario:

- Tivoli Service Request Manager V7.2
  - Service Catalog
  - Service Desk
- Tivoli Asset Management for IT V7.2
- IBM Maximo Asset Management V7.1
- Tivoli Identity Manager V5.1

20.3 Benefits

The potential benefits of this scenario are described in this section.

20.3.1 Paperless activities

Because this scenario is a strict integration between systems and UI interactions, it helps companies by reducing paper usage for requisitions. Tivoli Process Automation Engine products provide processes and tools to automate manual interactions and use web-based or email-based channels for notifications.

20.3.2 Speed up productivity

Given the strong integration between product and process-based activities, everything that is required to help a new employee be ready to start working can be set up quickly and automatically. The new employee can start actual work much sooner and can be productive faster in the new job. Also, the processes related to separation of an employee from the company can be completed more quickly. In this latter case, you have a limited window to take care of the separation tasks to do them properly.

20.3.3 Pattern for new hires

After building the workflows to hire specific roles, those workflows can reused anytime they are required for other roles or positions, such as temporary workers, and so on.

Archived

20.3.4 Increased sense of competence

The new employees, seeing that everything was set up prior to their start dates, can feel more secure, and consider themselves as part of a competent organization.

20.3.5 Integrated platform

Using products that are part of a unique portfolio, you expect to have a solution that contains full management of processes, tooling, automation and people.

These products also use a common infrastructure that allows better data sharing and consequently more integration between processes. This approach can lower the cost of ownership, training, and support during the lifetime of the products.

20.4 Roles

Figure 20-1 shows the roles used in this scenario.

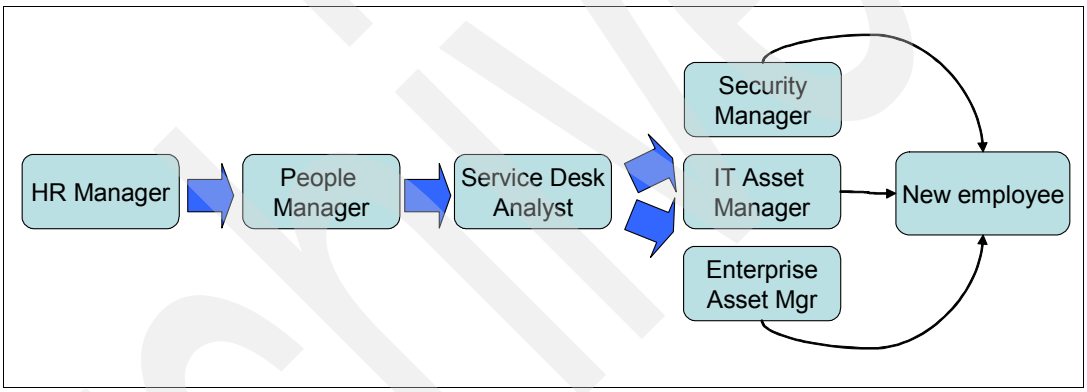


Figure 20-1 Roles

The following roles were identified during the development of this scenario and are used throughout the chapter. Notice that those roles might be modified in the field according the structure of the companies where this integration is implemented.

- *New employee* is person who is just hired and needs the basic work material to initiate work activities. This role also uses self-service applications to check status requests and execute security tasks such as password changes and resets.
- *HR Manager* is responsible to create the new employee record on the Human Resources (HR) system and inform the department (People Manager) about the employee ID just created.
- *People Manager* selects what is needed for the new employee and submits orders that fulfill those needs. This role can also help define the Service Catalog, to make sure that it reflects the company requirements.
- *IT Asset Manager* is engaged in the delivery of the IT requests made for the People Manager. This role plans and orders the execution of the activities that are generated by the team. Resources, inventory, and infrastructure are a subset of the manageable items for which this role is responsible.
- *Enterprise Asset Manager* is similar to IT Asset Manager. This role is responsible for managing non-IT assets such as facilities or other industry-specific assets.

Archived

- *Security Manager* is responsible for managing credentials requests and addressing them according the enterprise security rules. This role works closely with IT Asset Manager. Sometimes the roles can be combined into one position.
- *Service Desk Analyst* receives, filters, and distributes requests across the company. Certain job tasks can be automated to reduce the responsibilities of this role.

Additional roles can be added (or removed) from this scenario. Some of those can be combined or shared to achieve the result that enables the new employee to be productive more quickly.

20.5 Architectural diagram of integration

Figure 20-2 shows how each system interacts and the sequence of the steps taken, from the HR System request to the IT/Enterprise Asset Management tools delivery.

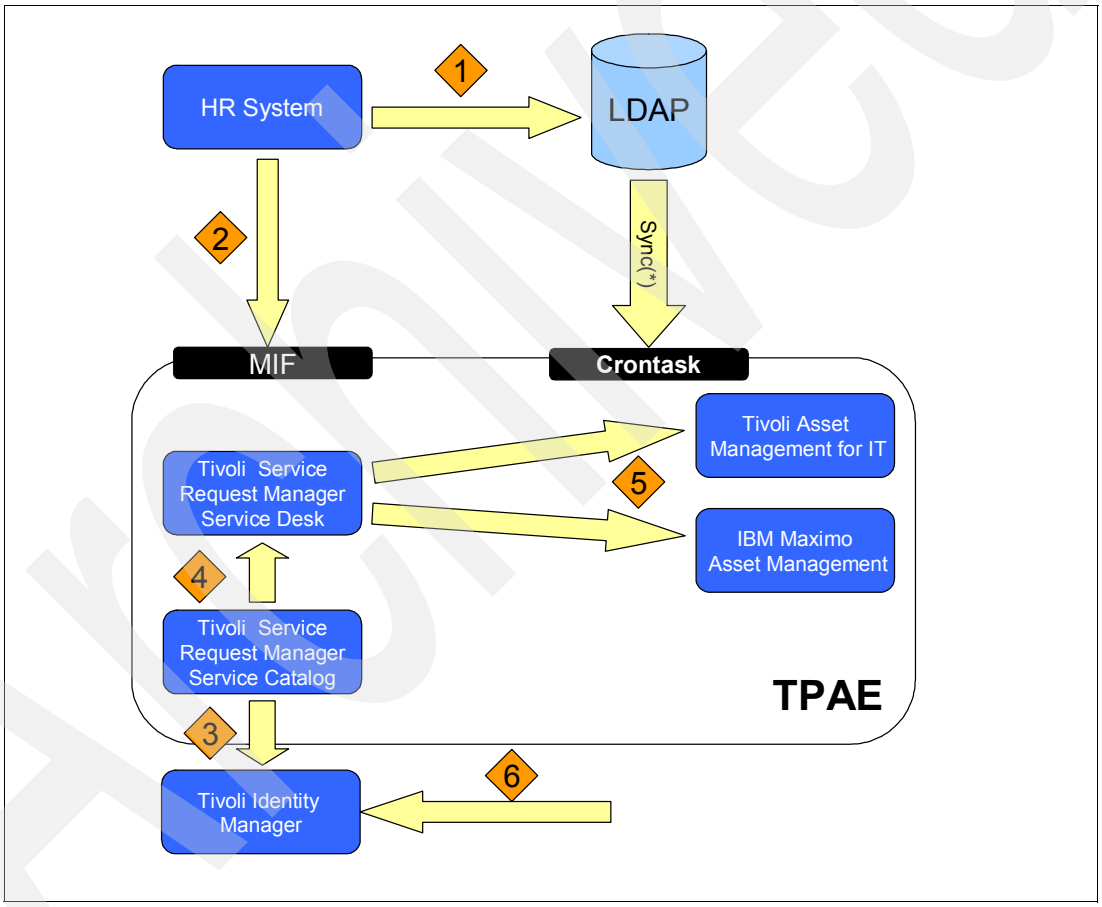


Figure 20-2 Architectural diagram of integration

Interactions indicated in the figure are as follows:

1. The HR System feeds the LDAP (LDIF format or through LDAP API) to ensure Authorization data (new employee ID) is created on central repository.
2. The HR System exports data (employee record) that is imported by the Tivoli Process Automation Engine through Maximo Integration Framework (person object).

3. Tivoli Service Request Manager Service Catalog pushes orders directly to Tivoli Identity Manager (Lotus Notes related) through embedded offerings.
4. Tivoli Service Request Manager Service Catalog orders IT (notebook) and Enterprise (mobile phone) Assets. Those requests are forwarded to Tivoli Service Request Manager Service Desk queues and are managed by appropriate analysts.
5. Tivoli Service Request Manager Service Desk analyzes the request, creates a related work order, and assigns it to IT asset manager (notebook) and Enterprise Asset Manager (mobile phone).
6. The user can use Tivoli Identity Manager self-service to check status of security requests while they are not yet completed.

20.6 Implementation steps

In this section, we cover the integration steps. Product installation is not covered.

This section provides a view of how roles (users) participate in the scenario and the how the products support the related actions.

20.6.1 Human Resource (HR) Manager

HR Manager uses the Human Resource System to create a new employee record and generate the Employee ID used throughout other systems. See Figure 20-3.

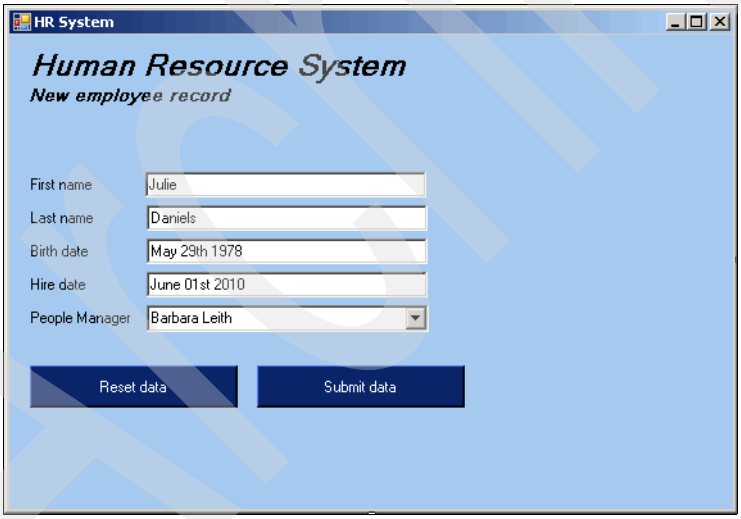


Figure 20-3 HR record creation

Information such as first name, last name, and birth date is used to create the unique ID. For example, Peter Knust, birth date of 12 May 1978, generates an ID such as KnustPe78.

With the new employee record already created, we move the data to an LDAP instance that is responsible to maintain the organization of all the authentication data.

LDAP entries are synchronized with the Tivoli Process Automation Engine database through VMMSync or LDAPSync cron tasks. For now, that is a unidirection synchronization and includes groups also.

Archived

The setup for VMM Mappings for User and Groups can be seen in the Crontask application (**Go To** → **System Configuration** → **Platform Configuration** → **Crontask setup**). Filter by VMMSYNC to get the appropriate entry and then you can see how XML mapping is stored and structured (partial view - Figure 20-4).

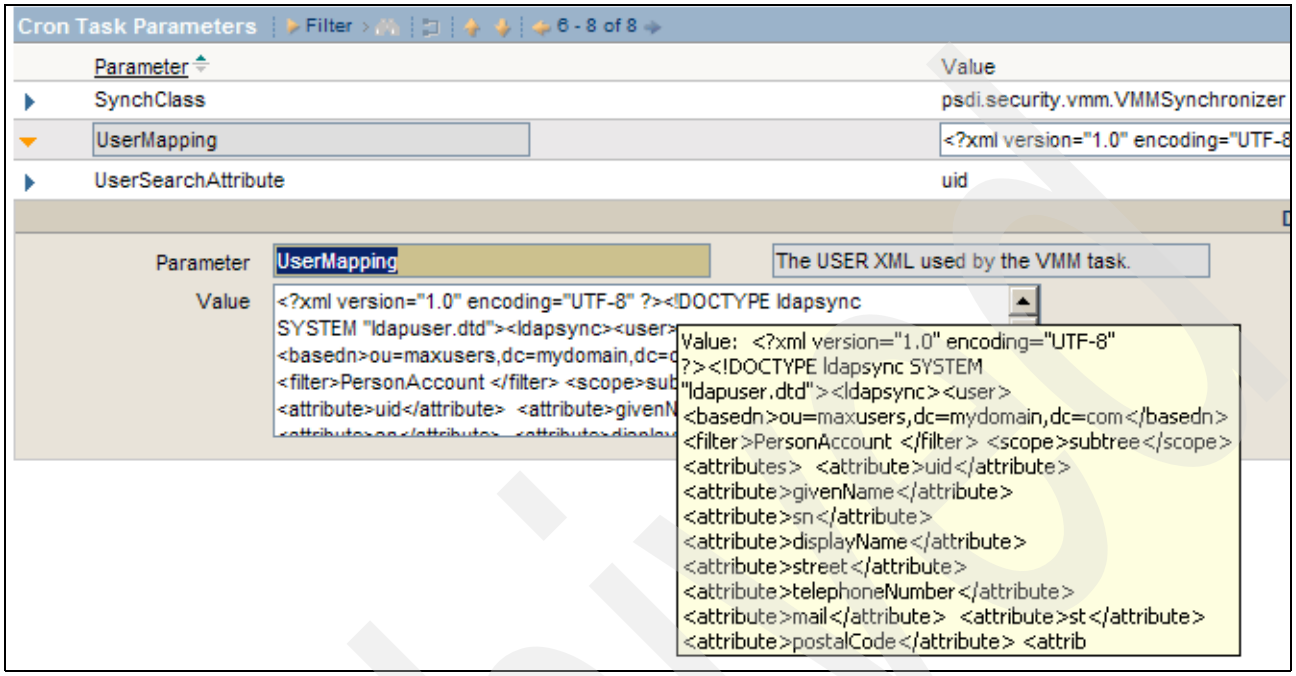


Figure 20-4 Cron Task Parameters

After the synchronization is complete, the new user can now log in to Tivoli Service Request Manager and use the services available, such as Self-service and Service Request applications.

If you are not using LDAP with the products in this scenario: In the case of not using LDAP, our suggestion is to export the HR system data and transform it to a comma-separated value (CSV) file or a Maximo Integration Framework supported XML format. Both can be loaded by Maximo Integration Framework (MIF) functions. In case you need synchronous integration (create user online), the HR system can access Maximo Web Services and request the user creation.

In both situations, the HR system can open and close a ticket in Tivoli Service Request Manager to track the new user creation. This integration can be done through either batch processing (CSV/XML files) or online services such Web Services.

The access level of the new user depends on the groups with which it has been associated in LDAP and what rights were defined for each group in Tivoli Process Automation Engine.

20.6.2 People Manager

This section shows what the People Manager must execute to be able to request a Lotus Notes Account and a notebook for the new employee.

First step is when the People Manager uses Tivoli Service Request Manager Service Catalog to request a new ID.

The People Manager should be part of a security group named PMSCSRU that is installed along with the Service Catalog content. That security group has the ability to grant permission for all applications and actions so that someone can view and request an offering.

To access the Offering Catalog application, the People Manager clicks the link on the Maximo StartCenter (Figure 20-5) or by clicking **Go To** → **Service Request Manager Catalog** → **Offering Catalog**.



Figure 20-5 Offering Catalog link on Startcenter

With the Offering Catalog application opened, the People Manager can select the offering to create the new Lotus Notes account.

Note: Depending on your company policy, you can also automate the creation a Lotus Notes account for the new hire by using a workflow. This activity can be in the workflow after creation of the user ID for the new hire.

Certain parameters are required to complete the request (Figure 20-6 on page 630). After completing the request, the People Manager can submit it or keep it on the cart while more offerings are added to it.

Tip: In addition to the parameters shown in Figure 20-6, a good approach for the People Manager is to identify the roles and responsibilities of the new employee to provision accounts, group memberships, and software provisioning needs.

Submitting the offering generates a new service request record with all the information registered during the Service Catalog process. This new record will be the entry point for the work execution and related processes, such as approve, purchase, install, and assign.

Archive

Figure 20-6 shows that the service request has been created.

.

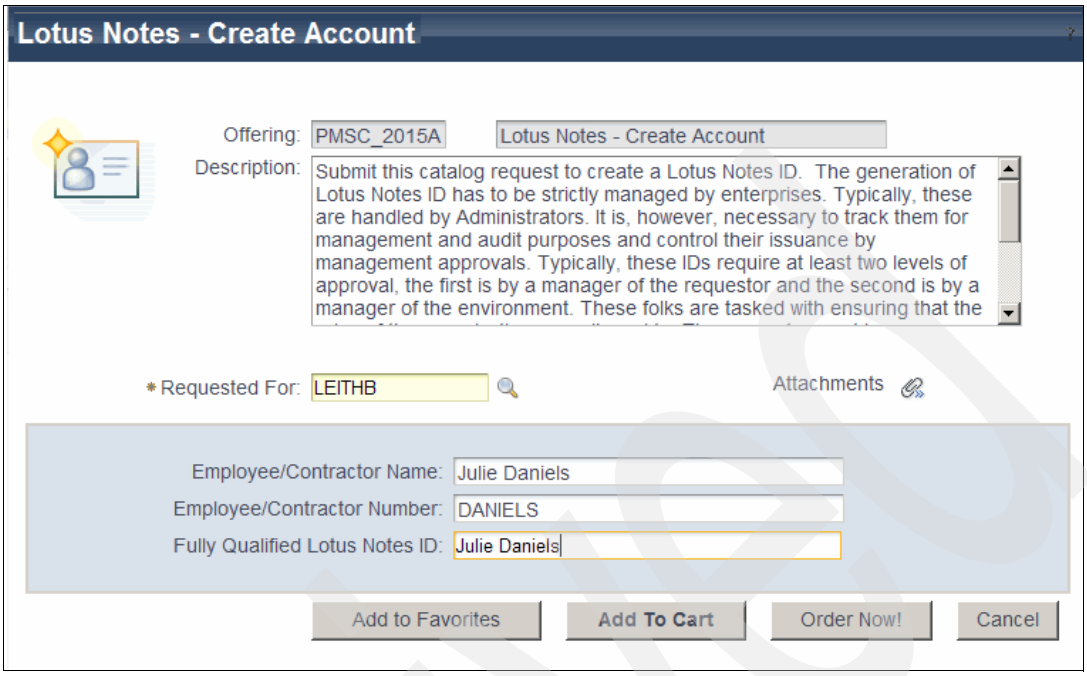


Figure 20-6 Create Account window

The service request created is assigned to the IT Manager and will be analyzed and resolved by that team.

Figure 20-7 shows the order submitted.

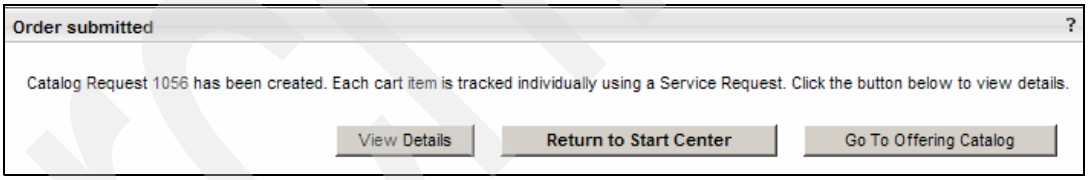


Figure 20-7 Order submitted

Additionally, Service Catalog and Tivoli Identity Manager have strategic integrations that allow Catalog Offerings to launch the Lotus Notes operations directly. Automating these processes reduces the time to deliver the following functions, after these processes are executed by the Tivoli Identity Manager API (invoked by Tivoli Service Request Manager).

See Figure 20-8 on page 631:

- Change password: Request to change a Lotus Notes password using Tivoli Identity Manager.
- Reset password: Request to reset a Lotus Notes password using Tivoli Identity Manager.

Archive

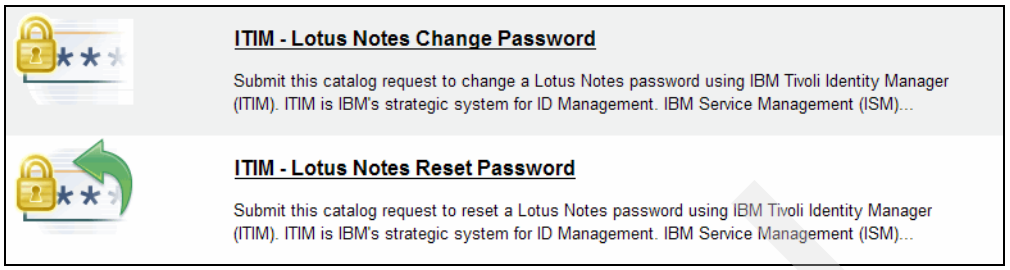


Figure 20-8 Change and reset password

For more details about this integration, see Chapter 19, “User and password management with Tivoli Identity Manager and Tivoli Service Request Manager” on page 595.

To complement the basic set of needs for the new employee to start to work, the People Manager role must request a notebook (with sales software installed) and a mobile phone. Two offerings were created specifically for this scenario (Figure 20-9). These are not available as part of the default product content.

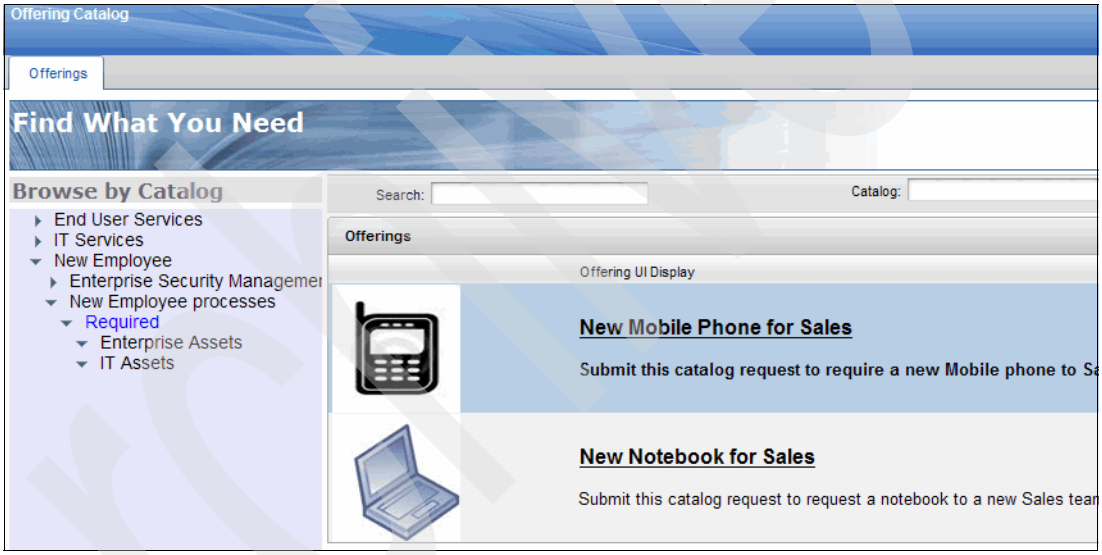


Figure 20-9 Requesting a notebook

When a People Manager selects the notebook offering, the person also must select the User ID that is created by the HR system. All assignments are performed based on that ID. See Figure 20-10 on page 632.

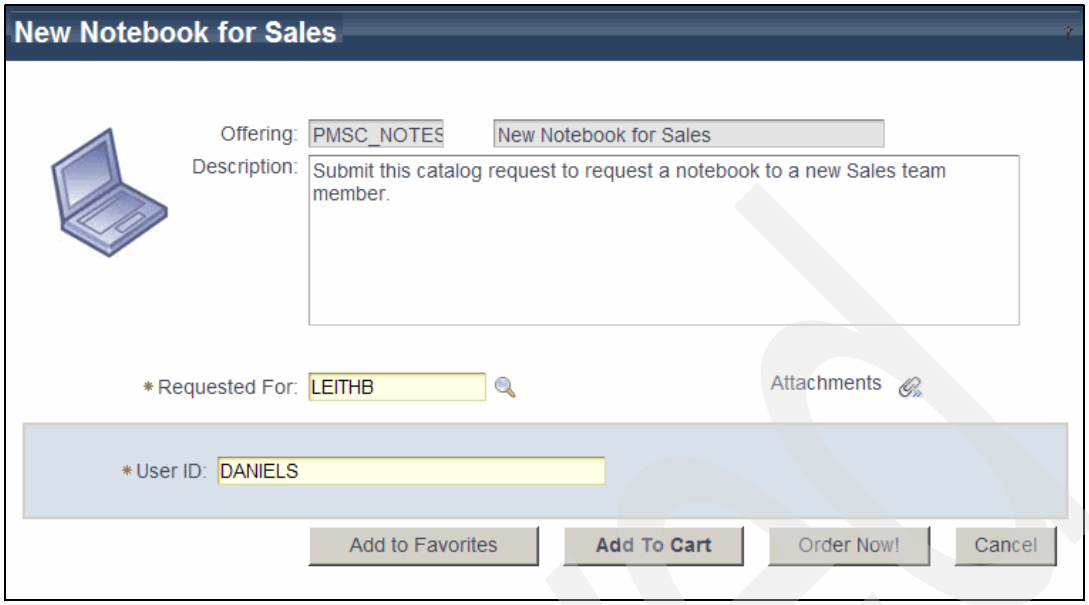


Figure 20-10 Notebook offering

For each cart-item offering that is submitted or direct ordered, a separate service request entry is created.

At this time, the integration between Tivoli Service Request Manager Service Catalog and Tivoli Service Request Manager Service Desk is initiated. A Catalog Request ID is included as an attribute of the newly created service request entry.

20.6.3 Service Desk Analyst

After the Catalog request is done, a new Service request is automatically created and associated with it. The new Service request is assigned to an IT Service Desk Analyst who performs the initial analysis and follows IT procedures to make the requests available to the user.

The new requests are available in the Service Desk Startcenter (Figure 20-11) within Service Catalog Orders result set.

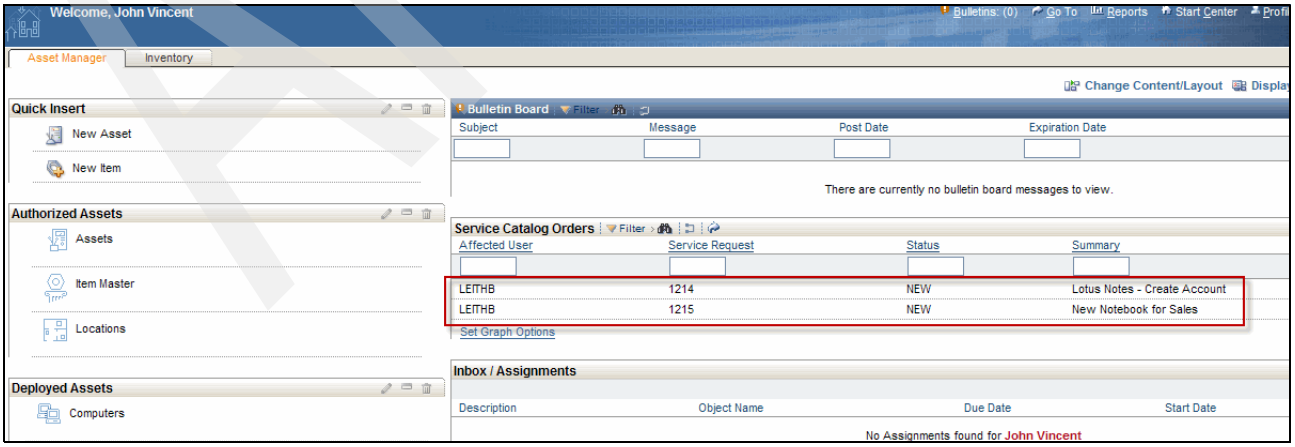


Figure 20-11 New requests are available

A detailed view of the catalog order is shown in Figure 20-12. You may verify information about the requester and the request itself. Select the **Specifications** tab to see the User ID that is formed during the catalog execution.

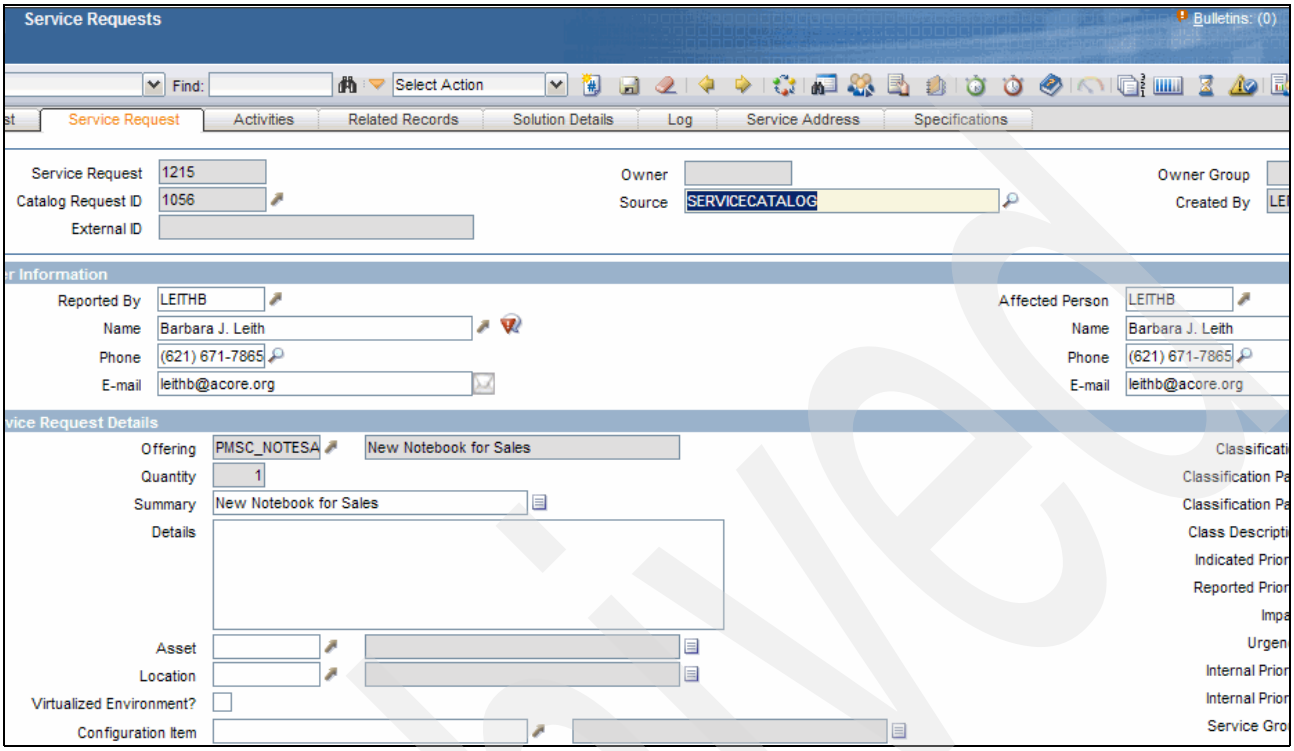


Figure 20-12 Detailed view of the catalog order

With the request verified and accepted, it is time to create a work order and manage the work plan and execution properly. Click **Select Action menu** → **Create** → **Work Order** to create the related record. The integration between the Tivoli Service Request Manager Service Desk and Tivoli Asset Management is initiated.

The newly created work order is not shown automatically. Go to the **Related Record** tab to view and select the appropriate work order (Figure 20-13).

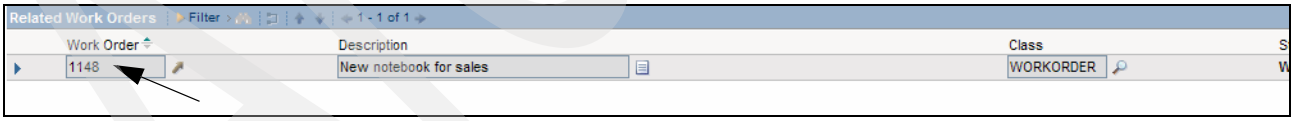


Figure 20-13 Work order

Use the application shortcut just to the right of the work order number to go to Work Order application.

20.6.4 IT Asset Manager

The IT Asset Manager must manage resources to complete the request. The work order just created is the where the required steps are addressed.

Applying a job plan to the type of work

The first step is to apply a job plan specific to this type of work. Figure 20-14 on page 634 details the activities used in this scenario. Job plans can help to determine the specific

activities and materials depending of the role of the new employee. For example, if the new employee is a software developer, the job plan would contain activities that are related to source code credentials and also hardware and software specifics. By expanding this job plan for more roles, you may create a powerful set of templates that can be used for onboarding of a new employee.

Sequence	Task	Summary
50	50	Assign Hardware to Employee
40	40	Assign License to Hardware
30	30	Install Software
20	20	Select Material (HW and SW)
10	10	Check material (HW and SW) availability

Figure 20-14 Job plan

Fulfilling the work order

With the job plan applied, we show you the activities to deliver the required items to the new employee. The activities are detailed in the following sections:

Checking the material (hardware, software) availability

The IT Asset Manager needs to verify if the materials required are in the storeroom and purchase (request) new items if necessary.

To verify the item availability, click the shortcut icon beside the Asset number. The View Item Availability window opens, as shown in Figure 20-15.

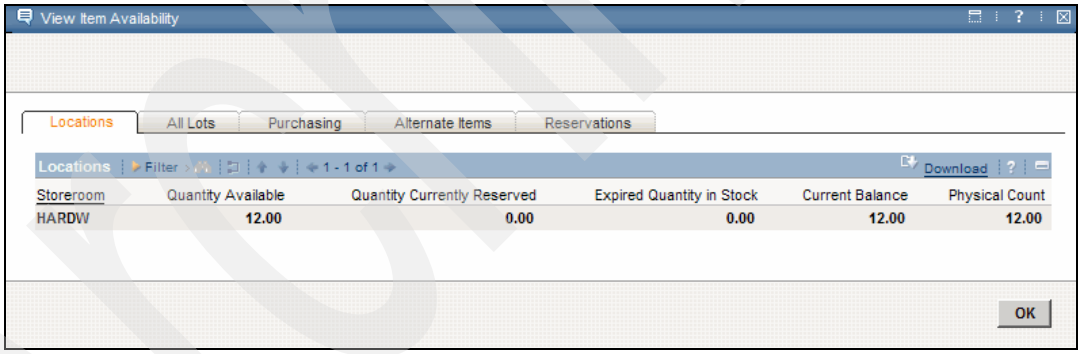


Figure 20-15 Checking Material (hardware and software) availability

The IT Asset Manager verifies the availability of the required software and hardware, which can also include planning of more resources, such as people, tools, and so on (when necessary). This activity is part of the plan phase and must be executed before approval of the work order. Assuming that all resources are available, the work order is ready to be approved.

Selecting materials (Actuals tab)

After finishing the planning phase, the IT Asset Manager selects the actual materials that are necessary to execute the work. This selection is initiated on the Materials subtab that is on the *Actuals* tab. The Select Materials button launches a dialog where the assets required for that request can be selected.

Those assets (hardware and software) are being allocated in an activity and are considered as reserved until the work order is completed or cancelled.

Installing the software

This is a step to consider during the work execution. The installation is performed by the IT Analyst and is related to the software and hardware that are listed as materials of the work order. The software can be automatically provisioned by a program such as Tivoli Provisioning Manager. The People Manager must indicate the roles that this new employee will have in the company so that appropriate software is provisioned on the employee's workstation.

Assigning license to the hardware

During this step, the IT Asset Manager modifies software asset attributes to change the asset location from storeroom to the user's office and also defines the hardware (notebook) as the software asset's parent asset.

The function Move/Swap/Modify on the Select Action menu opens a window when modified operations are available, as shown in Figure 20-16.

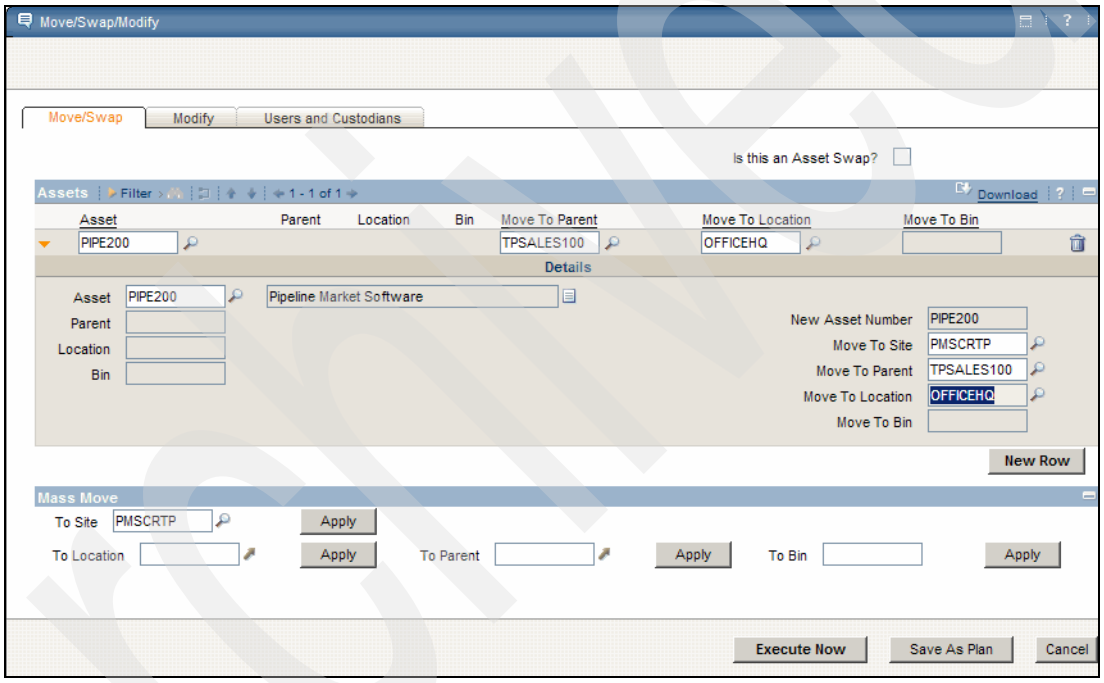


Figure 20-16 Assign license to the hardware

After completing the moving action, you may check the software as part of the hardware asset through the Asset application. This information is available on the Spare Parts tab.

Important: Starting with Tivoli Asset Management for IT V7.2.X, the preferred way to create software assets is to use the License application (Figure 20-17 on page 636). This application is where you can create a license, assign software products from the software catalog to the license, and then allocate the license to an asset, GL Account, or person. The License application provides a single platform to manage software license entitlements. The following applications reference the License application:

- PR/PO Applications: Links to procurement cycle.
- Deployed Software Application: Associates discovered instances.
- Software Contracts Application: Links to contract cycle.

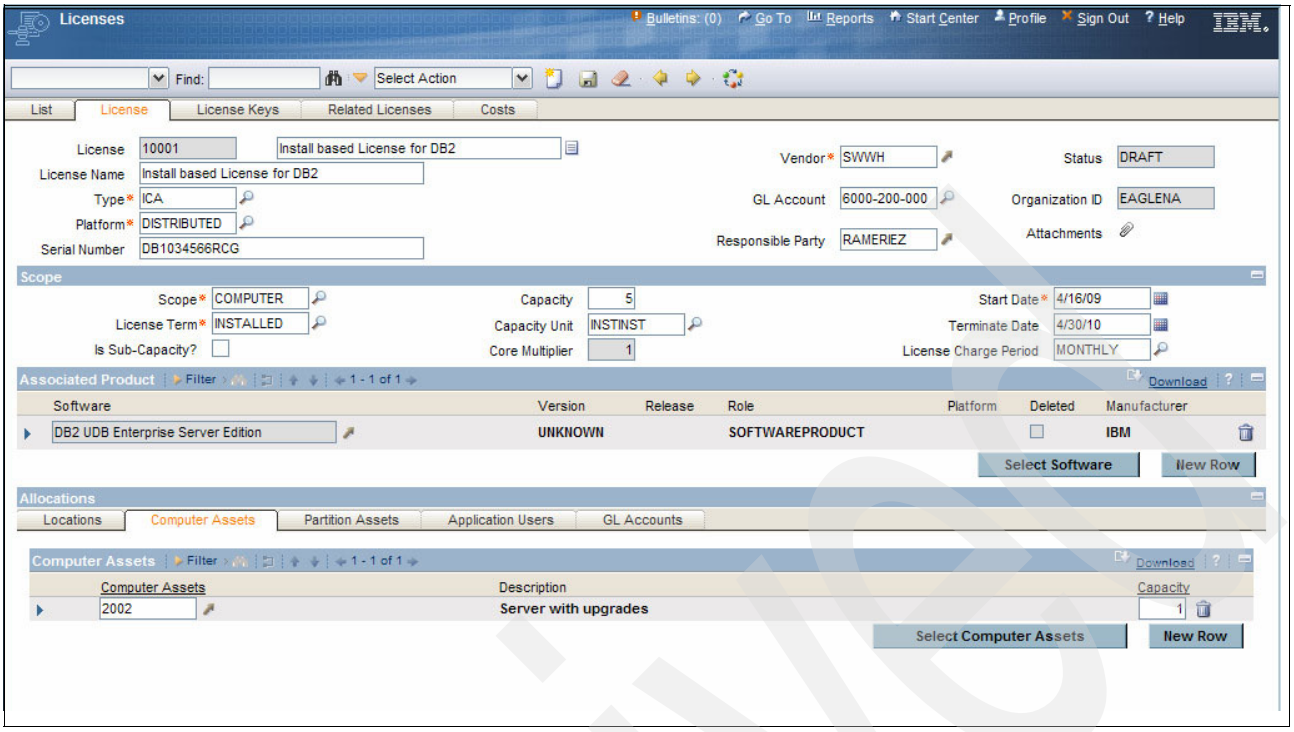


Figure 20-17 Licenses application

Assigning hardware to the employee

To complete the work order execution, IT Asset Manager must associate the hardware and the user that the hardware is delivered to. It is done in the Asset application and can be launched from the associated work order or accessed directly from Go To menu.

With the Asset application open and the asset (Hardware - notebook) already selected, the IT Manager uses the toolbar option as shown in Figure 20-18.

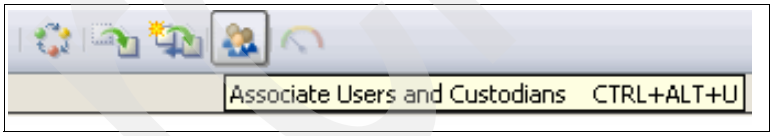


Figure 20-18 Asset application

Archive

The window shown in Figure 20-19 opens and the user can be associated with the notebook.

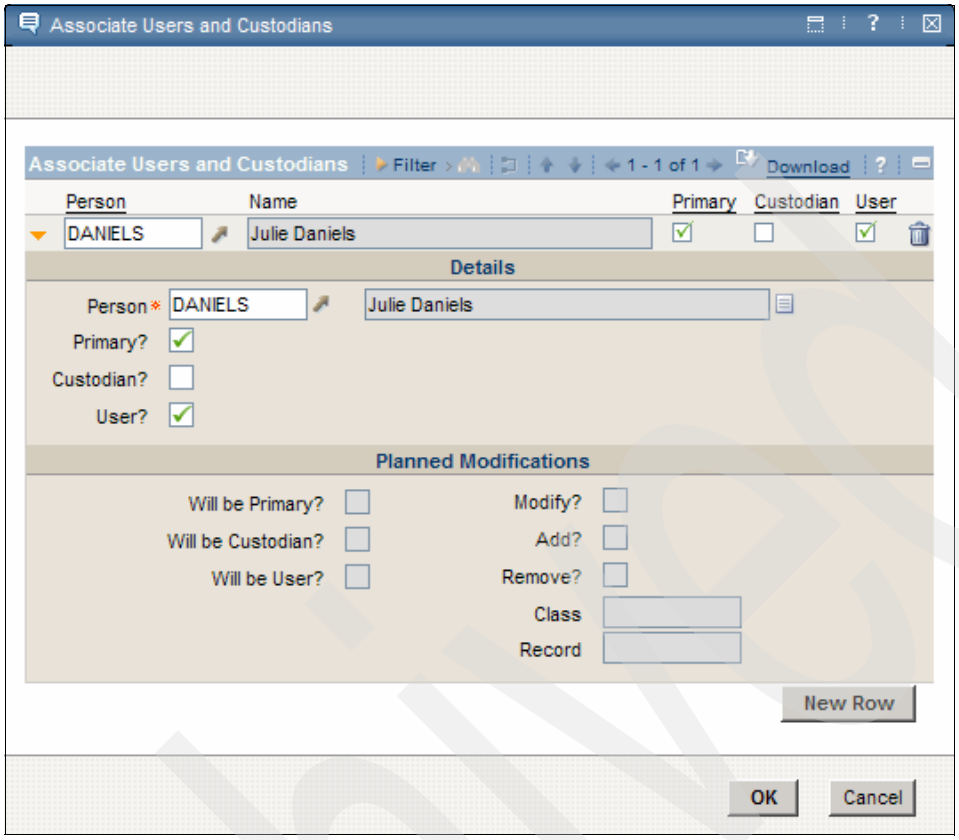


Figure 20-19 User association with the notebook

Notice that the check boxes (Primary and User) are selected. They allow multiple configurations during the assignment activity.

Completing the assignment, the hardware can be delivered to the user and the work order can be closed.

With that, the processes for onboarding a the new employee and delivering the IT assets to them have been completed.

20.6.5 Additional integration: Enterprise Asset Manager

Similar to the IT Asset Manager, the Enterprise Asset Manager can execute the activities (or delegate them) to fulfill the People Manager's requests according to expectations. The difference between the function is what is being delivered, in terms of material and services, and processes necessary to fulfill the request.

In our scenario, the new employee, from sales team, will need a desk, chair, and so on. For this reason, this information is forwarded to the Enterprise Asset Manager.

20.6.6 Additional integration: Security Manager

Several activities may be associated with the Security Manager role, but in this scenario we define the role only as the responsible person to create the new employee credentials (network, Lotus Notes, and so on) based on the HR Manager's input, and associate the new user to specific groups according the People Manager's requests.

Depending on the rules adopted at each client, even what might be automatic tasks, such as Lotus Notes create/reset password, might be incorporated as part of security processes and be approved manually by a manager.

20.6.7 Additional integration: Self-service

Certain services might be offered as self-services after the new employee receives a user ID and initial password.

As examples, Tivoli Identity Manager and Tivoli Service Request Manager integration allows the user to interact with services, such as password reset, service request creation, and service catalog orders.

20.6.8 Additional integration: HR informed that employee onboarding is complete

To end the cycle of request and delivery, HR might be advised that the new employee has received the work material (notebook, mobile phone, and credentials) and is ready to initiate the activities (such as training, for example).

This integration can be designed by using data that is exported from Tivoli Process Automation Engine platform and imported by the HR systems. Tivoli Process Automation Engine can supply the data in a variety of ways such as by using batch files (CSV, XML) or by online invocations using Web Services.

Archived

20.7 Scenario walk-through

Table 20-1 lists the sequence of steps, the role that performs each step, the product used, and the activities that are performed in this integration.

Table 20-1 Scenario walk-through

Step	Role	Product	Activities
1	HR Manager	HR System	1. Register new employee. 2. Get the employee ID. 3. Provide interface with LDAP and Tivoli Identity Manager to register new employee ID.
2	People Manager	Tivoli Service Request Manager - Service Catalog	1. Identify which catalog better fits employee needs and request what is necessary for employee's position. 2. Follow-up execution of requests through service request application.
3	Service desk Analyst	Tivoli Service Request Manager - Service Desk	1. Receive the request. 2. Generate work order based on service request. 3. Assign work order to appropriated team.
4.1	IT Asset Manager	Tivoli Asset Manager for IT	1. Receive IT work order. 2. Check item availability. 3. Select material (manage people, licenses, resources in general). 4. Install software. 5. Assign license to hardware. 6. Assign hardware to employee.
4.2	Enterprise Asset Manager	Maximo Asset Management	1. Receive Enterprise work order. 2. Check item availability. 3. Select material. 4. Assign Asset to employee.
4.3	IT Security Manager	Tivoli Identity Manager	1. Receive Security work order. 2. Create employee credentials. 3. Associate employee ID to appropriated rules.
5	User (new employee)	Tivoli Service Request Manager - Service Catalog	Request Lotus Notes password reset, service request creation, and service catalog orders.
N/A	People Manager	Tivoli Identity Manager	Check security request status.

Archived

20.8 Summary

Figure 20-20 shows the summary of this integration.

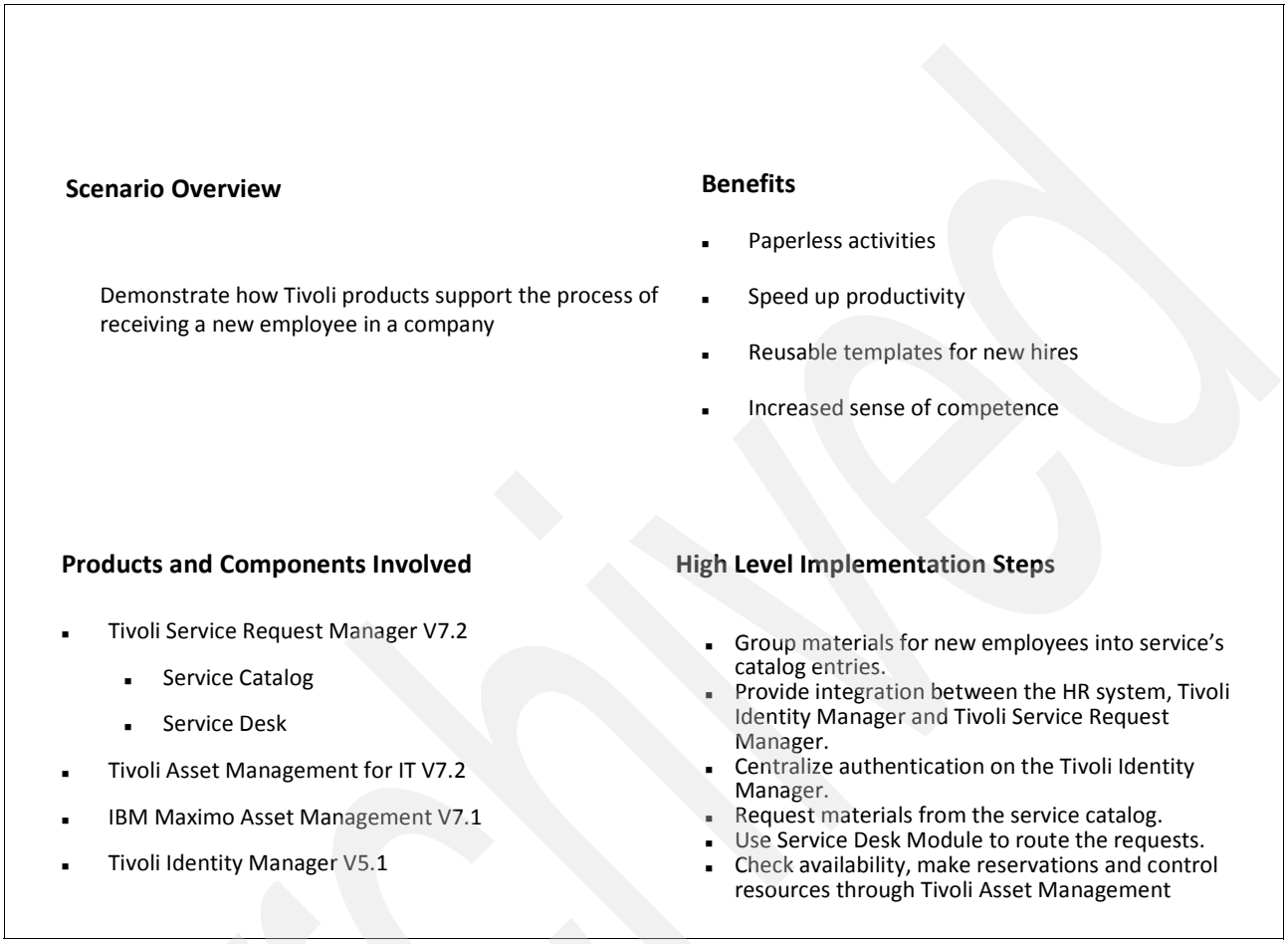


Figure 20-20 Quick summary

Archived

21

TADDM L2 discovery using IBM Tivoli Monitoring agents

This chapter describes the use of IBM Tivoli Monitoring agents to discover components in the IT infrastructure using Tivoli Application Dependency Discovery Manager (TADDM). In this chapter we describe how the IBM Tivoli Monitoring sensor of TADDM discovers configuration items in the IBM Tivoli Monitoring environment and with little configuration overhead.

This chapter contains the following topics:

- 21.1, “Scenario description” on page 642
- 21.2, “Products involved” on page 642
- 21.3, “Architecture” on page 643
- 21.4, “Benefits” on page 643
- 21.5, “Implementation steps” on page 644
- 21.6, “Scenario walk-through” on page 655
- 21.7, “Summary” on page 660

© Copyright IBM Corp. 2011. All rights reserved.

21.1 Scenario description

TADDM provides automated application dependency mapping and configuration auditing. The TADDM discovery process is driven by a discovery profile. TADDM supplies three default discovery profiles:

- Level 1 discovery (also called a *credential-less discovery*) discovers basic information about the active computer systems in the environment
- Level 2 discovery captures detailed information about the active computer systems in the environment, including operating-system details and shallow application information. This level of discovery requires a service account to be created on all computer systems that are discovered using SSH key-based and password-based connections
- Level 3 discovery captures information about the entire application infrastructure, including physical servers, network devices, virtual LANs, host configuration, deployed software components, application configuration, and host data used in the environment. This level of discovery requires application and operating system credentials

In a dynamic enterprise environment, maintaining a service account on all server assets for a TADDM L2 discovery becomes a tedious task. This scenario demonstrates the use of the IBM Tivoli Monitoring sensor to discover configuration items in the IBM Tivoli Monitoring environment.

The IBM Tivoli Monitoring sensor supports TADDM L1 and L2 discoveries.

We use the following steps to demonstrate the scenario:

1. Configure the Tivoli Monitoring Sensor.
2. Distribute the discovery target support bundle to the monitored infrastructure.
3. Install custom queries on the Tivoli Enterprise Portal Server.
4. Configure TADDM discovery profiles.
5. Run a TADDM L2 Discovery.

21.2 Products involved

The following products are used in this integration:

- IBM Tivoli Application Dependency Discovery Manager V7.2
- IBM Tivoli Monitoring V6.2.2

For more information, see the following resources:

- Installation and configuration of Tivoli Application Dependency Discovery Manager 7.2:

  http://publib.boulder.ibm.com/infocenter/tivihelp/v10r1/topic/com.ibm.taddm.doc_7.2/welcome_page/welcome.html

- Installation and configuration of IBM Tivoli Monitoring 6.2.2:

  http://publib.boulder.ibm.com/infocenter/tivihelp/v15r1/topic/com.ibm.itm.doc_6.2.2/itm_install03.htm#wq3

Archived

21.3 Architecture

Figure 21-1 shows the architectural diagram of this integration. The basic steps are as follows:

1. When invoked by TADDM discovery, the IBM Tivoli Monitoring Discovery sensor passes single discovery commands to the managed system name or IP address of the system where it is to be executed. The IBM Tivoli Monitoring agent then executes the command on the system to be discovered.
2. The system returns output streams.
3. IBM Tivoli Monitoring agent returns the standard output and standard error streams of the command.
4. IBM Tivoli Monitoring Discovery sensor parses the returned data and builds result objects.
5. The data is loaded into the TADDM database.

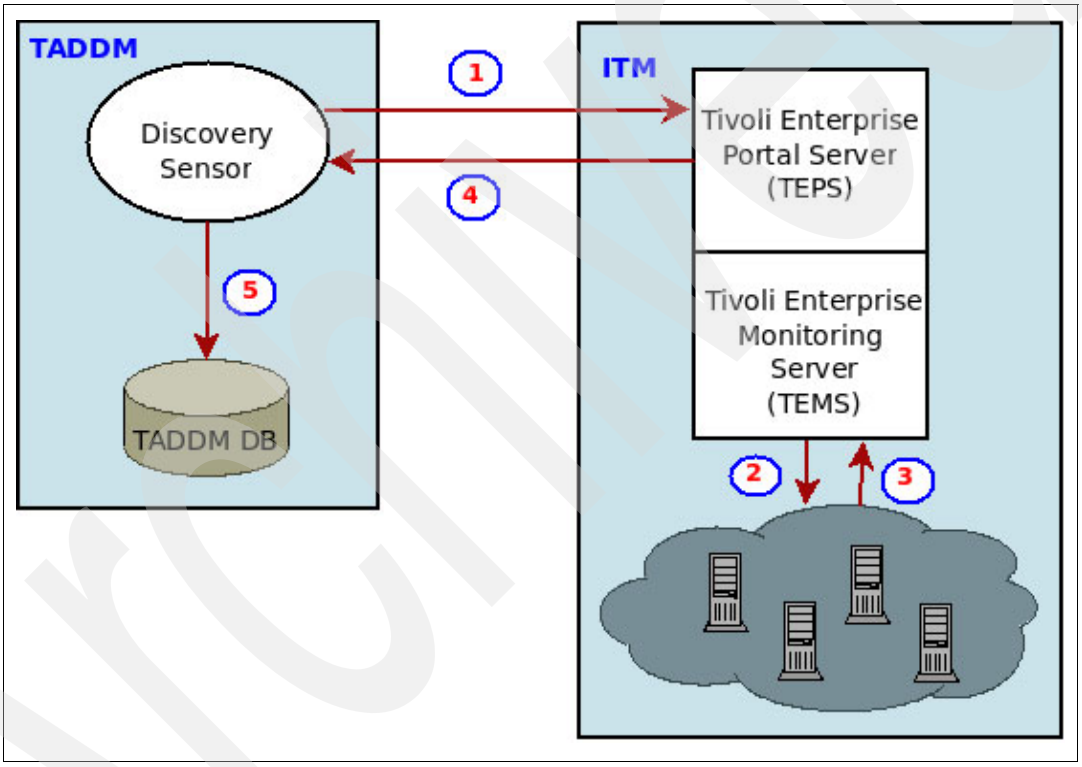


Figure 21-1 Implementation architecture

21.4 Benefits

This solution offers the following benefits:

- Discovers CI information in an infrastructure where IBM Tivoli Monitoring is already running by using the IBM Tivoli Monitoring agents.
- Only the Tivoli Enterprise Portal Server credential is required in performing the discovery; no operating-system level credentials are required for each target.

Archived

- L1 and L2 discovery can be performed by using IBM Tivoli Monitoring operating system agents. The sensor discovers Tivoli Enterprise Portal Server and Tivoli Monitoring agents for IBM Tivoli Monitoring Version 6.2.1 or later.
- AIX, HP-UX, Linux, Solaris, and Windows are supported. See Table 21-1.
- It is more secure because it requires only the IBM Tivoli Monitoring Tivoli Enterprise Portal Server credentials.

Table 21-1 Supported operating systems

Tivoli Monitoring Level 1 discovery	Tivoli Monitoring Level 2 discovery
Microsoft Windows 2000/2003, 2008	Microsoft Windows 2000/2003, 2008
HP-UX 11i	HP-UX 11i
Solaris 2.8, 2.9, and 2.10	Solaris 2.8, 2.9, and 2.10
AIX 5.x, 6.1	AIX 5.x, 6.1
Linux on x-32 or x-64 architecture	Linux on x-32 or x-64 architecture

21.5 Implementation steps

In this section, we describe the steps that are required to implement this scenario.

Configure the IBM Tivoli Monitoring Sensor

Copy the file in this process from the IBM Tivoli Monitoring Server to the TADDM Server:

1. On the TADDM server, create a `$COLLATION_HOME/lib/itm` directory.

   Note: The TADDM installation is $COLLATION_HOME. By default, it is in `C:\IBM\CMDB\DIST` on Windows and `/opt/IBM/cmdb/dist` on Linux and UNIX.

2. Copy the following files from the IBM Tivoli Monitoring Tivoli Enterprise Portal Server into the $COLLATION_HOME/lib/itm directory on the TADDM server:

   On a Windows system, copy the following files from the `$CANDLEHOME\CNB\classes` directory, on Linux and UNIX systems, copy the files from the `$CANDLEHOME/cq/classes` directory:

   - `browser.jar`
   - `cnp.jar`
   - `cnp_vbjorball.jar`
   - `kjrall.jar`
   - `util.jar`
   - `tap_cli.jar`

   Note: The `$CANDLEHOME` is the IBM Tivoli Monitoring installation directory. By default, it is in `C:\IBM\ITM` on Windows and `/opt/IBM/ITM` on Linux and UNIX.

3. From the `$CANDLEHOME/GSK7/classes` directory on the Tivoli Enterprise Portal Server, copy the `cfwk.zip` file.

Archived

4. On Linux and UNIX systems, use the following command to set the user and group of the previously copied files to the user and group that is used to run the TADDM server:

   ```
   chown -r taddmuser:taddmuser $COLLATION_HOME/lib/itm
   ```

5. Restart the TADDM server by using the following steps:

   a. Log in as the non-root user that was defined during the TADDM installation process.

   b. Go to the directory where you installed the TADDM server. Use one of the following commands to run the start script:

      - On Windows:

        ```
        %COLLATION_HOME%\bin\control.bat start
        ```

      - On Linux and UNIX:

        ```
        $COLLATION_HOME/bin/control start
        ```

Distributing the discovery target support bundle

During the discovery process, the TADDM must copy binary file data between itself and the discovery target using IBM Tivoli Monitoring as an intermediary. The support bundles must be distributed to the target client servers by using Tivoli Enterprise Monitoring Server depot. The support bundle must also be loaded into any remote Tivoli Enterprise Monitoring Server depots that exist in the IBM Tivoli Monitoring environment.

Sharing an agent depot across your environment: If your monitoring environment includes multiple monitoring servers (a hub monitoring server and remote monitoring servers), you can put your agent depot in a central location, such as a shared file system, and access the depot from all of the monitoring servers. See the steps at:

http://publib.boulder.ibm.com/infocenter/tivihelp/v15r1/topic/com.ibm.itm.doc_6.2.2fp2/share_depot.htm#share_depot

The necessary prerequisites for TADDM discovery should be installed on each IBM Tivoli Monitoring endpoint. The instructions to configure the target system for L2 discovery is at the following address:

http://publib.boulder.ibm.com/infocenter/tivihelp/v10r1/topic/com.ibm.taddm.doc_7.2/AdminGuide/r_cmdb_confighostsfordiscovery.html

Copy the TADDM support bundle from the TADDM DVD. The support bundle is in the `KD7.zip` file in the `/itm-discovery-support` directory. Extract it in a temporary folder.

To distribute the support bundle to the Tivoli Enterprise Monitoring Server depot, perform the following steps:

1. Log in to the Tivoli Enterprise Monitoring Server as a user who can run the **tacmd** command, shown (for Windows, and for Linux and UNIX) in Example 21-1.

Example 21-1 Log in to Tivoli Enterprise Monitoring Server

```
On Windows:
C:\IBM\ITM\BIN>tacmd.exe login -s localhost -u sysadmin

 Password?

Validating user...

KUIC00007I: User sysadmin logged into server on https://localhost:3661.
```

On Linux and UNIX:

```
<$CANDLEHOME>/bin/tacmd login -s localhost -u sysadmin

Password?

Validating user...

KUIC00007I: User sysadmin logged into server on https://localhost:3661.
```

2. Add the TADDM bundle to the Tivoli Enterprise Monitoring Server depot. See Example 21-2, which shows the process on Windows, and on Linux and UNIX.

Example 21-2 Add TADDM bundle

On Windows:

```
C:\IBM\ITM\BIN>tacmd.exe addBundles -i C:\TEMP\Support_Bundles\KD7\062001000\

 KUICAB023I: Are you sure you want to add the following bundles to the
C:\IBM\ITM\CMS\depot\ depot?

Product Code : d7
Deployable   : true
Version      : 062001000
Description  : TADDM Discovery through ITM enablement
Host Type    : aix526
Host Version : aix526,aix536
Prerequisites:

Product Code : d7
Deployable   : true
Version      : 062001000
Description  : TADDM Discovery through ITM enablement
Host Type    : aix613
Host Version : aix613
Prerequisites:

Product Code : d7
Deployable   : true
Version      : 062001000
Description  : TADDM Discovery through ITM enablement
Host Type    : hp11
Host Version : hp11,hp116
Prerequisites:

Product Code : d7
Deployable   : true
Version      : 062001000
Description  : TADDM Discovery through ITM enablement
Host Type    : li6213
Host Version : li6213
Prerequisites:

Product Code : d7
Deployable   : true
Version      : 062001000
Description  : TADDM Discovery through ITM enablement
```

Archived

```
Host Type    : li6263
Host Version : li6263
Prerequisites:

Product Code : d7
Deployable   : true
Version      : 062001000
Description  : TADDM Discovery through ITM enablement
Host Type    : lx8266
Host Version : lx8266
Prerequisites:

Product Code : d7
Deployable   : true
Version      : 062001000
Description  : TADDM Discovery through ITM enablement
Host Type    : sol506
Host Version : sol506
Prerequisites:

Product Code : d7
Deployable   : true
Version      : 062001000
Description  : TADDM Discovery through ITM enablement
Host Type    : sol603
Host Version : sol603
Prerequisites:

Product Code : d7
Deployable   : true
Version      : 062001000
Description  : TADDM Discovery through ITM enablement
Host Type    : sol606
Host Version : sol606
Prerequisites:

Product Code : d7
Deployable   : true
Version      : 062001000
Description  : TADDM Discovery through ITM enablement
Host Type    : WINNT
Host Version : WINNT
Prerequisites:
 KUICAB024I: Enter Y for yes or N for no: Y

 KUICAB020I: Adding bundles to the C:\IBM\ITM\CMS\depot\ depot. The time
required to complete this operation depends on the number and size of the added
bundles.

 KUICAB022I: The bundles were successfully added to the
C:\IBM\ITM\CMS\depot\depot
```

On Linux and UNIX:

```
<$CANDLEHOME>/bin/tacmd addBundles -i /tmp/TADDM/KD7/062001000/
```

Archived

3. Distribute the support bundle from the Tivoli Enterprise Monitoring Server to the discovery targets. See Example 21-3, which shows the process on Windows, and on Linux and UNIX.

Example 21-3 Distribute the TADDM support bundle

```
On Windows:
C:\IBM\ITM\BIN>tacmd.exe addSystem -t D7 -n Primary:ISLISIT64:NT

KUICAR010I: The agent type D7 is being deployed.

KUICAR028I: The operation has been successfully queued for deployment, the
trans
action id is 127838596650000000000041, use the getDeployStatus CLI to view the
status.

On Linux and UNIX:
<$CANDLEHOME>/bin/tacmd addSystem -t D7 -n Primary:ISLISIT64:NT
```

Important: Message similar to the following messages might occur after deploying the discovery target support bundle:

```
Error Message: KDY1024E: The command /opt/IBM/ITM/bin/CandleAgent
-h /opt/IBM/ITM start d7 did not start or stop agent.
The command returned a return code.

Error Message: KDY1008E: The agent action INSTALL failed with a return code of for
product code d7. The command /opt/IBM/ITM/tmaitm6/aix526/bin/kdy_xa -setCMS d7
produced the following error text: <Variable formatSpec="{4}">stdErr Text</Variable>.
The specified return code was received from the two way translator.
```

These messages do not indicate actual errors, because the discovery target support bundle is not intended to respond to the agent start or stop command. The Tivoli Monitoring **cinfo** command also does not list the support bundle, because the support bundle is an addition to the existing OS agent.

Verify that the discovery target support bundle is correctly installed on the discovery target. From the Tivoli Monitoring directory on the target computer, run the **find** command as follows:

```
[root@machine ITM]# cd /opt/IBM/ITM
[root@machine ITM]# find . -iregex .+D7
./tmaitm6/agentdepot/d7
[root@machine ITM]#
```

On Windows, TADDM support bundle is deployed at the following location:

```
C:\IBM\ITM\TMAITM6\agentdepot\d7\
```

On Linux and UNIX servers, it is deployed at the following location:

```
/opt/IBM/ITM/tmaitm6/agentdepot/d7
```

4. Check the status of the deployment using the **tacmd getDeployStatus** command. See Example 21-4 on page 649 (for both Windows, and Linux and UNIX).

Example 21-4 Check deployment status

```
On Windows:
C:\IBM\ITM\BIN>tacmd.exe getDeployStatus -g 1278385966500000000000041
Transaction ID : 1278385966500000000000041
Command        : INSTALL
Status         : SUCCESS
Retries        : 0
TEMS Name      : HUB_IMONITOR
Target Hostname: Primary:ISLISIT64:NT
Platform       : WINNT
Product        : D7
Version        : 062001000
Error Message  : KDY0028I: Request completed successfully. Deployment request
was processed successfully and is now completed.

On Linux and UNIX:
<$CANDLEHOME>/bin/tacmd -g <TRANSACTION_ID>
```

Installing custom queries on the Tivoli Enterprise Portal Server

To support the lookup of managed system MAC addresses and agent versions by the IBM Tivoli Monitoring sensor, custom queries must be installed on the Tivoli Enterprise Portal Server.

On the TADDM DVD, the custom queries are in the `TEPS_Query.zip` file in the `/itm-discovery-support` directory. Extract the custom queries to a temporary folder:

- `zkd7.Agent_Version.sql` (Returns the version number of the agent on each endpoint)
- `zkd7.MAC_Address_LZ.sql` (Returns the MAC address of each Linux endpoint)
- `zkd7.MAC_Address_UX.sql` (Returns the MAC address of each UNIX endpoint)

For detailed instructions about installing the custom queries, go to the following website:

`http://publib.boulder.ibm.com/infocenter/tivihelp/v10r1/index.jsp?topic=/com.ibm.taddm.doc_7.2/SensorGuideRef/r_cmdb_sensor_itm_install_custom_queries.html`

Configuring the discovery profile

If you want to create computer system objects for each discovered endpoint or to use a Tivoli Enterprise Portal Server port other than 1920, update the discovery profile for the IBM Tivoli Monitoring sensor.

By default, TADDM discovers Tivoli Monitoring endpoints and creates a scope set. The scope set contains the discovered endpoints and uses the default Tivoli Enterprise Portal Server port 1920. Computer system objects are not created for the Tivoli Monitoring endpoints.

The Tivoli Monitoring sensor provides both Level 1 and Level 2 data for TivoliMonitoring endpoints.

To update the discovery profile, complete the following steps:

1. Log in to Tivoli Application Dependency Discovery Manager. To log in to the TADDM Product Console, use the following web address, where <*TADDM_Server_Name*> is the IP address/DNS name of the server that is running the TADDM application:

   `http://<TADDM_Server_Name>:9430/`

2. Click **Start Product Console** and log in using the user credentials. See Figure 21-2 on page 650.

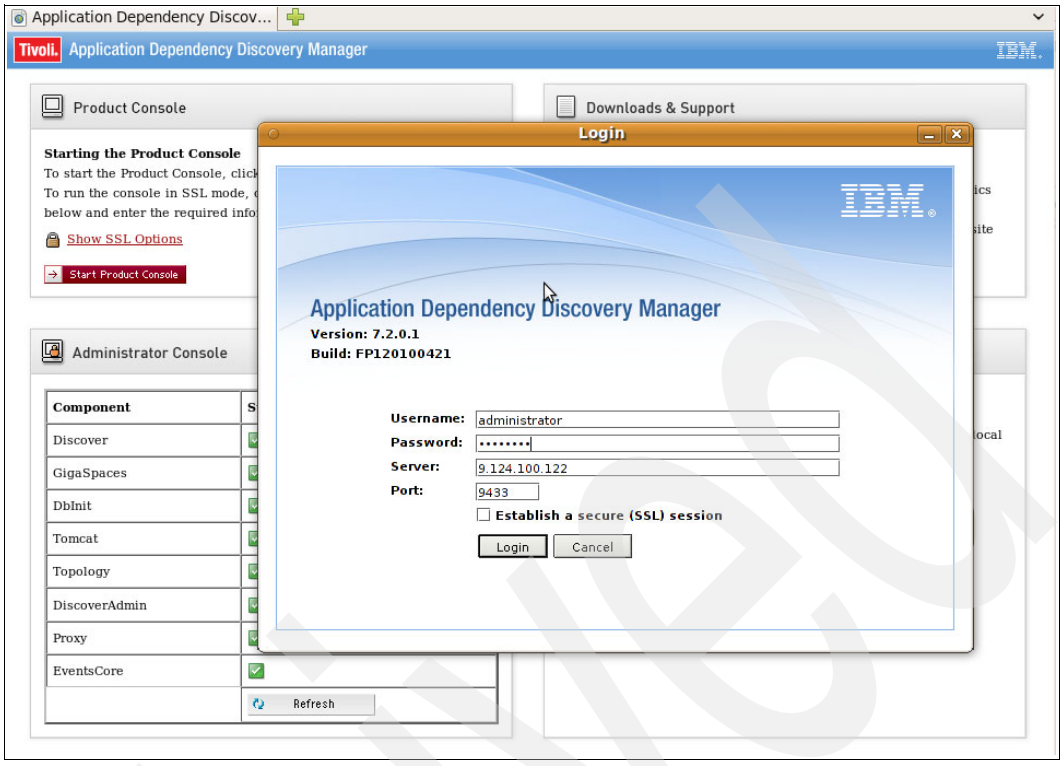


Figure 21-2 TADDM Login Console

3. From the Product Console, click the **Discovery Profiles** icon. See Figure 21-3.

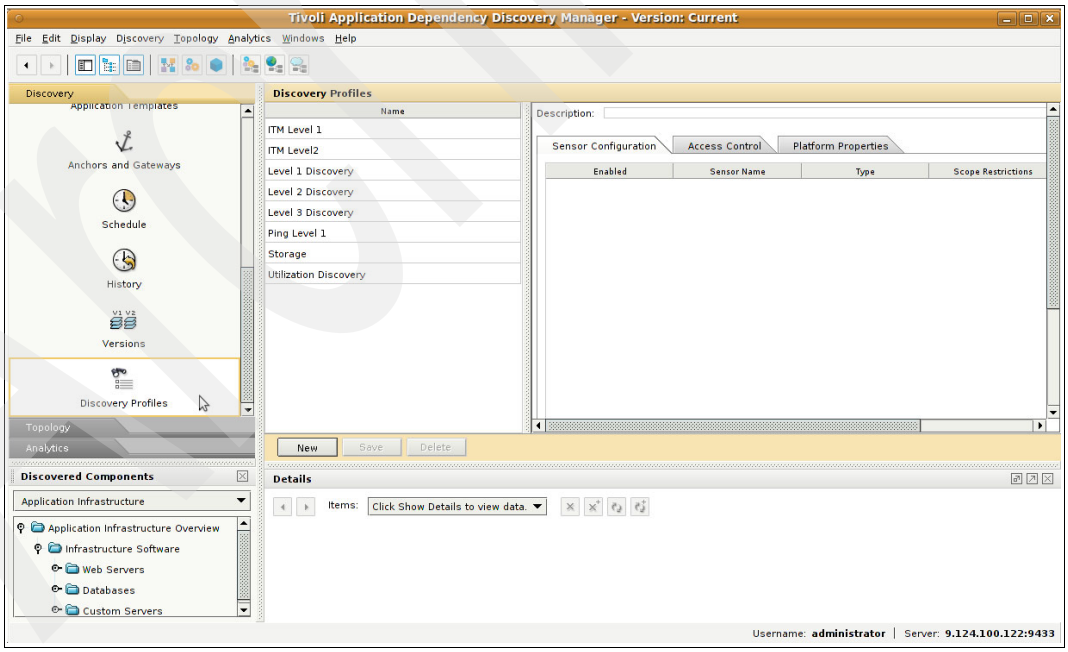


Figure 21-3 Click Discovery Profiles

4. In the Discovery Profiles window, click **New** as shown in Figure 21-4.

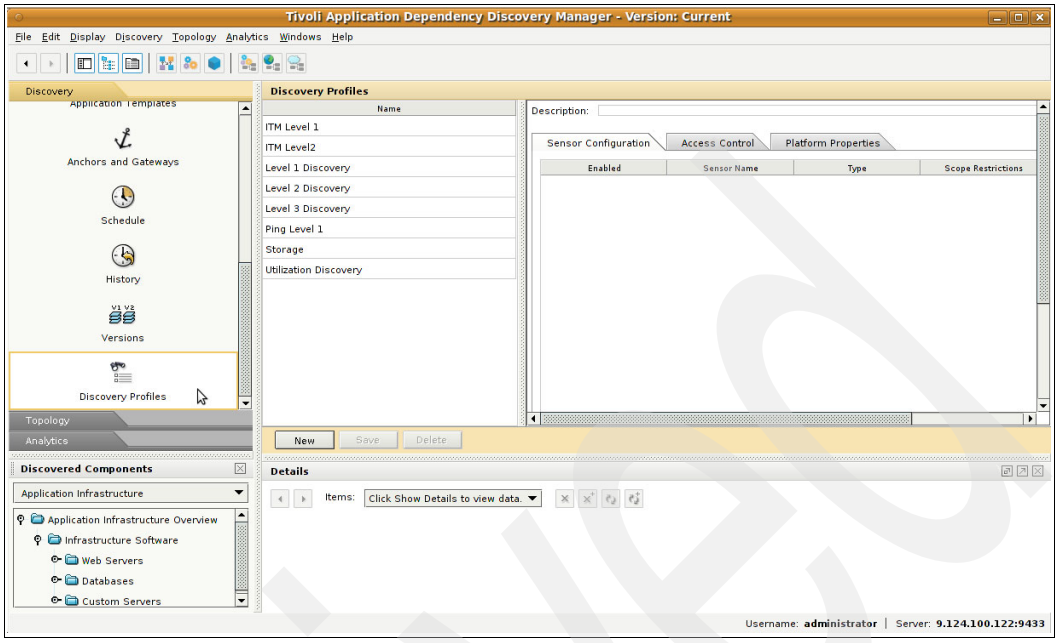


Figure 21-4 Create a new discovery profile

5. In the Create New Profile window, type the profile name and description. In the Clone existing profile field, click **Level 1 Discovery** or **Level 2 Discovery**, and click **OK**. See Figure 21-5.

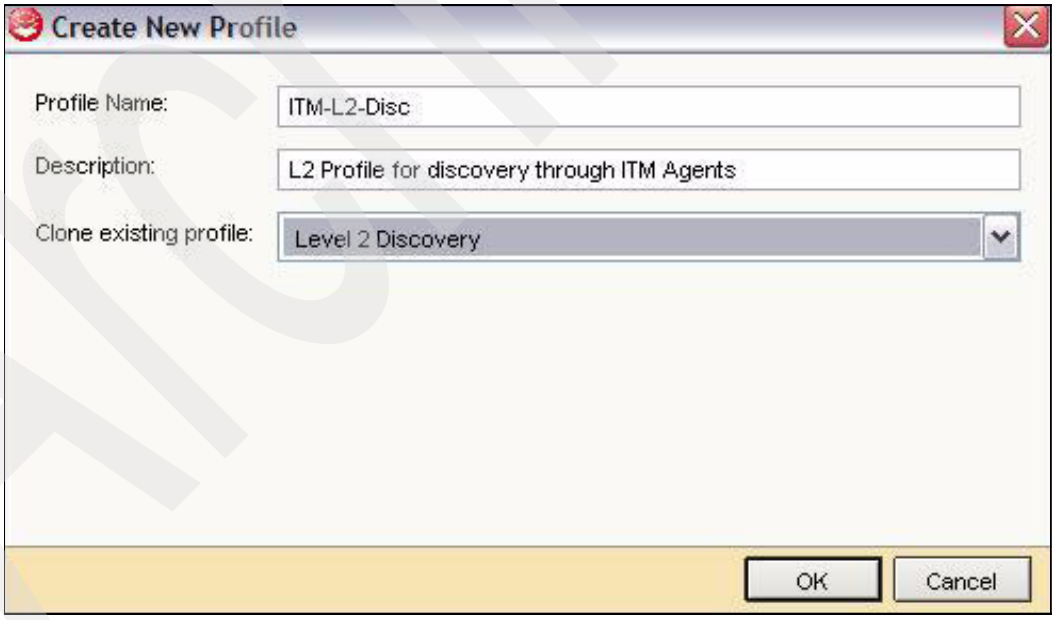


Figure 21-5 Create New Profile

Archived

6. In the list of sensors, click the I**TMScopeSensor**, clear the check box in the Enabled column for this sensor, and click **New**. See Figure 21-6.

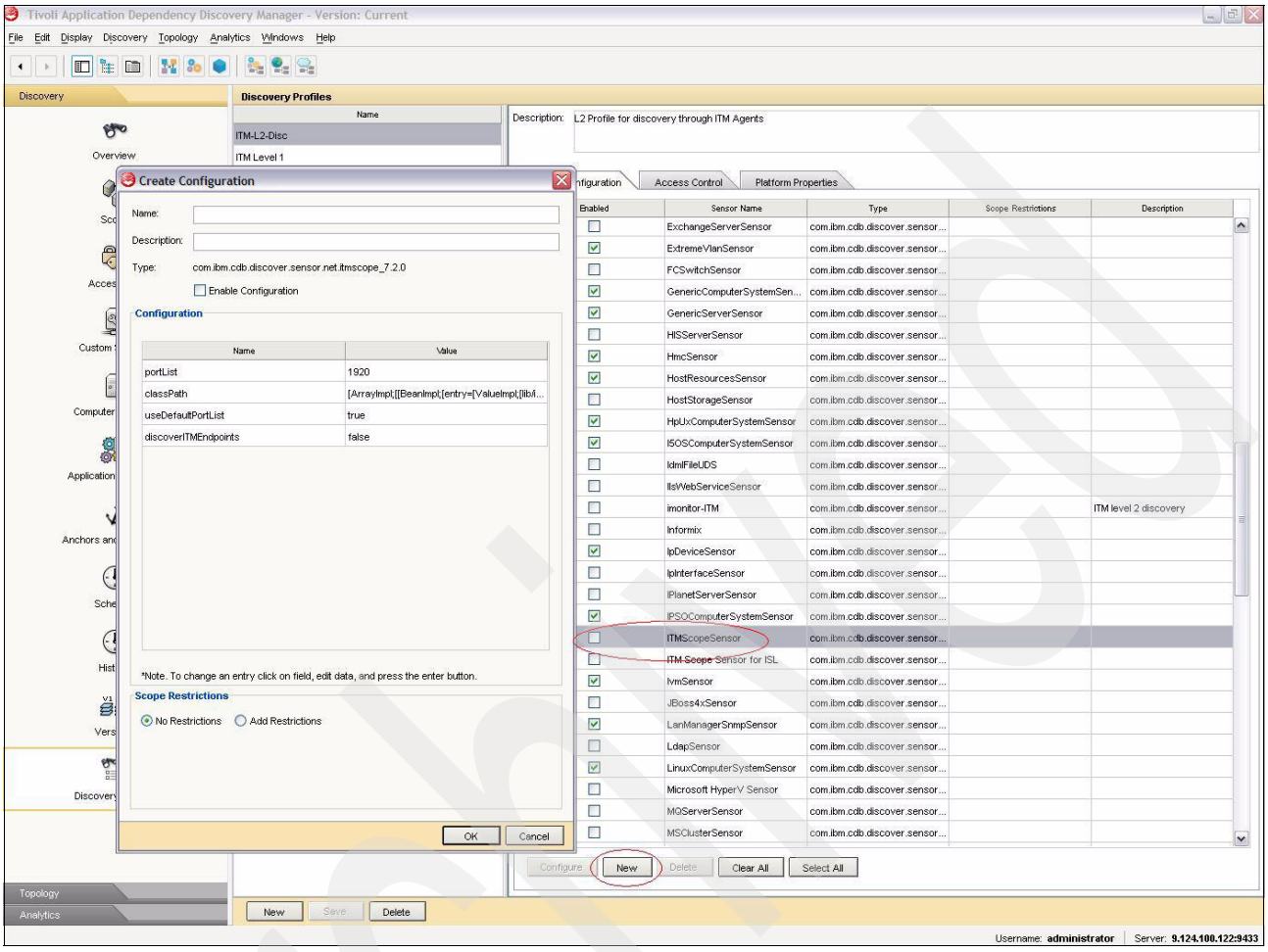


Figure 21-6 New ITM Scope Sensor

7. In the Create Configuration window, type the name and description for your configuration of the ITMScopeSensor, and select the **Enable Configuration** check box.
8. In the Configuration section of the Create Configuration window, to define a set of ports to look for the Tivoli Enterprise Portal Server, click **portList**. Then, double-click the **Value** field in the row, and type each port number value, separating each value with a comma.
9. To configure the sensor *not* to use port 1920, click **useDefaultPortList**, and then double-click the **Value** field in the row, and type `false`. The default value for `useDefaultPortList` is true. If a port list is provided and `useDefaultPortList` is set to true, port 1920 is added to the list of ports to look for the Tivoli Enterprise Portal Server.
10. To create computer system objects that display in the discovered components tree during a Level 1 discovery, click **discoverITMEndpoints.** Then, double-click the **Value** field in the row, and type `true`. If you do not want to create computer system objects during a Level 1 discovery, either do not type anything in the field, or type false.

11. Click **OK t**o return to the Discovery Profiles window.
12. In the Discovery Profiles window, click **Save**. See Figure 21-7.

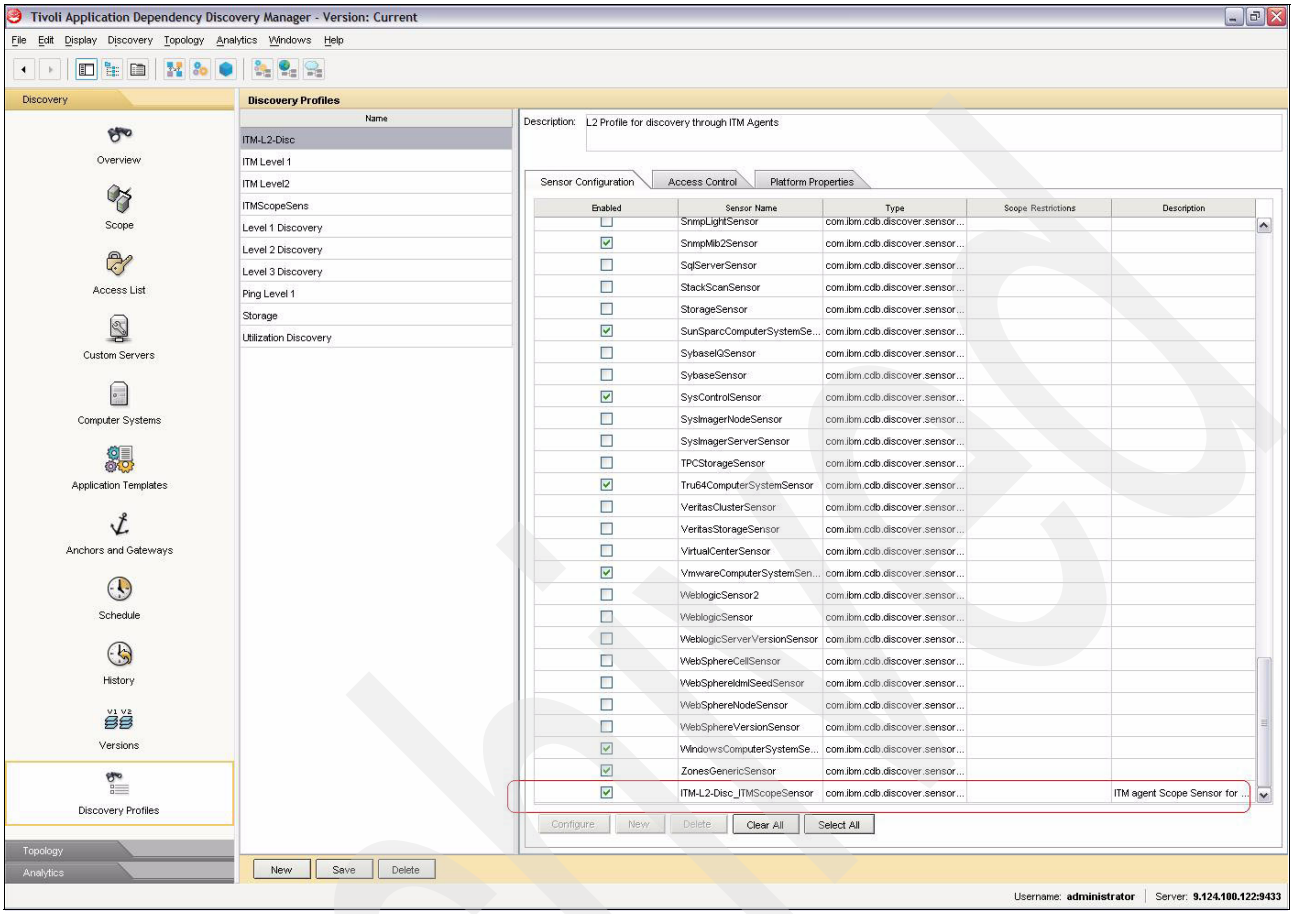


Figure 21-7 View ITM Scope Sensor created in the Discovery Profiles View

Configuring the access list

To give the IBM Tivoli Monitoring sensor access to the Tivoli Enterprise Portal server application, you must configure the access list:

1. From the Product Console, create a discovery scope set that contains your Tivoli Enterprise Portal Server, or use an existing scope that contains your Tivoli Enterprise Portal Server. See Figure 21-8.

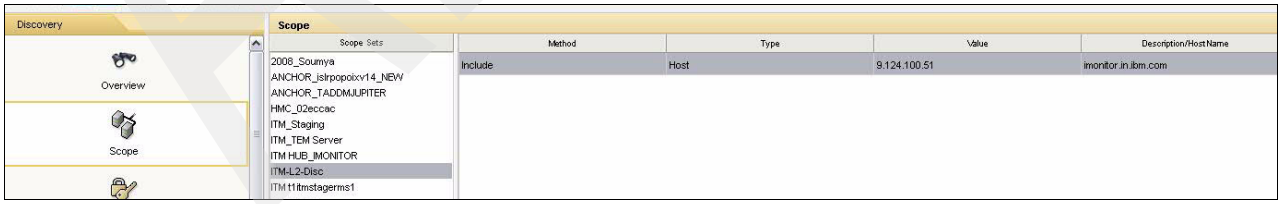


Figure 21-8 Create a TADDM Scope set containing Tivoli Monitoring Portal Server details

2. To create an access list, click the **Access List** icon.
3. In the Access List window, click **Add**.

4. In the Component Type field of the Access Details window, click **IBM Tivoli Monitoring**.
5. Type the credentials for the Tivoli Enterprise Portal Server. Use the credentials that are required to log in to the Tivoli Enterprise Portal Server rather than the credentials for the computer on which the Tivoli Enterprise Portal Server resides. See Figure 21-9.

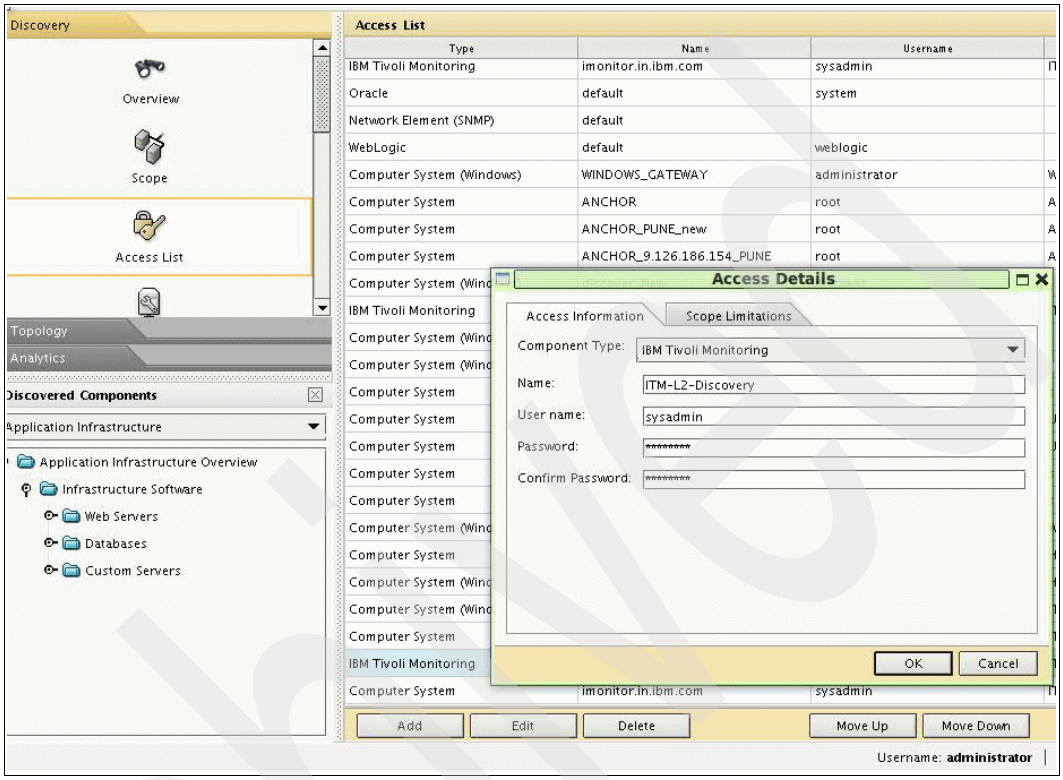


Figure 21-9 Create an Access List

Archive

21.6 Scenario walk-through

After performing all the prerequisite tasks in 21.5, “Implementation steps” on page 644, the integration allows the execution of the scenario in this section.

21.6.1 Deploy the TADDM discovery bundle using Tivoli Enterprise Portal

The TADDM Discovery Bundle can be deployed by using the command described in Example 21-3 on page 648 or by using the Tivoli Enterprise Portal.

Perform the following steps:

1. Log in to the IBM Tivoli Enterprise Portal by using the following URL:

   `http://<ITM_Server_Name>:1920///cnp/kdh/lib/cnp.html`

   In the URL address, replace *<ITM_Server_Name>* with the IP address/DNS name of the server running the IBM Tivoli Enterprise Portal application. See Figure 21-10.

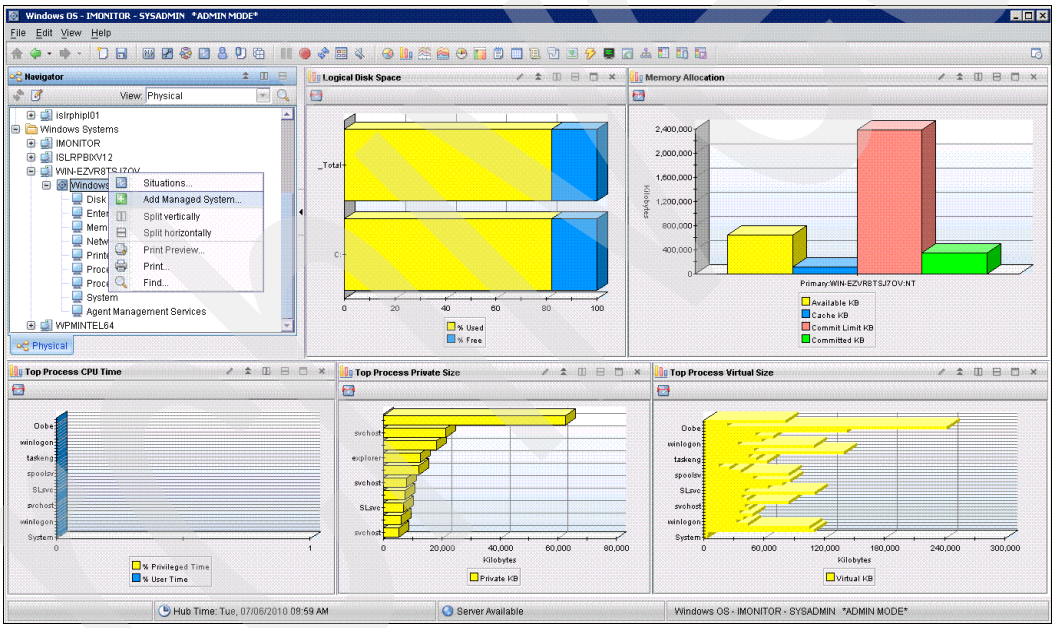


Figure 21-10 Add managed system

2. Click **Add Managed System**. Select the **TADDM monitoring agent bundle**. See Figure 21-11.

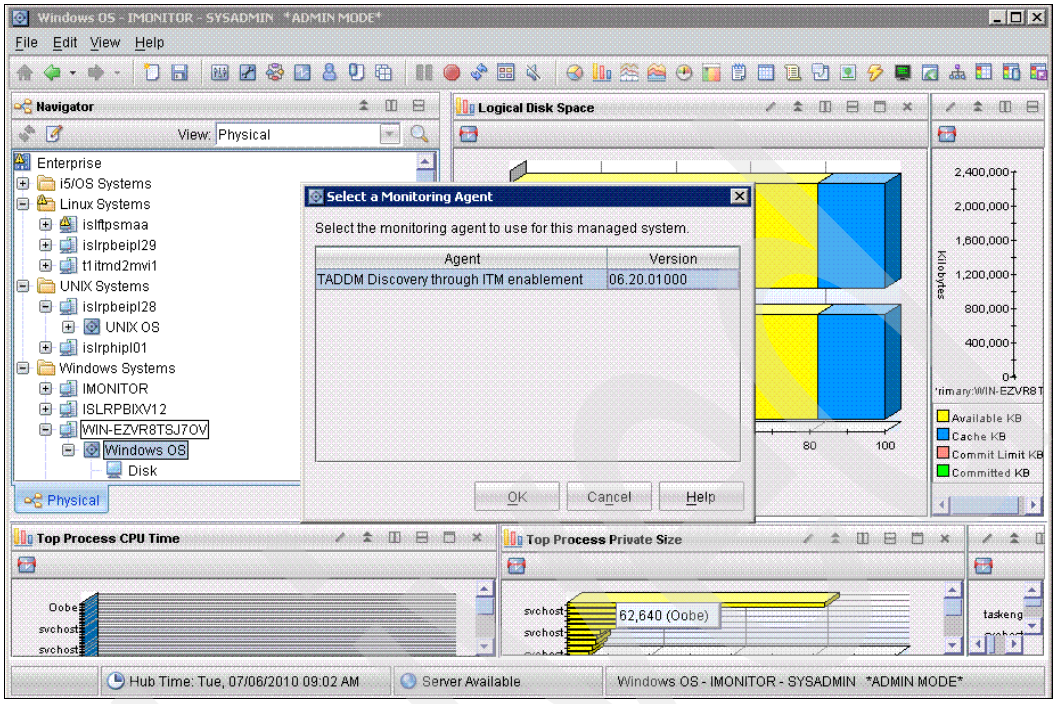


Figure 21-11 Select the monitoring agent bundle

3. View the deployment status on the Tivoli Enterprise Portal using the “Deployment Status by Product” workspace. See Figure 21-12.

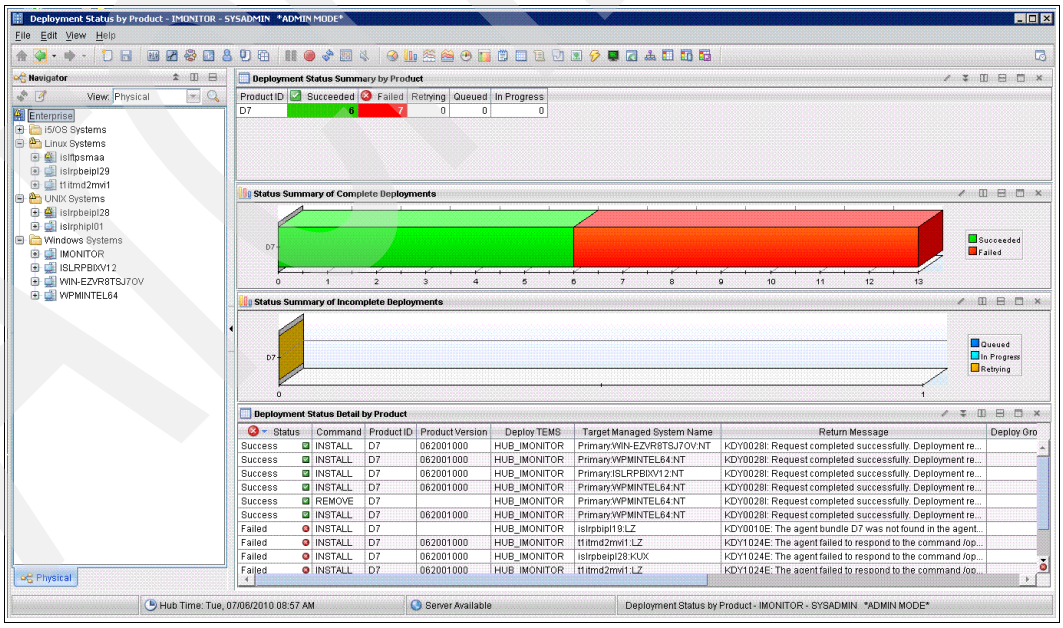


Figure 21-12 Tivoli Monitoring Deployment Status by Product Workspace

21.6.2 View TADDM L2 Discovery data

Perform the following steps to view the TADDM L2 Discovery data:

1. Log in to the TADDM Product Console. Click **Run Discovery**, as shown in Figure 21-13.

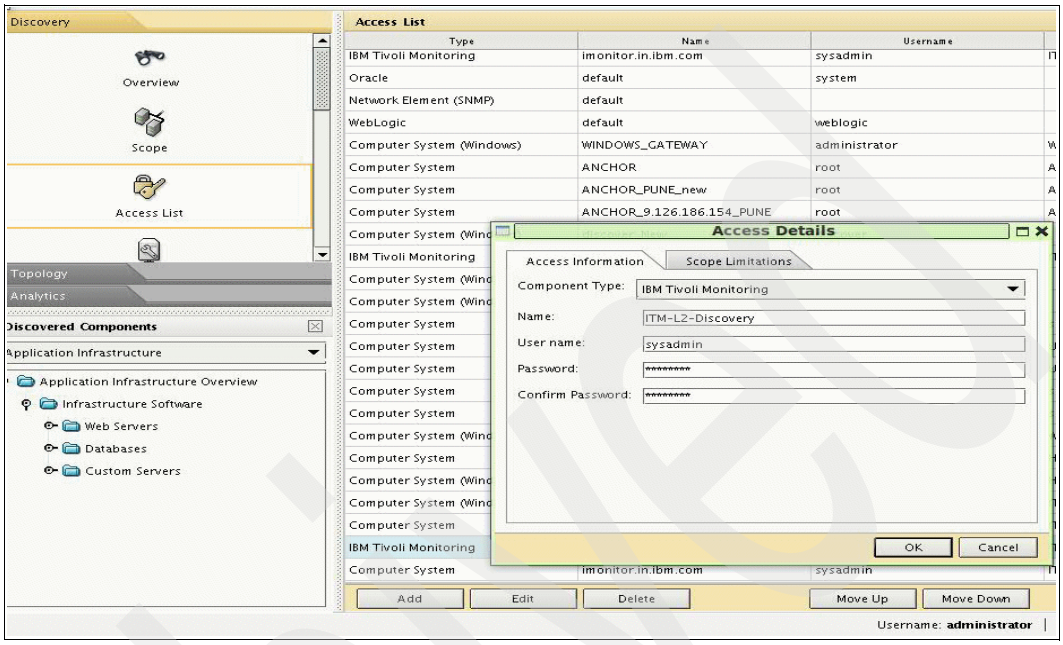


Figure 21-13 Run Discovery

The discovery runs as shown in Figure 21-14.

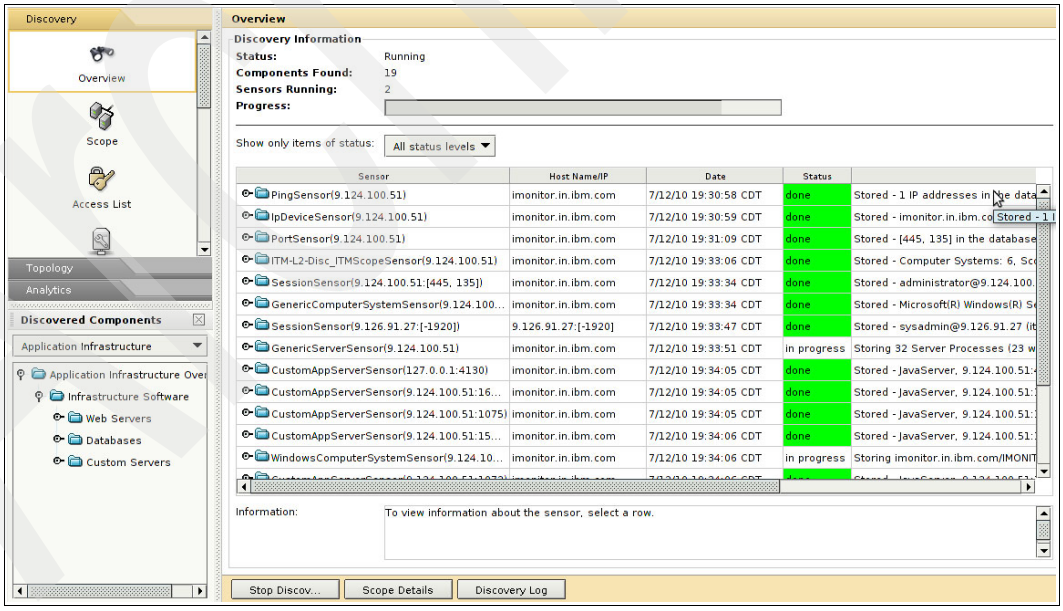


Figure 21-14 Discovery in progress

2. View the discovered components in the TADDM report. See Figure 21-15.

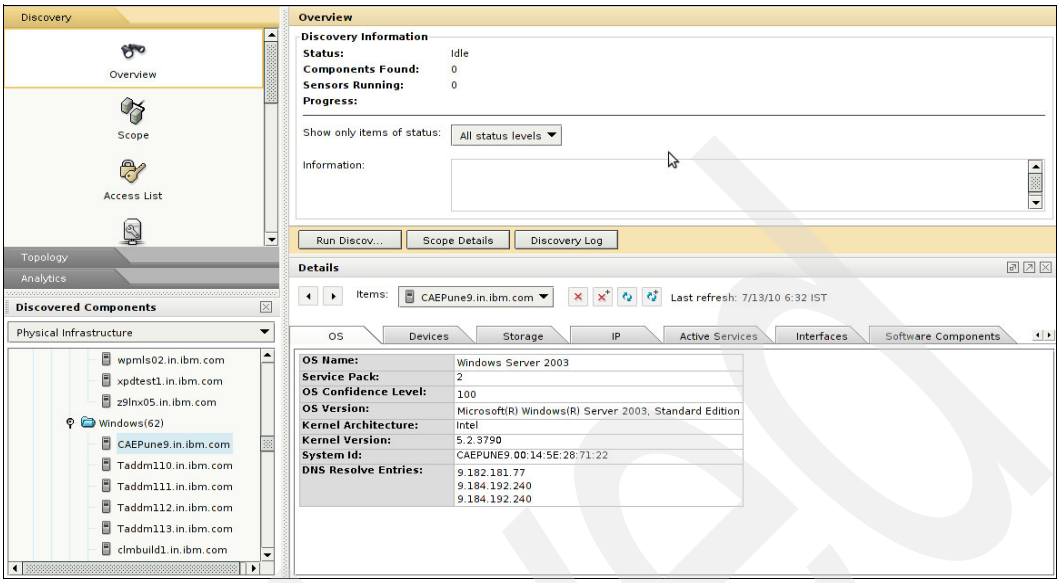


Figure 21-15 TADDM Discovered CI

Figure 21-16 shows TADDM L2 Discovery data for a sample server done without using server credentials; Figure 21-17 on page 659 shows the data discovered by TADDM using an Tivoli Monitoring Agent.

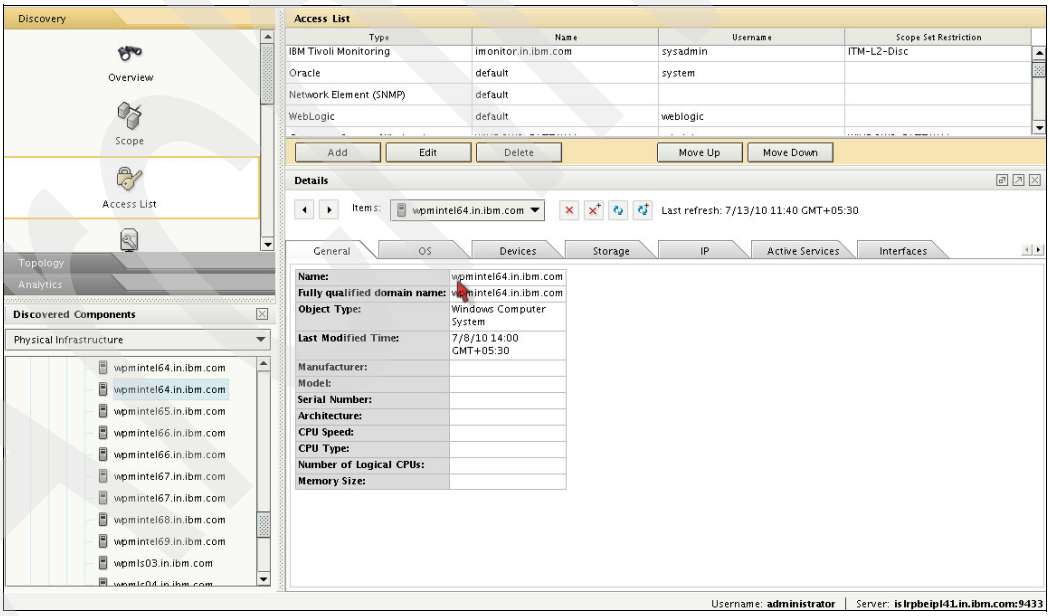


Figure 21-16 TADDM L1 discovery data for a sample server

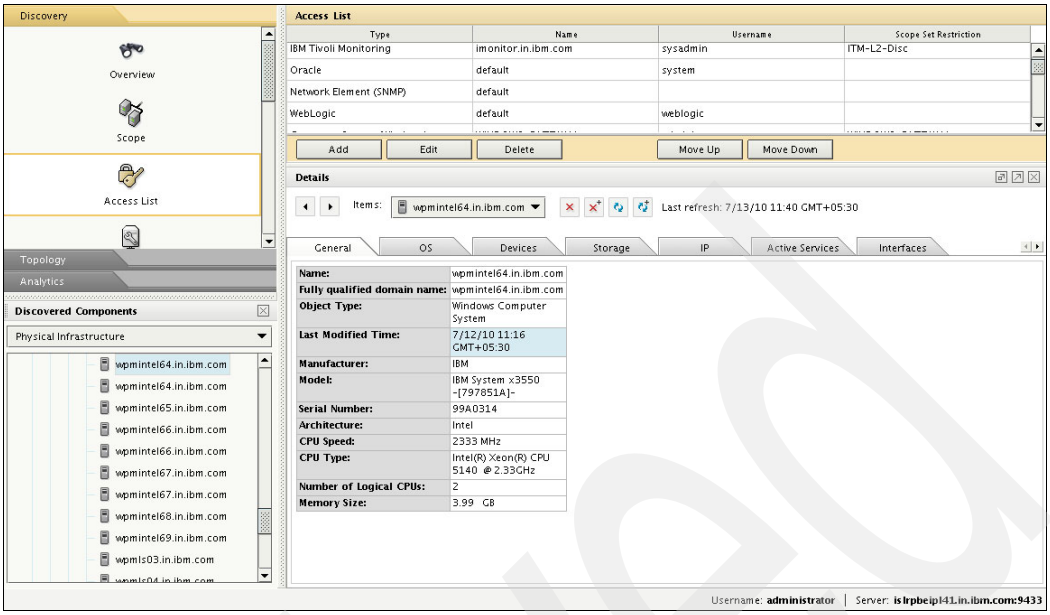


Figure 21-17 TADDM L2 discovery data using Tivoli Monitoring Agents for the sample server

Archived

21.7 Summary

Figure 21-18 shows a quick summary of the scenario.

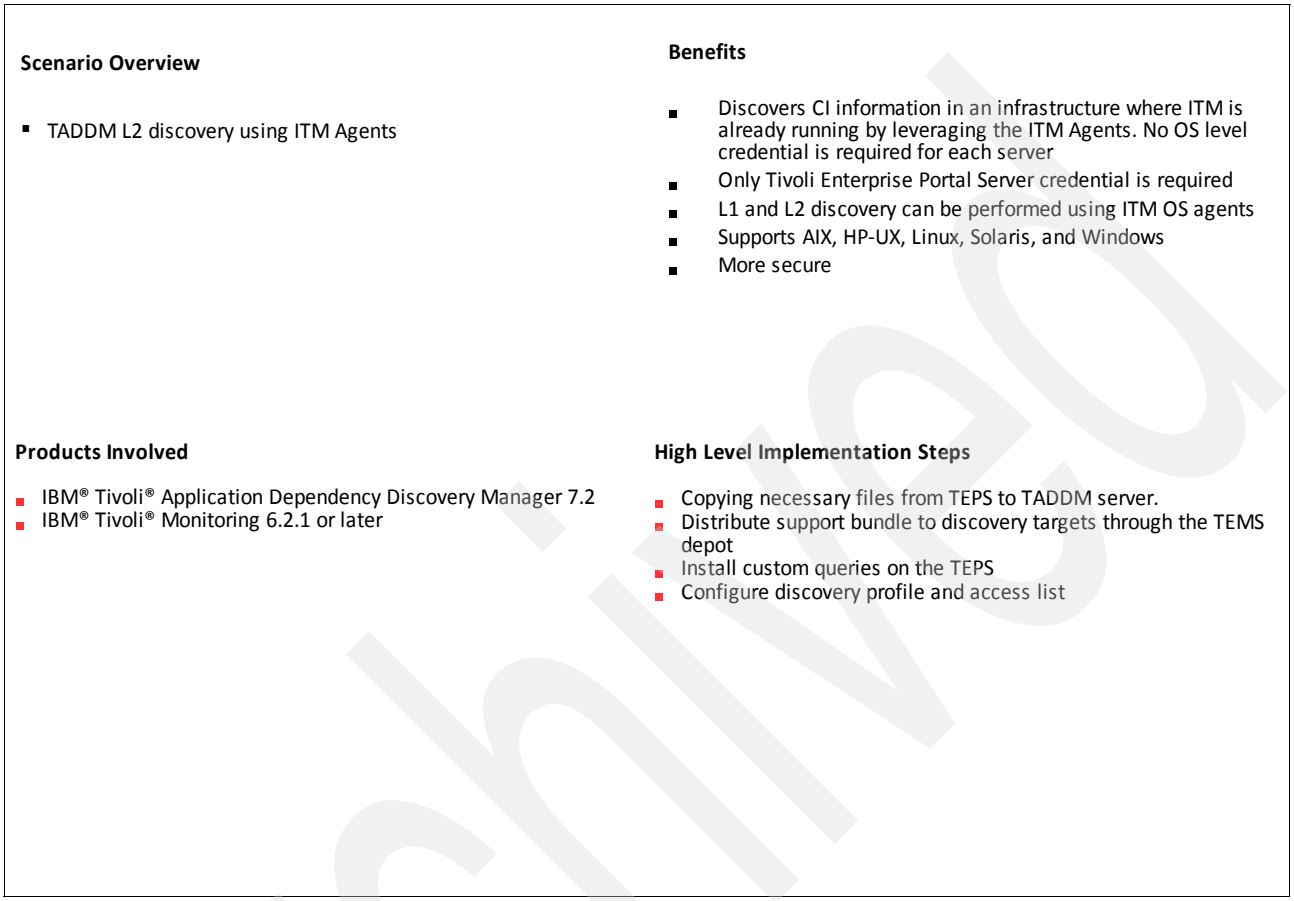


Figure 21-18 Quick summary

Archived

22

Managing business services in an integrated environment

This chapter demonstrates how an integrated environment that is implemented with Tivoli Application Dependency Discovery Manager (TADDM), CCMDB, and Tivoli Business Service Manager can improve the efficiency and effectiveness of a service operation. It also describes an additional integration with Tivoli System Automation Application Manager with Tivoli Business Service Manager, within the context of service management.

This chapter contains the following topics:

- 22.1, “Definitions” on page 662
- 22.2, “Scenario overview and products involved” on page 663
- 22.3, “Benefits” on page 664
- 22.4, “The Common Data Model” on page 665
- 22.5, “Integration and data transfer methods” on page 666
- 22.6, “Configuring the environment for integration” on page 667
- 22.7, “Additional integration: Integrating Tivoli System Automation Application Manager with Tivoli Business Service Manager” on page 695
- 22.8, “Summary” on page 697

© Copyright IBM Corp. 2011. All rights reserved.

22.1 Definitions

Before delving into the details of this scenario, first we define several terms and concepts that are used throughout this scenario.

22.1.1 What is a service

Defining a service may be done at various levels. A general definition can be obtained from the IT Infrastructure Library®.

Service definition: A *service* is a means of delivering value to customers by facilitating outcomes customers want to achieve without the ownership of specific costs and risks.

The value is achieved when a service provider seamlessly delivers an output, or in other words, a service, that is a necessary and acknowledged piece for the outcomes of the customers. As stated in the ITIL service definition, the providers are managing the elements that support the services and providing only the result, truly transferring to them the difficulties and risks of accomplishing it.

Thus, a service always consists of two parts: the *visible* part that is recognized by the customer, and the *encapsulated* part that is a series of elements that support it and that must be managed by the provider. By *manage*, we mean to operate and monitor the underlying infrastructure, and also to control the changes and configuration, and support problem and incident investigation, all from an integrated environment.

22.1.2 The service hierarchy

Depending on the target customer, services may be classified as *business* or *IT* services. Business services are entities that are recognized by the business as a *business enabler*. For example, a billing application can be recognized as a service by the business because it enables the billing business process. But in fact, the billing application is only the *visible* part of the service and many other elements such as hardware, backup, middleware, monitoring, firewall, and databases must be managed by the provider to support the business service. These technical elements form the *IT services* category and can be hierarchically positioned as the *supporting* elements of the business service.

Archive

Figure 22-1 shows a billing business service, supported by a hierarchy of IT services.

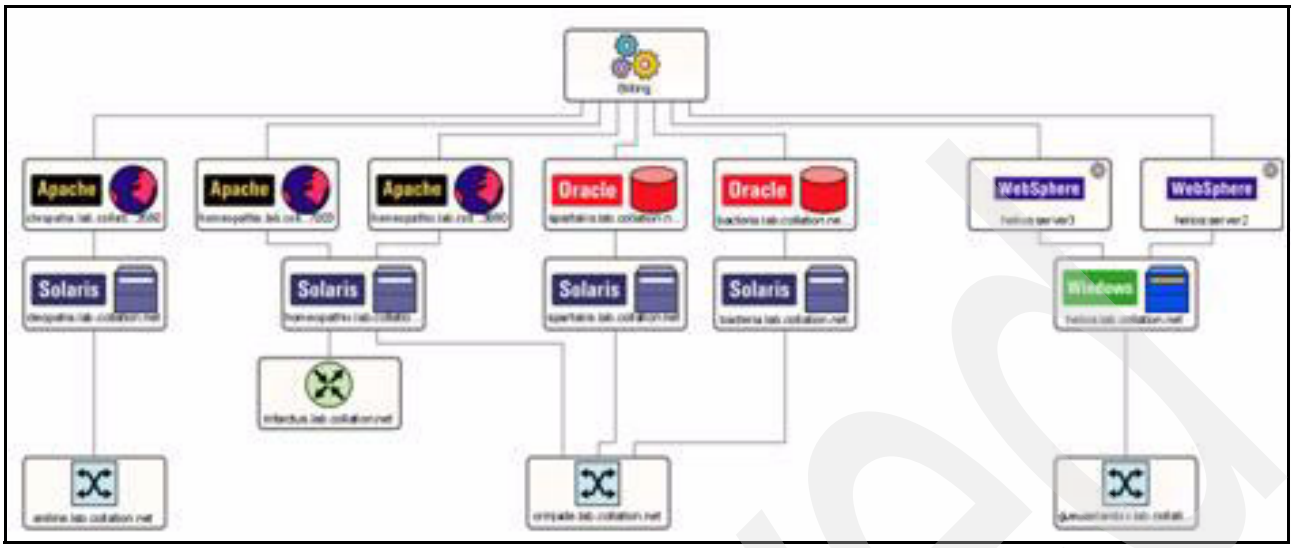


Figure 22-1 TADDM business service supported by a hierarchy of IT services

22.2 Scenario overview and products involved

The billing service, defined as a *business service* because it is a visible and recognized value element provided to the business by IT, must be managed to fulfill the business requirements. Table 22-1 shows the main usage of specific tools in the context of managing this service.

Table 22-1 Tools and their usage in the context of managing services

Objective	Related tool
Discover the IT components that support the business service and their dependencies	Tivoli Application Dependency Discovery Manager (TADDM) V7.2
Manage changes on the service configuration, avoiding unexpected outages and keeping it aligned with the business requirements	Change and Configuration Management Database (CCMDB) V7.2
Track the operational status of services and their components and take the necessary actions	Tivoli Business Service Manager V4.2.1

Archived

Figure 22-2 shows how the tools relate to each other.

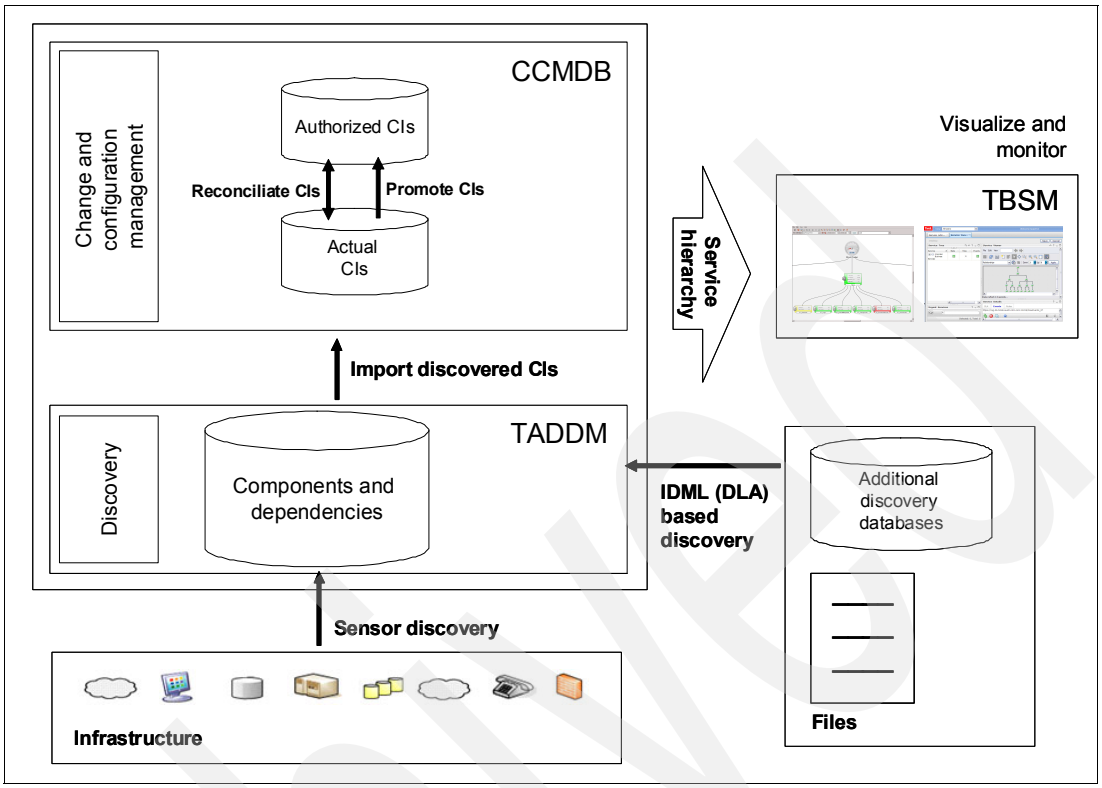


Figure 22-2 Functions and relations between tools

Assumptions: This chapter describes the integrations between the products in Table 22-1 but does not cover installation and methods to load components into TADDM. Thus, we assume that the products are installed and that a specific method, either sensor or Discovery Library Adapter (DLA) based, is already configured to load components into TADDM.

22.3 Benefits

The objective of the billing service scenario is to demonstrate how an integrated environment formed by TADDM, CCMDB and Tivoli Business Service Manager can improve the efficiency and effectiveness of service operation.

In this context, the following tasks are possible:

1. Define the service and its supporting infrastructure, through the discovery of its components.
2. Manage service outages through a proper visibility of the service operational status.
3. Manage service changes through the control of service configuration.

An integrated environment ensures that the necessary information is accessed at the right time, the right input is used to configure each tool and that, when needed, the tools are launched in the proper context.

Figure 22-3 shows that we have defined TADDM as the primary service definition source. This approach is a good practice because both Tivoli Business Service Manager and CCMDB will have the same consistent source for the service components and their relationships, using the Common Data Model (CDM) as the primary logical model. TADDM also generates *Globally Unique Identifiers (GUIDs)* that identify each component with a unique key through CCMDB and Tivoli Business Service Manager and enables the *launch-in-context* functionality. See 22.4, “The Common Data Model” on page 665 and 22.6.3, “Launch-in-context integration” on page 687 for more details.

Using this single service definition, each integration is responsible for a specific type of service management capability:

- TADDM - CCMDB integration controls the changes and promotes reconciliations between discovered and authorized components.
- TADDM - Tivoli Business Service Manager integration ensures that the monitored components are visualized and tracked as part of business services.

Both integrations together ensure that services are defined with an updated underlying infrastructure and that their changes are managed and outages are minimized. Figure 22-3 shows the integrated environment.

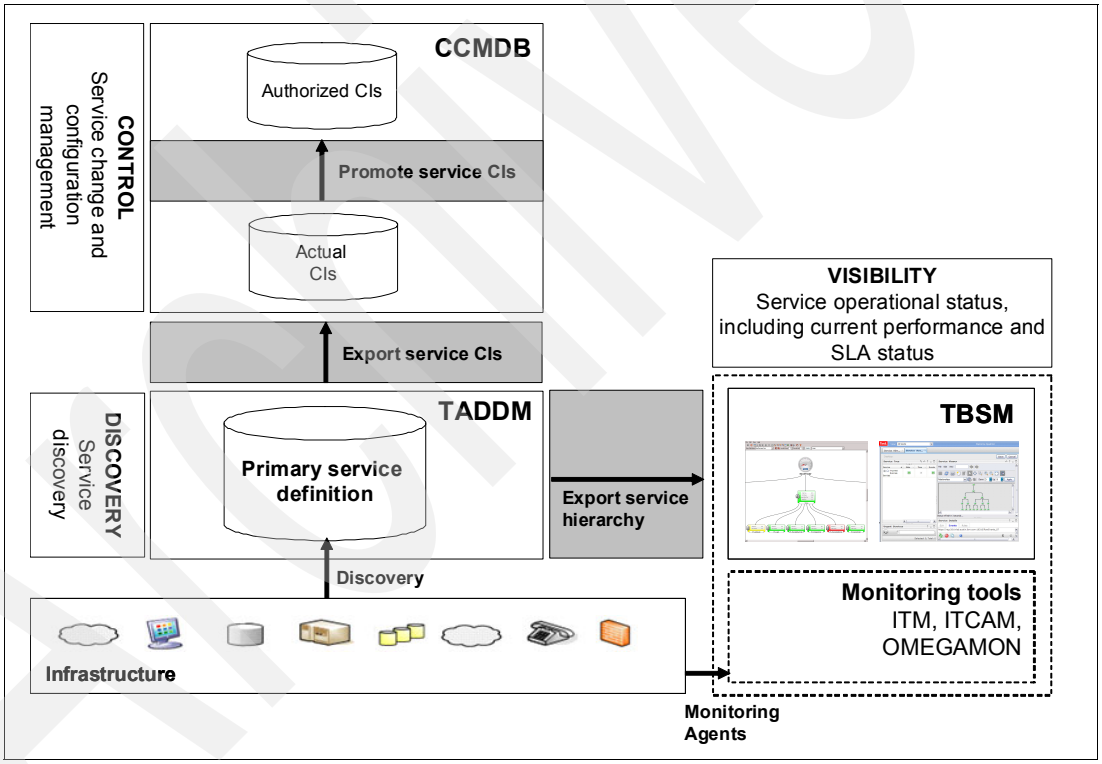


Figure 22-3 Primary source for service definition; how service is monitored and controlled

22.4 The Common Data Model

The Common Data Model (CDM) is a consistent, integrated, logical data model that defines the general characteristics of information communicated between our products. It is used as the model to integrate information from others sources into products such as TADDM and Tivoli Business Service Manager along with CCMDB. The model specifies how this data is organized to correspond to real-world entities and defines the relationships between the

entities. The CDM represents management information in a way that is easy for consuming management applications to use. TADDM discovers the components and their relationships and maps each one to a consistent CI and relationship CDM classification. Other products, such as Tivoli Monitoring, Tivoli Provisioning Manager, Tivoli Storage Productivity Center, Tivoli Network Manager all provide their information in the CDM format also. This information is shared with other products through TADDM such as Tivoli Provisioning Manager, Tivoli Asset Management for IT, CCMDB, and Tivoli Business Service Manager.

22.5 Integration and data transfer methods

Figure 22-4 shows the integration tools and methods between CCMDB, TADDM, and Tivoli Business Service Manager.

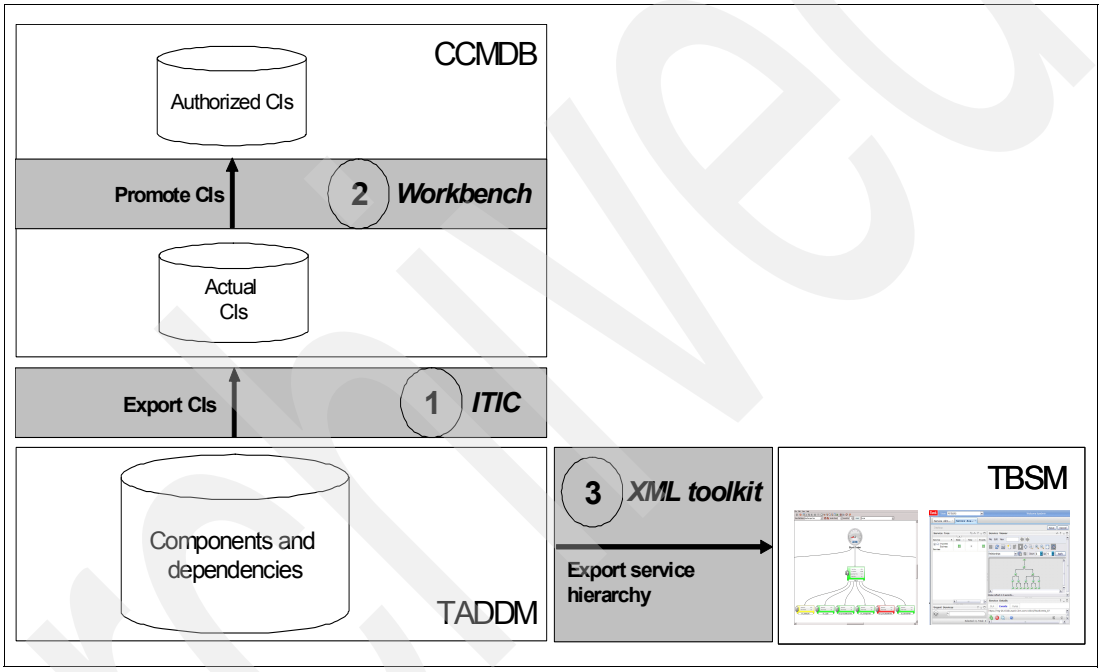


Figure 22-4 Integration tools and methods

The tools in the figure are as follows:

1. Tivoli Integration Composer

   Tivoli Integration Composer is an integration tool that you use to import data from a source to a target database. With Integration Composer, an enterprise can aggregate data collected by discovery tools and integrate it into a target database, creating a central repository for enterprise IT asset management, reporting, and decision support. It works with the following products:

   – IBM Tivoli Asset Management for IT
   – IBM Tivoli Change and Configuration Management Database (which includes TADDM)
   – IBM Tivoli Service Request Manager

2. CCMDB Deployer's Workbench

   Deployer's Workbench is a set of plug-ins for use within an Eclipse environment. These plug-ins provide a pluggable foundation for additional tooling for CCMDB. The CI space design tooling is used as a set of additional plug-ins for Deployer's Workbench to promote

the CIs from the CCMDB actual view to the authorized view. We use the CI space design tooling as a set of additional plug-ins for Deployer's Workbench to promote the CIs from the CCMDB actual view to the authorized view.

The authorized CI space consists of the following three basic pieces that provide CCMDB with the needed configuration to properly promote a CI from the actual space to the authorized space:

- Authorized Classification: A classification in CCMDB that has specific properties set that allow it to work with a CI.
- Promotion Scope (template): An object that contains a mapping of classifications between the actual and authorized classification spaces. This scope is the roadmap used by CCMDB to promote a CI from the actual to the authorized space.
- Relationship Rules: These rules define the relationships between the CIs in the authorized space. These rules must be set up correctly within the context of a promotion scope and the authorized classification set in order for CCMDB to properly promote a CI.

The CI space design tool provides a visual UI to help the user not only understand what the authorized CI space looks like, but also provides an intuitive drag and drop interface to help the user quickly build and edit their authorized CI space.

3. XML Toolkit

   The XML toolkit is used to populate Tivoli Business Service Manager with an inventory of resources from specific repositories using data in XML format. Additional discoveries (called *delta discoveries*) will then update or exclude components from the Tivoli Business Service Manager service hierarchy. The XML toolkit is part of the Tivoli Business Service Manager Base Services installation.

22.6 Configuring the environment for integration

We assume that an environment with a basic installation of Tivoli Integration Composer, TADDM, CCMDB, and Tivoli Business Service Manager is already in place. The following web addresses provide detailed guidance on product installation:

- Tivoli Integration Composer

  http://publib.boulder.ibm.com/infocenter/tivihelp/v32r1/index.jsp?topic=/com.ibm.ccmdb.doc/install/c_tamit_itic_installing.html

- CCMDB and TADDM

  http://publib.boulder.ibm.com/infocenter/tivihelp/v32r1/index.jsp?topic=/com.ibm.ccmdb.doc/install/c_install_intro.html

- Tivoli Business Service Manager

  http://publib.boulder.ibm.com/infocenter/tivihelp/v3r1/index.jsp?topic=/com.ibm.tivoli.itbsm.doc/install_guide_ic.html

We also assume that, as shown in Figure 22-3 on page 665, TADDM is the primary source for the service definition. In this way, we ensure that the most updated infrastructure components are used when tracking the service operational status in Tivoli Business Service Manager. Moreover, reconciliation tasks between the discovered CIs and the authorized view ensure that eventual gaps in change management process are detected and properly corrected.

Before setting up the integration, we must also ensure that the components that compose a service are defined in TADDM. There are many possible sources of integration, both

discovery and DLA based. The steps to configure discovery methods are not covered by this chapter. In our lab, we configured TADDM to discover the lab environment and used the discovered components and dependencies to define the underlying infrastructure of our service. Thus, our billing service consists of the following items:

- Two generic web servers
- One Apache server
- One Oracle database instance
- Two WebSphere application servers

22.6.1 Exporting CIs from TADDM to CCMDB

Perform the following steps to export CIs from TADDM to CCMDB:

1. If you are not running Integration Composer on the same machine as your CCMDB/Base, copy the SQL scripts that will create the mappings between TADDM and CCMDB databases, as follows:
   a. Copy `*.db2`, `*.ora`, `*.sqs`, and `*.schm` files (from the DVD or the extracted `.zip` file) to the `<ITIC root dir>\data\dataschema` location.
   b. Copy `*.fsn` file (from the DVD or the extracted `.zip` file) to the `<ITIC root dir>\data\mappings` location.
2. Configure a schema for TADDM CI Type:
   a. Log in to the Maximo database and run the following SQL scripts, which are stored in `<ITIC_root_directory>\data\dataschema`:

      ```
      createTADDM72ActualCIDataSchema.db2
      qualifierCCMDB72ActualCI.db2
      ```

      File extensions of the SQL scripts: In our example, we use a DB/2 database that has `.db2` as the file extension. When using other databases, the extension of the script files can vary. Oracle databases have the `.ora` file extension. This note applies to all SQL scripts that are shown in this chapter.

   b. Create a TADDM data source in IBM Tivoli Integration Composer (see Figure 22-5 on page 669):
      i. Start the Tivoli Integration Composer application, executing the following command:

         Windows system:

         ```
         <ITIC_root_directory>\bin\startFusion.bat
         ```

         Linux system:

         ```
         <ITIC_root_directory>\bin\startFusion.sh
         ```

      ii. Go to **Data Source** → **Define New Data Source**.

Archived

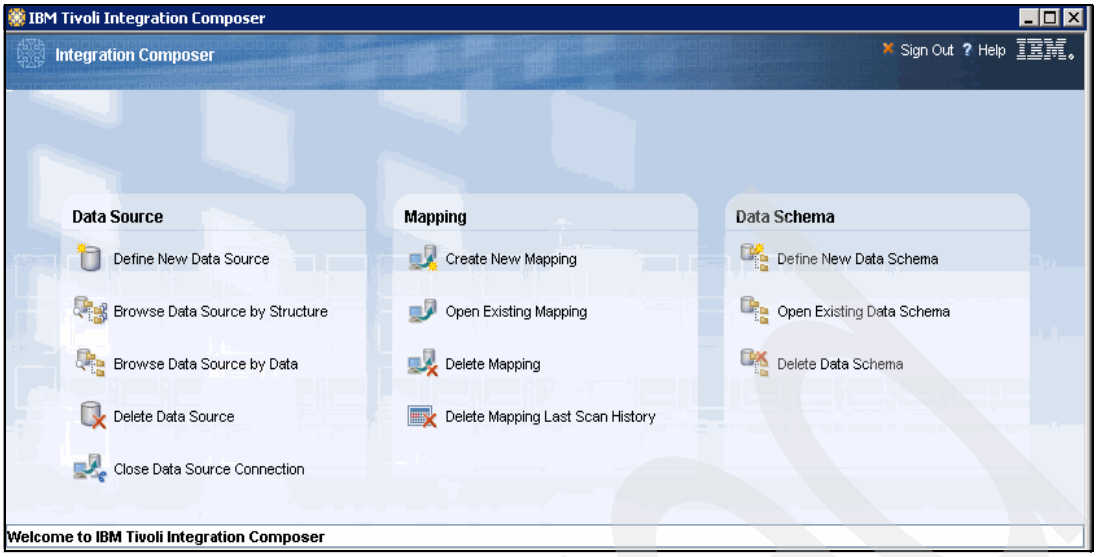


Figure 22-5 Tivoli Integration Composer main menu

iii. Select **Define New Data Schema** and type **CCMDB72Classification**. See Figure 22-6.

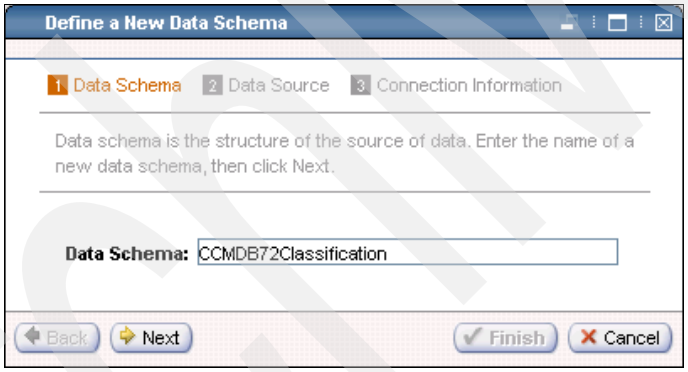


Figure 22-6 Defining a new data schema

iv. Type `CCMDB72Classification` for the name of the data source.

v. On the Connection Information page, select **IBM DB2 JDBC Driver** as the connection method**.** See Figure 22-7 on page 670. We assume that the standard DB2 database is being used.

vi. Type the host name for the TADDM server (not the database server). We use the IP address of our server: 10.100.100.84. Accept all other standard values.

vii. Click **Finish.**

Archive

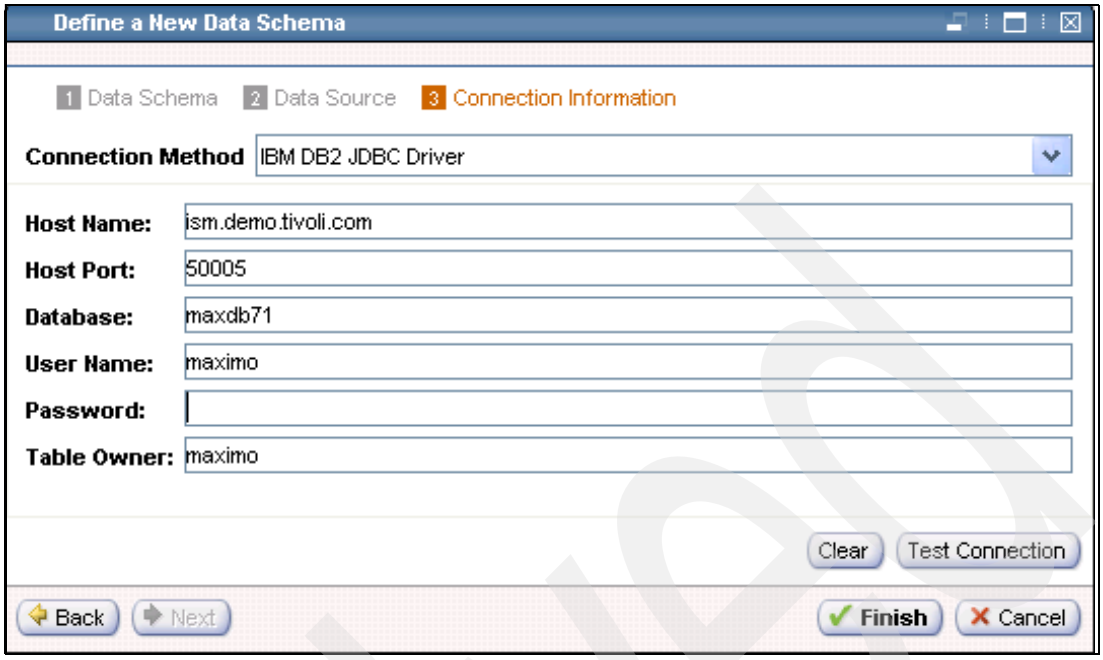


Figure 22-7 Connection information for the new data source

c. Go to **Select Action** → **Import Data Schema.** See Figure 22-8 and Figure 22-9.

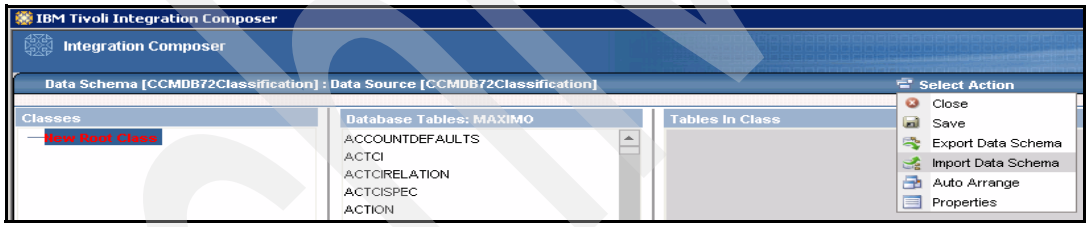


Figure 22-8 Importing the data schema.

d. Navigate to the directory where the Tivoli Integration Composer adapter files were extracted and select **CCMDB72Classification.schm**. See Figure 22-9.

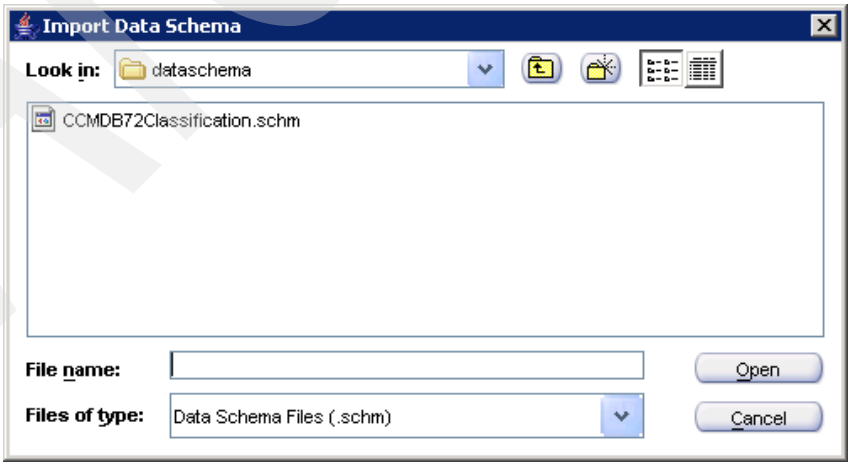


Figure 22-9 The .schm file

Archived

When the schema is imported, you normally see errors as a result of differences in data types. For example, big integer versus small integer versus integer. These data types have various lengths. Another typical reason for errors is a mismatch in the text case. See Figure 22-10.

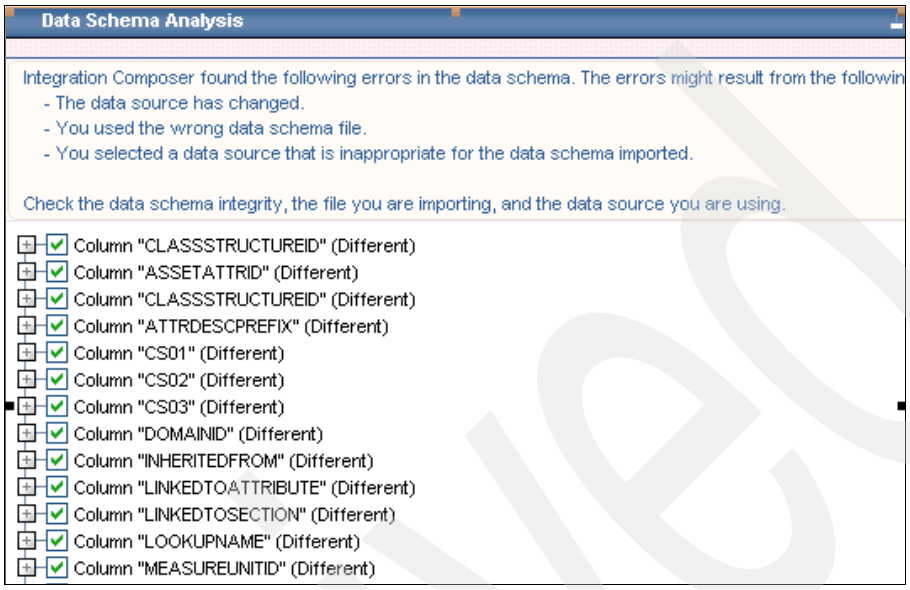


Figure 22-10 The data schema analysis

e. To see the details of each error, expand the tree of the analysis.

f. Click **Fix Errors**.

At this point, the Data Schema Analysis reports No Errors. See Figure 22-11.

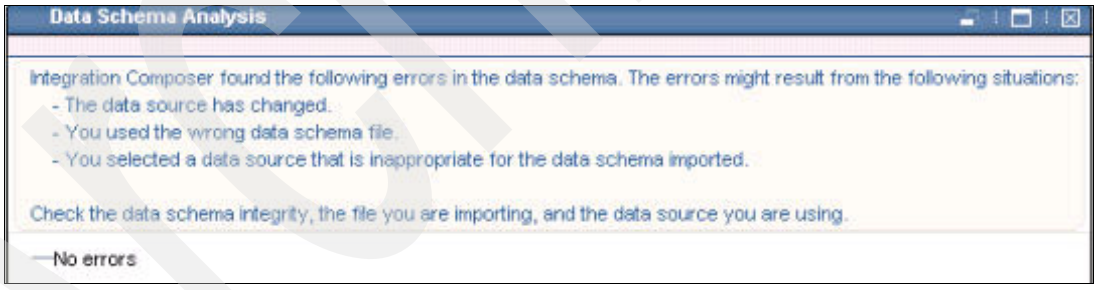


Figure 22-11 Final analysis report.

g. Save the Schema and close the window.

h. Go to your CCMDB database command prompt and run the following script:

```
qualifierCCMDB72Classification.db2
```

i. Select the Tivoli Integration Composer menu option **Define New Data Source**, choose **TADDM CI Type**, and click **Next**.

j. In the Data Source name field type `TADDMCIType`.

k. On the Connection information window, provide the configuration shown on Figure 22-12 on page 672. Change the host name for the IP address of your server.

About SSL: When using SSL, you must copy the `jssecacerts.cert` file from your TADDM server to your Tivoli Integration Composer machine. Use SSL for production environments but *not* for demonstration environments. The steps are as follows:

1. From the server where Integration Composer is installed, point a web browser to the TADDM server at port 9430, for example:

   `http://hostname:9430`

2. Make sure the status of all TADDM components is green, which means started.
3. Click **Show SSL Options**.
4. Click **Download Trust Store**.
5. Save the `jssecacerts.cert` file in the `etc` folder in the Integration Composer installation directory.

Be sure your TADDM server is running and you can log in to the TADDM console.

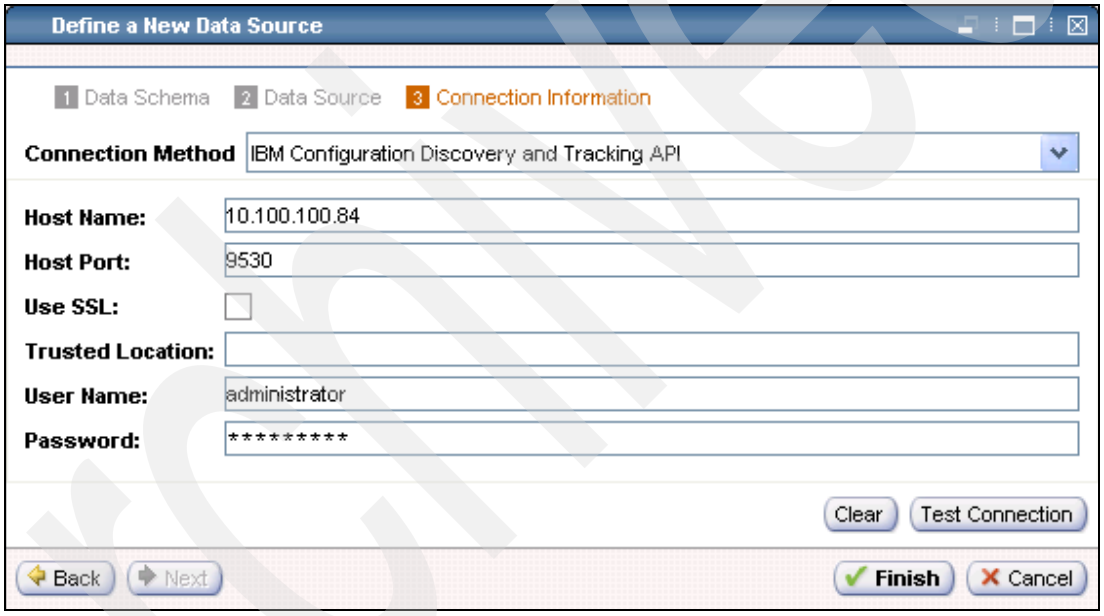


Figure 22-12 Configuring the TADDM data source

l. On Tivoli Integration Composer main menu, go to **Create New Mapping**. Select the TADDM data source you just created as the source and select your new CCMDB data source as the target. See Figure 22-13.

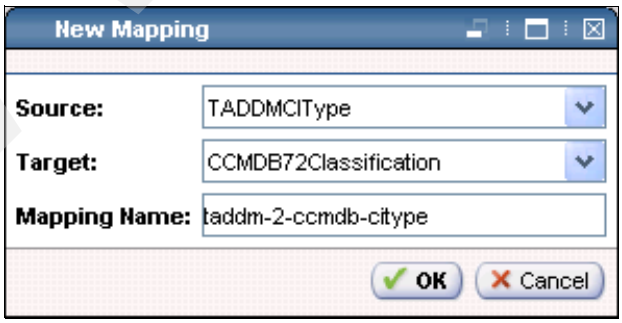


Figure 22-13 Source and target mapping

m. When you sign into the data sources, the window, shown in Figure 22-14 opens. Click **Select Action** → **Import**.

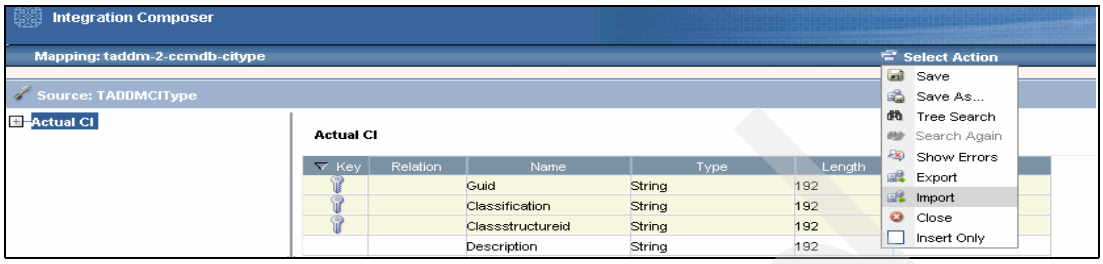


Figure 22-14 Importing the mapping

n. Access the directory where the adapter files were extracted to.

o. Select the `TADDM72CITypeToCCMDB72Classification.fsn` file. See Figure 22-15.

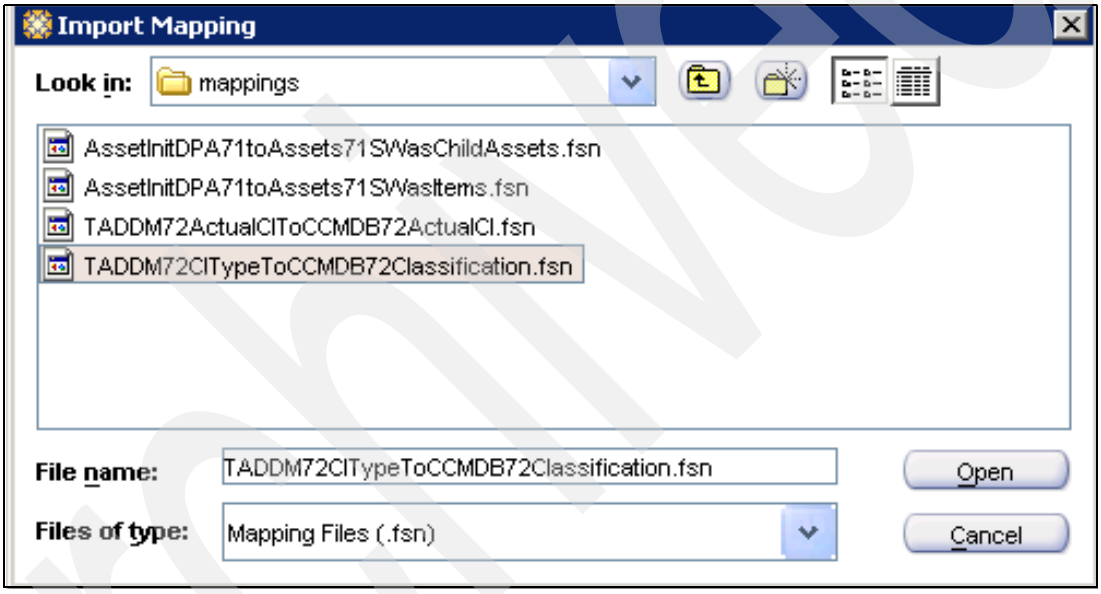


Figure 22-15 Mapping file between TADDM and CCMDB

p. Save the mapping and close Tivoli Integration Composer.

6. Configure the mapping for the actual CI adapter

a. Go back to your CCMDB database command prompt and run the following script:

```
createTADDM72ActualCIDataSchema.db2
```

b. Open Tivoli Integration Composer and select **Define new Data Schema**. Type `CCMDB72ActualCI` as the data schema and data source names. This step defines the *target* schema that Tivoli Integration Composer will use to receive the imported data.

Archived

c. Provide the same database connection information that was previously entered. Note that the connection method is determined by the type of the CCMDB database. Click **Finish**. See Figure 22-16.

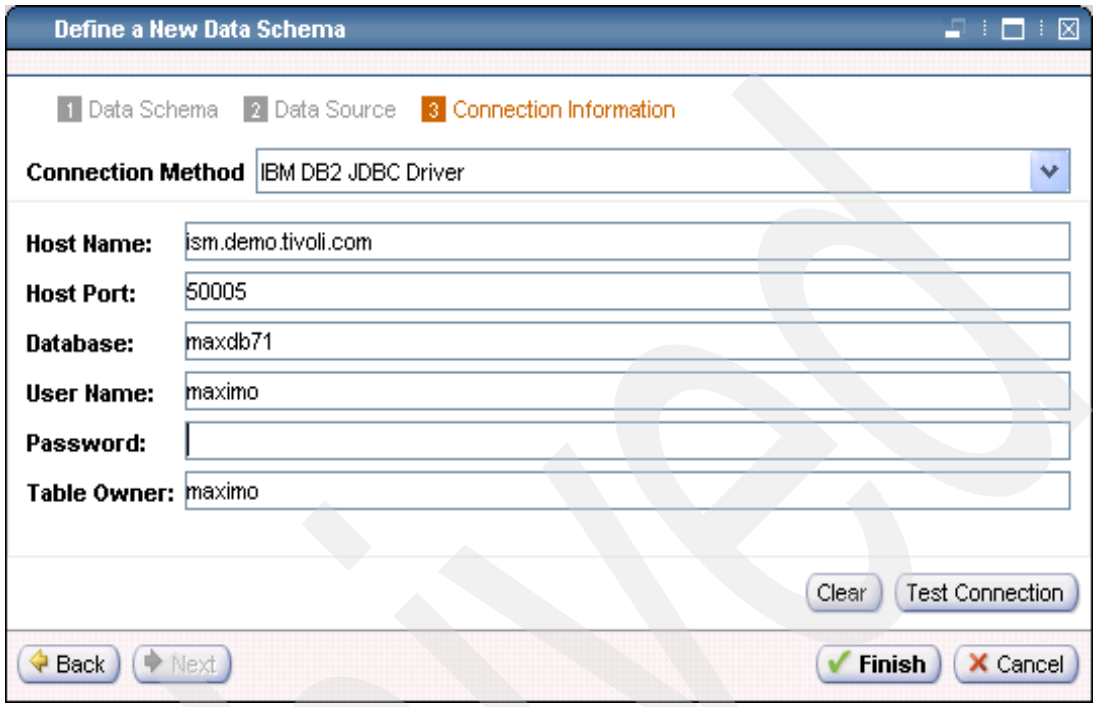


Figure 22-16 Database connection information

d. Go to **Select Action** → **Import Data Schema**.

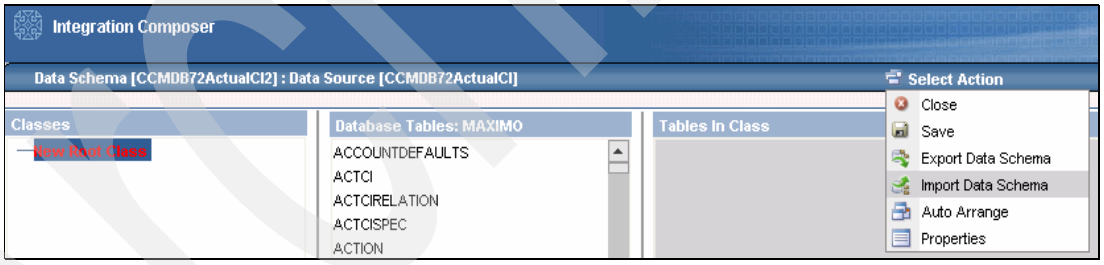


Figure 22-17 Importing the data schema

Archived

e. Access the Tivoli Integration Composer adapter directory and select the `CCMDB72ActualCI.schm` file. See Figure 22-18.

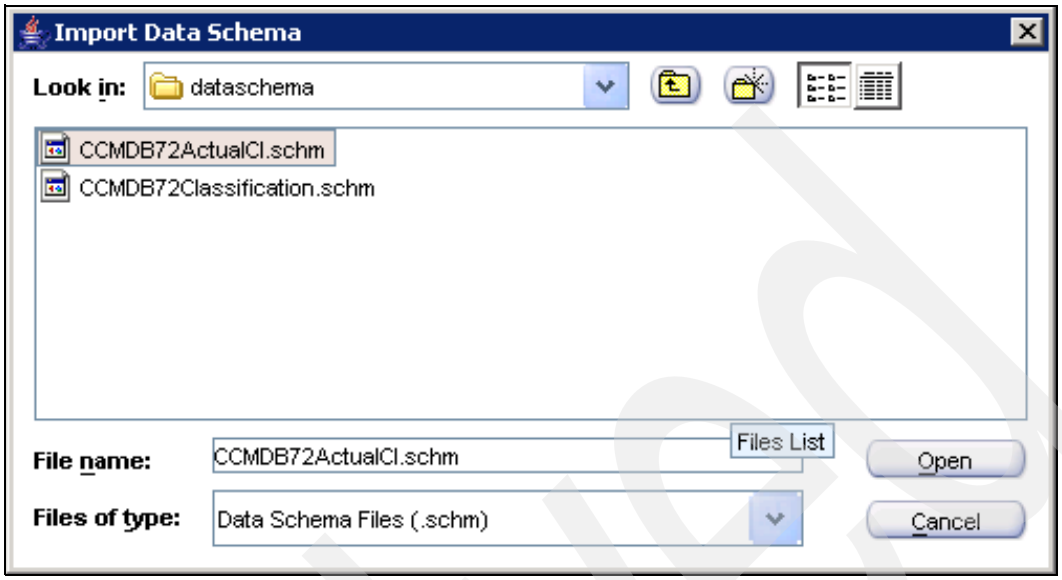


Figure 22-18 Data schema file

f. As with the previous import, you may see errors as a result of differences in data types. Click **Fix errors** if any error occurs. Figure 22-19 shows an example.

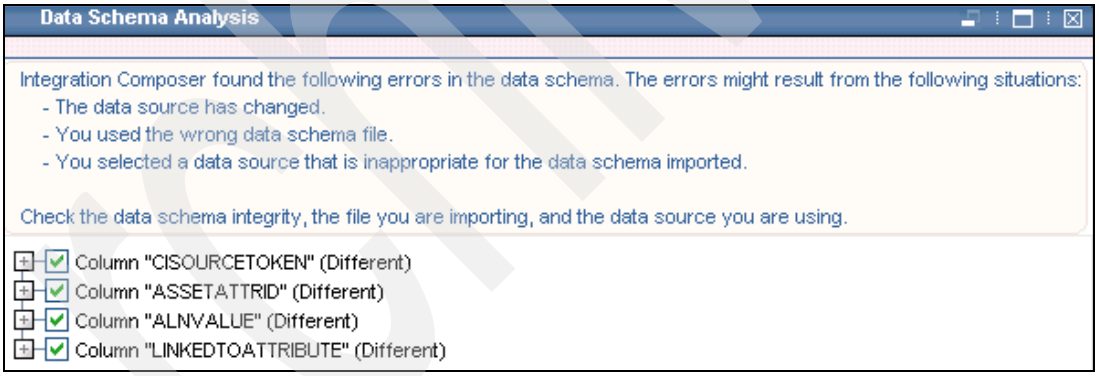


Figure 22-19 The data schema analysis

g. Save the schema and close the dialog.

h. On the Tivoli Integration Composer main menu, go to **Define New Data Source**, choose **TADDM 7.2 Actual CI**, and click **Next**.

i. Provide a name such as `taddm72-actual-ci`.

j. Provide connection parameters for the TADDM server (the data source). See Figure 22-20 on page 676 for details. Save and close the new data source.

Archived

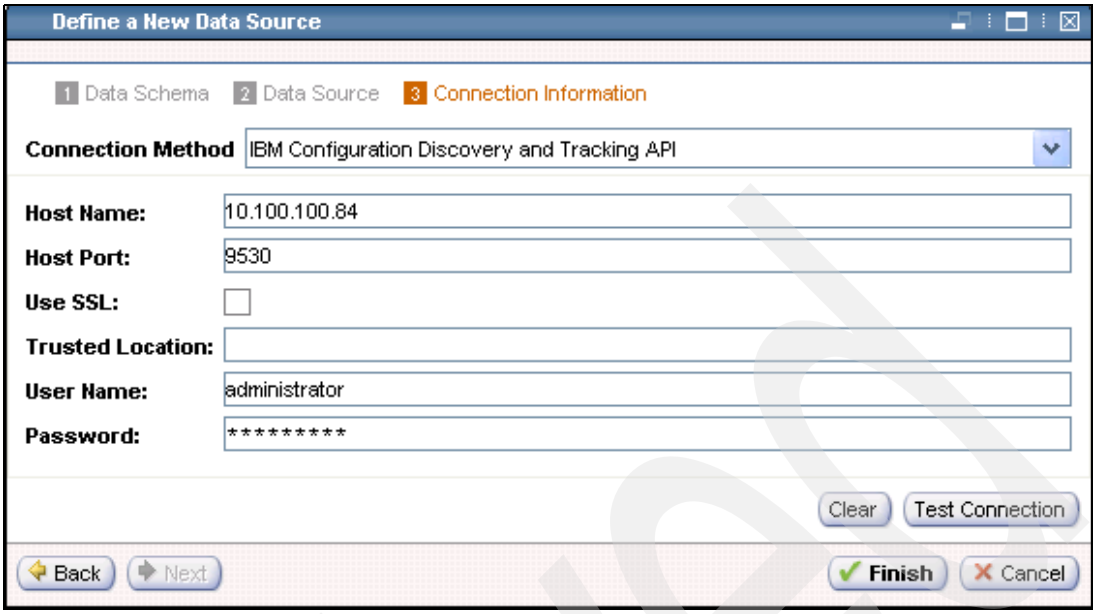


Figure 22-20 TADDM connection parameters

k. On the Tivoli Integration Composer main menu, go to **Create New Mapping**. Select the source and target as shown in Figure 22-21.

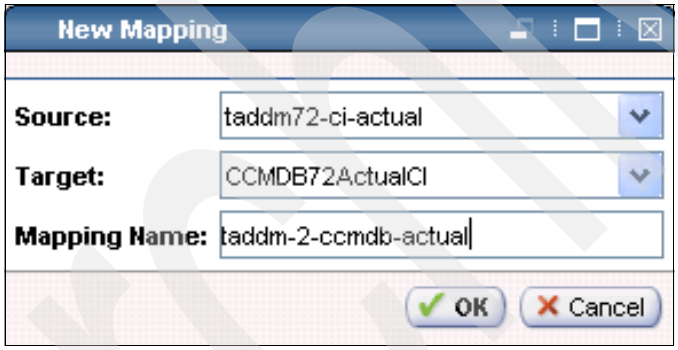


Figure 22-21 Source and target for the data mapping

l. Sign into both data sources. The window shown in Figure 22-22 opens. Click **Select Action** → **Import**.

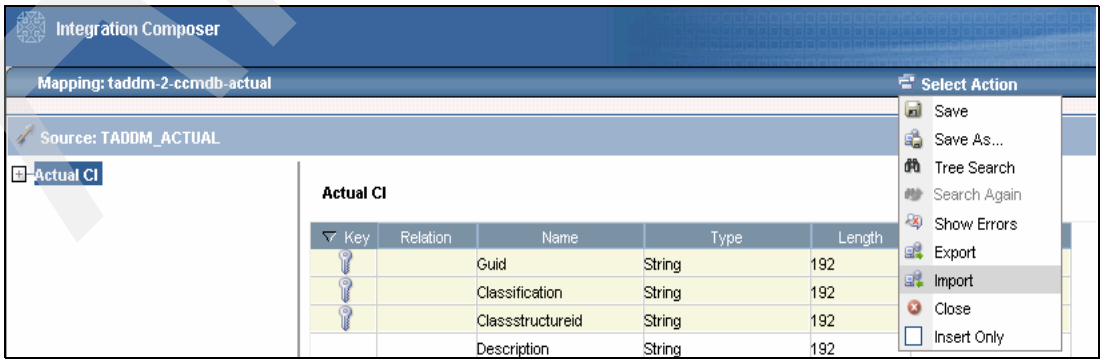


Figure 22-22 Importing the mapping between TADDM and CCMDB

Archived

m. Access the adapter files adapter and select the `TADDM72ActualToCCMDBActualCI.fsn` file. See Figure 22-23.

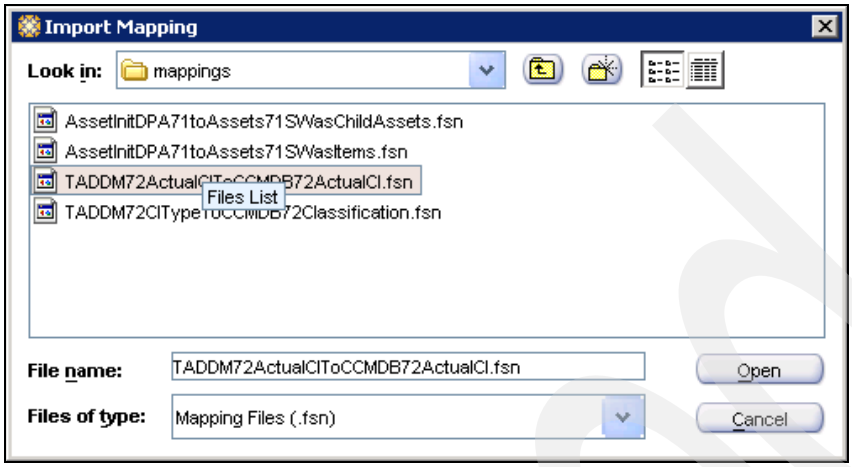


Figure 22-23 The mapping file import

n. Save the **Mapping** and close the dialog window.

7. Map the execution for the CI Types adapter:

a. Open a command prompt window and move to the integration composer `bin` directory, as follows:

```
C:\>cd Integration composer\bin
```

Figure 22-24 shows the contents of the directory.

```
C:\Integration Composer\bin>dir
 Volume in drive C has no label.
 Volume Serial Number is AC61-E006

 Directory of C:\Integration Composer\bin

07/16/2010  10:57 PM    <DIR>          .
07/16/2010  10:57 PM    <DIR>          ..
06/24/2010  04:27 PM                94 class.bat
06/24/2010  04:59 PM             2,860 commandLine.bat
06/24/2010  03:44 PM             2,426 init.bat
06/24/2010  03:28 PM               339 startFusion.bat
06/24/2010  03:27 PM             2,238 startFusion.ico
               5 File(s)          7,957 bytes
               2 Dir(s)  110,331,256,832 bytes free

C:\Integration Composer\bin>_
```

Figure 22-24 Directory contents

b. Run **`commandLine.bat`** (import engine) without any parameters to retrieve the usage information. See Figure 22-25 on page 678 for details.

Archived

```
C:\Integration Composer\bin>commandLine.bat
Integration Composer will now start.
The [mappingName] was not specified.

Usage:
EngineMain [-delete] mappingName repositoryUser repositoryPwd sourceUser sourceP
wd targetUser targetPwd
where:
  -delete         - (optional) Delete mapping last scan history prior to mapping
execution
  mappingName     - The name of the predefined mapping; i.e., SMS 2.0
  repositoryUser  - The login user name of the Integration Composer repository
  repositoryPwd   - The login password of the Integration Composer repository
  sourceUser      - The login user name of the source data source
  sourcePwd       - The login password of the source data source
  targetUser      - The login user name of the target data source
  targetPwd       - The login password of the target data source
```

Figure 22-25 Running the commandLine.bat without parameters

c. Using the parameters of your environment, run the fully qualified **commandLine.bat** command to start importing CI types from TADDM into CCMDB, for example:

```
commandLine.bat taddm-2-ccmdb-citype maximo your_password administrator
collation maximo your_password
```

d. Wait for the completion of the import process. It may take several minutes depending on the amount of data.

8. Activate CI types in CCMDB:

a. Log in to CCMDB application as maxadmin.

b. Go to **Administration** → **CI types**. See Figure 22-26.

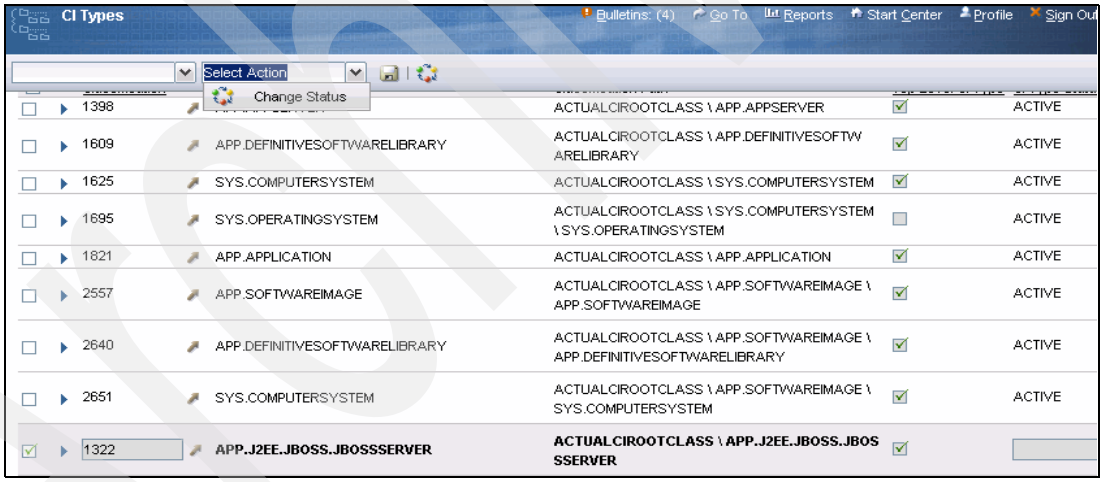


Figure 22-26 The CI types list

c. Select the classifications that will be used when importing CIs from TADDM. The discovery process automatically associates specific Common Data Model (CDM) classes to each component and classifications reflect the classes that apply to the listed components. Only the CIs that apply to the selected classifications will be managed by the application.

Archived

Note: In our example, we excluded the SYS.OPERATING SYSTEM classification and included the APP.APPLICATION. *That means that we are not going to manage the operating systems as CIs, but we do want to manage the applications as CIs.* This definition varies depending on the management and control needs of each organization and is usually part of their configuration management policies. More details about CDM and data models are in *IBM Tivoli Common Data Model: Guide to Best Practices,* REDP-4389.

d. Click **Select Action** → **Change Status**. See Figure 22-26 on page 678.

Note: The first time this action is performed, a warning message indicates that a one-time process is being invoked in the background. This process generally takes 10 - 20 minutes to complete.

9. Map the execution for the CI Actual Adapter:

   a. Open a command prompt window and go to the Integration Composer `bin` directory:

   ```
   C:\>cd Integration composer\bin
   ```

   Figure 22-27 shows an example of the contents of the directory.

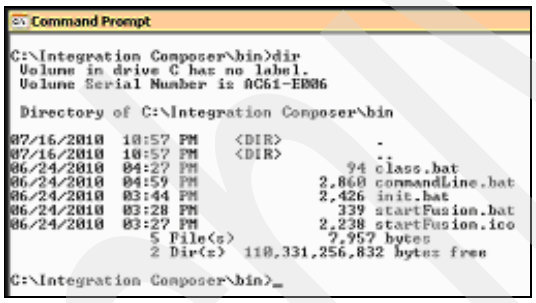


Figure 22-27 The integration composer directory

   b. Run **`commandLine.bat`** command (import engine) without any parameters to retrieve the usage information.

   c. Run the fully qualified **`commandLine.bat`** command to start importing the CIs from TADDM to CCMDB. The “mappingName” must be the same name that was set for Actual CI Adapter, for example:

   ```
   commandLine.bat taddm-2-ccmdb-actual maximo your_password administrator
   collation maximo your_password
   ```

Note: You can create a script (`.bat` or `.sh` for UNIX systems) to automate the process to import the classification.

When the process completes, the window is similar to Figure 22-28.

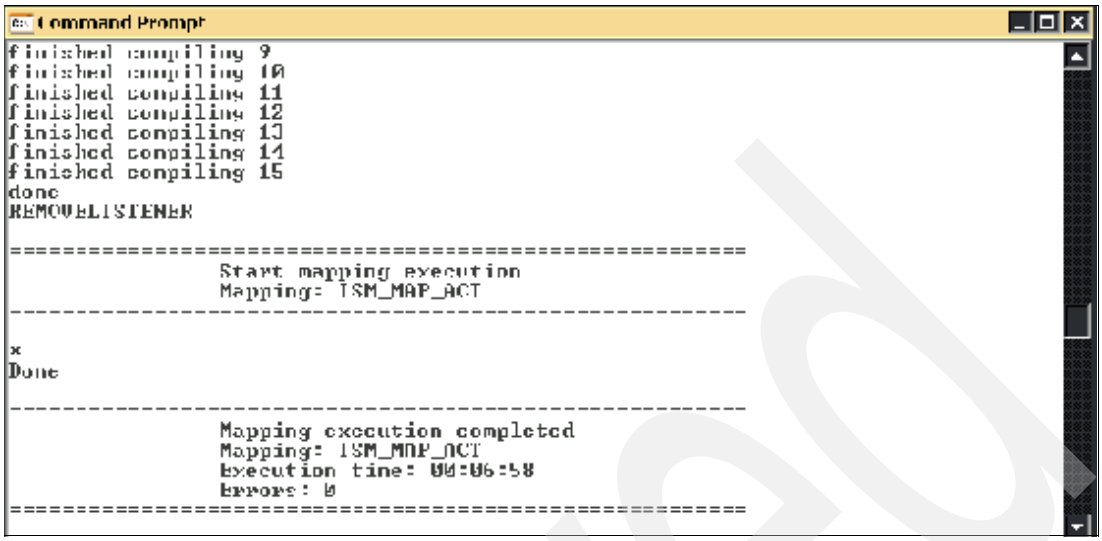


Figure 22-28 The completion of the mapping between TADDM and CCMDB

Notes:

- Be sure to schedule the mapping execution to keep CCMDB data in sync with your discovered data in TADDM.
- The authorized CI view of the CCMDB must also be configured and linked to the actual view. This chapter does not cover the authorized view configuration, only the integration between TADDM and CCMDB actual view. Detailed information about the creation of authorized CI classifications, relationships, and attributes is at the following address:

  http://publib.boulder.ibm.com/infocenter/tivihelp/v32r1/topic/com.ibm.ccmdb.doc/config/t_create_authorized_cis.html

Archived

22.6.2 Exporting the service definition from TADDM

After installing Tivoli Business Service Manager, transferring the service definition from TADDM is possible. To complete this transfer, the XML toolkit must be installed. The following web page contains detailed installation instructions:

http://publib.boulder.ibm.com/infocenter/tivihelp/v3r1/index.jsp?topic=/com.ibm.tivoli.itbsm.doc/installguide/bsmi_t_installing_discovery_library_toolkit.html

During installation, be sure to input specific configurations for the integration. The following steps show the necessary installation parameters:

1. Input the initial installation parameters as shown on Figure 22-29. If needed, adjust the Tivoli Business Service Manager data server host name. If a DNS is not properly configured, an IP address must be provided. Click **Next**.

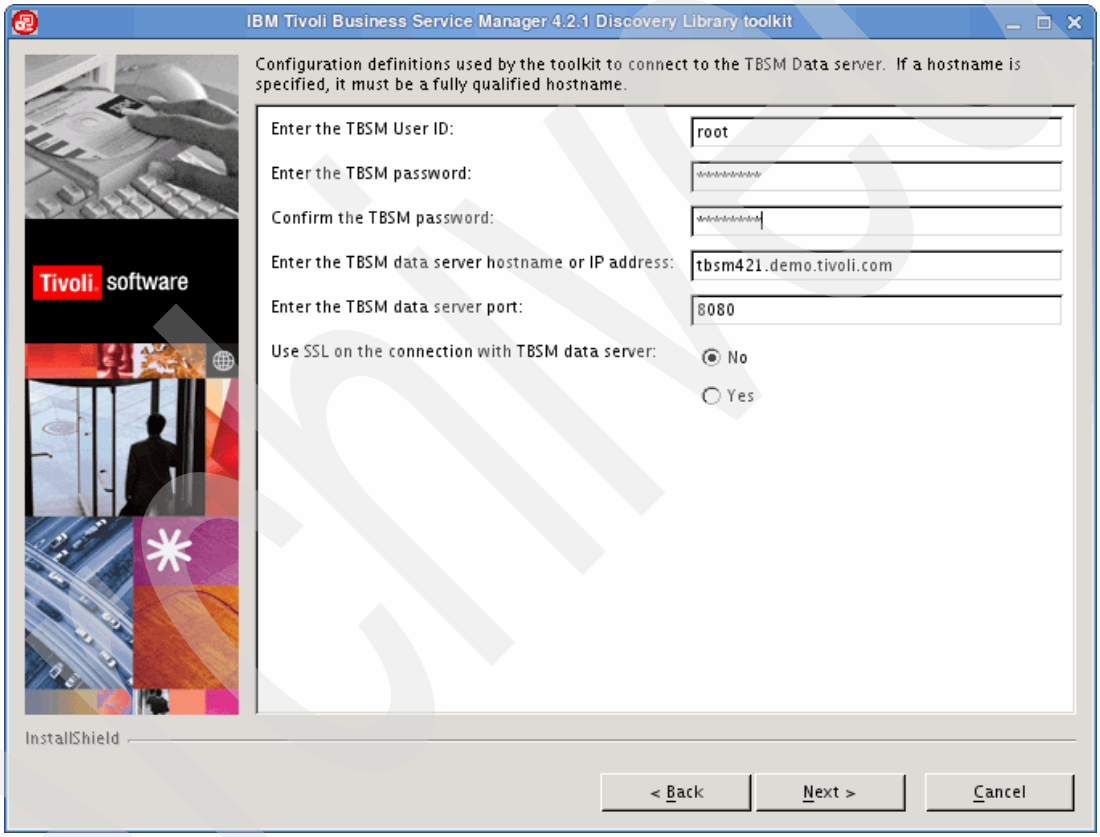


Figure 22-29 Initial installation window

2. Select **TADDM** as your data source and **PostgreSQL** as the database. PostgreSQL is the standard database used by Tivoli Business Service Manager to store the service hierarchy.

Figure 22-30 shows an example. Click **Next**.

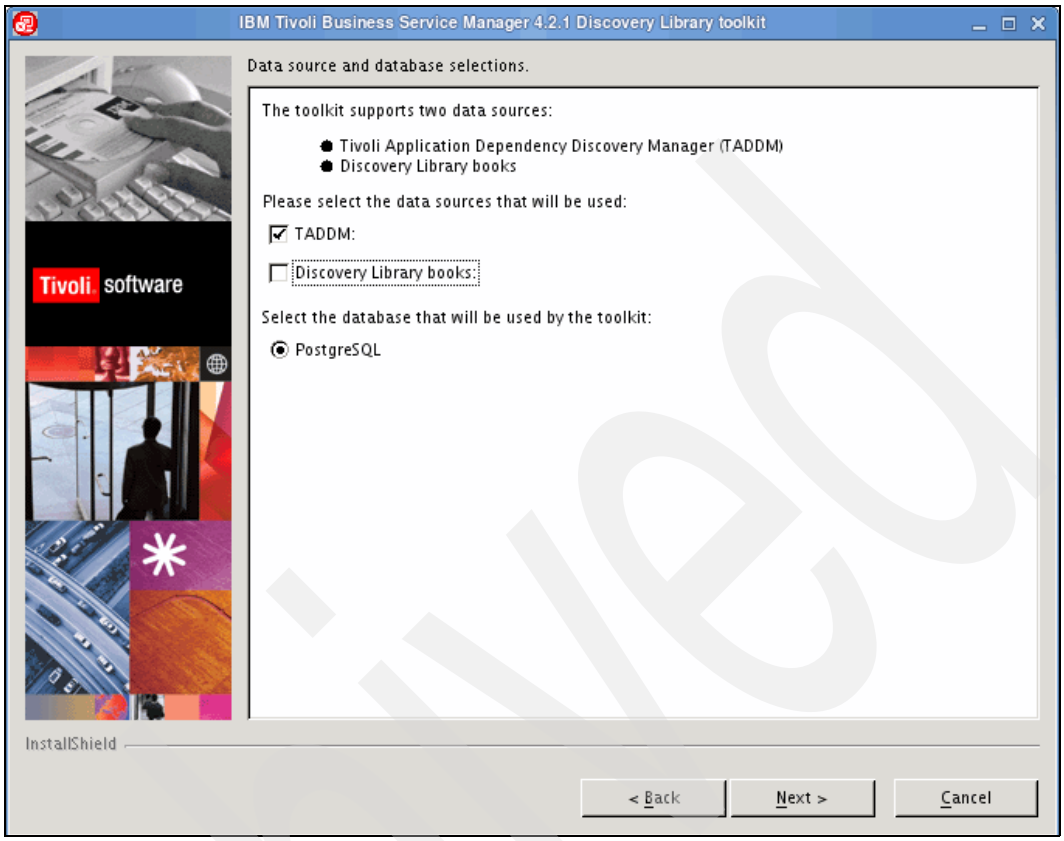


Figure 22-30 The selection of the XML toolkit data source

Archived

3. On the next window, input the TADDM connection configurations. See Figure 22-31. Click **Next**.

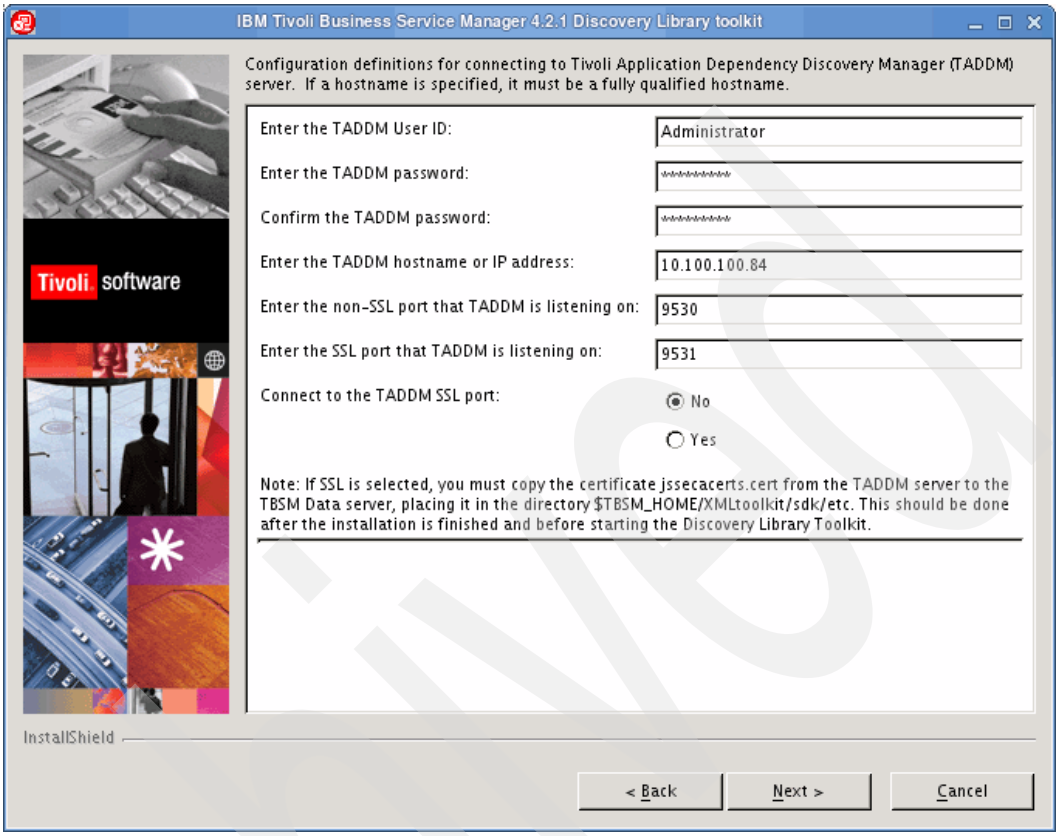


Figure 22-31 TADDM connection configuration

Archive

4. Input the TADDM database configurations. We are configuring a DB2 database in our example. See Figure 22-32. Click **Next**.

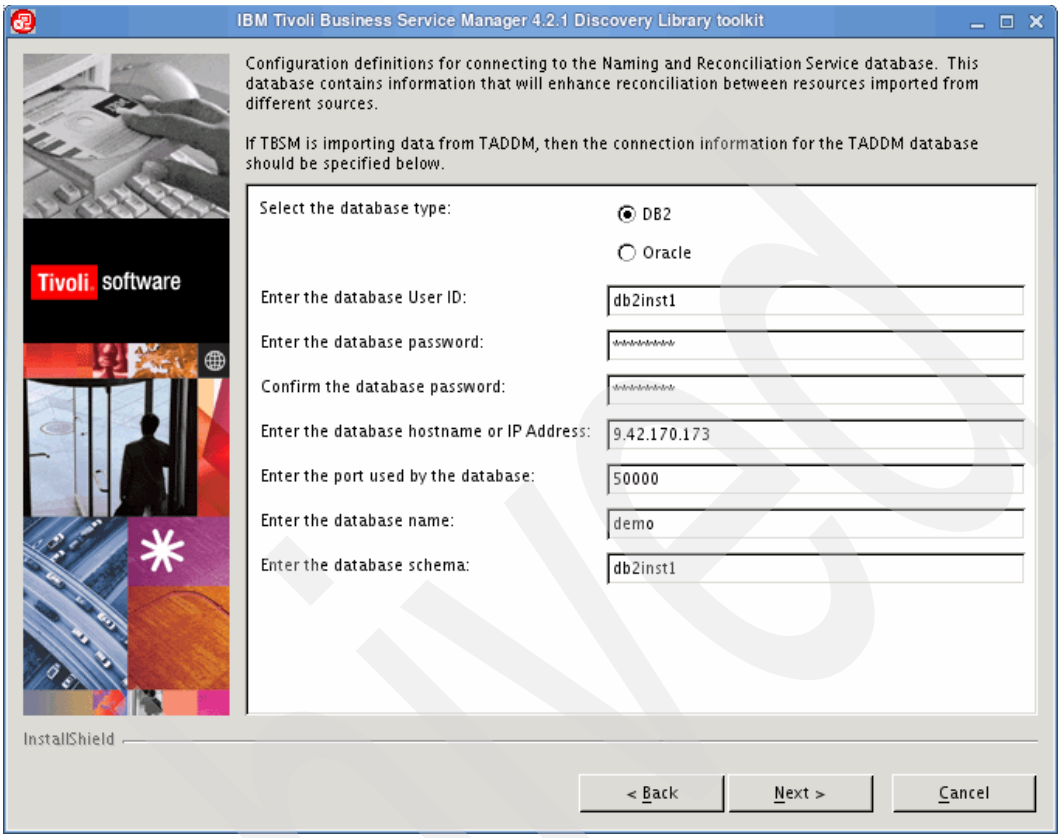


Figure 22-32 TADDM database configurations

Archived

5. Define a directory to store the data that comes from TADDM and input a Tivoli Integrated Portal dashboard host name. If Tivoli Integrated Portal is not configured, you can use Tivoli Business Service Manager own front end. See Figure 22-33. Click **Next**.

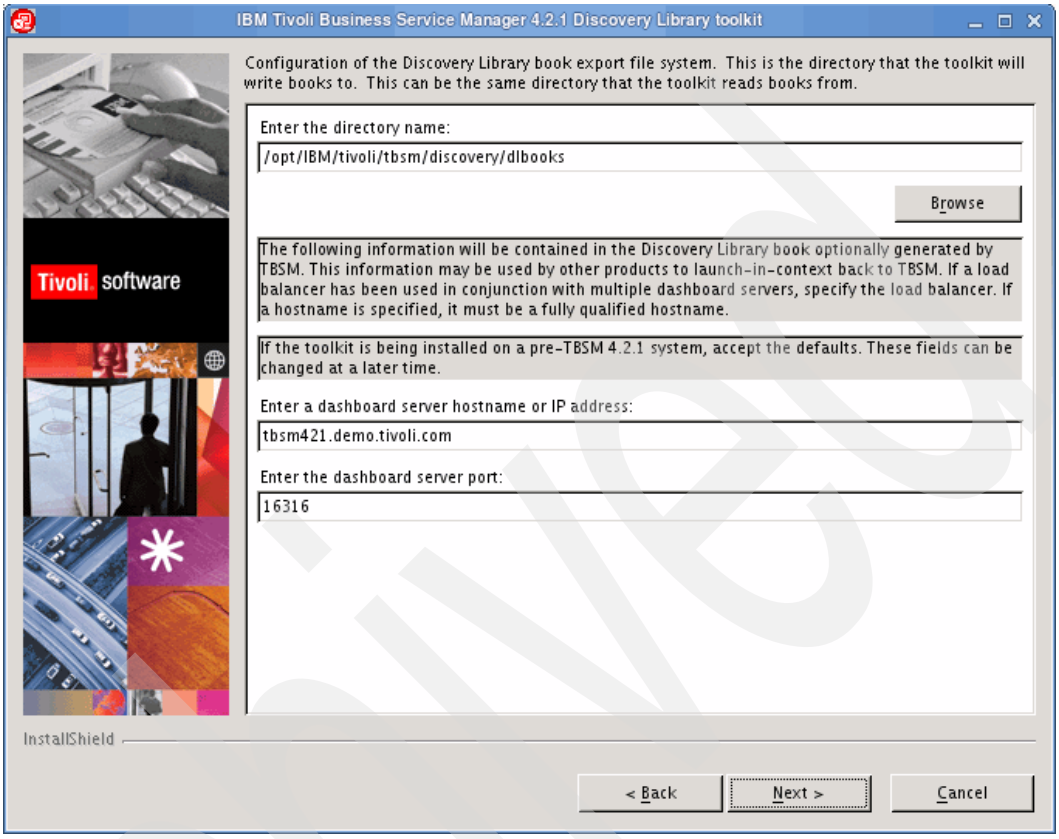


Figure 22-33 XML toolkit directory and TIP host name

6. Open the following file by using a text editor:

   `$TBSM_HOME/XMLToolkit/bin/xmltoolkitsvc.properties`

7. Look for the **DL_TADDM_Hostname** line.
8. Complete the attribute with the IP address of your TADDM server. Figure 22-34 on page 686 shows the details.
9. Save the file.

Archive

```
# will instead wait DL_PollTaddmIntervalSeconds before making its
# first request for data.
#
# Default value is true
# ------------------------------------------------------------------
DL_TADDM_Immediate_Poll=true
#
# ------------------------------------------------------------------
# The TADDM that the service will connect to if the property
# DL_TADDM_Connect is set to true.
# ------------------------------------------------------------------
DL_TADDM_HostName =10.100.100.84
#
# ------------------------------------------------------------------
# The TADDM port. The default is 9530.
# ------------------------------------------------------------------
DL_TADDM_Port =9530
#
# ------------------------------------------------------------------
```

Figure 22-34 The DL_TADDM_HostName parameter

10. Load the data from TADDM to Tivoli Business Service Manager:

    a. On the command prompt, execute the following commands:

    ```
    ~ cd $TBSM_HOME/XMLtoolkit/bin
    ~ $ ./tbsmrdr_start.sh
    ```

    b. After loading TADDM data for the first time, XML toolkit checks for TADDM updates every 60 minutes, by default. If you prefer, stop this process by executing the following commands:

    ```
    ~ cd $TBSM_HOME/XMLtoolkit/bin
    ~ $ ./tbsmrdr_stop.sh
    ```

    c. After the initial load, Tivoli Business Service Manager processes the TADDM data and a service hierarchy is configured. See Figure 22-35 for an example.



Figure 22-35 The service hierarchy directory

Archive

22.6.3 Launch-in-context integration

The launch-in-context functionality provides a way to easily switch between TADDM, Tivoli Business Service Manager or CCMDB in the context of the same component or service. The launch occurs through the directions indicated by Figure 22-36. Note that launching TADDM both from Tivoli Business Service Manager (TBSM) and CCMDB is a standard feature. Launching Tivoli Business Service Manager and CCMDB from each other demands a manual configuration that is shown in this section. Typical scenarios that can take advantage of this functionality include the following situations:

- From a specific service in CCMDB, launch Tivoli Business Service Manager to check for the operational status of the service.
- From a specific service in CCMDB or Tivoli Business Service Manager, launch TADDM to check for the most current discovered components.
- From a specific service in Tivoli Business Service Manager, launch CCMDB to open a change request or check for the history of changes.

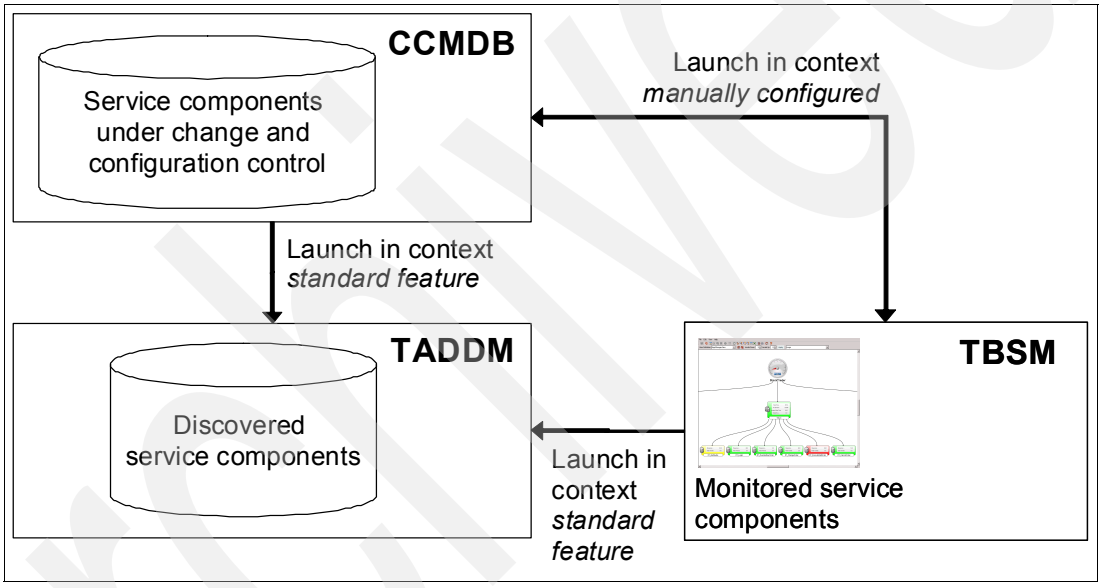


Figure 22-36 Directions for the launch-in-context functionality

Launching CCMDB from Tivoli Business Service Manager

These steps show how to add two actions to the right-click menu option **Launch to**:

- One action opens a process request for the selected service (Open process request action). Process requests can be used as request for changes (RFCs) in CCMDB.
- The other action shows the changes for the selected service (View changes action).

Figure 22-37 on page 691 shows the final menu that is obtained.

Perform the following steps

1. Define the CCMDB address in Tivoli Business Service Manager:
   a. Open the following file, located on the Tivoli Business Service Manager dashboard server:

      `<PRODUCT_INSTALLATION_PATH>\systemApps\isclite.ear\sla.war\etc\rad\RAD_sla.props`

      If you are using a Linux system, the file is in the following location:

      `<PRODUCT_INSTALLATION_PATH>/systemApps/isclite.ear/sla.war\etc/rad/RAD_sla.props`

b. Search for the line `impact.sla.ccmdb.sourceContactInfo`.

c. Uncomment the property and enter the URL with the host address name and port for the CCMDB console, for example:

```
impact.sla.ccmdb.sourceContactInfo=http://ism.demo.tivoli.com
```

Note: The domain name server (DNS) must be properly configured.

d. Save and close the file.

e. Start and stop the Tivoli Business Service Manager server.

2. Enable the launch-in-context function.

Three files must be edited to enable the function:

– Windows:

- `<PRODUCT_INSTALLATION_PATH>\profiles\TBSMProfile\installedApps\TBSMCell\tbsm.ear\sla.war\av\xmlconfig\canvasDynamicSubMenuActions.xml`
- `<PRODUCT_INSTALLATION_PATH>\systemApps\isclite.ear\sla.war\WEB-INF/classes/custom_canvas_actions.properties`
- `<PRODUCT_INSTALLATION_PATH>\profiles\TBSMProfile\installedApps\TBSMCell\tbsm.ear\sla.war\av\xmlconfig\canvasOpenURLActions.xml`

– Linux or UNIX:

- `<PRODUCT_INSTALLATION_PATH>/profiles/TBSMProfile/installedApps/TBSMCell/tbsm.ear/sla.war/av/xmlconfig/canvasDynamicSubMenuActions.xml`
- `<PRODUCT_INSTALLATION_PATH>/systemApps/isclite.ear/sla.war/WEB-INF/classes/custom_canvas_actions.properties`
- `<PRODUCT_INSTALLATION_PATH>/profiles/TBSMProfile/installedApps/TBSMCell\tbsm.ear/sla.war/av/xmlconfig/canvasOpenURLActions.xml`

Note: Always back up the files before editing them. The original Tivoli Business Service Manager configuration can be restored if any serious issue occurs while executing the steps.

a. Using the text editor of your preference, open the `canvasDynamicSubMenuActions.xml` file.

b. In the file, search for the section listed in Example 22-1.

Example 22-1 canvasDynamicSubMenuActions.xml

```
<!-- Old Menu Items -->
    <nextAction name = "ShowManagedSystem"/>
    <nextAction name = "SoaLaunch"/>
    <nextAction name = "ShowHOPViewLocal"/>
    <nextAction name = "ShowHOPViewRemote"/>
    <nextAction name = "ShowPhysicalTopology"/>
    <nextAction name = "ShowChangeHistory"/>
    <nextAction name = "ShowCIDetails"/>
    <nextAction name = "ShowOpenServiceRequest"/>
    <nextAction name = "TPCLaunch"/>
    <!-- End: Old Menu Items -->
```

Archived

c. Add the three menu options, as shown in Example 22-2.

Example 22-2 Add those lines

```
<nextAction name = " ShowOpenServiceRequest "/>
<nextAction name = "ShowOpenProcessRequest"/>
<nextAction name = "ShowViewChanges"/>
```

The updated section looks like Example 22-3.

Example 22-3 Updated file

```
<!-- Old Menu Items -->
    <nextAction name = "ShowManagedSystem"/>
    <nextAction name = "SoaLaunch"/>
    <nextAction name = "ShowHOPViewLocal"/>
    <nextAction name = "ShowHOPViewRemote"/>
    <nextAction name = "ShowPhysicalTopology"/>
    <nextAction name = "ShowChangeHistory"/>
    <nextAction name = "ShowCIDetails"/>
    <nextAction name = "ShowOpenServiceRequest"/>
    <nextAction name = "ShowOpenProcessRequest"/>
    <nextAction name = "ShowViewChanges"/>
    <nextAction name = "TPCLaunch"/>
    <!-- End: Old Menu Items -->
```

d. Save the file.

3. Add the descriptions for the menu items:

a. Using a text editor, edit the `custom_canvas_actions.properties` file.

b. Look for the following line:

```
ShowOpenServiceRequest_description=Show the screen to open a service request
in IBM Tivoli CCMDB
```

c. Add the lines that are listed in Example 22-4, after the following line located in step b:

```
ShowOpenServiceRequest_description
```

Example 22-4 Add these lines

```
ShowOpenProcessRequest=Open Process Request (CCMDB)
ShowOpenProcessRequest_description=Show the screen to open a process request
in IBM Tivoli CCMDB
ShowViewChanges=View Changes (CCMDB)
ShowViewChanges_description=Show the open process requests in IBM Tivoli
CCMDB
```

d. Save the file.

4. Add the launch actions to be performed:

a. Using a text editor, edit the `canvasOpenURLActions.xml` file.

b. Look for the section shown in Example 22-5 on page 690.

Archived

Example 22-5 Section to look for

```
<openURLAction description="Open Service Request (CCMDB)" displayName="Open
Service Request (CCMDB)"
    enableDisableExpression="'__TADDM_sourceToken__' != 'NULL'"
    name="ShowOpenServiceRequest"
permissionCheckerClassName="com.micromuse.sla.map.AVCheckRADInstancePermissi
onsImpl"
roleRequired="tbsmViewService"
    target="CCMDB"
visibleInGUI="true">__CCMDB_sourceContactInfo__/maximo/ui/?event=loadapp&amp
;value=createsr
  </openURLAction>
```

c. Immediately after this section, add the two new menu actions (to open a process request and to view changes) shown in Example 22-6.

Example 22-6 Add these lines

```
<openURLAction description="Open Process Request (CCMDB)" displayName="Open
Process Request (CCMDB)"
    enableDisableExpression="'__TADDM_sourceToken__' != 'NULL'"
    name="ShowOpenProcessRequest"

permissionCheckerClassName="com.micromuse.sla.map.AVCheckRADInstancePermissi
onsImpl"
    roleRequired="tbsmViewService"
    target="CCMDB"
visibleInGUI="true">__CCMDB_sourceContactInfo__/maximo/ui/?event=loadapp&amp
;value=pmcomsr&amp;additionalevent=insert
  </openURLAction>
<openURLAction description="View Changes (CCMDB)" displayName="View Changes
(CCMDB)"
    enableDisableExpression="'__TADDM_sourceToken__' != 'NULL'"
    name="ShowViewChanges"
permissionCheckerClassName="com.micromuse.sla.map.AVCheckRADInstancePermissi
onsImpl"
    roleRequired="tbsmViewService"
    target="CCMDB"
visibleInGUI="true">__CCMDB_sourceContactInfo__/maximo/ui/?event=loadapp&amp
;value=change&amp;additionalevent=sqlwhere&amp;additionaleventvalue=exists
(select multiid from multiassetlocci where cinum in (select cinum from ci
where actcinum = (select ACTCINUM from actci where
GUID='__TADDM_sourceToken__')) and (recordkey=wochange.wonum and
recordclass=wochange.woclass))&amp;forcereload=true
  </openURLAction>
```

d. Save the file.

e. Restart the Tivoli Business Service Manager 4.2.1 Data Server and the Tivoli Business Service Manager 4.2.2 Dashboard Server. See Figure 22-37.

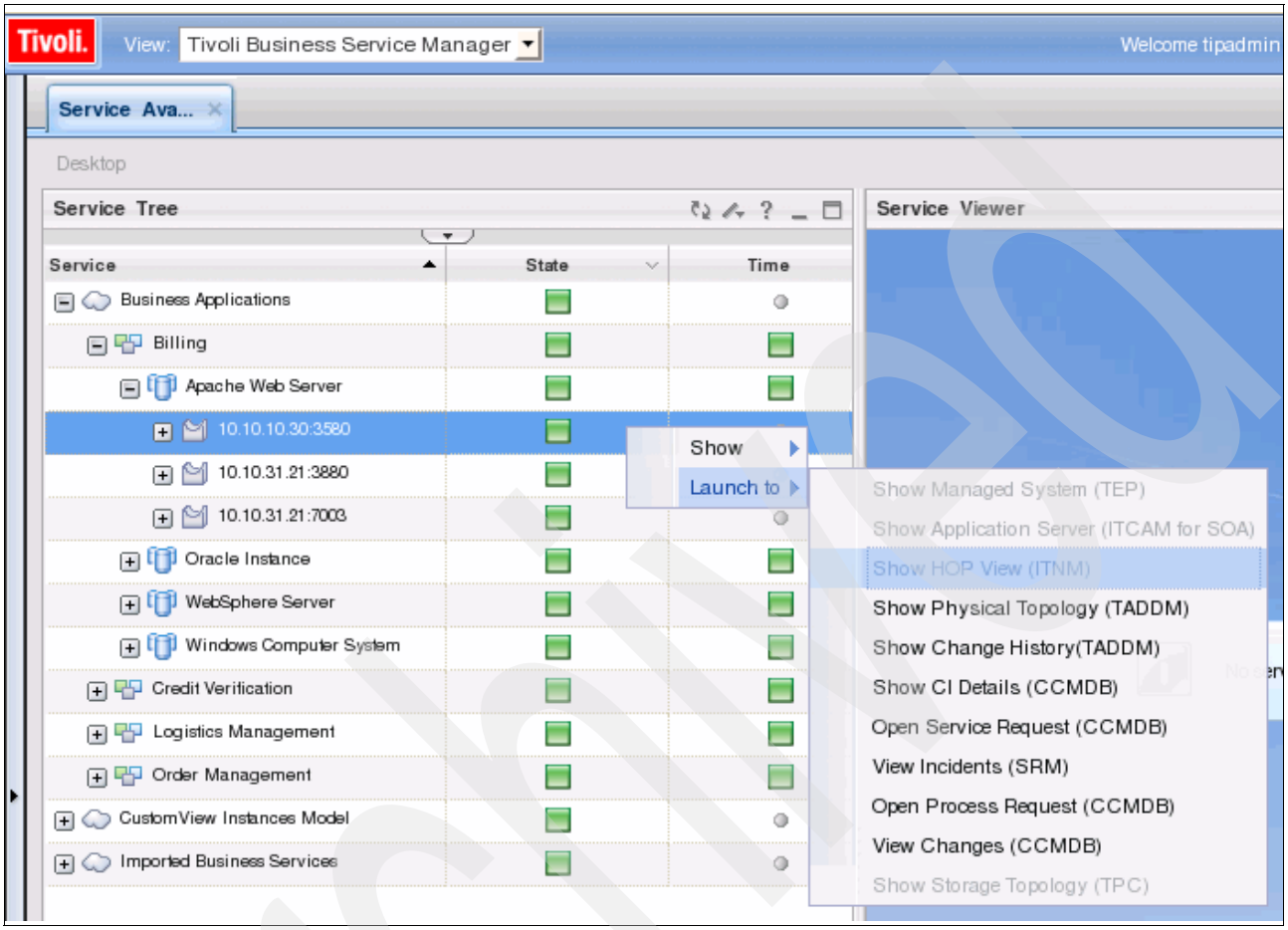


Figure 22-37 The launch menu in Tivoli Business Service Manager

Launching Tivoli Business Service Manager from CCMDB

The steps in this section show how to add three actions to the Select Action menu of CCMDB. Each action opens a separate service view in Tivoli Business Service Manager:

- Show Service Viewer: Basic service instance relationships that are near to this configuration item. It is referred to as service instance within Tivoli Business Service Manager.
- Show All Services Affected: Shows the Service Viewer, but instead of showing a basic relationship topology view, a view shows all Tivoli Business Service Manager business services that contain this particular service is displayed.
- Show Impacted Services: Shows the Service Viewer, but instead of showing basic relationship topology view, a view shows all non-green Tivoli Business Service Manager business services that contain this particular service is displayed.

Perform the following steps:

1. From the Windows Start menu, click **Integration** → **Launch in Context** → **Launch entry** and click **New launch entry.**
2. Enter the launch entry names, descriptions, and URLs. If needed, change the host names and ports of our examples to the values of your environment. Each of the following items represents a separate entry:
   - LIC Entry Name: TBSMVIEWER

     Description: Show Service Viewer (Tivoli Business Service Manager)

     `URL:https://tbsm421:16316/ibm/action/launch/com.ibm.tbsm.navigationElement.desktop&Guid={actualci.guid}`
   - LIC EntryName: TBSMALLB

     Description: Show All Services Affected (Tivoli Business Service Manager)

     `URL:https://tbsm421:16316/ibm/action/launch/com.ibm.tbsm.navigationElement.desktop&Guid={actualci.guid} &View=BusinessImpactAll`
   - LIC EntryName: TBSMIMPBS

     Description: Show Impacted Services (Tivoli Business Service Manager)

     `URL:https:/tbsm421:16316ibm/action/launch/com.ibm.tbsm.navigationElement.desktop&Guid={actualci.guid}&View=BusinessImpact`
3. Add a **configuration item** resource object for Launch Contexts section. Figure 22-38 shows an example of a complete entry.

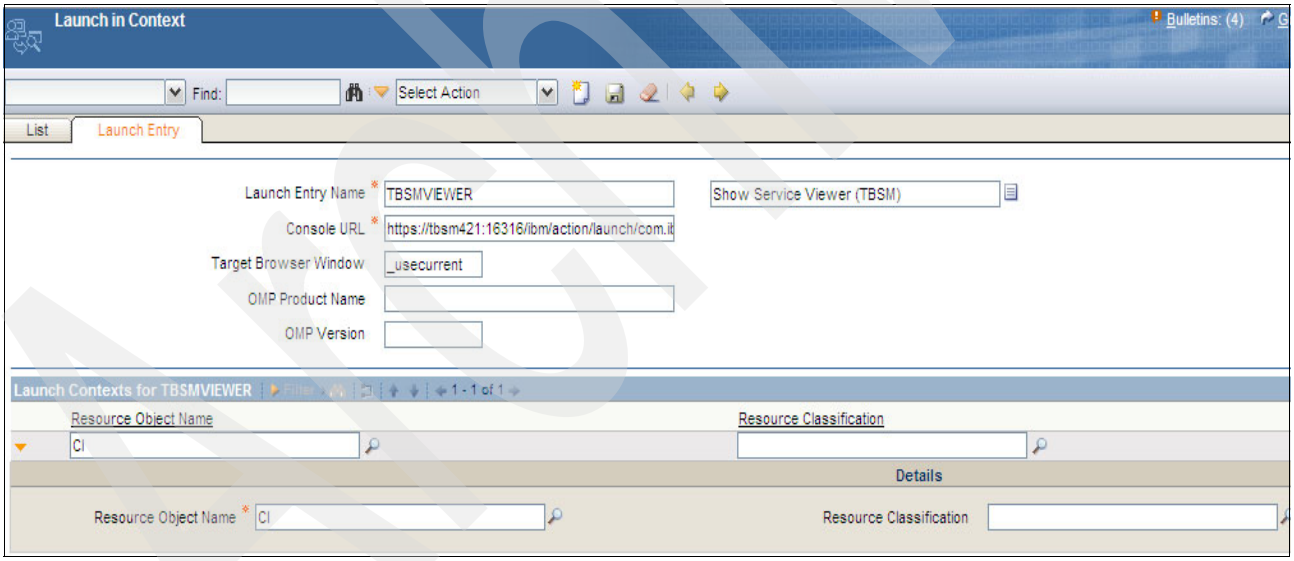


Figure 22-38 The launch-in-context configuration

4. Associate the launch entry with a signature option with the application designer:
   a. Go to **System Configuration** → **Platform Configuration** → **Application Designer.**
   b. List all applications and select **Configuration Items** as the application to be linked to the Tivoli Business Service Manager entries.
   c. Select **Add/Modify Signature Options** from the drop-down list.
   d. Expand the **Advanced Signature Options** section. See Figure 22-39 on page 693.

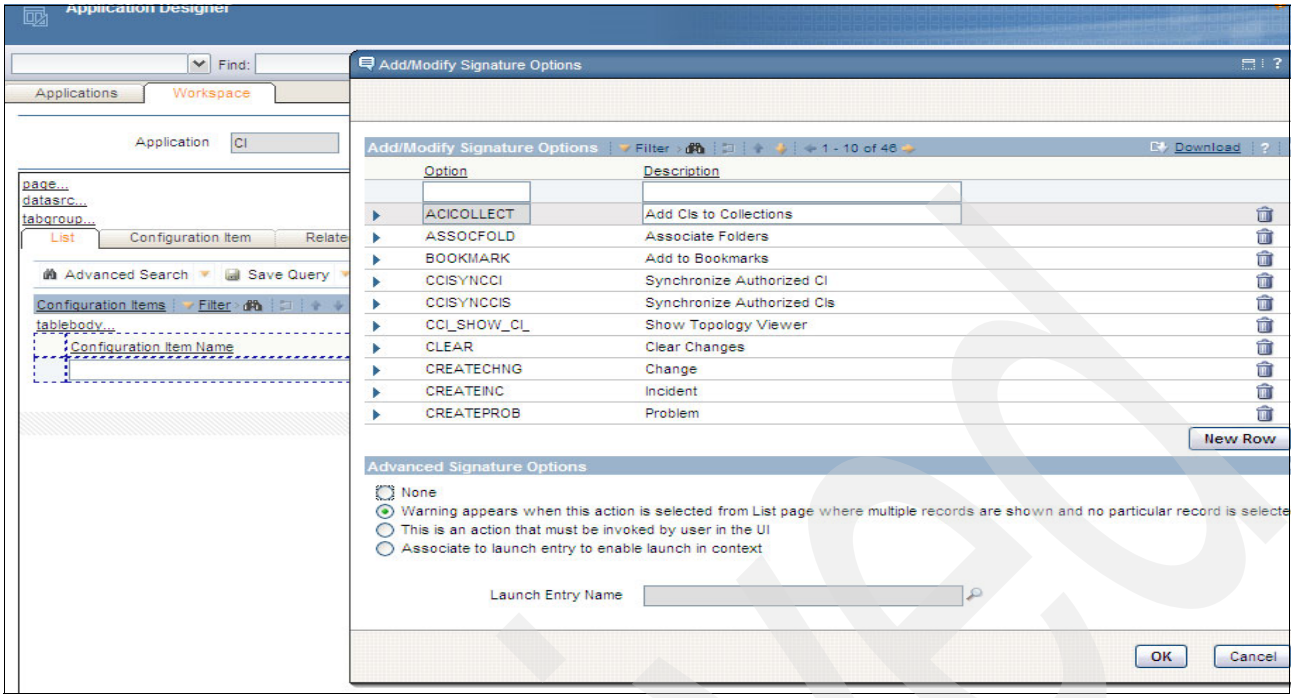


Figure 22-39 Configuration Item application signature options

e. Enter option name (reference used in other panels) and a description (string that appears to user on the action menu).

f. Select **Associate to launch entry to enable launch in context**. Enter the **Launch in Context** entry name. Create one entry for each line of Table 22-2.

Table 22-2 Signature options configurations

Option	Description	Launch-in-context entry name
SVIEWER	Show Service Viewer (Tivoli Business Service Manager)	TBSMVIEWER
SALLBS	Show All Services Affected (Tivoli Business Service Manager)	TBSMALLBS
SIMPBS	Show Impacted Services (Tivoli Business Service Manager)	TBSMIMPBS

g. After creating all signature options, click **Select Action** → **Add/Modify Select Action Menu**.

h. Create entries in accordance with the information presented in Table 22-3. (Figure 22-40 on page 694) shows an example.

Table 22-3 Select Action configurations

Key value (option)	Position	Tabs	Sub-position
SVIEWER	85	Main	0 (means that it is positioned at the main level, not in a menu sub-directory)
SALLBS	86	Main	0
SIMPBS	87	Main	0

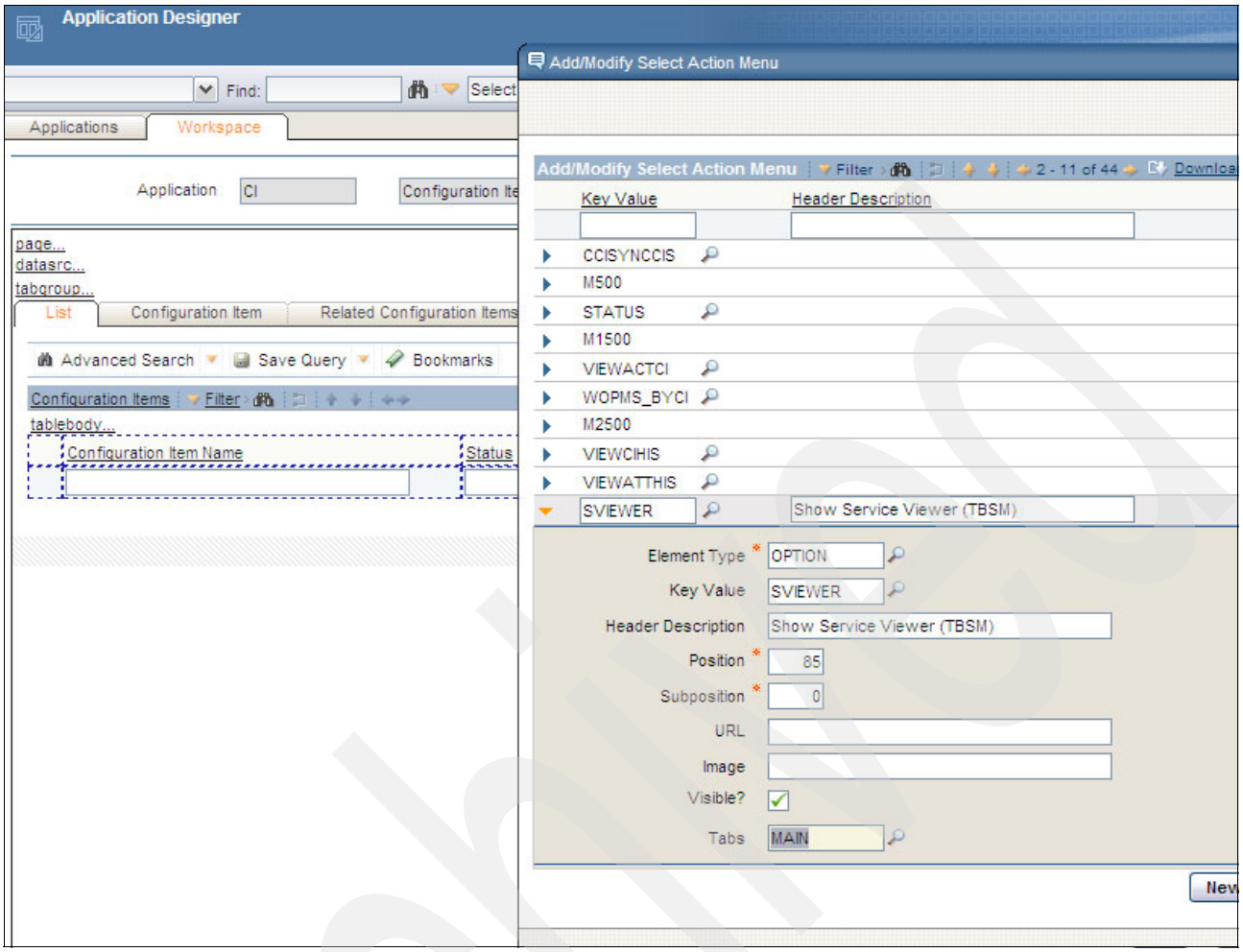


Figure 22-40 Select Action menu

5. Associate a condition with the signature option and configure security settings for user group access to launch entry:
   a. Go to **Security** → **Security Groups.**
   b. Find the security group **Everyone**. If needed, specific access constraints can be configured to other groups.
   c. Click **Applications** tab and find the **Configuration items** application.
   d. Find the signature options on the lower section of the page and for each Tivoli Business Service Manager signature option of Table 22-2 on page 693 add the `VACTCINUM` condition. See Figure 22-41 on page 695 for an example.

Note: The *VACTCINUM condition* verifies whether an actual CI is linked to the CI that is used as a reference to launch Tivoli Business Service Manager. If no actual CI is found, launching Tivoli Business Service Manager in the CI context is not possible and no *Globally Unique Identifier* is associated to the CI.

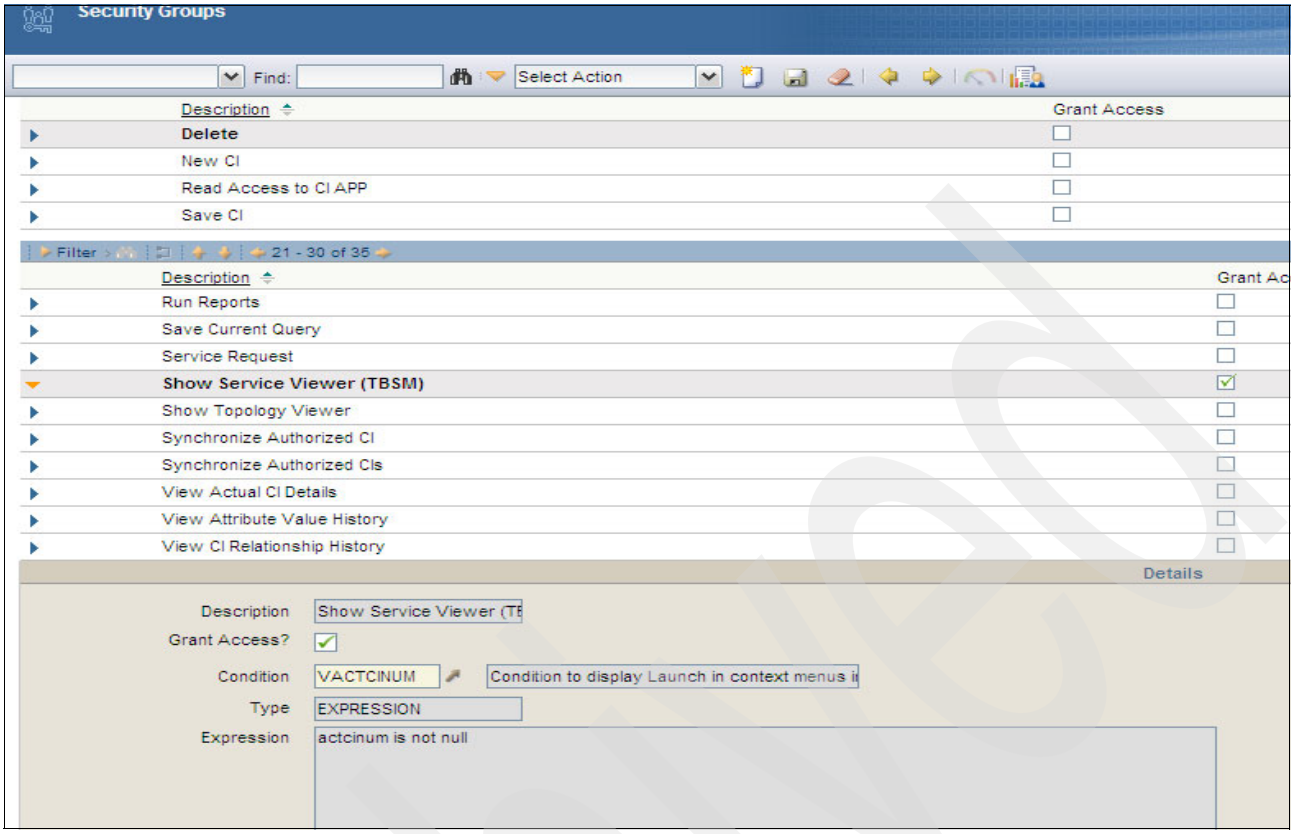


Figure 22-41 The signature option security configuration

e. Save the record.

22.7 Additional integration: Integrating Tivoli System Automation Application Manager with Tivoli Business Service Manager

In this chapter so far, we have seen integration of CCMDB and TADDM with Tivoli Business Service Manager that is used to define and manage services. In this section, we describe another integration within the context of service management: Tivoli System Automation Application Manager with Tivoli Business Service Manager integration.

Typically, business applications consist of various middleware components, are multitiered, and run on heterogeneous platforms. Tivoli Business Service Manager is used to provide health information of the multitiered application and to provide monitoring of service-level agreements (SLA) based on information coming from numerous sources

Tivoli Netcool/OMNIbus is used to collect all events that are related to the business application landscape and Tivoli Business Service Manager uses these events to determine the status of the business applications.

System Automation Application Manager is used in business application landscapes to automate start-stop dependencies, provide a common operating, provide automatic recovery in failure situations, and obtain an aggregated availability status.

System Automation for Multiplatforms and System Automation for z/OS are used to make individual components of the business application highly available, for instance a critical database.

You can create a solution that integrates Tivoli Business Service Manager and Tivoli System Automation Application Manager to monitor, control and automate your business applications from end-to-end.

The integration solution of Tivoli System Automation Application Manager and Tivoli Business Service Manager addresses the following areas:

- Visualize service scorecards, key performance indicators, and SLAs of business applications in Tivoli Business Service Manager. Use System Automation Application Manager to control and automate these business applications and to provide automatic recovery of failure situations.
- Data from System Automation Events can be used to enrich Tivoli Business Service Manager service views with information from System Automation. System Automation Application Manager delivers a Tivoli Business Service Manager service template containing pre-configured rules how to map states from System Automation to Tivoli Business Service Manager service instances.

  As an example, in this integration, when a business application was brought offline because of maintenance reasons Tivoli Business Service Manager can now be aware that this is a planned downtime of the business application and not an unplanned failure situation.
- The integration of System Automation Application Manager resources with Tivoli Business Service Manager can be automated and simplified using the Discovery Library Adaptor (DLA). With this approach System Automation Application Manager resources are automatically imported into a Tivoli Business Service Manager service model, the right Tivoli Business Service Manager status rules are automatically assigned to those services, and System Automation Application Manager events are automatically matched with the correct Tivoli Business Service Manager service instances without manual configuration.
- Launch-in- context from a Tivoli Business Service Manager service view to the System Automation Operations Console and vice versa allows an operator to easily see the information about a specific application from both products. For example, with this capability Tivoli Business Service Manager can be used to monitor the health of a business application and an operator can use the launch in context capability to jump to the System Automation Operations Console to start or stop this business application.

This integration solution is described in detail in the white paper called *Integrating Tivoli System Automation Application Manager with Tivoli Business Service Manager*, written by Wolfgang Schaeberle from IBM Germany. The paper is available at the following location:

https://www.ibm.com/developerworks/wikis/display/tivoli/Tivoli+System+Automation+Application+Manager+Integration+Scenarios.

22.8 Summary

Figure 22-42 shows the summary of this integration.

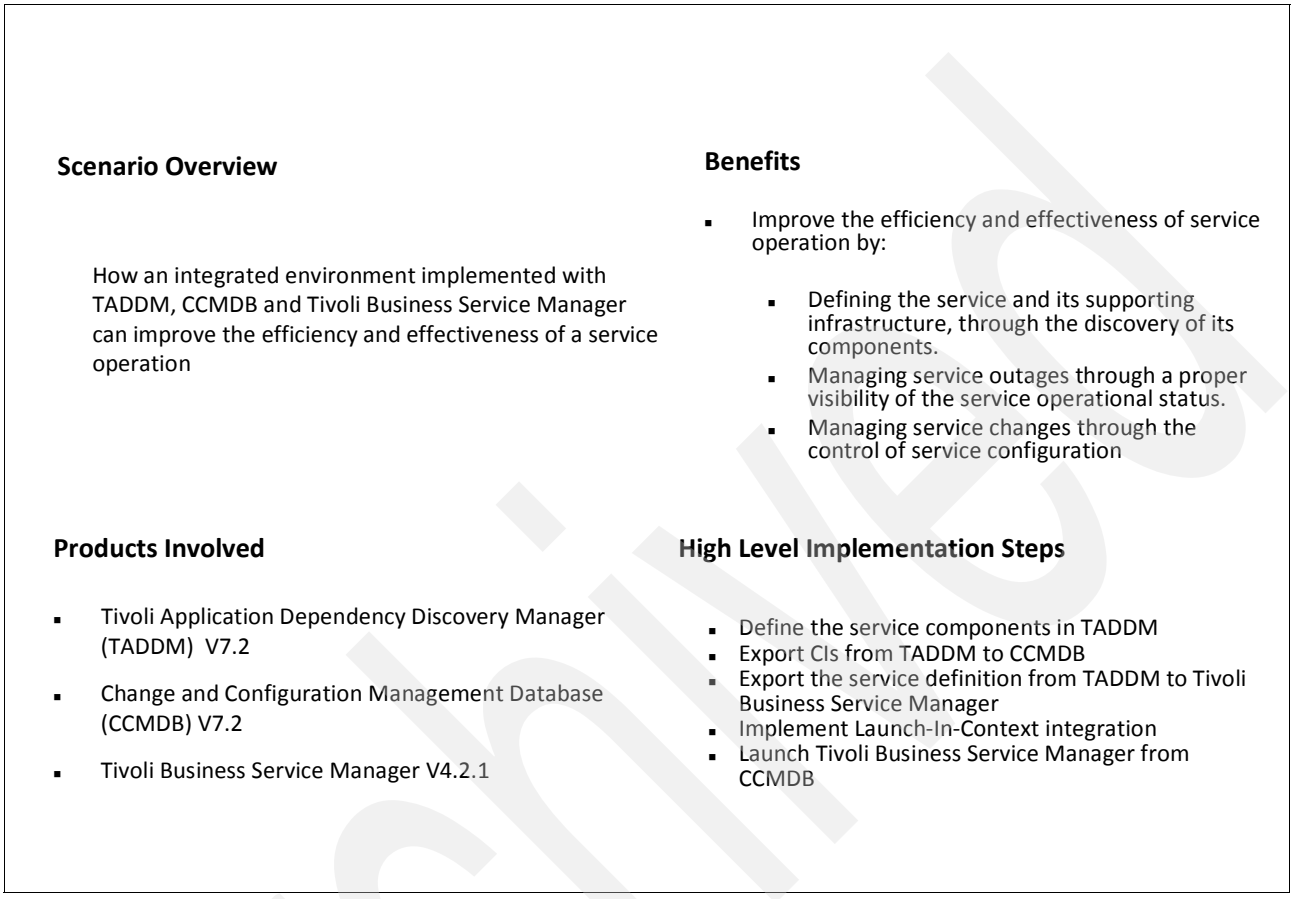


Figure 22-42 Quick summary

Archived

Archived

23

CCMDB V7.2.1: New launch-in-context technology

This chapter, which is the final scenario of this part, describes the launch-in-context (LIC) technology that is available with CCMDB V7.2.1.

CCMDB V7.2.1 enables new LIC technology that addresses certain client requirements with the existing technology. The current LIC technology will continue to be available in CCMDB V7.2.1 along with the new technology.

This chapter contains the following topics:

- 23.1, “Scenario overview and benefits” on page 700
- 23.2, “Components of the new LIC technology” on page 701
- 23.3, “Naming and Reconciliation Service” on page 702
- 23.5, “Deployment and configuration of DIS/CMS” on page 704
- 23.6, “CMS Registry Loader externals” on page 708
- 23.7, “Summary” on page 715

© Copyright IBM Corp. 2011. All rights reserved.

23.1 Scenario overview and benefits

Launch-in-context (LIC) allows integration of various Tivoli products, increasing operational productivity and reducing cost of ownership of these products. Although this feature is powerful with the current implementation, the products must create product-to-product agreement for launching URLs, with every change or new release LIC must by synchronized and revisited. In addition, the launches must be manually configured.

The new LIC technology provides the following benefits:

- Better utilization of the existing products: You can use multiple Tivoli products in their environment, each providing information. Using the new LIC technology, you see a comprehensive view to help with decision-making and can be used to solve issues, thus providing more value for investments.
- Fast time to value: You do not have to configure launch points manually. Also, you do not have to go to each product to view the information that is related to some component in the environment. The new LIC technology provides a single repository from where each product can fetch all information quickly without any configurations.
- Navigation across products using launch points: LIC enables a common cross product registration mechanism for launch points thus providing integration point for all CI-related products. This benefit enhances the operational experience across the Tivoli products, each having information related to the CI.

Many integration scenarios contain the need for LIC, in the following examples:

- Business Service Impact analysis through Incident and Availability Management
- Change Lifecycle Management
- IBM Tivoli Monitoring with Systems Director Monitoring
- IBM Tivoli Network Manager for IP Edition with Systems Director Monitoring

To explain this need, consider the “sub-scenario” example of a Link Failure at Edge of Network:

1. Service operator analyzes the event and evaluates service impact in Tivoli Business Service Manager.
2. Service Operator validates the problem in IBM Tivoli Monitoring.
3. Service Operator opens ticket and assigns to Network Operator.
4. Network Operator views network event in IBM Tivoli Netcool/OMNIbus.
5. Network Operator views network topology in IBM Tivoli Network Manager.
6. Network Operator assigns ticket to Network Subject Matter Expert (SME).
7. SME views network data in IBM Tivoli Network Manager for IP Edition.

Archived

As shown in Figure 23-1, by using the new LIC technology, one can seamlessly navigate across various products, instead of going to each launch entry of the product and navigating.

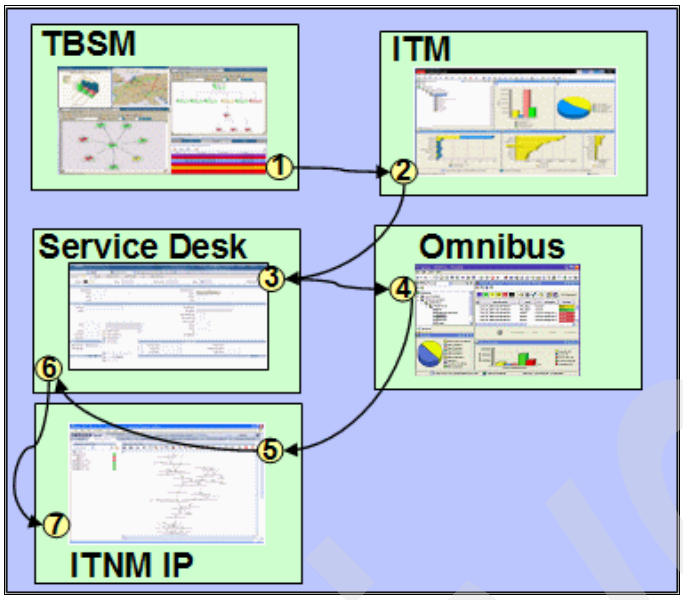


Figure 23-1 Navigation across products

Note: IBM Director, TADDM and CCMDB are initial adopters of the new LIC technology. The products explained in the demo are yet to adopt this technology. The example is given only to show users how the LIC would work with the new technology.

23.2 Components of the new LIC technology

The components involved in the new LIC technology are as follows:

- Data Integration Services (DIS) component handles multiple naming contexts for resources through enhanced reconciliation with a centralized database of resources and all their naming contexts and naming attributes.
- Context Menu Service (CMS) component manages a centralized registry of LIC entries and can build the context menus for resources using the naming information in DIS.
- Operational Management Products (OMP) register their naming context for resources that they manage with DIS and use the CMS to build context menus for those resources in the OMP user interfaces.

Figure 23-2 on page 702 illustrates these components that are used in the new LIC technology.

Archived

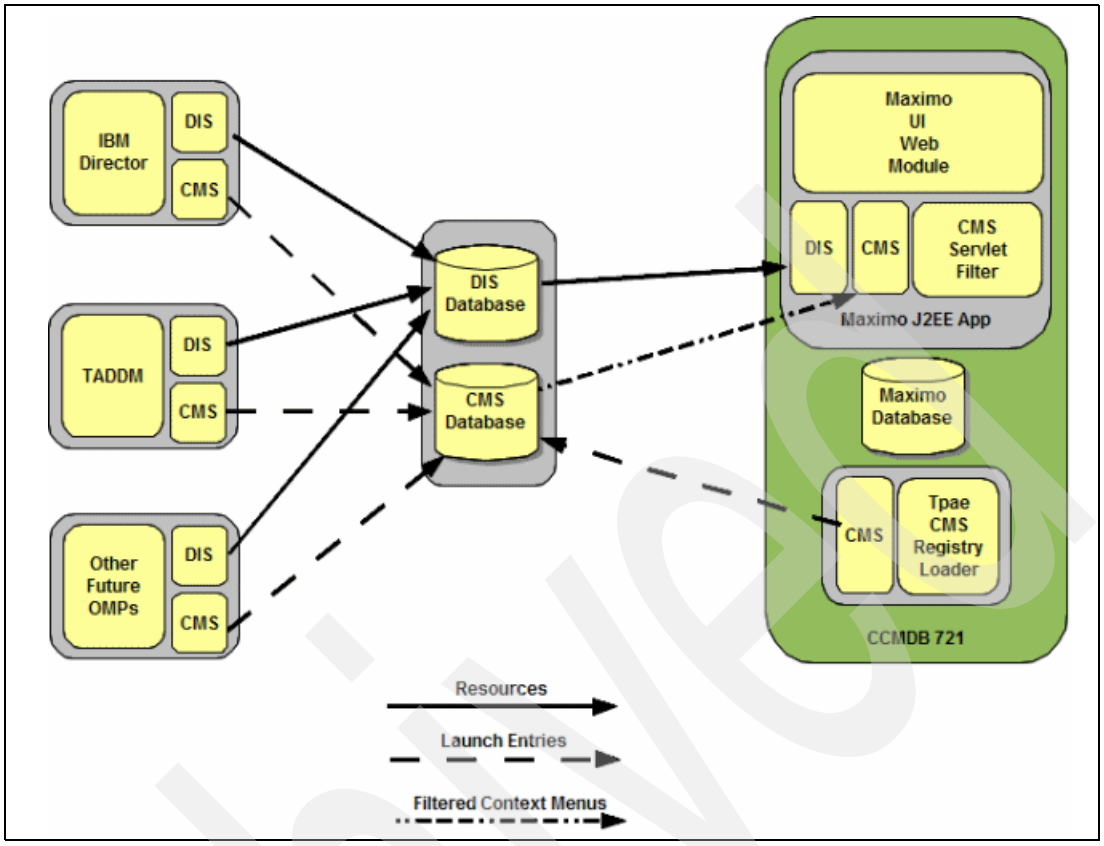


Figure 23-2 LIC components

23.3 Naming and Reconciliation Service

Naming and Reconciliation Service helps OMPs get and set attributes that CMS uses. It provides a common “CDM-aware” data source for all OMPs and CMS to use. It also provides basis for common language across products.

An OMP stores attributes it “knows” about a resource through DIS. CMS uses DIS to look up the attributes (resource context filters). See Figure 23-3.

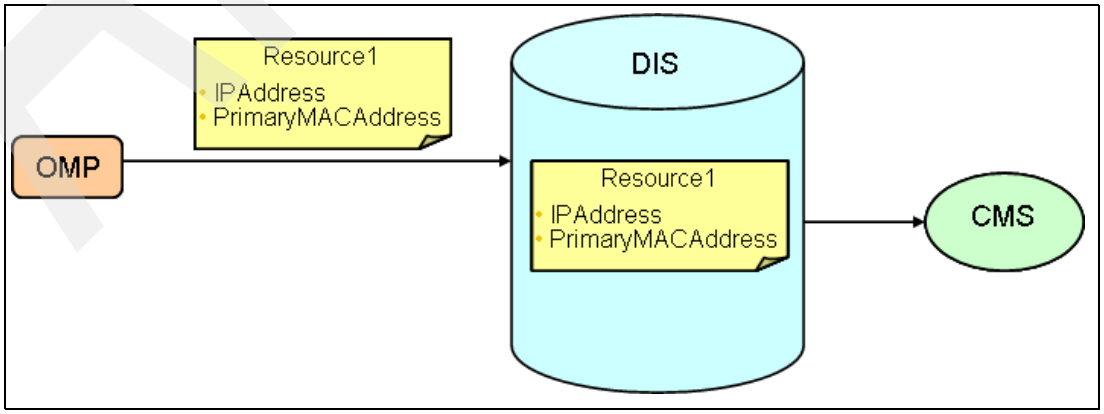


Figure 23-3 Naming

Reconciliation is a fundamental and critical function of DIS. DIS can “reconcile” resource information that is provided by multiple OMPs. DIS identifies a resource as being the same when reported by two separate OMPs. Reconciliation then merges the attributes that are provided by multiple OMPs. See Figure 23-4.

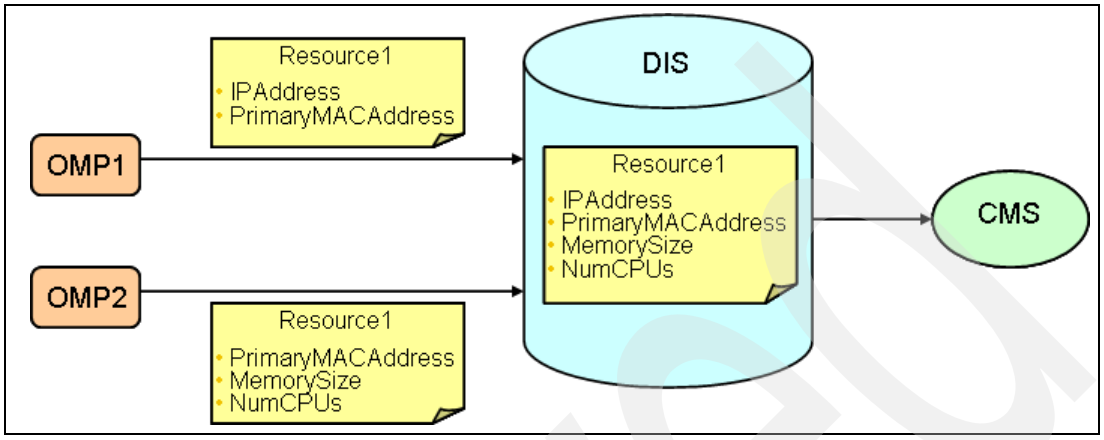


Figure 23-4 Reconciliation

23.4 Context Menu Services

The use of CMS decouples OMPs from each other for LIC. As additional OMPs are instrumented to use CMS to build their context menus, these OMPs are able to launch into the CCMDB CI applications with no updates required to CCMDB and with no product-to-product interlock between the launching OMP and the OMP into which the launch-into is performed (CCMDB).

The use of CMS allows launching products to determine launch points at run time. When the user selects the launch point, the other product UI starts with context (or previously started UI gets new context), which is dynamic without “hard wiring.” The launched product can provide integrations without waiting for synchronization with the launching products’ release schedule.

The use of CMS in CCMDB V7.2.1 provides the following capabilities:

- Ability to launch into the CCMDB CI applications from any OMP that adopts DIS and CMS.
- Ability to launch from the CCMDB CI applications into any OMP that adopts DIS and CMS.

Archive

23.5 Deployment and configuration of DIS/CMS

This section describes deployment and configuration of DIS/CMS:

23.5.1 Deployment components

The following components are included for this integration:

- The *TADDM DIS/CMS Integration Package* is a TADDM Interim Fix Package that can be used for the following tasks:
  - Creating the database tables that are required for DIS and CMS.
  - Importing and registering resources from TADDM into DIS.
  - Registering CMS Launch Entries for launching into TADDM from other OMPs, such as CCMDB.
- The *Base Services Installer* provides functions for configuring the TPAE environment to integrate with DIS and CMS and optionally for creating the DIS and CMS tables in the Maximo Database.
- The *CI Applications CMS Integration PSI Package* configures the CCMDB CI Applications to support the following launch directives:
  - Launching out from the CI Applications to other OMPs (such as TADDM) that have registered launch entries with CMS
  - Launching into the CI Applications from other OMPs

23.5.2 Options for creating the CMS and DIS tables

The DIS and CMS tables are shared resources with which multiple Tivoli products can integrate. Based on the Tivoli products that you have deployed and the installation sequence of those products, you have three options to deploy the DIS and CMS as follows:

- The DIS and CMS tables have already been set up. In this case, configure the Base Services Installer and the TADDM DIS/CMS Integration Package to use the already-created tables.
- Use the Base Services Installer to create the DIS and CMS tables in the Maximo Database along with the other CCMDB and Tivoli Process Automation Engine tables. In this option, configure the TADDM DIS/CMS Integration Package to load resources and launch entries into those tables in the Maximo Database.
- Use scripts in the TADDM DIS/CMS Integration Package to create the tables. In this option, use the Base Services Installer to configure the Tivoli Process Automation Engine Environment to use the tables setup by TADDM.

Archived

CCMDB and TADDM deployment considerations:

- DIS and CMS tables must be in the same database. TADDM does not support an environment where the DIS and CMS tables are created in separate databases. CCMDB support for CMS-based LIC depends on TADDM registering resources in DIS. Therefore, you must not use the Base Services Installer functions that allow the Tivoli Process Automation Engine environment to be configured to connect to DIS and CMS in separate databases. On the panel labeled “Data Integration Service Database Information,” you must always select the check box labeled “Use the same database for both data integration and context menu services.”
- CCMDB and TADDM do not support CMS LIC using SQL Server. TADDM does not support SQL Server. CCMDB support for CMS-based LIC depends on TADDM registering resources in DIS. Therefore, you must use DB2 or Oracle as the databases for the DIS and CMS tables. You may not select the Base Services Installer option labeled “Deploy data integration and context menu services into the same database that you created for the product” on the panel labeled “Data Integration Service and Context Menu Service Deployment Options” if you have configured the Maximo Database to use SQL Server.

23.5.3 Deployment scenarios

Users performing a base installation of CCMDB V7.2.1 or upgrading an existing CCMDB version to CCMDB V7.2.1 have several options related to DIS/CMS deployment:

- Set up DIS and CMS tables in the database used by CCMDB and configure Tivoli Process Automation Engine to use those tables.
- Configure Tivoli Process Automation Engine to use a previously deployed DIS and CMS.
- Defer or skip DIS and CMS setup and configuration.
- Set up DIS/CMS or configure Tivoli Process Automation Engine for DIS and CMS after installing or upgrading to V7.2.1. After installing or upgrading CCMDB V7.2.1, the user may need the ability to set up Tivoli Process Automation Engine for DIS and CMS. Again, because of the TADDM dependency, a customer may need to initially defer the DIS and CMS setup. This capability allows them to add this integration subsequent to point in time where CCMDB V7.2.1 is upgraded or base installed.

Set up DIS and CMS tables in the database used by CCMDB and configure Tivoli Process Automation Engine to use those tables

This option can be used if the DIS and CMS databases have not previously been deployed by another Tivoli product or OMP. In this scenario, the DIS and CMS Tables are stored in the Maximo Database. For this, first install or upgrade CCMDB before running the TADDM DIS/CMS Integration Package to load launch entries into CMS and resources into DIS:

1. Run the Base Services Installer that is available with CCMDB V7.2.1.
2. In the Base Services Installer UI on the “Data Integration Service and Context Menu Service Deployment Options” panel, select the **Deploy data integration and context menu services into the same database that you created for the product** option.
3. From the “Run Configuration Step” panel, select the **Deploy application files manually later** check box.
4. After the Base Services Installer completes, use the Process Solution Installer (PSI) GUI to install the CI Applications CMS Integration PSI Package.

This package is located in the `pmp` sub-directory of the product installation directory and is named `cci_cms_lic_7.2.1.0.zip` file. From the Package Options panel, ensure that the **Defer Maximo Application Redeployment** and **Defer the Update of the Maximo Database** check boxes are *not selected*.

5. Run the TADDM DIS/CMS Integration Package to populate the CMS Registry database with TADDM Launch Into Entries and to register actual configuration item resources into DIS.
6. Import CI Types and Actual CIs into CCMDB using Tivoli Integration Composer (corresponding to the resources registered into DIS by the TADDM DIS/CMS Integration Package in step 5).

Configure Tivoli Process Automation Engine to use a previously deployed DIS and CMS

If CCMDB is being added on or upgraded in an environment where another product (or the customer) has set up the DIS and CMS databases, this option can be used. Connection information about the existing DIS and CMS databases can be collected and Tivoli Process Automation Engine specific actions (such as registration of Launch Into entries for CCMDB) can be initiated against the current DIS and CMS.

In this scenario, the DIS and CMS Tables are not created in the Maximo Database. The Base Services Installer configures the Tivoli Process Automation Engine environment to use a database already set up by another product to hold the DIS and CMS tables. Two typical scenarios that clients have, where DIS and CMS is already deployed, are as follows:

- CCMDB installed after TADDM creates DIS and CMS tables
- CCMDB installed after another IBM product creates DIS and CMS tables

CCMDB installed after TADDM creates DIS and CMS tables

In this typical scenario, the Base Services Installer configures the Tivoli Process Automation Engine environment to use a database already set up by TADDM to hold the DIS and CMS tables.

The steps are as follows:

1. Run the TADDM DIS/CMS Integration Package to create the DIS and CMS tables (these must be in the same database and only support DB2/Oracle).
2. Run the TADDM DIS/CMS Integration Package to populate the CMS Registry database with TADDM Launch Into Entries and to register actual configuration item resources into DIS.
3. Run the Base Services Installer that comes with the CCMDB 721 product.
4. In the Base Services Installer UI on the “Data Integration Service and Context Menu Service Deployment Options” panel, select the to **Configure data integration and context menu services for this product using a previously deployed data integration and context menu service instance** option.
5. On the subsequent “Data Integration Service Database Information” panel, select the **Use the same database for both data integration and context menu services** check box. On that panel, specify the database connection properties for the DIS/CMS database that you created using the TADDM DIS/CMS Integration Package.
6. From the Run Configuration Step panel, select the **Deploy application files manually later** check box.

7. After the Base Services Installer completes, use the IBM Process Solution Installer (PSI) GUI to install the *CI Applications CMS Integration* PSI package. This package is located in the `pmp` sub-directory of the product installation directory and is named `cci_cms_lic_7.2.1.0.zip`. From the Package Options panel, ensure that the **Defer Maximo Application Redeployment** and **Defer the Update of the Maximo Database** check boxes are *not* selected.
8. Import CI Types and Actual CIs into CCMDB using Tivoli Integration Composer (corresponding to the resources registered into DIS by the TADDM DIS/CMS Integration Package in step 2 on page 706).

CCMDB installed after another IBM product creates DIS and CMS tables

This scenario is similar to the scenario in "CCMDB installed after TADDM creates DIS and CMS tables" on page 706, with the exception that another product (other than TADDM) has already created the DIS and CMS Tables when CCMDB is installed or upgraded. The deployment steps from that scenario are reused with the following exceptions:

- Step 1 on page 706 is not required because the DIS and CMS Tables have already been created.
- In step 5 on page 706, the DB connection properties specified must be for those that are associated with the DIS and CMS database created previously by another IBM product.

Defer or skip DIS and CMS setup and configuration

The default option is to set up CCMDB to take advantage of DIS and CMS. However, an option exists to install or upgrade CCMDB without this support. CCMDB has a dependency on TADDM to register resources with NRS. This option is required to enable the LIC using CMS. The customer installing or upgrading CCMDB may not have already upgraded or installed TADDM at the required V7.2.1 level.

Based on your requirements, you might need to install or upgrade CCMDB and elect to defer the configuration of DIS or CMS until a later time. For example, you might want to coordinate the deployment of DIS and CMS in your organization to a variety of IBM products on a deployment schedule separate from the schedule for rolling out configuration and change management in CCMDB. You can do this by electing to defer DIS and CMS configuration.

The steps are as follows:

1. Run the Base Services Installer.
2. In the Base Services Installer UI on the "Data Integration Service and Context Menu Service Deployment Options" panel, select the **Defer or skip data integration and context menu services setup and configuration** option.
3. In the "Run Configuration Step" panel, do *not* select the **Deploy application files manually later** check box.
4. At a later time, when you want to deploy data integration and context menu services within CCMDB, rerun the Base Services Installer.
5. The Base Services Installer detects that data integration and context menu services have not been configured, and displays the "Data Integration Service and Context Menu Service Deployment Options" panel.
6. From this panel, you may continue using one of the previously described deployment scenarios.

23.6 CMS Registry Loader externals

To support launch-into Tivoli Process Automation Engine applications from other products using CMS, those Tivoli Process Automation Engine applications must be registered as launch points in the CMS Registry. Tivoli Process Automation Engine provides a common program for performing this registration. The CMS Registry Loader is a command-line interface program for loading launch entry definitions for Tivoli Process Automation Engine applications into the CMS Registry.

The CMS Registry Loader provides the following functions:

- Creating and updating launch entries in CMS.
- Listing currently registered launch entries in CMS.
- Removing launch entries from CMS.
- Validating launch entry XML files.

23.6.1 Reasons to use CMS Registry Loader

The CMS Registry Loader is typically used internally during deployment of a Tivoli Process Automation Engine product. You manually invoke this program if necessary. The reasons to use CMS Registry Loader are as follows:

- Update the host or port for the Maximo application. You modify the launch entry XML file (or files) for the currently registered launch entries to correct the host or port and then use the CMS Registry Loader create action to modify the launch entries in CMS with the updated host/port.
- Debug CMS-based LIC If you are having problems with launching into or out of Tivoli Process Automation Engine applications using CMS. The CMS Registry Loader list action can then be used to debug the problem. This action displays information about currently registered launch entries.
- Create new launch entries for other Tivoli Process Automation Engine-based applications.

23.6.2 Syntax

The CMS Registry Loader is deployed in the `bin` directory of the base services installation directory (by default, this is `\IBM\SMP\bin`) and is implemented through the following scripts:

- **`registryLoader.bat`** for Windows-based administrative workstations
- **`registryLoader.sh`** for UNIX-based administrative workstations

An example of running the script is as follows:

```
registryLoader.bat(.sh)
   -action create|remove|list|validate
   <action-specific parameter>
```

Supported actions

Each invocation of the **`registryLoader`** script must specify the **`-action`** parameter, which identifies the action to be performed. Support actions are defined in Table 23-1 on page 709.

Table 23-1 Supported actions

Action	Description
create	Creates and updates launch entries in CMS.
list	Displays currently registered launch entries.
remove	Removes launch entries from CMS.
validate	Validates a launch entry XML file.

Supported parameters

Based on the action specified using the **-action** parameter, additional parameters are required. These parameters are defined in Table 23-2.

Table 23-2 Supported parameters

Parameter	Description	Supported action
-xmlfile <path-to-launch-entry-xml>	Identifies the launch entry XML file that defines the launch entries to be created, updated, or validated.	create validate
-bundle <path-to-nls-bundle-dir>	Identifies the directory containing the resource bundle properties files that contain the localized information for the launch entries being created or updated.	create
-applid <application-id-of-entries-to -remove>	Removes from CMS any launch entries whose CMS owner is set to the specified application identifier.	remove
-dbuser <CMS-database-userid>	Optional. Specifies the user ID used to access the CMS database. If not provided, the user ID collected by the Base Services Installer is used.	create remove list
-dbpwd <CMS-database-userid>	Optional. Specifies the password used to access the CMS database. If not provided, the password collected by the Base Services Installer is used.	create remove list

Launch entry XML file

A CMS launch entry defines the application URL, substitutable URL parameters, filters, and localized information (display name and description) for a cross-product launch in context point.

A launch-entry XML file is an XML file that contains a collection of CMS launch entries that are associated with a single application identifier. The application identifier is used to group a collection of launch entries belonging to an application or product.

All the launch entries in a single launch entry XML file will have the same application identifier.

Launch entries can be grouped into a hierarchical fashion using the XML tags in the launch entry XML file. This permits the creation of nested sub-menus.

An XML schema descriptor is shipped with the base services installer. This XSD is used to validate the contents of the launch entry XML file.

Archived

Creating or updating launch entries

The create action of the CMS Registry Loader is used to create or update CMS launch entries. The key input is the launch entry XML file which is specified using the -xmlfile parameter.

Inside the launch entry XML file, the AppId attribute specifies the application identifier of the launch entries to create. If launch entries already exist with the specified application identifier in CMS, those entries are first removed and then the entries defined in the remainder of the launch entry XML file are created in CMS. Therefore, the create action can be used multiple times to, in effect, replace the currently registered launch entries associated with a particular application identifier.

In addition to the **-xmlfile** parameter, you must also specify the **-bundle** parameter, which identifies the directory containing the resource bundle properties files containing localized information for the launch entries contained in the launch entry XML file.

The create action is typically driven from a PSI Package during installation of a product to install the CMS launch entries associated with a product. You use it to modify currently registered launch entries or to add new launch points for other Tivoli Process Automation Engine applications. A sample invocation is as follows:

```
registryLoader.bat -action create -xmlfile c:\CCICMSLaunchEntries.xml bundle
c:\CCI_Properties
```

Listing currently registered launch entries

The list action lists information about all the currently registered launch entries in CMS. This action can be useful when debugging issues that are associated with CMS-based LIC scenarios. Information about all registered launch entries is printed to standard output. A sample invocation is as follows:

```
registryLoader.bat -action list
```

Removing currently registered launch entries

The remove action of the CMS Registry Loader is used to remove all the launch entries associated with a specific application identifier from CMS. The key input is the applid parameter which specifies the application identifier of the launch entries that are to be removed.

The remove action is typically reserved for invocation by product uninstaller programs or PSI Packages. It permits products to remove the product launch entries from CMS, so that launches into the product being uninstalled are no longer available through CMS. A sample invocation is as follows:

```
registryLoader.bat -action remove -applid com.ibm.ism.cci.app.ci
```

Validating a launch entry XML file

The validate action of the CMS Registry Loader is used to validate a launch entry XML file for well-formedness and for adherence to the XML schema associated with launch entry XML files. The action requires the -xmlfile parameter which specifies the path to the launch entry XML file to be validated.

You should use this command if you are making customizations to a launch entry XML file or creating new launch entry XML files. Validation is also performed automatically when the create action is specified. A sample invocation is as follows:

```
registryLoader.bat -action validate -xmlfile c:\CCICMSLaunchEntries.xml
```

Archive

23.6.3 LIC using TADDM

This section describes an LIC scenario from CCMDB into TADDM.

The first step involved is the setting up of the TADDM and CCMDB servers. After TADDM 7.2.1 is installed, use TADDM Integration Package to register the TADDM launch points.

TADDM Integration Package provides utilities for the following items:

- Creating the CMS and DIS Database tables.
- Register CMS launch entries for launching in to TADDM.
- Register resources in DIS.

Perform the following steps to complete the DIS/CMS setup:

1. Run TADDM DIS/CMS Integration Package to create the DIS and CMS tables.
2. Run the TADDM DIS/CMS Integration Package to populate the CMS Registry database with TADDM Launch Into Entries and to register resources in DIS.
3. Run the Base Services Installer (BSI) with the CCMDB 7.2.1Payload.
4. Select the BSI option to configure DIS/CMS for Tivoli Process Automation Engine and specify the DB connection properties for the DIS/CMS databases.
5. CMS and DIS binaries are extracted and CMS servlet filter is configured.
6. Use the PSI GUI to install the CCI LIC package to enable the CCMDB-specific support for CMS LIC.

   The MAXIMO EAR is rebuilt and redeployed.
7. Import CI Types and Actual CIs into CCMDB using the Tivoli Integration Composer.
8. Promote the CIs.

Test the LIC function as follows:

1. Click **Go To** → **IT Infrastructure** → **Configuration Items**.
2. Select one of the CIs that you had promoted earlier and click the Details Menu of the Actual Configuration Item. See Figure 23-5 on page 712.

Archived

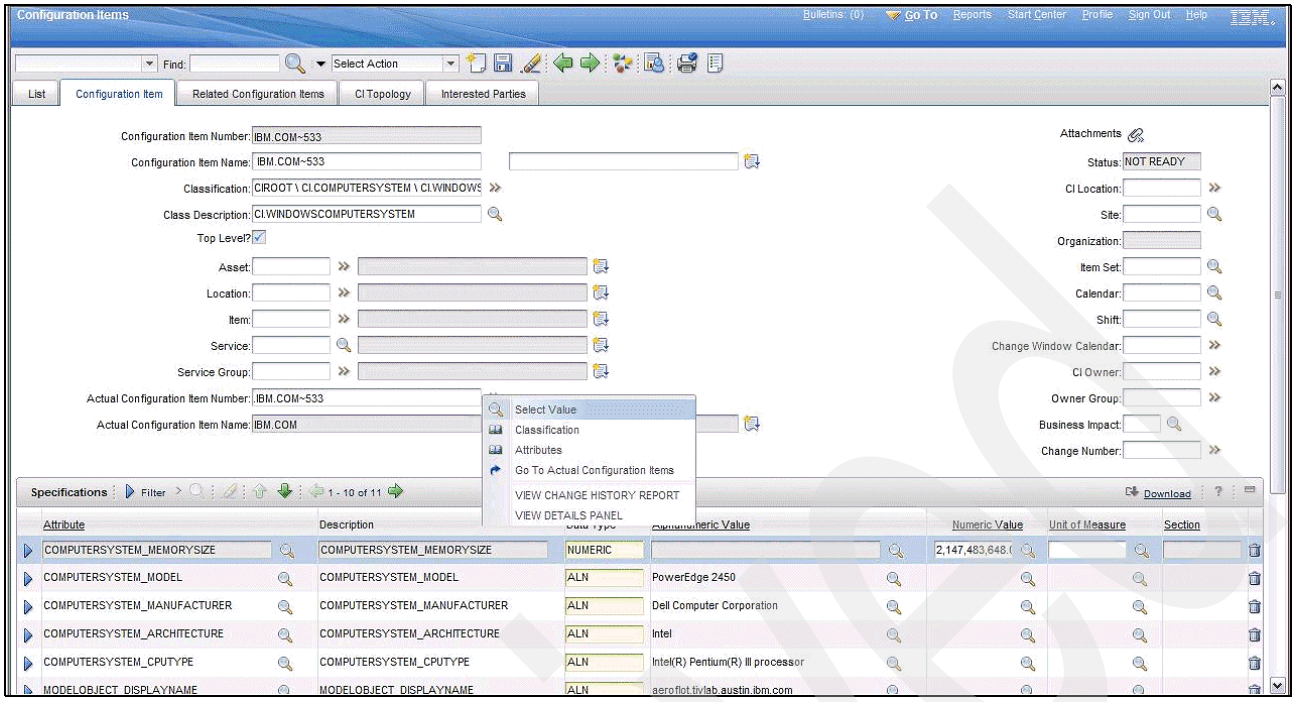


Figure 23-5 Details Menu of the Actual Configuration Item

3. Use the View Change History Report and View Details panels to launch into TADDM CI. You may also select the LIC items from the Select Action menu of the CI.

Archived

4. Click **View Change History Report** to open the TADDM authentication window as shown in Figure 23-6.



Figure 23-6 TADDM authentication window

After providing the required credentials, the change history is listed as shown in Figure 23-7.

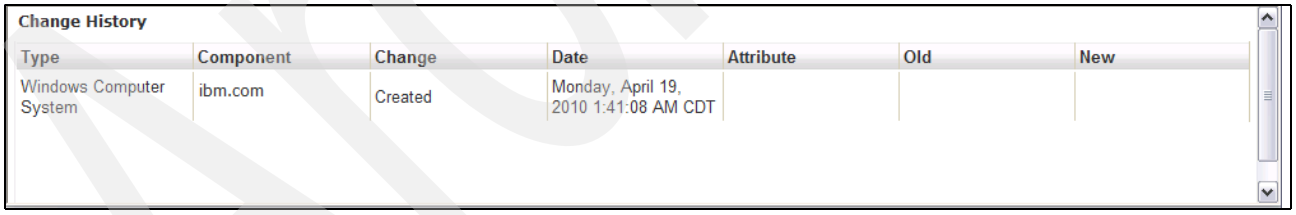


Figure 23-7 Change history

Archive

The View Details panel from CCMDB is shown in Figure 23-8.

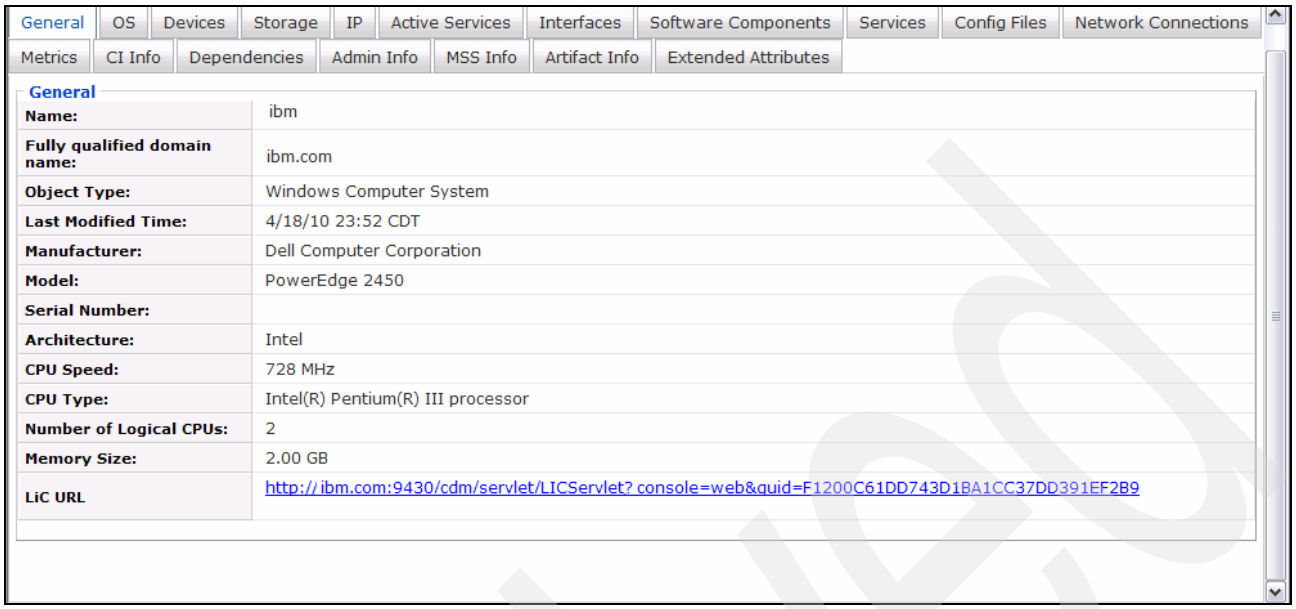


Figure 23-8 View Details panel

Archive

23.7 Summary

Figure 23-9 shows the quick summary of this integration.

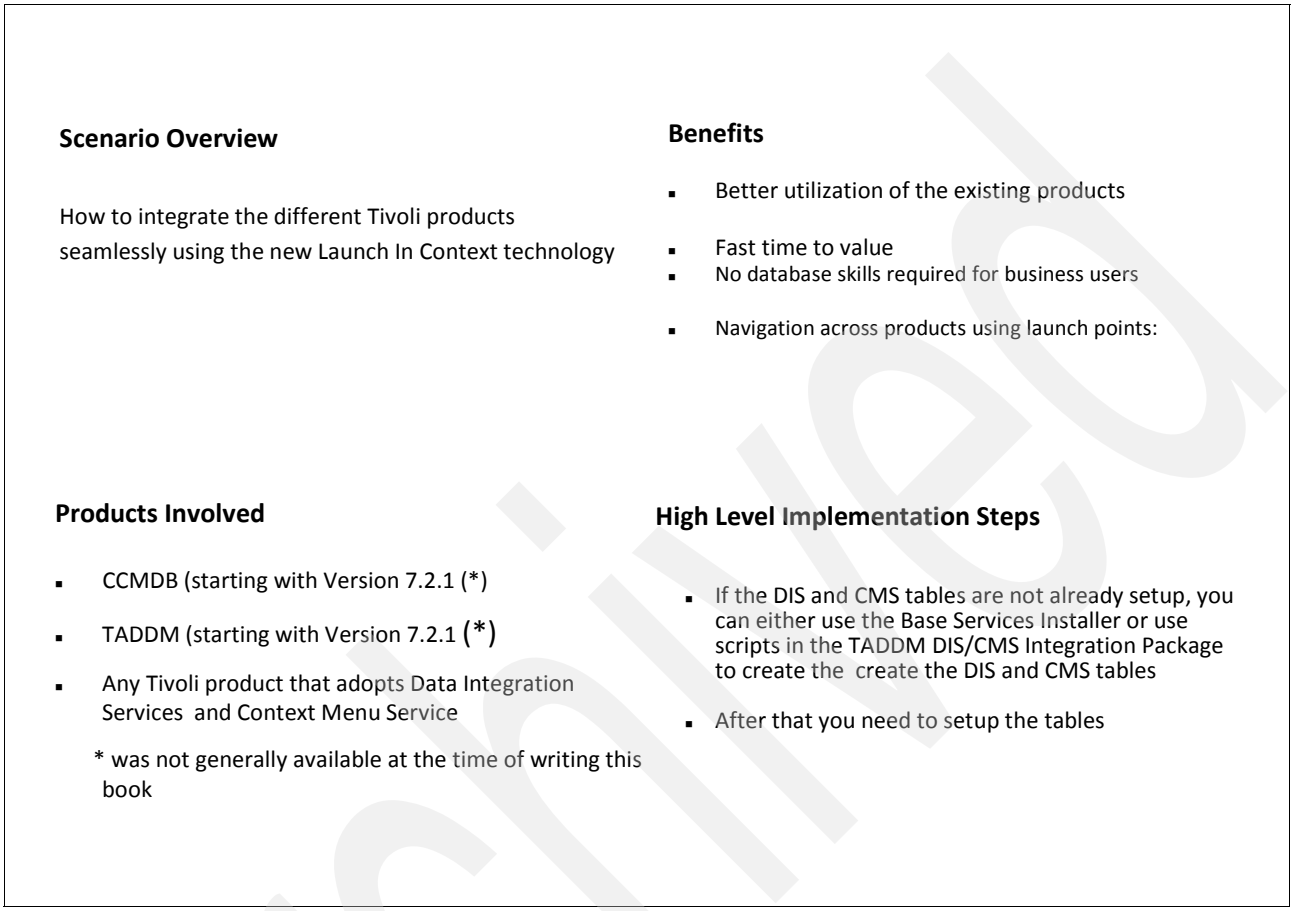


Figure 23-9 Quick summary

Archived

Archived

Part 6

Client integration scenarios

In this part, we cover an integration scenario implemented by an IBM client: ArcelorMittal Gent (`www.arcelormittal.com/gent`). This scenario involves integration of Tivoli Workload Scheduler and Tivoli System Automation for Multiplatforms to implement a highly available scheduling environment.

In this scenario, we took a slightly different approach and covered some of the details of Tivoli Workload Scheduler and Tivoli System Automation for Multiplatforms product functionality related to this scenario. This scenario is designed to help you understand these concepts before implementing this integration.

This part contains the following chapters:

- Chapter 24, “Tivoli Workload Scheduler: Architecture and concepts” on page 719
- Chapter 25, “Tivoli System Automation for Multiplatforms: Concepts” on page 745
- Chapter 26, “Implementing high availability for the Tivoli Workload Scheduler master domain manager” on page 763
- Chapter 27, “Implementing high availability for Tivoli Workload Scheduler dynamic workload” on page 793

© Copyright IBM Corp. 2011. All rights reserved.

Archived

24

Tivoli Workload Scheduler: Architecture and concepts

This chapter focuses on important availability aspects in the architecture of Tivoli Workload Scheduler that are related to the development of the Tivoli System Automation for Multiplatforms policies. Knowing about the policies can help you react to certain events in the network.

This chapter contains the following topics:

- 24.1, “Components” on page 720
- 24.2, “Network communication” on page 723
- 24.3, “High availability” on page 733
- 24.4, “Virtualization” on page 741

© Copyright IBM Corp. 2011. All rights reserved.

24.1 Components

This section concentrates on the availability of critical components in a Tivoli Workload Scheduler network. If a specific part is not available, certain services that are provided to the user can be affected. Understanding how a particular failure might affect the network and which symptoms can arise from that error is important. This knowledge can significantly improve problem determination.

24.1.1 Engine

The Tivoli Workload Scheduler engine is the core component installed on each workstation. The most important role is fulfilled by the engine on the master domain manager, which generates the production plan and routes messages to all workstations.

If the master domain manager is not available, the production plan is not extended, and any message is not routed in the network. An unavailability of the engine must therefore be resolved as quickly as possible. Migrating the role of master domain manager to any full status agent in the top-level domain can be achieved by running **conman switchmgr** command. See 24.3.4, “Switch Manager” on page 739 for a detailed description of the mechanism.

24.1.2 Database

The Tivoli Workload Scheduler database contains the definitions of all scheduling objects and event rules. Moreover, for each triggered event rule, an instance is created in the database and the result of the triggered action is logged.

If the database is unavailable, the following problems might arise:

- Extension of the plan fails because *planman* is unable to establish a connection with the database.
- All *composer* and *optman* commands break down.
- Ad-hoc submission from the database fails because the scheduling objects cannot be retrieved. As a result, **conman sbs** and **conman sbj** commands do not function properly.
- Creation, modification, deployment and triggering of event rules stops functioning.
- Submission of dynamic jobs breaks down.

Although planned workload does not suffer from a problem with the database, a problem with it must certainly be fixed prior to the start of the following production day. If not, planned jobs are affected because the extension of the plan fails. The issue might even be more pressing because event rules are no longer triggered. Moreover, dynamic scheduling is also unavailable during an outage of the database.

If the issue cannot be fixed quickly, a switch to a standby database must be made. For an in-depth discussion of how this switch can be achieved for a DB2 database, see 24.3.1, “DB2: ACR and HADR” on page 733.

24.1.3 The embedded WebSphere Application Server

To interact with the engine or the database, the client interfaces use a connector which is based on embedded WebSphere Application Server. By default, the embedded WebSphere Application Server instance is installed on the master domain manager and backup master

domain managers. Managing the connector is handled by the *appservman* process which reads its requests from the `Appserverbox.msg` message queue.

If the connector is unavailable on the master domain manager, the following symptoms might be observed:

- Extension of the plan fails because planman cannot contact the database.
- All composer and optman commands break down because no connection with the database can be established.
- Ad-hoc submission from the database fails because the scheduling objects cannot be retrieved. As a result, **`conman sbs`** and **`conman sbj`** commands do not function.
- If the *Event Processor* was running inside the application server, it is also unavailable. As a result, no event rules are triggered.
- The *Dynamic Workload Broker Component* fails if it was hosted by the failing embedded WebSphere Application Server instance. As a result, no dynamic jobs are launched. See 24.1.6, “Dynamic Workload Broker Component” on page 723 for a detailed discussion.

All planned workload during the production day does not experience any issues in case of a problem with the connector. Similar to an issue with the database, a problem with the application server must be addressed prior to the start of the following production day to avoid affecting the planned workload. If the error cannot be resolved, a connection to a different embedded WebSphere Application Server instance must be established. For command-line tools (composer, conman, optman), this step can be achieved by altering the embedded WebSphere Application Server connection properties in the `localopts` configuration file, as shown in Example 24-1.

Example 24-1 Embedded WebSphere Application Server connection properties in localopts

```
#HOST            =
#PROTOCOL        = https
#PROTOCOL        = http
#PORT            =
#PROXY           =
#PROXYPORT       =
#TIMEOUT         = 3600
#CLISSLSERVERAUTH = yes
#CLISSLCIPHER     = MD5
#CLISSLSERVERCERTIFICATE =
#CLISSLTRUSTEDDIR =
#DEFAULTWS       =
#USEROPTS        =
```

The state of the application server can be observed in the local Symphony™ file by using the **`conman showcpus`** command, as shown in the reduced output of Example 24-2. State ‘A’ indicates that an embedded WebSphere Application Server instance is active on the particular workstation.

Example 24-2 The embedded WebSphere Application Server and Event Processor state

```
TWSPOC1      UNIX MASTER     LTI JW   EA
FTA_1       *UNIX FTA          I J  MD
FTA_2        UNIX FTA
TWSPOC2      UNIX FTA                 A
XA_VMDM      UNIX X-AGENT
```

The appservman process regularly checks the status of the embedded WebSphere Application Server instance. The interval can be controlled by the *Appserver check interval* localopts tunable. Moreover, as the status of the application server is visible in the Symphony file on each workstation, the appservman process is also responsible for generating state messages. These messages are put in the local `Mailbox.msg` message queue where the connector is installed. Subsequently, state messages are transferred to the master domain manager that will broadcast them further in the network. For more details about the interprocess communication, see 24.2.2, "Workstation interprocess communication" on page 725.

24.1.4 Event Processor

The Event Processor is a J2EE application running inside an embedded WebSphere Application Server in the network. Any Full Status workstation in the top-level domain that is running the connector is eligible for its hosting. The most important components of the Event Processor are as follows:

- The *Event Correlation Engine* is the core component of the Event Processor that receives all events from the monitoring agents. Each incoming event is sequentially passed to all active rules to verify the correlation conditions.
- The *Rule Builder* is responsible for building event rules and deploying them on the target workstations. See 24.2.4, "Event driven workload automation" on page 728 for more details about the deployment process.
- After all correlation conditions are satisfied, the *Action Helper* makes an instance of the event rule in the database and executes the associated actions. The result of the triggered actions is also logged in the Tivoli Workload Scheduler database.

During an unavailability of the Event Processor, triggering and deployment of event rules no longer function properly; creation and modification of existing rules experience no issues if the database is available. The monitoring agents notice the unavailability of the Event Processor and queue events in a local cache. For more information about this mechanism, see 24.2.6, "Store-and-forward " on page 732. In case of a permanent failure, another embedded WebSphere Application Server instance in the network can be promoted to house the Event Processor. This promotion can be achieved by issuing the **`conman switchevtproc`** command. See 24.3.5, "Switch Event Processor" on page 740 for an in-depth discussion of the mechanism.

Like the embedded WebSphere Application Server, the state of the Event Processor can be examined by the **`conman showcpus`** command**.** State 'E' implies that the Event Processor is running on the specific workstation. Similar to the application server, the Event Processor will put a message in the local Mailbox.msg queue when it is started. Next, the local engine will forward the message to the master domain manager, which will distribute it further in the network. As a result, all workstations in the network will be able to determine the location of the Event Processor and update their configuration files.

Note: If Event Processor is switched to another embedded WebSphere Application Server instance, unlinked workstations are not aware of this migration. As a result, connections to the wrong Event Processor instance are attempted.

24.1.5 Tivoli Dynamic Workload Console

As of Tivoli Workload Scheduler version 8.5, running the Tivoli Dynamic Workload Console inside a single Tivoli Workload Automation instance is possible. In case of a failure of the

embedded WebSphere Application Server, the web console will not be accessible. Therefore, be sure to separate the Tivoli Dynamic Workload Console from the embedded WebSphere Application Server instance.

24.1.6 Dynamic Workload Broker Component

Starting with version 8.5.1, the Dynamic Workload Broker Component can be installed as part of the Tivoli Workload Scheduler infrastructure. The workload broker extends the capabilities of the traditional engine by enabling dynamic scheduling. The Dynamic Workload Broker Component runs inside an embedded WebSphere Application Server instance and is installed by default on the master domain manager and backup master domain managers. The workload broker consists of the following important components:

- The *Resource Advisor* component of the workload broker server receives real-time information from the *Resource Advisor Agent* component of the workload broker agents that periodically scan for available physical resources (both hardware and operating system information). Along with logical resources, this data is stored in the *Resource Repository,* which is located in the Tivoli Workload Scheduler database.
- When a dynamic job is submitted, the *Job Dispatcher* interacts with the Resource Advisor and allocates the required resources in the *Allocation Repository*. This repository keeps track of the resources that are assigned to each dynamic job and is also located in the Tivoli Workload Scheduler database.
- The Job Dispatcher on the workload broker server instructs the workload broker agent to launch and monitor the dynamic job.

The sequence outlined shows that the workload broker server is highly dependent on the Tivoli Workload Scheduler database. As a result, if the database or the embedded WebSphere Application Server that is hosting the workload broker server becomes unavailable, no more dynamic jobs are launched. In case of a failure, the Dynamic Workload Broker Component can be migrated to a backup master domain manager in the top-level domain. See 24.2.5, “Dynamic scheduling” on page 731 for an in-depth discussion on the dynamic scheduling internals.

Note: The Dynamic Workload Broker Component must be collocated with the master domain manager. If a failure of the Dynamic Workload Broker Component occurs, the master domain manager role will be migrated also.

24.2 Network communication

Before developing Tivoli System Automation for Multiplatforms policies for Tivoli Workload Scheduler, a deeper understanding of how the network operates and how the workstation interprocess communication works internally are required. This section will discuss the details of the building blocks in a Tivoli Workload Scheduler network and their mutual relationships. The integration of *Event Driven Workload Automation* and the Dynamic Workload Broker Component will also be dealt with thoroughly. The study of these critical components and their common interaction is important when developing the intelligence of the automation policies.

24.2.1 Engine initialization

When the plan is extended, the entire network is restarted and the new Symphony file is distributed to all workstations and subordinate domain managers. The following operations take place during network initialization:

1. On the domain manager, the *mailman* process examines the Symphony file and learns which workstations it needs to connect to.
2. For each discovered fault tolerant agent (FTA), the mailman process on the domain manager establishes a TCP/IP connection to the remote *netman* process on the workstation.
3. The netman process on the FTA notices that the request is originating from a remote mailman process on the domain manager and subsequently creates a *writer* child process to handle the incoming connection.
4. The spawned writer process on the workstation examines the local Symphony file (if it exists) and reports the current run number back to the mailman process on the domain manager. If *wr enable compression* is set to *yes* in the `localopts` configuration file, compression of the Sinfonia file is also requested by the writer process on the FTA.
5. The mailman process on the domain manager compares its Symphony run number with the run number of the Symphony file on the workstation. If the run numbers differ, the mailman process on the domain manager sends the Sinfonia file to the writer process on the FTA using the unidirectional link.
6. The writer process on the workstation decompresses the Sinfonia file, if needed, and copies it to Symphony.
7. The mailman process on the domain manager issues a start operation for the workstation from the netman process on the FTA.
8. The netman process on the workstation starts the local mailman process which in turn examines the local Symphony file. Subsequently, the local mailman process on the FTA discovers its domain manager and links back to it, resulting in a writer process for that workstation on the domain manager.
9. Finally, the *batchman* process on the FTA is started by mailman and the *jobman* process is started by batchman.

Figure 24-1 on page 725 shows a graphical representation of these operations.

Archive

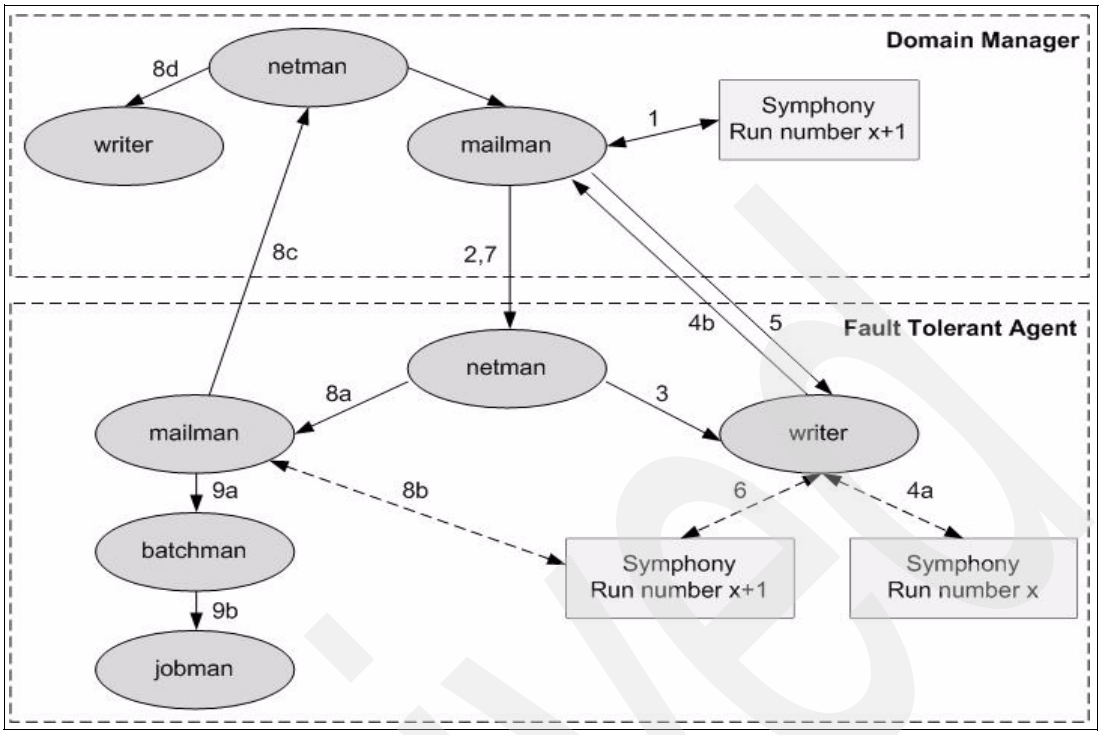


Figure 24-1 Engine initialization

Notes:

- To balance the load, there can be multiple mailman processes on the domain manager. The main mailman process on the domain manager remains in charge of distributing the Sinfonia file to all workstations and subordinate domain managers. The use of additional mailman processes might reduce the time to initialize a workstation.
- The sequence assumes that the workstation has the autolink flag set in its definition. See 24.2.6, “Store-and-forward ” on page 732 for more details about autolink.

24.2.2 Workstation interprocess communication

Now that the plan has been distributed, each workstation can start processing jobs. The following sequence of events occurs on a workstation throughout the production day:

1. The batchman process periodically checks the Symphony file to determine which jobs are ready to run. The interval used by the batchman process is controlled by the *bm look* localopts parameter.
2. If a job is ready to run, the batchman process writes a record in the *Courier.msg* message queue. The jobman process periodically scans Courier.msg for new requests. The interval used by the jobman process is regulated by the *jm read* localopts tunable.
3. The jobman process discovers the new message in the Courier.msg message queue. Subsequently, the requested job is launched and an entry is written in the *Mailbox.msg* queue indicating the job has started. The mailman process periodically scans Mailbox.msg for new messages. The time-out used by the mailman process is governed by the *mm read* localopts parameter. The jobman process will monitor the progress of the executing job. The interval used by the jobman process is controlled by the *jm look* localopts parameter.

4. The mailman process detects the new entry in the Mailbox.msg queue and puts a record in both the Mailbox.msg on the domain manager and the local *Intercom.msg* queue. The local batchman process periodically checks the `Intercom.msg` queue for new records. The interval used by the batchman process is regulated by the *bm read* localopts tunable. The mailman process on the domain manager propagates the message to remote writer processes in the network that are interested in the event.
5. The batchman process retrieves the message in the Intercom.msg queue and updates the local copy of the Symphony file indicating the job is executing.
6. When the job finishes, the jobman process puts a job termination record in the `Mailbox.msg` queue.
7. The mailman process discovers the entry in `Mailbox.msg` and puts a message in both the `Mailbox.msg` on the domain manager and the local Intercom.msg queue. Again, the mailman process on the domain manager further propagates the change in the network.
8. The batchman process retrieves the message in Intercom.msg and updates the Symphony file indicating the job has finished.

Figure 24-2 gives a schematic overview of the steps.

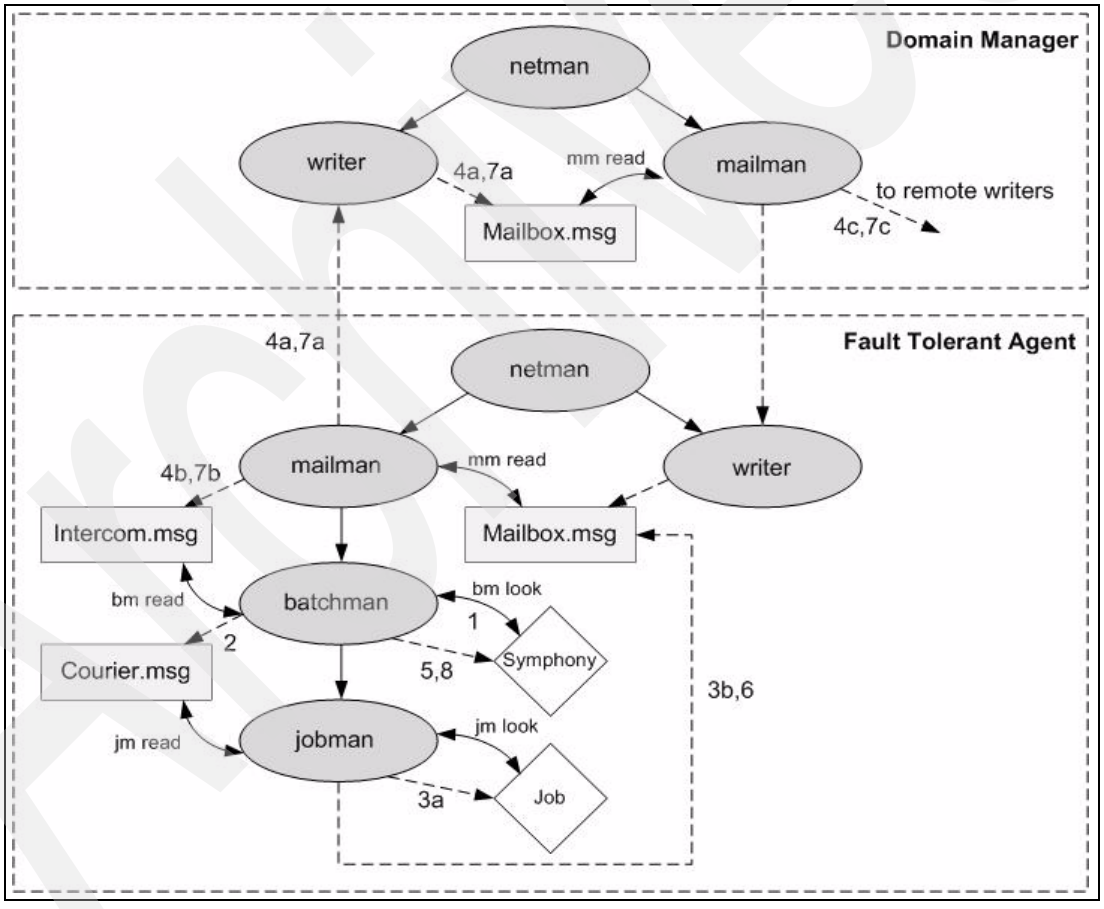


Figure 24-2 Workstation interprocess communication

Archived

Notes:

- The writer process on the domain manager receives messages from the remote mailman process on the FTA and puts them into the local `Mailbox.msg`queue. The writer process on the workstation receives messages from a remote mailman process on the domain manager and puts them into the local `Mailbox.msg`. Together, a bidirectional link is formed in which state messages are exchanged.
- The *mm read* tunable is currently not present in the `localopts` file. APAR IZ74286 was created to address this issue. The default is 15 seconds.

24.2.3 Job execution

When batchman instructs jobman to launch a job, the following events take place:

1. The jobman process reads the BL record from Courier.msg message queue and forks a copy of itself. The effective user ID of the forked child process (Job Monitor) will be the STREAMLOGON user that is defined for the job.
2. The spawned jobman process forks again. The child process performs an execve system call to run the **`jobmanrc`** script. The script or command that must be executed, is passed as an argument to the **`jobmanrc`** script.
3. If a local configuration **`jobmanrc`** script exists for the user and its use is allowed, it is also executed.
4. The job is launched.
5. Upon successful completion (the job's return code matches the return code condition, or zero if no condition is defined), the jobman process puts a JS (Job Successful) message in the `Mailbox.msg` message queue.
6. Regardless of the return code, a JT (Job Termination) message is written in the `Mailbox.msg` queue.

Figure 24-3 summarizes the events graphically.

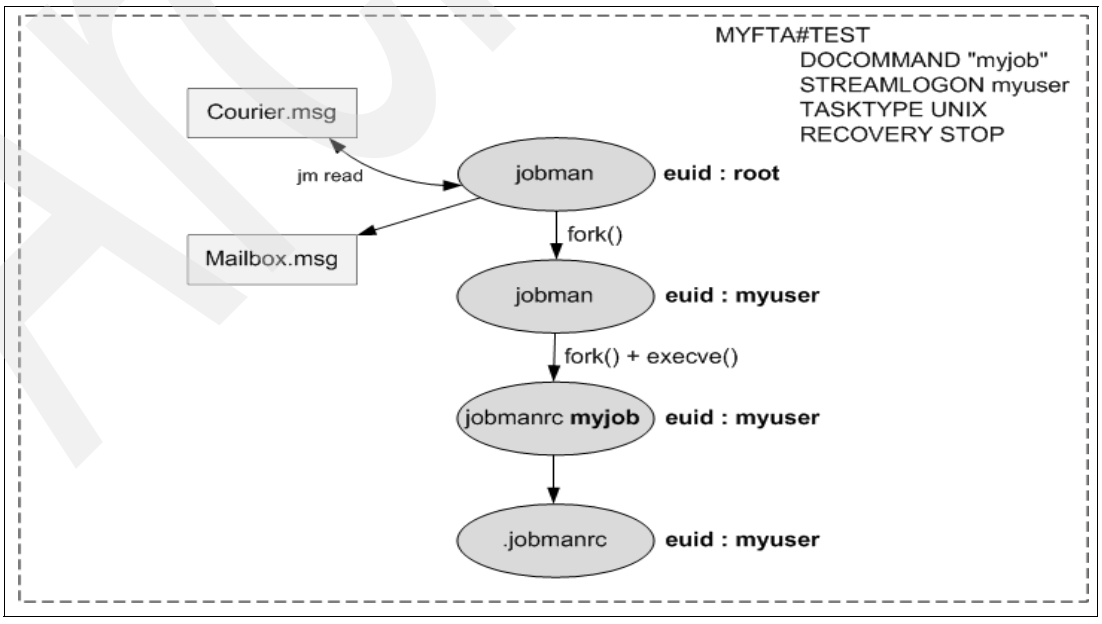


Figure 24-3 Job execution details

Notes:

- If the job ends unsuccessfully, no JS record is written in the `Mailbox.msg` queue. The job will be marked as ABEND.
- The jobman executable is a **`setuid`** root program.
- The maximum number of concurrent Job Monitor processes is governed by the CPU limit of the workstation.

24.2.4 Event driven workload automation

After an event rule is saved in the database and marked as complete, it will be activated and deployed by the Rule Builder component of the Event Processor. The deployment process can be initiated either manually, using the **`planman deploy`** command, or automatically, which is governed by the *deploymentFrequency* global option. The deployment process consists of the following steps:

1. The Rule Builder generates configuration files for each specific workstation and compresses them. A dedicated compressed file is created for each FTA. The event rule is activated in the Event Correlation Engine by the Rule Builder.
2. The *monman* process on the affected workstation is notified about the new available configuration.
3. The monman process downloads the compressed file using the HTTP/HTTPS protocol.
4. The monman process decompresses the content of the `deployconf.zip` file into the monconf subfolder on the FTA.
5. The monman process starts the *ssmagent* process.

Figure 24-4 illustrates the deployment process.

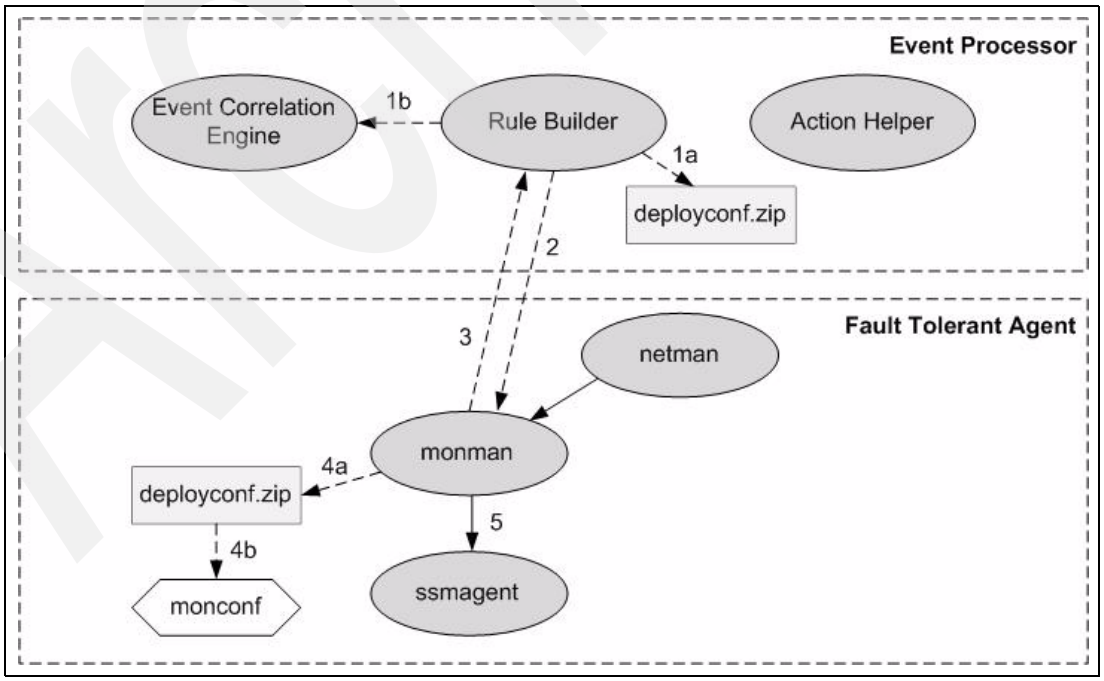


Figure 24-4 Event Driven Workload Automation: deployment process

Note: Although the ssmagent process is started by the monman process, it is not a child of monman process.

The communication mechanism employed for the exchange of status messages between the Event Processor and the monitoring agents relies on the *Event Integration Facility (EIF),* which originates from Tivoli Enterprise Console. Note that the communication mechanism that is used by the EIF has no affinity with traditional Tivoli Workload Scheduler engine communication. As a result, event rules can still be deployed or triggered even if the workstation is unlinked.

Tivoli Workload Scheduler Object Monitor

The Tivoli Workload Scheduler Object Monitor built-in event provider is responsible for monitoring specific events related to the current plan, which are categorized as internal events. The objects that can be monitored include workstations, jobs, job streams, prompts and application server instances. These objects are monitored on the following locations in the network:

- Job streams are monitored on the master domain manager.
- Jobs are monitored by the workstation on which they run. If a job is run on a standard agent, the status will be monitored by the corresponding domain manager. If a job is run on an extended agent, the status will be monitored by the host on which the extended agent runs.
- A workstation is monitored by itself.
- Global prompts are monitored by the master domain manager.
- Local prompts are monitored by the workstation that is responsible for the job of job stream that has a dependency on it.
- An application server is monitored by the workstation on which it is running.

State changes for jobs, job streams, and prompts are monitored by the local batchman process. Workstation state changes are watched by the mailman process and the state of an application server is examined by the appservman process. All events are written to the `Monbox.msg` message queue, which is read by the monman process. Finally, the monman process forwards the events to the Event Correlation Engine on the Event Processor.

Archived

Figure 24-5 displays all components on a single workstation.

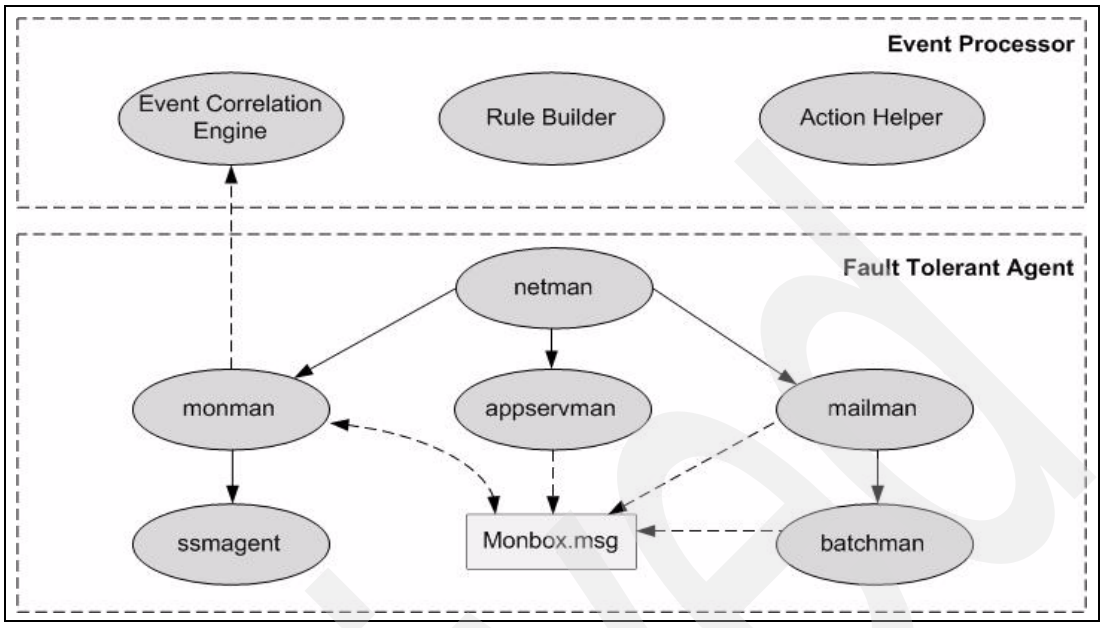


Figure 24-5 Tivoli Workload Scheduler Object Monitor

File Monitor

The File Monitor built-in event provider delivers file-based monitoring on the workstation where it is deployed. These events are classified as external events. The Netcool SSM (System Service Monitors) agent provides several subagents which are capable of monitoring and parsing text files. These subagents generate events, which are dispatched to the Event Correlation Engine on the Event Processor, as illustrated in Figure 24-6.

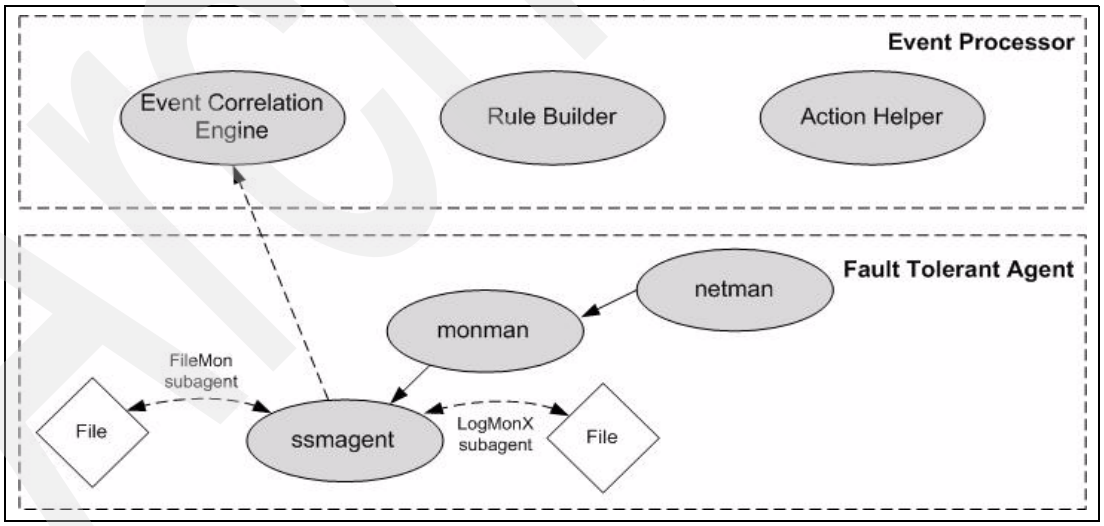


Figure 24-6 File Monitor

Note: The monman process merely delivers an interface to control the ssmagent process (start/stop). There is no affinity between these processes and each one provides distinct monitoring functions.

Archived

24.2.5 Dynamic scheduling

The Dynamic Workload Broker Component extends the capabilities of the traditional Tivoli Workload Scheduler engine by allowing a dynamic election of the workstation on which a job can run. The corresponding Tivoli Workload Scheduler job is launched on the workload broker server, which emulates a standard agent in the Tivoli Workload Scheduler network. The following steps explain briefly how a dynamic job is launched:

1. The Resource Advisor Agent component of the *JobManager* process periodically scans the local system for physical resources. The interval used by the JobManager process is regulated by the *ScannerPeriodSeconds* in the `JobManager.ini` configuration file. The Resource Advisor Agent will send the result from the scan to the Resource Advisor on the workload broker server every *NotifyToResourceAdvisorPeriodSeconds* seconds.
2. When a dynamic job is submitted, the Job Dispatcher interacts with the Resource Advisor to find the most suitable candidate for running the dynamic job. After a candidate is found, the job submission request is sent to it. The JobManager process on the elected agent launches a *taskLauncher* process.
3. The taskLauncher process launches and monitors the job.
4. After the dynamic job has completed, the taskLauncher process informs the JobManager process about the exit code.
5. Finally, the JobManager process reports the status of the dynamic job back to the Job Dispatcher component of the workload broker server.

Figure 24-7 illustrates the sequence graphically.

Important: The regular Tivoli Workload Scheduler network is not used by the Dynamic Workload Broker Component for delivering messages. As a result, dynamic jobs can be launched even if the workstation is unlinked.

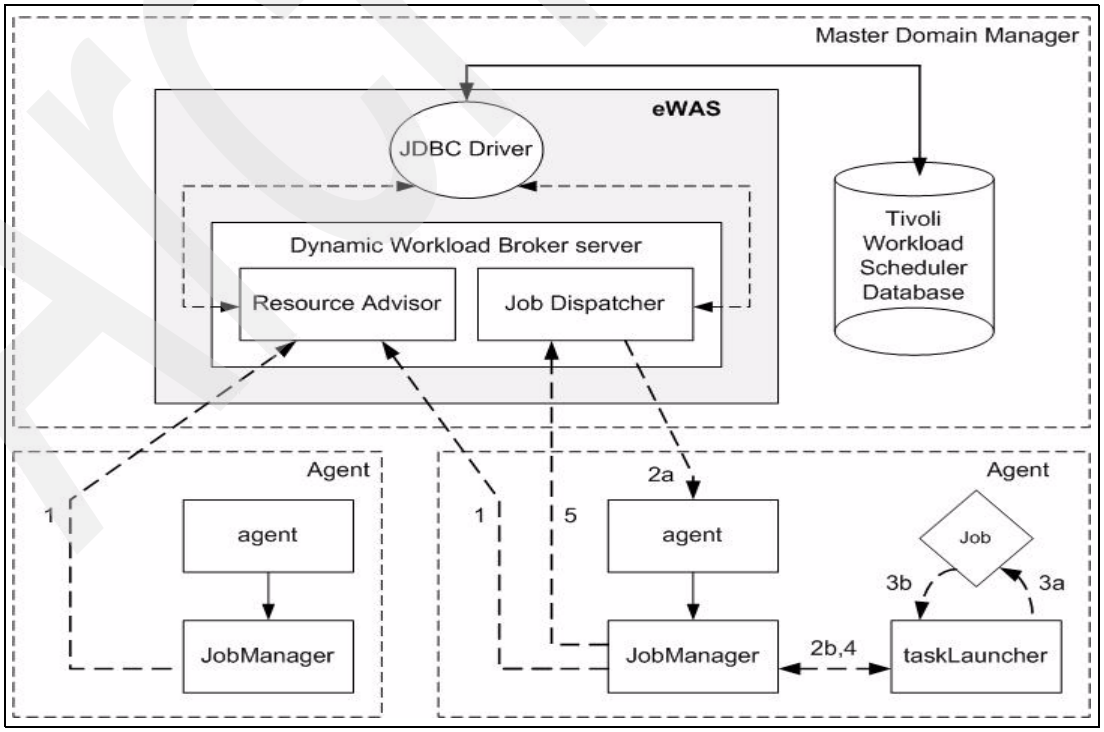


Figure 24-7 Dynamic scheduling

24.2.6 Store-and-forward

To guarantee a consistent network, Tivoli Workload Scheduler employs a store-and-forward mechanism by queuing messages while a TCP/IP connection is not available. If the link with the domain manager is not available, the workstation queues outgoing messages in the `tomaster.msg` message queue located in the pobox subdirectory. If a link with a workstation is broken, the domain manager queues outbound messages in a `XYZ.msg` file in the `pobox` subfolder, where `XYZ` is name of the unlinked FTA.

An important aspect in the store-and-forward technology is the opening and closing of links, which is mainly controlled by the autolink flag in the workstation definition. This flag indicates whether the domain manager should open the link to the FTA when the domain manager starts. As mentioned, this will be important when the plan is extended, which causes a restart of the entire network. Moreover, if a workstation is stopped gracefully, its link to the domain manager will be closed because the mailman process on the FTA is halted. As a result, the corresponding writer process on the domain manager for this particular workstation will also cease to exist. However, the link from the domain manager to the FTA remains open until one of the following situations occurs:

- The mailman process on the domain manager times out the connection to the workstation. As a result of this unlink operation, the writer process on the FTA is halted. The time-out used by the mailman process on the domain manager is regulated by the *mm unlink* localopts tunable.
- A manual unlink operation is issued on the domain manager.

Alternatively, if the link from the domain manager is closed unexpectedly (such as by killing the writer process on the FTA), the mailman process on the domain manager tries to reestablish the link to the workstation. The interval that is used by the mailman process is governed by the *mm retrylink* localopts parameter. If the workstation is unlinked manually, the remote mailman process does not try to restore the link.

The event driven workload automation framework has store-and-forward techniques similar to the ones of a regular Tivoli Workload Scheduler network. If the Event Processor cannot be contacted, the monitoring agents will queue outbound messages for a later delivery. Each monitoring agent uses a separate queue. The monman process uses the `monmancache.dat` file in the EIF subfolder; the ssmagent process employs the `ssmcache.dat` file. The **`sendevent`** command, which is used to generate events from the command line, has its own queue `sendeventcache.dat`. Figure 24-8 and Figure 24-9 on page 733 each show a view of all store-and-forward queues during a network outage.

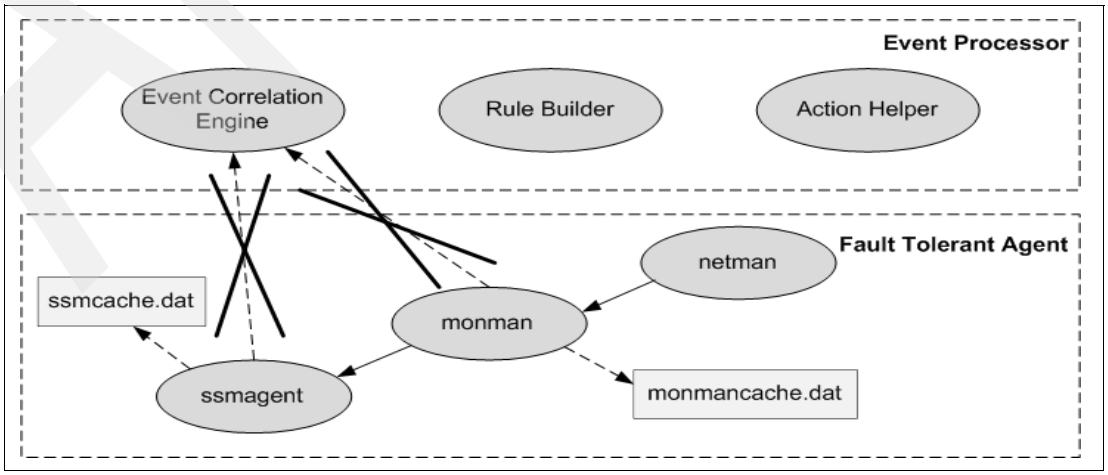


Figure 24-8 Store-and-forward Event Driven Workload Automation

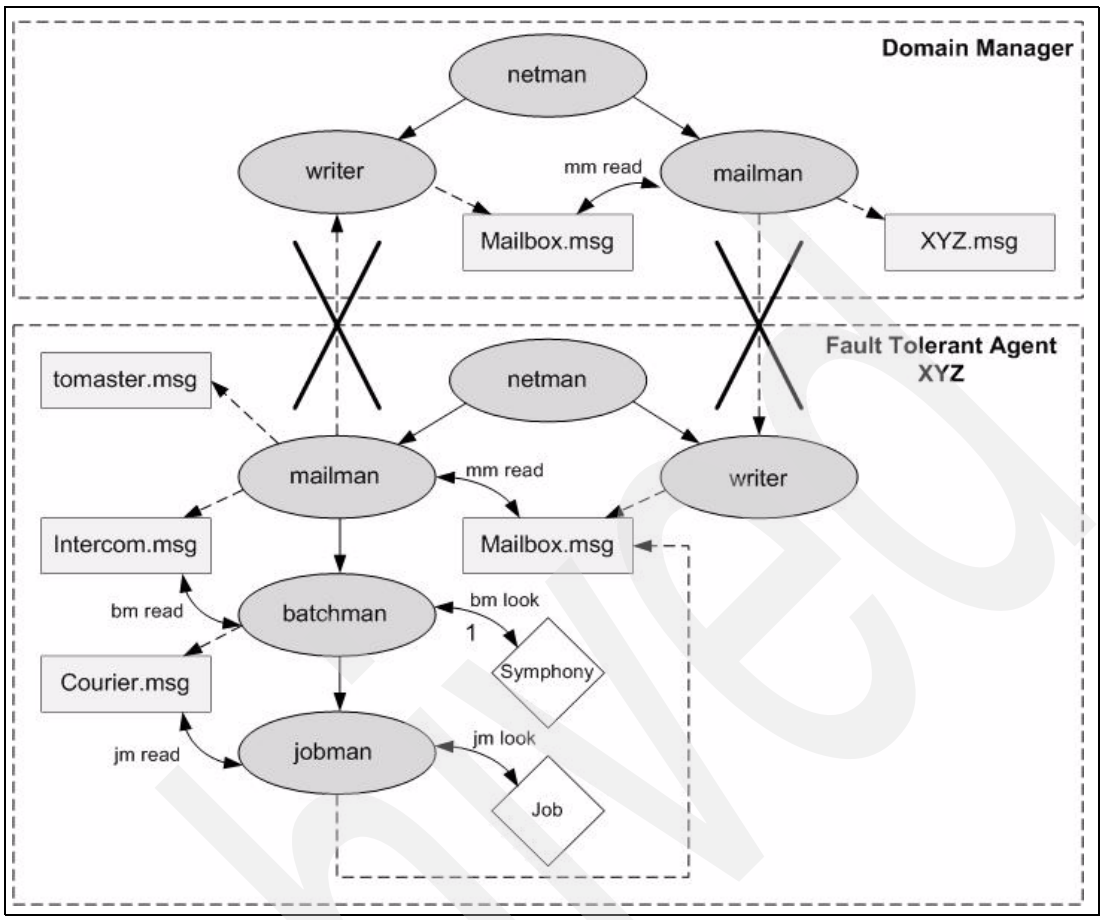


Figure 24-9 Store-and-forward Tivoli Workload Scheduler

24.3 High availability

This section will concentrate on the concepts of high availability features for the master domain manager, which are delivered ready to use. To guarantee a transparent failover to a backup master domain manager, these features will be used by the automation policies. The use of DB2 as backend RDBMS will be assumed.

24.3.1 DB2: ACR and HADR

By default, DB2 includes several features that can be used to improve availability. The most important are *automatic client reroute (ACR)* and *high availablity disaster recovery (HADR)*:

- ACR enables the DB2 JDBC driver in the embedded WebSphere Application Server to recover from a loss of communication with the primary database instance by rerouting the connection to an alternate server. This connection rerouting occurs automatically without any manual intervention. However, ACR does not synchronize the databases between the primary and standby DB2 servers.
- HADR is a database replication technology which provides high availability and disaster recovery for both partial and complete site failures. Both ACR and HADR can be combined to achieve a higher degree of availability.

Archive

Important: If ACR and HADR are combined, be sure to verify that the initial connection is pointing to the primary database. If the primary and standby databases switch roles, the embedded WebSphere Application Server data source properties should be updated accordingly.

ACR

The location of the alternate DB2 server is propagated to the JDBC driver when the embedded WebSphere Application Server makes an initial connection to the primary DB2 server. If the communication between the connector and the database server is interrupted, the JDBC driver on the embedded WebSphere Application Server attempts to reestablish the connection automatically by using the alternate server information. The following sequence occurs when the embedded WebSphere Application Server is started:

1. The JDBC driver of the embedded WebSphere Application Server connects to the primary DB2 server.
2. The primary DB2 server returns information about the standby DB2 server to the JDBC driver.
3. The JDBC driver caches this information in memory.
4. If a network interruption or failure of the primary DB2 server occurs, the JDBC driver of the embedded WebSphere Application Server automatically establishes a connection to the standby DB2 server.

Figure 24-10 depicts the sequence.

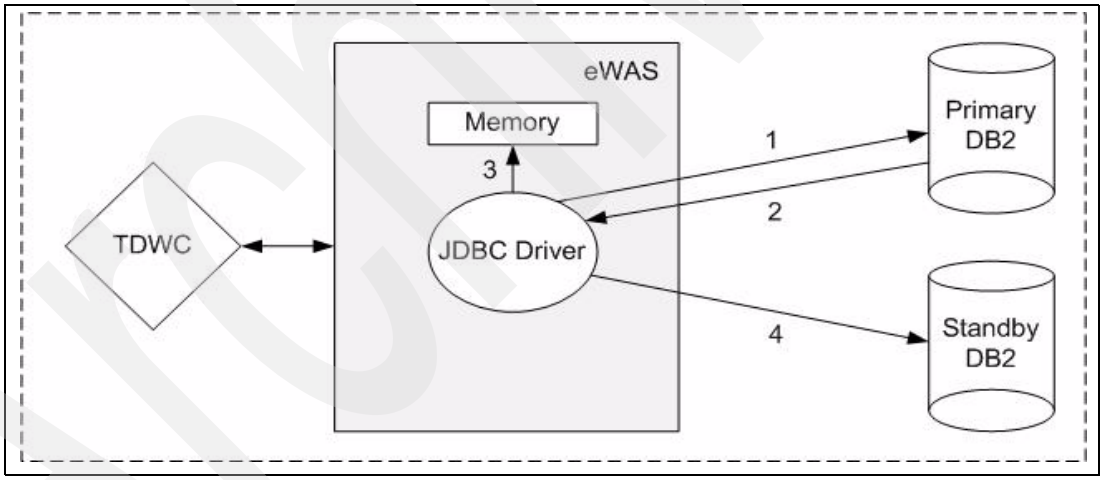


Figure 24-10 DB2 ACR

Note: ACR is enabled, by default, in the JDBC driver. No additional configuration is required on the client side. On the server side, the alternate server must be defined in the database directory.

HADR

DB2 HADR is a database replication mechanism in which log records use TCP/IP to transmit from the primary database server to a standby server. The standby server is in a continuous roll-forward mode and replays all log records to its copy of the database, keeping it tightly synchronized with the primary instance. The standby database in a HADR pair cannot be used by clients, not even for read-only access. If the primary database is not available, the

standby database can take over its role. Be sure to either manually perform this operation or employ an automation tool.

DB2 HADR delivers three modes synchronization modes, each providing a level of protection:

- SYNC

  In synchronous mode, log writes are considered successful only when the following situations occur:

  – Logs are written to log files on the primary database.

  – The primary database has received an acknowledgement from the standby database that the logs have successfully been written to log files.

  Using this mode, ensures no loss of data if the HADR pair is synchronized.

- NEARSYNC

  In near synchronous mode, log writes are considered successful only when the following situations occur:

  – Log records are written to log files on the primary database.

  – The primary database has received an acknowledgement from the standby database that the logs are in main memory.

  If the primary database goes down and the standby also fails before takeover is completed, there will be loss of data, though this is unlikely to occur.

- ASYNC

  In this mode, the primary database does not wait for an acknowledgement from the standby database. Log writes are considered successful only when the following situations occur:

  – Log records are written to log files on the primary database.

  – Log records have been delivered to the TCP/IP stack on the primary database. No acknowledgement is expected from the standby database.

  Data loss is more likely to happen when using this synchronization mode.

Figure 24-11 illustrates the HADR synchronization modes and the point at which DB2 commits the transaction.

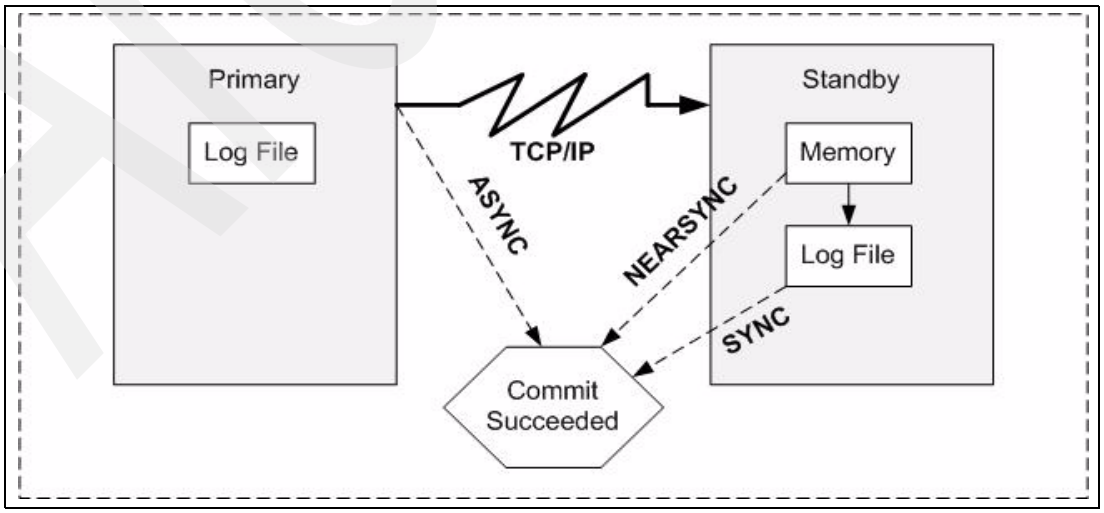


Figure 24-11 DB2 HADR synchronization modes

24.3.2 Fault tolerance

If the TCP/IP connection between the domain manager and the workstation is interrupted during the production day, the workstation is still able to submit local workloads to the workstation because it has a copy of the Symphony file. As described in 24.2.6, “Store-and-forward ” on page 732, the local workstation queues the message indicating the job has finished in the `tomaster.msg` queue if the connection with the domain manager is broken. When the connection is restored, the message is forwarded to the domain manager, which in turn broadcasts the message to other workstations that are interested in the event. When using dynamic scheduling, the concepts of fault tolerance differ because there is no production plan on the local workstation for dynamically submitted workloads. As illustrated in Figure 24-7 on page 731, the Job Dispatcher on the workload broker server contacts the agent synchronously if a job needs to run on the workstation. Therefore, while the communication between the workload broker server and the broker client is broken, the job does not start on the client.

The following example scenario illustrates the subtle differences between both technologies (dynamic and non-dynamic scheduling). Assume that job ABC, which has no predecessor dependencies, must run at 8:00 p.m. on workstation XYZ. Job ABC will take five minutes to complete. At 7.50 p.m., a planned network outage will last for 20 minutes. If job ABC is a traditional Tivoli Workload Scheduler job, the following chronological events take place:

1. (7:50 p.m.) The TCP/IP connection between the domain manager and XYZ is broken. Workstation XYZ is unlinked from the domain manager.
2. (8:00 p.m.) Workstation XYZ starts the job because it is in the local Symphony file. The state of the job in the local Symphony file changes from `HOLD` to `READY/INTRO+/EXEC+`.
3. (8:05 p.m.) Job ABC ends successfully. The state of the job in the local Symphony file is updated to `SUCC`. The state of the job in the Symphony file on the domain manager remains in the state `READY`.
4. (8:10 p.m.) TCP/IP connectivity between the domain manager and XYZ is restored.
5. (8.10 p.m. - 8:20 p.m.) After a maximum of `mm retrylink` seconds (default 600), the link with the domain manager is restored. The contents of the `tomaster.msg` queue on workstation XYZ, which contains the state changes of job ABC, is transferred to the domain manager.
6. (8:10 p.m. - 8:20 p.m.) The domain manager updates the local Symphony file and broadcasts the successful completion of job ABC to workstations that are interested in the event.

Figure 24-12 summarizes the events.

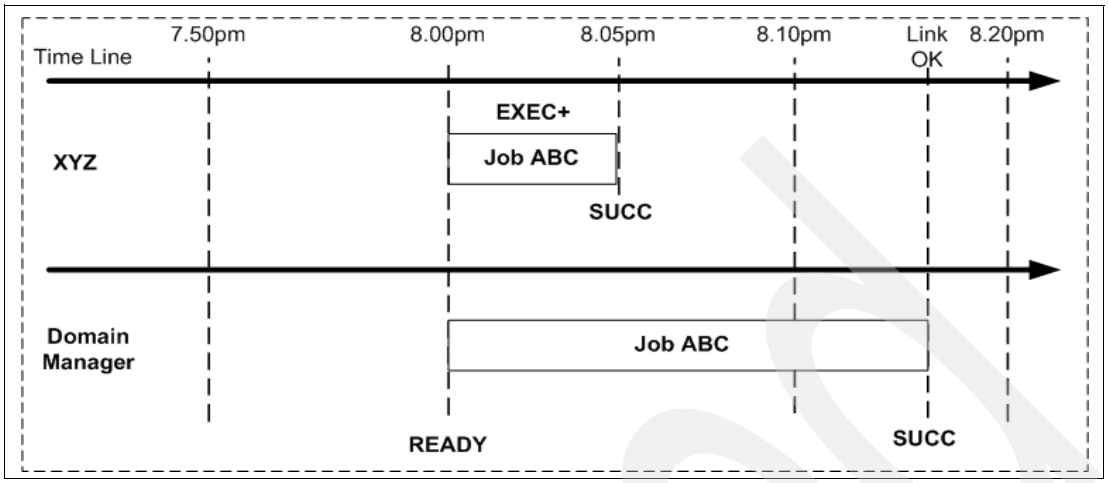


Figure 24-12 Fault tolerance I

If job ABC is a dynamic job that is bound to computer XYZ, the following chronological events take place:

1. (7:50 p.m.) The TCP/IP connection between the dynamic workload broker server and computer XYZ is interrupted.
2. (7:56 p.m. - 7:59 p.m.) From 360 seconds (`MissedHeartBeatCount * NotifyToResourceAdvisorAgentSeconds`) to 540 seconds (`(MissedHeartBeatCount +1) * NotifyToResourceAdvisorAgentSeconds`) after the network interruption, computer XYZ is marked `Unavailable` in the dynamic workload broker server. The `MissedHeartBeatCount` tunable (default 2) in `ResourceAdvisorConfig.properties` specifies the number of missed heartbeats after which a computer is listed as `Unavailable`.
3. (8:00 p.m.) The Tivoli Workload Scheduler engine on the master domain manager will instruct the dynamic workload broker server to start the job. Because the broker server cannot contact the broker client, the dynamic job will enter the state `Waiting for resources` for a maximum of `MaxWaitingTime` seconds. The global MaxWaitingTime tunable in ResourceAdvisorConfig.properties on the dynamic workload broker specifies the number of seconds a dynamic job can wait for a resource to become available. The state of the corresponding Tivoli Workload Scheduler job changes from `HOLD` to `WAIT`.
4. (8:10 p.m.) TCP connectivity between the dynamic workload broker server and XYZ is restored.
5. (8:10 p.m. - 8:13 p.m.) After a maximum of `NotifyToResourceAdvisorPeriodSeconds` seconds (default 180), the broker client will be able to connect to the broker server again. As a result, the dynamic workload broker server updates the state of computer XYZ to `Available`. Next, the Job Dispatcher will dispatch the job to computer XYZ. The state of the dynamic job changes to `Running` whereas the state of the corresponding Tivoli Workload Scheduler job changes to `EXEC+`.
6. (8:15 p.m. - 8:18 p.m.) After five minutes of run time, job ABC ends successfully. The state of the dynamic job is updated to `Completed Successfully`; the state of the corresponding Tivoli Workload Scheduler job is changed to `SUCC`.

Figure 24-13 illustrates the events.

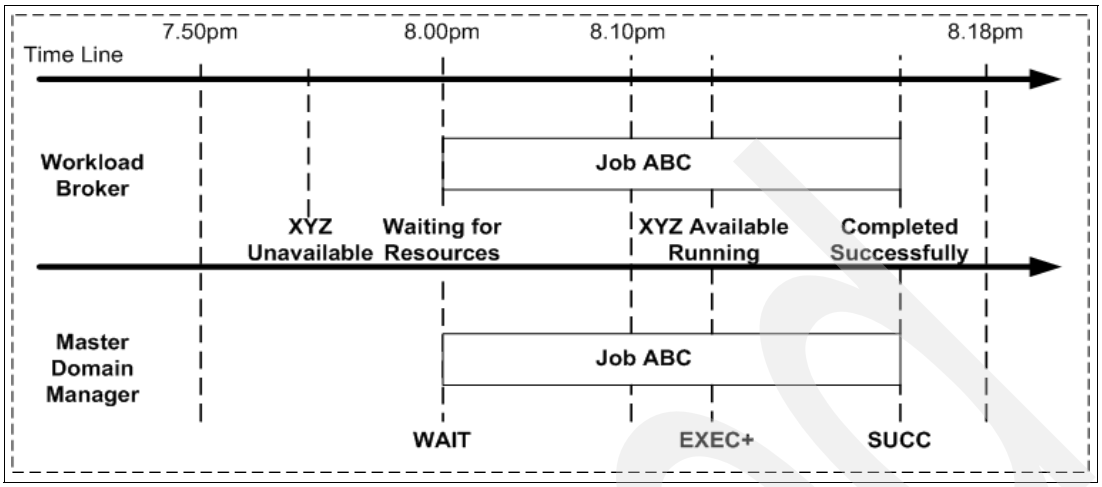


Figure 24-13 Fault tolerance II

Important: Alternatively, the maximum resource waiting time for a dynamic job can be defined in the JSDL. If defined, the global tunable `MaxWaitingTime` in the `ResourceAdvisorConfig.properties` on the workload broker server is overridden.

24.3.3 Switch Fault Tolerance

In fullstatus mode, a workstation is updated with the status of jobs and job streams running on all other FTAs in its domain and in subordinate domains. As a result, the Symphony file on the fullstatus FTA contains the same information as the Symphony file on the domain manager, which acts as a message broker.

As described in 24.2.6, “Store-and-forward ” on page 732, each workstation stores outgoing messages in the tomaster.msg message queue if the domain manager is not available. If the domain manager is available, the mailman process on the workstation sends the message to the writer process on the domain manager, which will put the message in the local `Mailbox.msg` queue. Subsequently, the mailman process on the domain manager retrieves the record and forwards it to all workstations that are interested in that particular event, including all fullstatus agents. In the unlikely event that the domain manager fails before broadcasting the message to the fullstatus agents in the domain, the following situations might occur:

- The message is located in the `Mailbox.msg` queue on the domain manager, which is temporary unavailable. As soon as the domain manager is available again, it forwards the message.
- The message is located in the `Mailbox.msg` queue on the domain manager, which is permanently unavailable. The message is lost.

To overcome the possible loss of messages, *Switch Fault Tolerance* can be enabled by issuing **`optman chg sw=YES`**. The change will become effective when the Symphony plan is extended or created. If enabled, messages are no longer routed from the domain manager to fullstatus agents. When a workstation sends a message to its domain manager, it also sends the event to all fullstatus agents in the domain. If the FTA is unable to deliver the message to any of them, the event is stored in the `pobox` subdirectory on the workstation. When a fullstatus agent receives a direct message from the workstation, the local Symphony is

updated and the event is buffered in a cyclical queue in the `ftbox` subdirectory, which acts as a recovery queue. As a result, all processed or partially processed messages are stored on at least two workstations. Events can be resent and reprocessed, thus eliminating the single point of failure of communication from the domain manager to fullstatus agents in the same domain.

Figure 24-14 shows the setup when Switch Fault Tolerance is enabled.

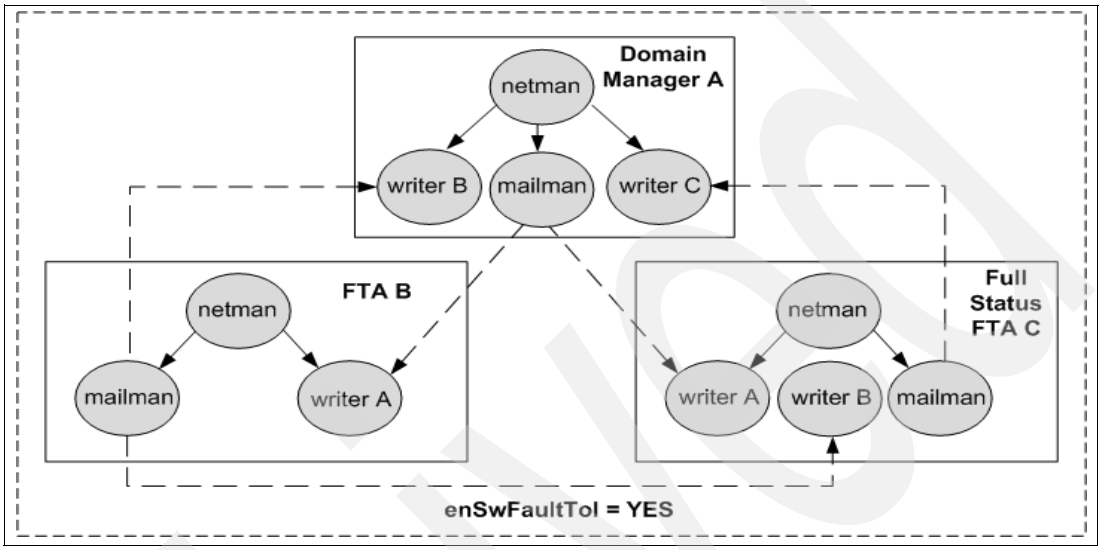


Figure 24-14 Switch Fault Tolerance

Tip: Limit the number of Full Status agents in the domain because having a high number of agents implies increased network traffic.

24.3.4 Switch Manager

If the domain manager is unavailable, any fullstatus agent in the domain can take over its role because the Symphony file on a fullstatus agent has the same content as the Symphony file on the domain manager. The fullstatus agent will be restarted and assume the role of the domain manager until the next plan extension, often referred as a *short term switch*. Subsequently, the old domain manager will be converted to a fullstatus workstation in the domain. The mechanism consists of the following steps and can be triggered from any FTA in the domain:

1. The **`conman switchmgr DOMAIN;XYZ`** command is issued on a workstation in the domain. XYZ is the name of the target workstation that will become the new domain manager of the domain.
2. The FTA establishes a TCP/IP connection to the netman process on the fullstatus agent XYZ, requesting the workstation to become the new domain manager.
3. Fullstatus workstation XYZ is restarted and becomes the new domain manager. A special message will be put in the local `Mailbox.msg` message queue indicating the domain manager switch.
4. The new domain manager XYZ links to all workstations in the domain and delivers the message.

5. On each workstation in the domain, the mailman process retrieves the event in the `Mailbox.msg` queue, unlinks from the old domain manager and relinks to the new domain manager XYZ. Messages that are stored in the `tomaster.msg` queue are transferred to the new domain manager XYZ. Moreover, the local batchman process is also notified to update the local Symphony file using the Intercom.msg queue.
6. The batchman and jobman process on the old domain manager are restarted.

The next time the plan is extended, the domain acts as though another **`switchmgr`** command was run and the old domain manager automatically resumes domain management responsibilities. If the old domain manager will not be available by the next plan extension, additional actions must be executed. The following steps are of a *long term switch*:

1. The definition of the *old* domain manager must be altered in the database. Its role should change from MANAGER to FTA.
2. The definition of the *new* domain manager must be altered in the database also. Its role needs to change from 'FTA' to 'MANAGER'.

If the affected domain is the top-level domain, the following additional steps are required to rectify the plan extension:

1. The old FINAL job stream needs to be cancelled.
2. The definitions of jobs contained in the FINAL job stream need to be linked to the new domain manager.
3. The updated FINAL job stream needs to be submitted to the plan.

This procedure is quite cumbersome and can be optimized. See 24.4, "Virtualization" on page 741 for a detailed discussion.

Important: The network is not restarted during a **`switchmgr`** operation.

24.3.5 Switch Event Processor

Any workstation in the top-level domain that is running an embedded WebSphere Application Server instance is eligible for hosting the Event Processor. If a problem occurs, the Event Processor's function can be easily migrated by running the **`conman switchevtproc`** command. The mechanism consists of the following steps:

1. The **`conman switchevtproc XYZ`** command is issued on workstation ABC in the network. XYZ is the name of the target FTA that will host the Event Processor. DEF is the name of the workstation that is currently hosting the Event Processor.
2. FTA ABC connects to the netman process on workstation XYZ, requesting it to become the Event Processor.
3. Subsequently, FTA XYZ connects to the netman process on workstation DEF, requesting it to release its role as the Event Processor.
4. FTA ABC puts a special message in the local `Mailbox.msg` message queue, indicating that workstation XYZ has become the new Event Processor.
5. The message is delivered to the upper domain manager and broadcasted on the entire network.

Archived

6. On each workstation in the network, the mailman process retrieves the event in the `Mailbox.msg` and asks batchman through the Intercom.msg queue to update the local Symphony file. Moreover, the mailman process will also put a message in the local `Moncmd.msg` message queue, which is examined by the monman process.
7. The monman process picks up the message in the `Moncmd.msg` queue and updates the location of the Event Processor in the configuration files.

Figure 24-15 presents a schematic overview.

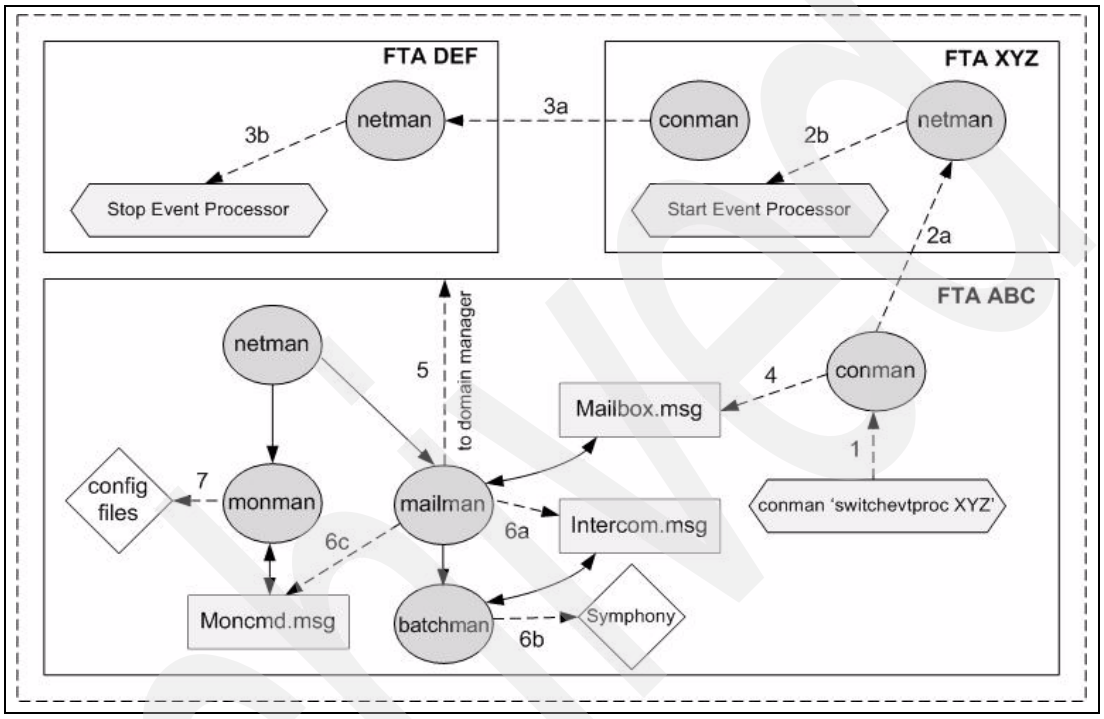


Figure 24-15 Switch Event Processor

Tip: Before running the **`conman switchevtproc`** command, deploy the event rules on the old Event Processor by using the **`planman deploy`** command. As a result, the latest changes will be deployed before the Event Processor is switched.

24.4 Virtualization

After the product is installed, the **`composer add Sfinal`** command adds the FINAL stream and associated jobs to the database. These objects are linked to the master domain manager workstation. In case the master is running on UNIX, the FINAL stream can be virtualized. As a result, the long-term switch operation can be simplified significantly.

24.4.1 The $MASTER keyword and unixlocl method

When defining an extended agent, the $MASTER keyword can be used to indicate that the agent's host workstation is the master domain manager. If the role of the master is migrated to another workstation, the $MASTER keyword automatically sees the new master domain manager.

Instead of defining multiple FTAs on the same UNIX node, the *unixlocl* extended agent access method can be used to virtualize workload. Its use implies reduced network traffic, improved network initialization and less CPU consumption of the Tivoli Workload Scheduler processes on the node, as shown in Figure 24-16.

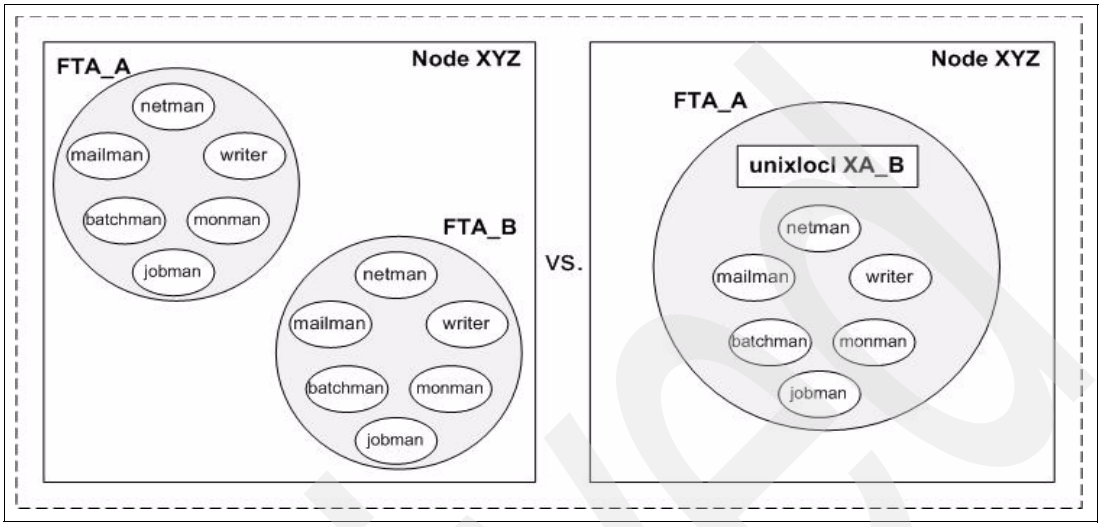


Figure 24-16 unixlocl access method

When launching a job on a unixlocl extended agent, the access method starts by running the standard configuration script (`.jobmanrc`). If the job's logon user is allowed to use a local configuration script (`.jobmanrc`),it is also run. Finally, the job is executed.

24.4.2 The mm resolve master

The localopts tunable *mm resolve master* controls if the $MASTER keyword is expanded to the real value of the master domain manager by the mailman process. When set to 'yes', the $MASTER keyword is resolved when the new plan is distributed. As a result, any extended agent that is linked to $MASTER switches to the new master domain manager when the network is restarted. When set to `no`, the keyword is not translated, resulting in an immediate switch for extended agents linked to $MASTER, triggered by a **`switchmgr`** operation.

Be careful when using $MASTER because it has a limitation. However, for this limitation, an enhancement request has been created. When the network is restarted during the plan extension, the jobman process will look at the jobtable to examine which jobs are still running. Only jobs that were started on FTAs and standard agents are currently taken into account. If a job was started on an extended agent linked to $MASTER prior to a switchmgr operation, the job will be marked ABEND immediately after the plan extension even though it is still running smoothly on the old master domain manager. Enhancement request MR011007250 was created to address this limitation.

24.4.3 FINAL stream

Using the $MASTER keyword and the unixlocl access method, the standard FINAL stream can now be modified as follows:

1. Define an extended agent linked to $MASTER using the unixlocl access method, as shown in Example 24-3.

Example 24-3 Virtual Master

```
CPUNAME XA_VMDM
  OS OTHER
  NODE NotApplicable TCPADDR 31111
  FOR MAESTRO HOST $MASTER ACCESS "unixlocl"
    TYPE X-AGENT
    AUTOLINK OFF
    BEHINDFIREWALL OFF
    FULLSTATUS OFF
END
```

2. Link the FINAL jobs CREATEPOSTREPORTS, MAKEPLAN, STARTAPPSERVER, SWITCHPLAN and UPDATESTATS to the virtual master, as shown in Example 24-4.

Example 24-4 FINAL jobs

```
XA_VMDM#CREATEPOSTREPORTS
 SCRIPTNAME "/opt/IBM/TWA/TWS/CreatePostReports"
 STREAMLOGON twsusr
 TASKTYPE UNIX
 RECOVERY CONTINUE

XA_VMDM#MAKEPLAN
 SCRIPTNAME "/opt/IBM/TWA/TWS/MakePlan"
 STREAMLOGON twsusr
 TASKTYPE UNIX
 RCCONDSUCC "(RC=0) OR (RC=4)"
 RECOVERY STOP

XA_VMDM#STARTAPPSERVER
 SCRIPTNAME "/opt/IBM/TWA/TWS/../wastools/startWas.sh"
 STREAMLOGON twsusr
 TASKTYPE UNIX
 RECOVERY CONTINUE

XA_VMDM#SWITCHPLAN
 SCRIPTNAME "/opt/IBM/TWA/TWS/SwitchPlan"
 STREAMLOGON twsusr
 TASKTYPE UNIX
 RECOVERY STOP

XA_VMDM#UPDATESTATS
 SCRIPTNAME "/opt/IBM/TWA/TWS/UpdateStats"
 STREAMLOGON twsusr
 TASKTYPE UNIX
 RECOVERY CONTINUE
```

Archived

3. Modify the FINAL stream as outlined in Example 24-5.

Example 24-5 FINAL stream

```
SCHEDULE XA_VMDM#FINAL
DESCRIPTION "FINAL stream"
ON RUNCYCLE RULE1 "FREQ=DAILY"
AT XXXX
CARRYFORWARD
:
XA_VMDM#STARTAPPSERVER

XA_VMDM#MAKEPLAN
 FOLLOWS STARTAPPSERVER

XA_VMDM#SWITCHPLAN
 FOLLOWS MAKEPLAN

XA_VMDM#CREATEPOSTREPORTS
 FOLLOWS SWITCHPLAN

XA_VMDM#UPDATESTATS
 FOLLOWS SWITCHPLAN
END
```

Archived

25

Tivoli System Automation for Multiplatforms: Concepts

This chapter introduces the basic concepts of Tivoli System Automation for Multiplatforms. It describes only a subset of the components that are available in Tivoli System Automation for Multiplatforms, and which are required for automating Tivoli Workload Scheduler.

This chapter contains the following topics:

- 25.1, “Overview” on page 746
- 25.2, “Resources” on page 746
- 25.3, “Resource managers” on page 747
- 25.4, “Resource groups” on page 748
- 25.5, “Managed relations” on page 750
- 25.6, “Equivalencies” on page 752
- 25.7, “Quorum” on page 755
- 25.8, “Behavior patterns” on page 757

© Copyright IBM Corp. 2011. All rights reserved.

25.1 Overview

Tivoli System Automation for Multiplatforms delivers a high-availability environment for AIX, Linux, Solaris, and Windows in which systems and applications are continuously monitored. The self-healing features of Tivoli System Automation for Multiplatforms alleviates downtime that is caused by various kinds of problems. High availability is achieved through the use of automation policies in which common relationships between various components are defined. As a result, the time that is required to recover from an outage can improve significantly. The following sections describe the most important cornerstones that can be used in the automation policies for Tivoli Workload Scheduler.

25.2 Resources

A *resource* is the most basic building block in a Tivoli System Automation for Multiplatforms cluster that represents an instance of a resource class. A resource is an abstract view of either a physical or logical component in the system, such as a file system, a network adapter or a piece of software. A resource class defines the set of common characteristics that instances of the class can have. A resource can have *persistent* and *dynamic* attributes.

25.2.1 Persistent resource attributes

Persistent attributes describe enduring properties of a resource, such as the name of a file system or the speed of a network adapter. The most important persistent attributes of a resource in the context of this chapter are *NodeNameList* and *ResourceType*:

- The NodeNameList persistent attribute represents the collection of nodes in the cluster on which the resource is eligible to run.
- The ResourceType persistent attribute specifies whether a resource is fixed or floating:
  - Fixed

    The NodeNameList attribute of the resource contains a single entry. As a result, the resource can only run on the specified node in the cluster.
  - Floating

    The NodeNameList attribute of the resource contains multiple entries. Although multiple nodes are defined, only one instance of the resource may be active at any time.

25.2.2 Dynamic resource attributes

Dynamic attributes represent changing characteristics of a resource. The most important dynamic attribute of a resource in the current context is *OpState*, which specifies the operational state of a resource.

Archive

Table 25-1 lists the possible states.

Table 25-1 Dynamic resource attributes

State	Description
Offline	The resource is not started.
Pending Online	The resource has been started but is not ready for work.
Online	The resource is ready for work.
Pending Offline	The resource is in the process of being stopped.
Failed Offline	The resource is broken and cannot be used.
Stuck Online	The resource cannot be brought offline.
Unknown	Unable to obtain reliable state information of the resource.

25.3 Resource managers

A *resource class* is a collection of resources of the same type which defines a common set of attributes for its instances. Resource classes are managed by their respective resource managers. The two most important resource classes are as follows, both of which are managed by the `IBM.GblResRM` global resource manager:

- IBM.Application

  This resource class is a generic class that can be used to manage any application on the system. The resource class provides the persistent attributes StartCommand, StopCommand, and MonitorCommand, which are used by the global resource manager to control and monitor the application.

- IBM.ServiceIP

  This resource class is used to manage virtual IP addresses that can be started, stopped, and moved between network adapters and nodes within a cluster. Each instance of this class is a floating resource that identifies one virtual IP address. These addresses are typically provided to clients that are connecting to some service offered by the cluster.

The `IBM.GblResRM` global resource manager only provides an interface to control resources. If a problem occurs with a specific resource, the IBM.RecoveryRM recovery resource manager automates the failover of the resource, based on the defined automation policies.

Archive

Figure 25-1 illustrates the most important resource managers.

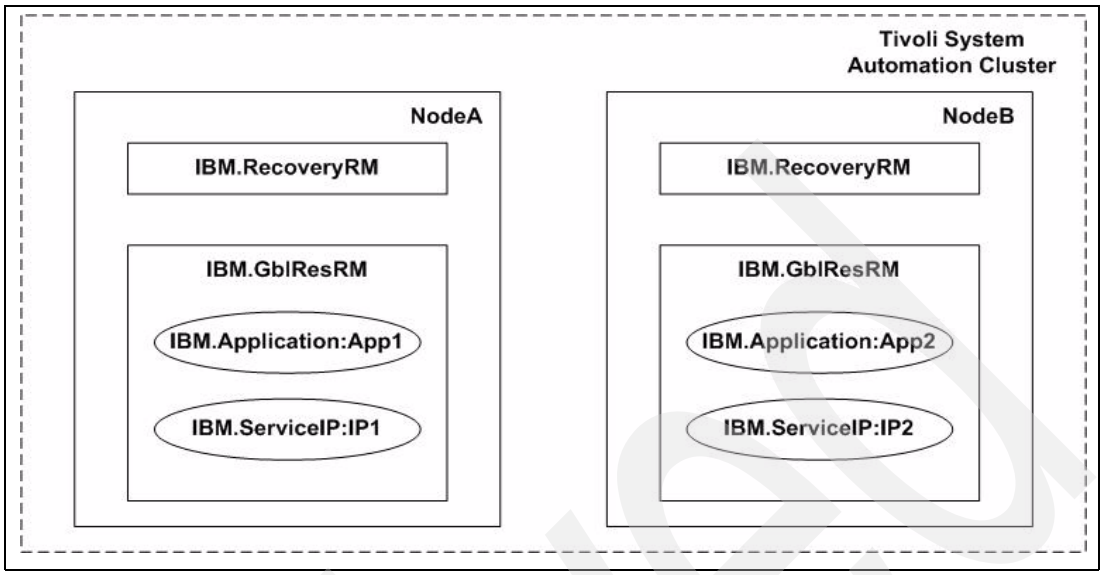


Figure 25-1 Resource managers

Note: The Tivoli System Automation for Multiplatforms commands cannot be used to start or stop resources directly. These actions can only be performed on resource groups.

25.4 Resource groups

Resource groups are logical containers that contain a collection of resources and other nested resource groups. All members of the resource group can be treated as one logical instance. As a result, resource groups can be used to control all members collectively. A resource group is an instance of the IBM.ResourceGroup class and has persistent and dynamic attributes similar to a resource.

25.4.1 Persistent resource group attributes

A resource group has the following important persistent attributes:

- MemberLocation

  This persistent attribute indicates whether all resource group members have to be collocated on the same node in the cluster.

Important: By default, all members of a resource group are collocated on the same node in the cluster.

- NominalState

  This persistent attribute defines the state of the resource group. Tivoli System Automation for Multiplatforms will try to start and keep the resource group in this state.

25.4.2 Dynamic resource group attributes

Similar to a resource, a resource group also has an OpState dynamic attribute. Tivoli System Automation for Multiplatforms uses this attribute to indicate the aggregate operational state of the managed resources that are member of the resource group.

The possible values of the OpState dynamic attribute are listed in Table 25-2.

Table 25-2 OpState dynamic resource group attribute

State	Description
Offline	Indicates that all member resources are Offline, (their OpState is set to Offline).
Pending Online	Indicates that the resource group's NominalState has just been set to Online. All resources in the resource group are started.
Online	Specifies that all *Mandatory* resources are Online. See 25.5, "Managed relations" on page 750 for more details.
Pending Offline	Indicates that resource group will be brought Offline.
Failed Offline	Specifies that one or more member resources are in the Failed Offline state.
Stuck Online	Indicates that one or more member resources are in the Stuck Online state.
Unknown	Specifies that one or more member resources are in the Unknown state.

25.4.3 Managed resources

A resource becomes a *managed resource* as soon as the resource has been inserted into a resource group or an equivalency, resulting in an instance of the IBM.ManagedResource class. The Mandatory persistent attribute of a managed resource specifies whether the resource is mandatory for the resource group. When a resource group is started, all managed resources within the group that are mandatory must be started. If a mandatory resource fails to start, the entire resource group is stopped and started on another node in the cluster. Non-mandatory resources may be sacrificed to activate the resource group.

Figure 25-2 on page 750 shows all components.

Important: The global resource manager IBM.GblResRM starts monitoring resources as soon as they become managed resources.

Archived

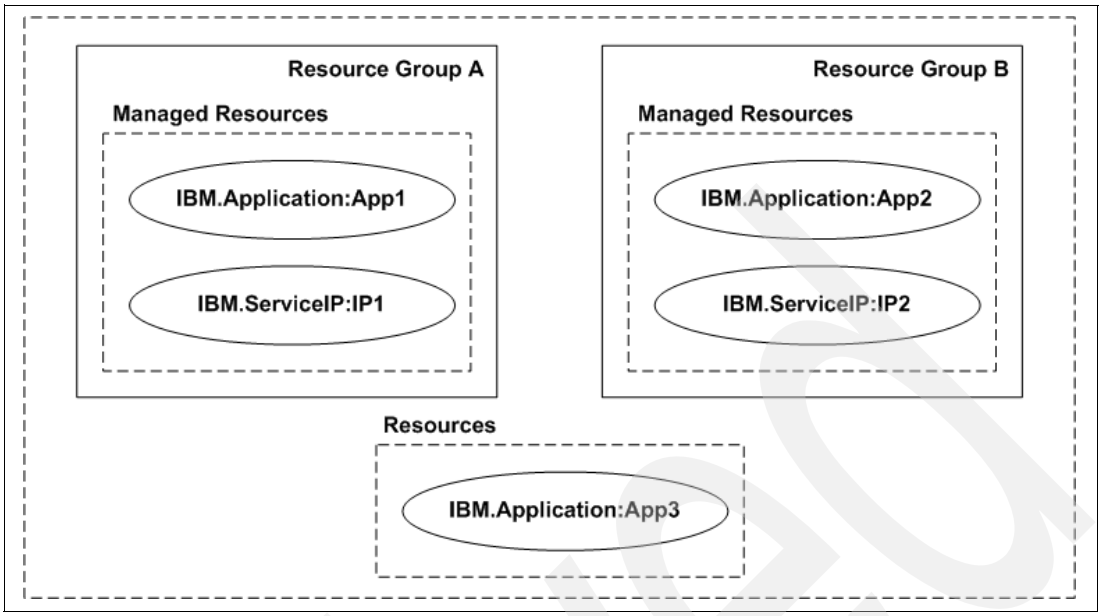


Figure 25-2 Managed Resources

25.5 Managed relations

A managed relationship can exist between a source resource or resource group and one or more target resources or resource groups. This section describes the most interesting relationships (because they are the most common) for the automation of Tivoli Workload Scheduler.

25.5.1 Start and stop dependencies

Tivoli System Automation for Multiplatforms provides relationships that can be used to regulate the starting and stopping of resources. The source of a start or stop relationship can be one of the following sources:

- Member of a resource group (managed resource)
- Resource group

The target of a start or stop relationship can be one of the following targets:

- Member of a resource group (managed resource)
- Resource group
- Equivalency (See “Equivalencies” on page 752 for an in-depth discussion)

The source and target can belong to separate resource groups.

StartAfter relationship

The *StartAfter* relationship ensures that the source resource is only started when the target resource is available, as shown in Figure 25-3 on page 751.

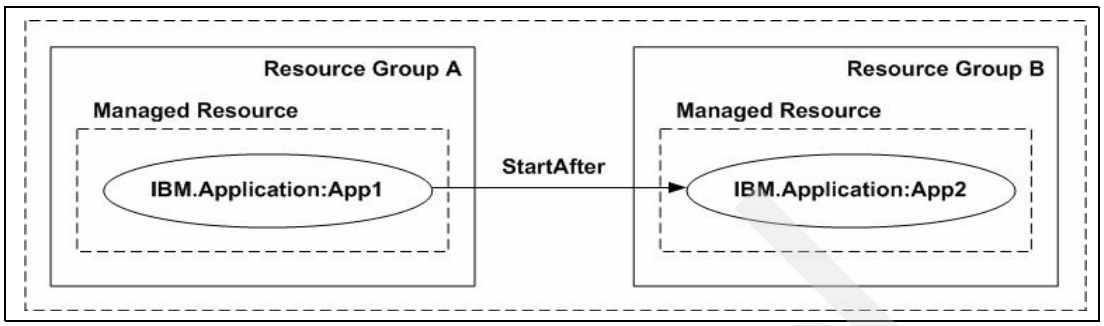


Figure 25-3 StartAfter relationship

When the App1 source resource has to be started, the App2 target resource is started first. As soon as App2 target resource reaches the Online operational state, App1 source resource is started. The start order only acts in the forward direction of the relationship. Setting the NominalState persistent attribute of resource group B to Online does not cause any action on resource App1 because resource App2 has no forward relationship with resource App1.

The StartAfter relationship will be used to automate the startup of the embedded WebSphere Application Server and the database.

DependsOn relationship

Similar to the StartAfter relationship, the *DependsOn* relationship is used to ensure that a source resource can only be started when the target resource is online. However, several important differences exist:

- A DependsOn relationship implies an implicit collocation between the source and the target resources. See 25.5.2, “Location dependencies” on page 751 for more details.
- If the target resource fails (OpState = Failed Offline) or is stopped gracefully (OpState = Offline), the source resource is also stopped.

The DependsOn relationship is used to automate the Event Processor and the Dynamic Workload Broker Component, which depend on the embedded WebSphere Application Server that is hosting them.

Important: Unlike the DependsOn relationship, the StartAfter relationship does not bring down the source resource if the target resource becomes unavailable.

25.5.2 Location dependencies

Tivoli System Automation for Multiplatforms also provides relationships that are used to define location constraints between *floating* resources. For automating Tivoli Workload Scheduler, the two relationships that are used are *Collocated* and *Affinity*:

- Collocated relationship

  The *Collocated* relationship is employed to ensure that the source resource and the target resource are located on the same node in the cluster. A DependsOn relationship between the source resource and the target resource implies this relationship automatically.

- Affinity relationship

  The *Affinity* relationship is used to indicate that, if possible, a source resource should run on the same node where the target resource is running. If another location relationship is inhibiting this, the source resource can also run on another node in the cluster.

Therefore, the Affinity relationship defines a soft-location relationship; the Collocated relationship is a hard-location relationship, as illustrated in Figure 25-4. The Affinity relationship between App4 and App1 is sacrificed because App4 has a Collocated relationship with App5, which is already bound to a separate node in the cluster.

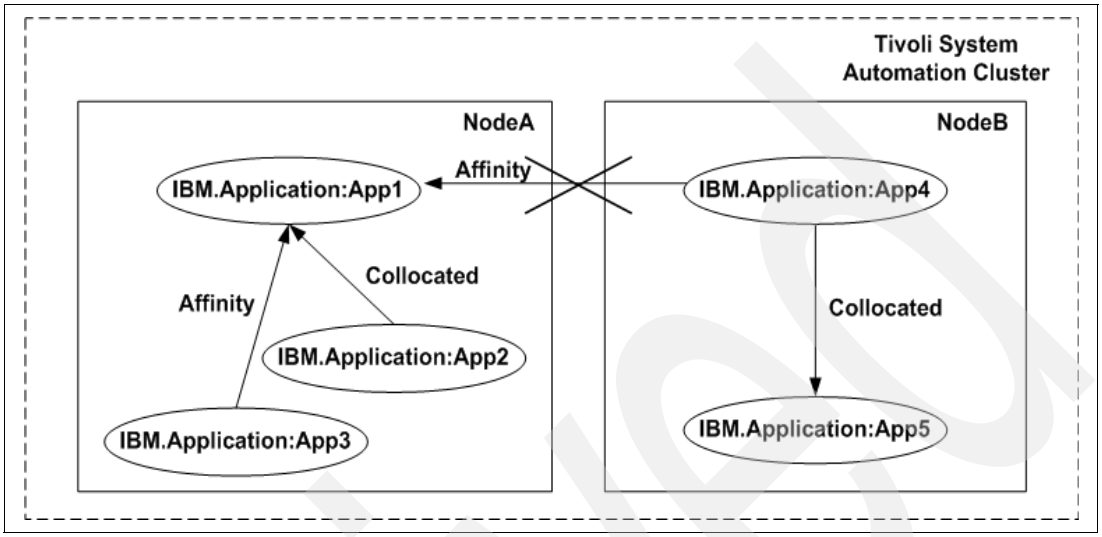


Figure 25-4 Collocated / Affinity relationship

25.6 Equivalencies

Resources that provide similar functionality can be grouped into an equivalency, which can be the target of a managed relationship. The collection of resources that embrace an equivalency must be *fixed* resources of the same resource class.

25.6.1 Example

Assume three applications, App1, App2 and App3, that have the following characteristics:

- App1 is an embedded application that can run in either App2 or App3. However, App1 cannot run in both App2 and App3 at the same time. As a result, App1 is a floating resource, and App2 and App3 are fixed resources.
- Each application can be started and stopped independently.
- If App1 is running in App2, and App2 suddenly fails, App1 must be restarted on App3.
- If App1 is running in App2, and App3 suddenly fails, Tivoli System Automation for Multiplatforms attempts to restart only App3.

At first appearance, this automation problem can be solved easily by creating a resource group for each application and using a DependsOn relationships between the resource groups, as illustrated in Figure 25-5 on page 753.

Archive

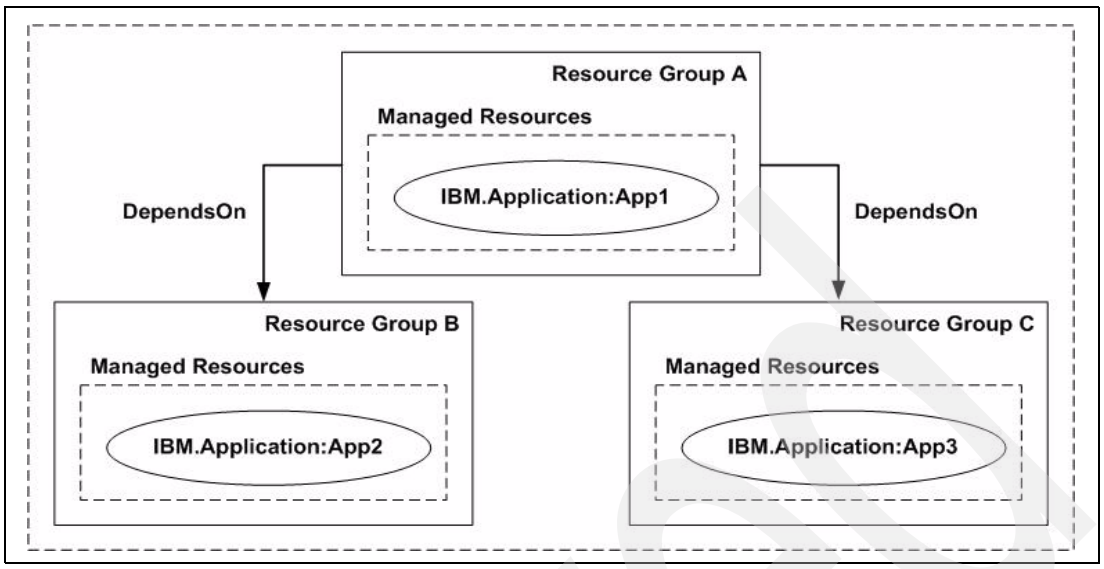


Figure 25-5 Automation problem: incorrect solution

Using this setup, Tivoli System Automation for Multiplatforms introduces the following unexpected automation behavior:

- App1 is started only if both App2 and App3 are available. This requirement is too strict.
- If App1 is running in App2, and App3 suddenly fails, App1 is stopped also. The cause is because App1 depends on both App2 and App3, but it should depend only on either App2 or App3, depending on the application that is currently hosting App1.

To achieve the automation behavior, the use of shadow resources and shadow equivalencies is required.

25.6.2 Shadow resources and shadow equivalencies

A *shadow resource* is an instance of the IBM.Application class that has the following characteristics:

- It is a fixed resource that monitors the OpState of another fixed resource.
- It is a member of a shadow equivalency that instructs Tivoli System Automation for Multiplatforms to evaluate only the OpState of the member resources. The implication is that the shadow resources will not be started or stopped.

Using shadow resources and shadow equivalencies, the automation problem described in section 25.6, “Equivalencies” on page 752 can now be solved as follows:

1. Create a resource group for each application.
2. Define shadow resources of App2 and App3. These shadow resources monitor the OpState of App2 and App3, respectively.
3. Create a shadow equivalency, which contains the shadow resources.
4. Define a DependsOn relationship between the resource group of App1 and the shadow equivalency.

Archive

Figure 25-6 illustrates the correct solution.

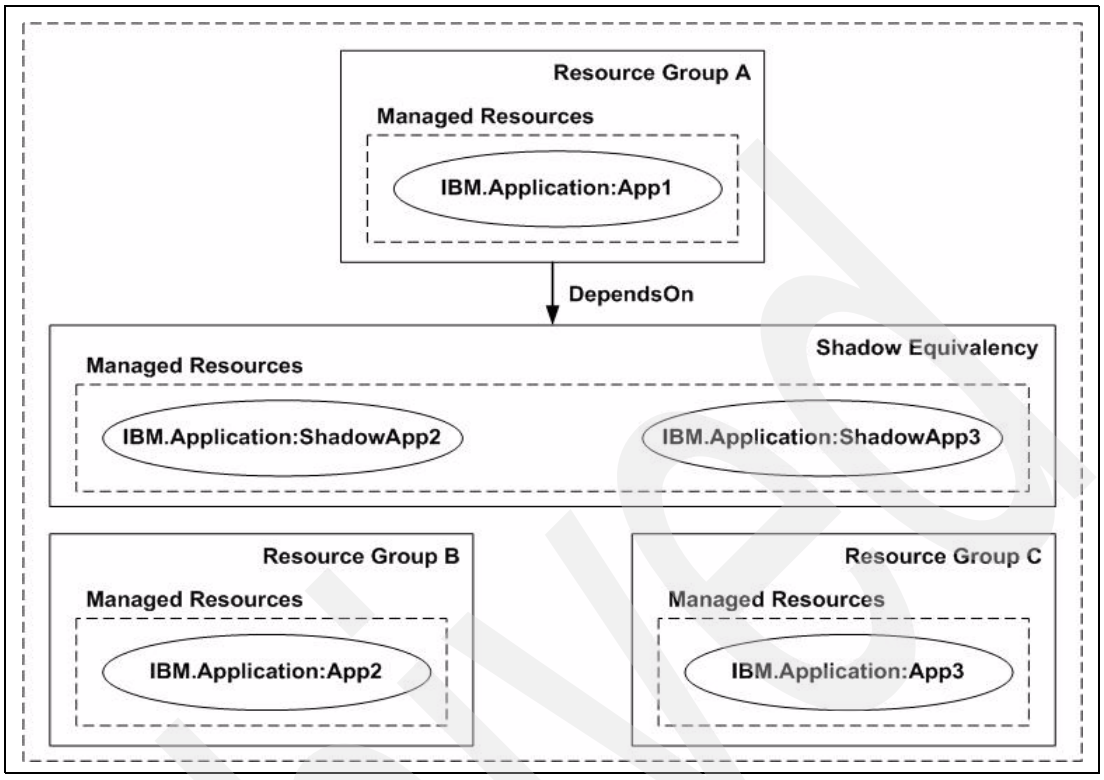


Figure 25-6 Automation problem: correct solution

The solution bypasses the undesired automation behavior as follows:

- App1 is started if either App2 or App3 is available.
- If App1 is running in App2 and App3 suddenly fails, Tivoli System Automation for Multiplatforms attempts to restart App3. No action is performed on App1.
- If App1 is running in App2 and App2 suddenly fails, App1 is restarted on App3.

Archive

25.7 Quorum

A cluster may split into two or more subclusters if no more communication is possible between the nodes in the cluster. If subclusters are unaware of each other's existence, Tivoli System Automation for Multiplatforms might start a new instance of an application that is already running in one of the other subclusters. If that application is a critical resource that is not allowed to run on multiple nodes in the cluster simultaneously, data integrity of the cluster might be endangered.

If the cluster splits into two or more subclusters, Tivoli System Automation for Multiplatforms determines which of the subclusters has the majority of member nodes. The subcluster with the majority of nodes (more than 50%) has the operational quorum and becomes the active cluster. Other subclusters are dissolved, as shown in Figure 25-7.

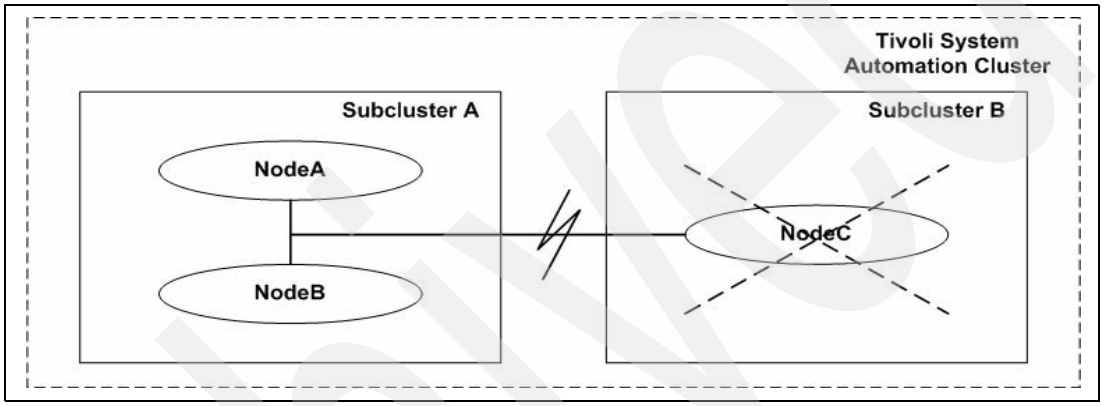


Figure 25-7 Quorum

The dynamic attribute OpQuorumState of the IBM.PeerDomain class identifies the operational quorum of the cluster. The attribute has the following possible values:

- OpQuorumState = 0 indicates that quorum is achieved. As a result, resources may be started.
- OpQuorumState = 1 indicates that a tie situation occurs and is not yet resolved. See 25.7.1, "Tie breaker" on page 755 for more details.
- OpQuorumState = 2 indicates that quorum is not achieved. As a result, Tivoli System Automation for Multiplatforms is not allowed to start resources.

Alternatively, the IBM.RecoveryRM can also be queried to obtain the Quorum State, as shown in Example 25-1.

Example 25-1 Quorum State

```
# lssrc -ls IBM.RecoveryRM | grep 'Operational Quorum State'
   Operational Quorum State: HAS_QUORUM
```

25.7.1 Tie breaker

If a tie occurs in which the cluster is partitioned into subclusters with an equal number of nodes, Tivoli System Automation for Multiplatforms employs a tie breaker to determine which subcluster has operational quorum. Operational quorum becomes effective as soon as the subcluster reserves the tie breaker. Several types of tie breakers exist, of which the network tie breaker is the most common type in the context of automating Tivoli Workload Scheduler.

To resolve a tie situation, the *network tie breaker* uses an external IP address which has to be reachable from all nodes in the cluster, as illustrated in Figure 25-8. This external IP address must be able to reply to Internet Control Message Protocol (ICMP) echo requests (**ping** command). If a firewall rule is installed that blocks ICMP traffic between the cluster nodes and the external IP address, the network tie breaker does not function properly. Using the network tie breaker has the following advantages over other tie breakers:

- No additional hardware is required.
- Unlike other tie breakers (for example, disk based), the network tie breaker evaluates the availability of TCP/IP communication between the cluster nodes.

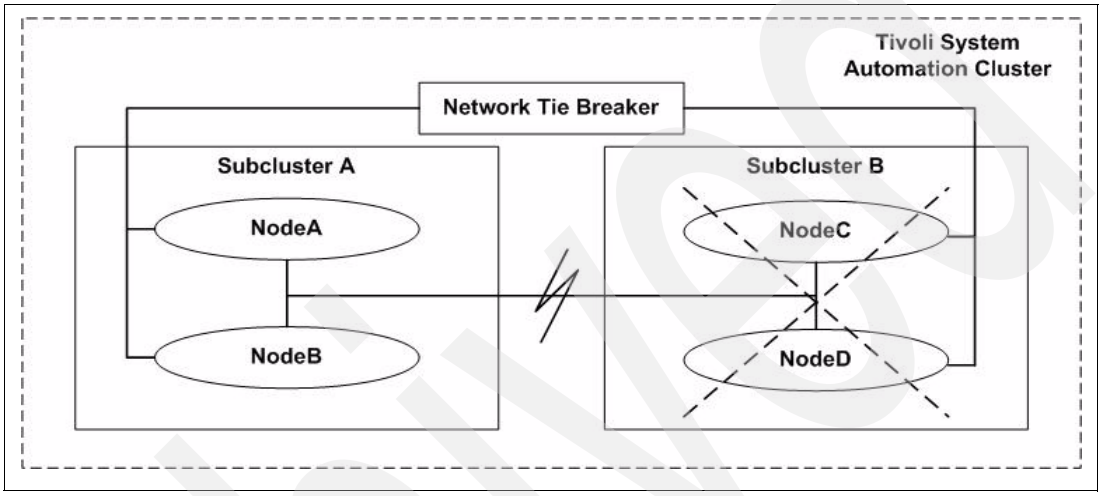


Figure 25-8 Network tie breaker

Important: The network tie breaker must be used only when all cluster nodes are part of the same subnet. By having the nodes in separate subnets, the nodes are more likely to **ping** the network tie breaker, although they cannot communicate to each other. As a result, a cluster split goes undetected.

25.7.2 Critical resource protection

The *ProtectionMode* persistent attribute of a resource specifies whether the resource is a critical resource. If critical resources are active in the dissolved subcluster, Tivoli System Automation for Multiplatforms uses the *CritRsrcProtMethod* attribute of each node in the dissolved subcluster to determine in which way the system should be terminated. The default (`CritRsrcProtMethod = 1`) is to simulate a kernel panic, as shown in Example 25-2.

Example 25-2 Critical resource protection

```
# lsrsrc -c IBM.PeerNode | grep CritRsrcProtMethod
        CritRsrcProtMethod = 1
```

Note: By default, instances of the IBM.Application resource class are non-critical; instances of the IBM.ServiceIP resource class are critical.

25.8 Behavior patterns

This section demonstrates the most common behavior patterns of Tivoli System Automation for Multiplatforms, which are based on a change of the operational state of one or more managed resources. Figure 25-9 describes the behavior and consists of two floating resources in a single resource group, which must be collocated on the same node in the cluster. The cluster in the example consists of two nodes, NodeA and NodeB.

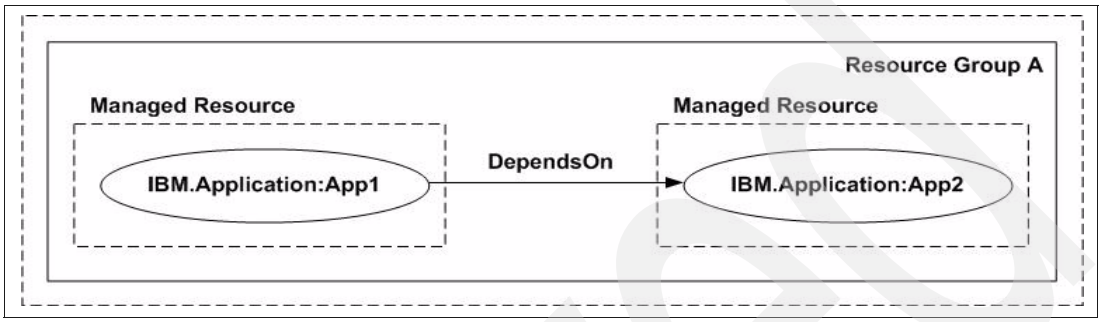


Figure 25-9 Behavior patterns

25.8.1 MonitorCommand

The OpState of an instance of the IBM.Application class is determined by the return code of the **`MonitorCommand`** command, which is run every *MonitorCommandPeriod* seconds. As described in 25.4, “Resource groups” on page 748, the operational state of a resource group is an aggregation of the OpState attributes of all managed resources that it comprises. If the NominalState of the resource group has been set to Online, the OpState of the resource group is set to Pending Online until all the managed resources in the resource group are Online. Finally, if all resources of the resource group have reached the NominalState of the resource group, the OpState of the resource group changes to Online. The next sections describe how Tivoli System Automation for Multiplatforms reacts to OpState changes of the managed resources.

OpState change App1

Assuming that the current operational status of both App1 and App2 is Online on NodeA and that the NominalState and OpState of resource group A is set to Online, two situations might occur: Offline and Failed Offline.

Case 1: OpState = Offline

If the operational state of App1 changes to Offline, Tivoli System Automation for Multiplatforms attempts to restart it on NodeA. If restarting App1 completes successfully, the operational state of resource group A changes from Pending Online to Online.

Archive

Table 25-3 shows the sequence that Tivoli System Automation for Multiplatforms follows.

Table 25-3 OpState change App1: Offline

NodeA		NodeB		GroupA
MonitorCommand (OpState App1)	**MonitorCommand (OpState App2)**	**MonitorCommand (OpState App1)**	**MonitorCommand (OpState App2)**	**OpState Resource Group A**
RC=1 (Online)	RC=1 (Online)	RC=2 (Offline)	RC=2 (Offline)	Online
RC=2 (Offline)	RC=1 (Online)	RC=2 (Offline)	RC=2 (Offline)	Pending Online
RC=1 (Online)	RC=1 (Online)	RC=2 (Offline)	RC=2 (Offline)	Online

Case 2: OpState = Failed Offline

An operational state change to Failed Offline of App1 indicates that the resource is broken on NodeA and cannot be restarted there. As a result, App1 must be restarted on NodeB. However, because App1 depends on App2 and both resources are part of the same resource group, first App2 must be stopped on NodeA and restarted on NodeB. Assuming that the restart is successful for both resources on NodeB, App1 is next started after App2. Finally, the OpState of resource group changes back to Online. Table 25-4 outlines the sequence in detail.

Table 25-4 OpState change App1: Failed Offline

NodeA		NodeB		Group A
MonitorCommand (OpState App1)	**MonitorCommand (OpState App2)**	**MonitorCommand (OpState App1)**	**MonitorCommand (OpState App2)**	**OpState Resource Group A**
RC=1 (Online)	RC=1 (Online)	RC=2 (Offline)	RC=2 (Offline)	Online
RC=3 (Failed Offline)	RC=1 (Online)	RC=2 (Offline)	RC=2 (Offline)	Pending Offline
RC=3 (Failed Offline)	RC=2 (Offline)	RC=2 (Offline)	RC=2 (Offline)	Offline
RC=3 (Failed Offline)	RC=2 (Offline)	RC=2 (Offline)	RC=1 (Online)	Pending Online
RC=3 (Failed Offline)	RC=2 (Offline)	RC=1 (Online)	RC=1 (Online)	Online

Note: The OpState of App1 on NodeA remains in the state Failed Offline. To change the state to Offline, resource App1 on NodeA must be reset with the `resetrsrc` command.

OpState change App2

Assuming that the current operational status of both App1 and App2 is Online on NodeA and that the NominalState and OpState of resource group A is set to Online, two events might occur: Offline and Failed Offline.

Case 1: OpState = Offline

If the OpState of resource App2 changes from Online to Offline, Tivoli System Automation for Multiplatforms stops App1 because of the DependsOn relationship. Subsequently, App2 is restarted on the same node in the cluster. Finally, App1 is restarted on NodeA also. Table 25-5 summarizes the events.

Table 25-5 OpState change App2: Offline

NodeA		**NodeB**		**Group A**
MonitorCommand (OpState App1)	**MonitorCommand (OpState App2)**	**MonitorCommand (OpState App1)**	**MonitorCommand (OpState App2)**	**OpState Resource Group A**
RC=1 (Online)	RC=1 (Online)	RC=2 (Offline)	RC=2 (Offline)	Online
RC=1 (Online)	RC=2 (Offline)	RC=2 (Offline)	RC=2 (Offline)	Pending Offline
RC=2 (Offline)	RC=2 (Offline)	RC=2 (Offline)	RC=2 (Offline)	Offline
RC=2 (Offline)	RC=1 (Online)	RC=2 (Offline)	RC=2 (Offline)	Pending Online
RC=1 (Online)	RC=1 (Online)	RC=2 (Offline)	RC=2 (Offline)	Online

Case 2: OpState = Failed Offline

If the operational state of resource App2 changes to Failed Offline while App1 is Online, Tivoli System Automation for Multiplatforms stops App1 on NodeA first. Next, App2 is started on NodeB. To honor the dependency, App1 is launched on NodeB after, as shown in Table 25-6.

Table 25-6 OpState change App2: Failed Offline

NodeA		**NodeB**		**Group A**
MonitorCommand (OpState App1)	**MonitorCommand (OpState App2)**	**MonitorCommand (OpState App1)**	**MonitorCommand (OpState App2)**	**OpState Resource Group A**
RC=1 (Online)	RC=1 (Online)	RC=2 (Offline)	RC=2 (Offline)	Online
RC=1 (Online)	RC=3 (Failed Offline)	RC=2 (Offline)	RC=2 (Offline)	Pending Offline
RC=2 (Offline)	RC=3 (Failed Offline)	RC=2 (Offline)	RC=2 (Offline)	Offline
RC=2 (Offline)	RC=3 (Failed Offline)	RC=2 (Offline)	RC=1 (Online)	Pending Online
RC=2 (Offline)	RC=3 (Failed Offline)	RC=1 (Online)	RC=1 (Online)	Online

Preview

25.8.2 StartCommand

Another important trigger for automation actions, is the result of the **StartCommand** command. This section gives more details about how this command affects the behavior of Tivoli System Automation for Multiplatforms.

StartCommand completed successfully

If the **StartCommand** command completes successfully (RC=0), Tivoli System Automation for Multiplatforms acts on the result of the subsequent **MonitorCommand** command.

Return code of MonitorCommand = 1 (Online)

The automation goal is reached. The OpState of the resource is set to Online.

Return code of MonitorCommand = 2 (Offline)

Tivoli System Automation for Multiplatforms sets the resource in the Pending Online state and waits for the outcome of the subsequent **MonitorCommand** command executions. If the resource is still Offline after *StartCommandTimeout* seconds, the resource is reset. If the reset is successful, an attempt to restart the resource is made and a counter is incremented. If the counter reaches a configurable threshold, the OpState of the resource is set to Failed Offline and will therefore trigger a failover.

The upper threshold is regulated by the RetryCount tunable, which can be set by the **samctrl** command. The default value is set to 3 as shown in Example 25-3.

Example 25-3 RetryCount tunable

```
# lssamctrl | grep RetryCount
        RetryCount = 3
```

StartCommand completed with an error

If the **StartCommand** command completes unsuccessfully (RC=1), Tivoli System Automation for Multiplatforms acts on the result of the previous **MonitorCommand** command.

Figure 25-10 on page 761 shows a state diagram of the **StartCommand** command.

Return code of MonitorCommand = 1 (Online)

In this case, the return code of the **StartCommand** command is ignored. The OpState of the resource is set to Online, which implies that the automation goal is reached.

Return code of MonitorCommand = 2 (Offline)

If the **MonitorCommand** command indicates that the resource is Offline, the OpState of the resource is put in a Failed Offline state and therefore triggers a failover.

Archive

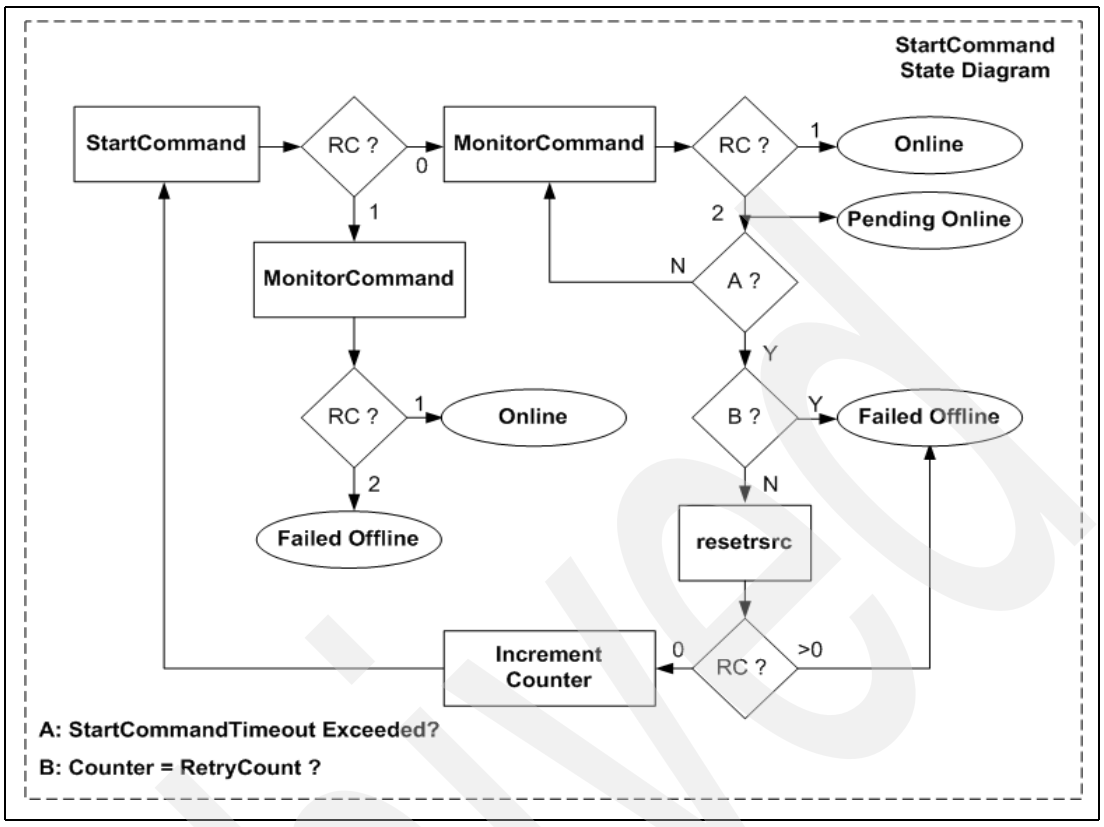


Figure 25-10 StartCommand state diagram

25.8.3 StopCommand

The logic behind the **StopCommand** command is similar to the behavior of the **StartCommand** command. If a stop or reset request is issued to an instance of the IBM.Application class, the **StopCommand** command is executed. This section describes the logic more in detail.

StopCommand completed successfully

If the **StopCommand** command completes successfully (RC=0), Tivoli System Automation for Multiplatforms acts on the outcome of the subsequent **MonitorCommand** command.

Return code of MonitorCommand = 2 (Offline)

In this case, the automation goal is reached. The OpState of the resource is set to Offline.

Return code of MonitorCommand = 1 (Online)

Tivoli System Automation for Multiplatforms sets the resource in the Pending Offline state and acts on the result of the subsequent **MonitorCommand** commands. If the resource is still Online after *StopCommandTimeout* seconds, the resource is reset. If after the reset timeout (equal to StopCommandTimeout seconds) the resource is still in the Online state, Tivoli System Automation for Multiplatforms will put the resource in the Stuck Online state. Operator intervention will be required to unblock the situation.

StopCommand completed with an error

If the **StopCommand** command completes unsuccessfully (RC=1), Tivoli System Automation for Multiplatforms will act as follows.

Return code of MonitorCommand = 2 (Offline)

The return code of the **`StopCommand`** command is ignored. The OpState of the resource is set to Offline. The automation goal is reached.

Return code of MonitorCommand = 1 (Online)

In this case, Tivoli System Automation for Multiplatforms sets the resource in the Pending Offline state. After, a reset on the resource is performed. If the resource is still Online after the reset timeout (equal to StopCommandTimeout seconds), the resource is put in the Stuck Online state. Figure 25-11 illustrates the states.

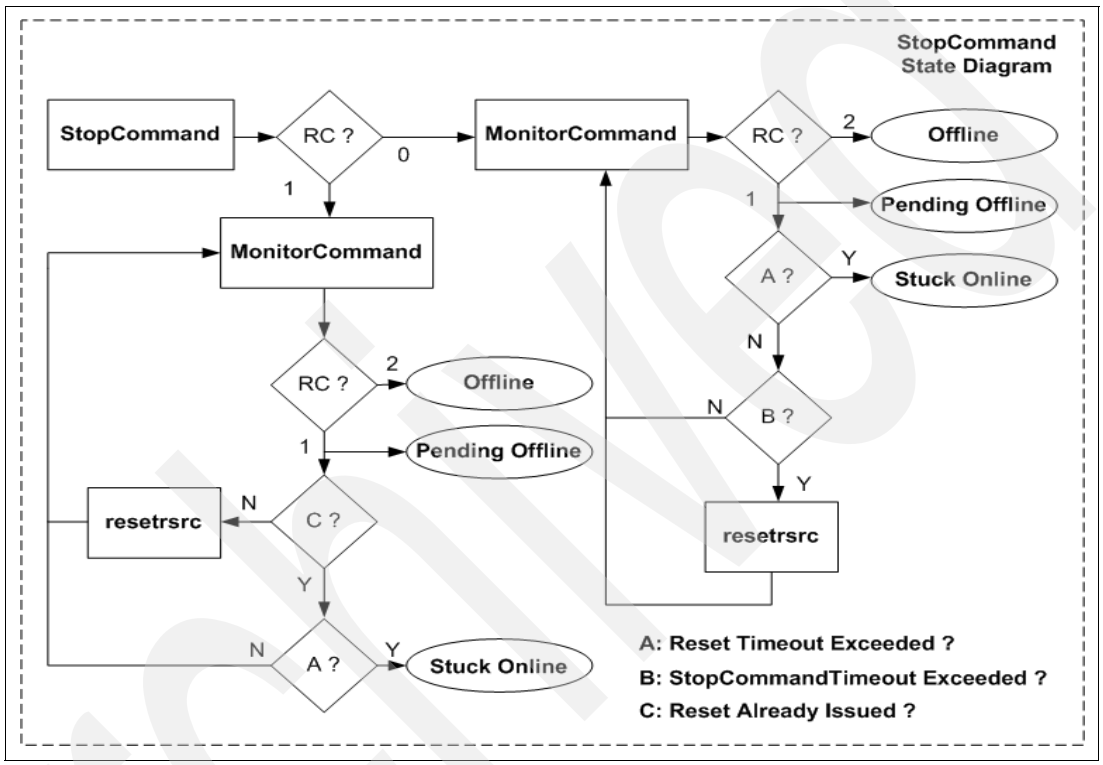


Figure 25-11 StopCommand state diagram

Archive

26

Implementing high availability for the Tivoli Workload Scheduler master domain manager

This chapter describes implementing high availability for the Tivoli Workload Scheduler master domain manager by using Tivoli System Automation for Multiplatforms. The design of the automation policies is based on the concepts described in Chapter 24, “Tivoli Workload Scheduler: Architecture and concepts” on page 719 and Chapter 25, “Tivoli System Automation for Multiplatforms: Concepts” on page 745.

The policies used in this chapter can be downloaded from the ITSO FTP site. For download instructions, see Appendix A, “Additional material” on page 803.

This chapter contains the following topics:

- 26.1, “Overview of the setup” on page 764
- 26.2, “Benefits” on page 764
- 26.3, “Installing DB2” on page 764
- 26.4, “Installing Tivoli Workload Scheduler” on page 767
- 26.5, “Installing and configuring Tivoli System Automation for Multiplatforms” on page 774
- 26.6, “Integrating DB2 HADR and Tivoli System Automation for Multiplatforms” on page 775
- 26.7, “Integrating Tivoli Workload Scheduler and Tivoli System Automation for Multiplatforms for high availability” on page 778
- 26.8, “Testing the policies” on page 790

© Copyright IBM Corp. 2011. All rights reserved.

26.1 Overview of the setup

The setup used in this chapter consists of two AIX 6.1 nodes: twspoc1 and twspoc2. Each is located in a separate computer room. The nodes have two logical volume manager (LVM) mirrored-volume groups: rootvg and twsvg.

The following software is used:

- AIX 6.1 TL 4 SP 4
- Tivoli System Automation 3.2 GA
- Tivoli Workload Scheduler 8.5.1 FP 1
- Tivoli Dynamic Workload Console 8.5.1 FP 1
- DB2 9.5 FP 4

26.2 Benefits

Using Tivoli System Automation for Multiplatforms, for automating the failover of the master domain manager, has the following advantages:

- By employing granular automation policies, the dependencies between all components are honored.
- The use of an external tool ensures that all components are monitored independently.
- The intelligence of the policies ensures that the required corrective actions are made.
- Compared to a manual failover of all impacted components, the amount of time required to perform a failover to the backup master domain manager are reduced significantly.

26.3 Installing DB2

This section describes the steps to install the DB2 instances on the master domain manager and the backup master domain manager. These instances host the Tivoli Workload Scheduler database.

26.3.1 Creating DB2 instance owner and file systems

Before installing DB2, the DB2 instance owner and file systems must be created on the twspoc1 and twspoc2 nodes. For demonstration purposes, the DB2 instance owner *db2tws* will be created. Figure 26-1 on page 765 shows the file system structure that is created in the twsvg volume group.

Archived

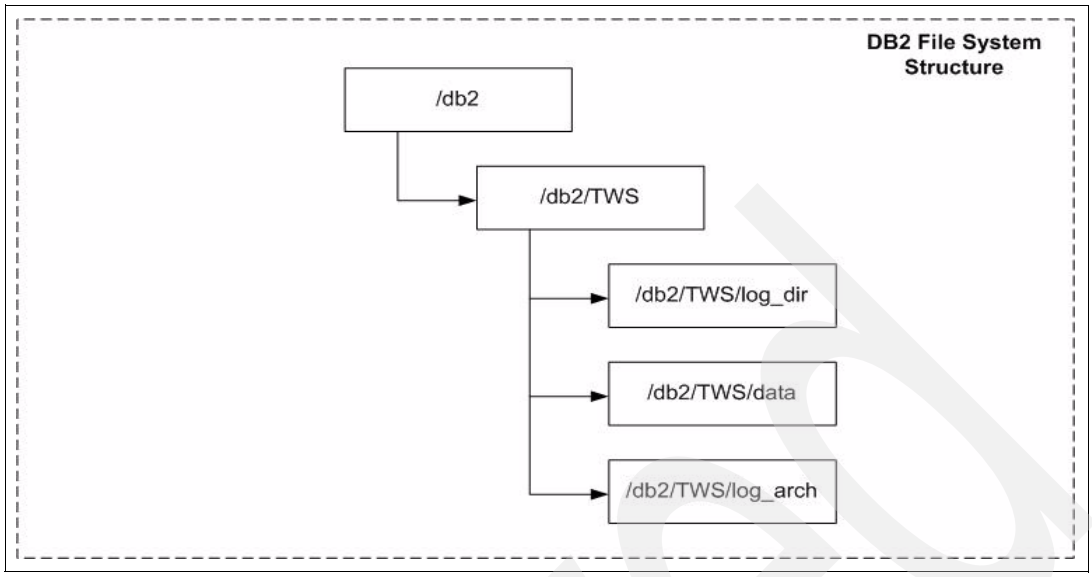


Figure 26-1 DB2 file system structure

The /db2 file system is contained in the DB2 9.5 binaries; the DB2 instance is created in the /db2/TWS file system. The table spaces are created in the /db2/TWS/data file system. Finally, the /db2/TWS/log_dir file system is used for the log files; the /db2/TWS/log_arch will contain the archived log files. The tasks must be performed as root on twspoc1 and twspoc2, as shown in Example 26-1.

Example 26-1 DB2 instance owner and file system creation

```
# crfs -v jfs2 -g twsvg -m /db2 -A yes -a size=2G
# crfs -v jfs2 -g twsvg -m /db2/TWS -A yes -a size=1G
# crfs -v jfs2 -g twsvg -m /db2/TWS/log_dir -A yes -a size=512M
# crfs -v jfs2 -g twsvg -m /db2/TWS/data -A yes -a size=2G
# crfs -v jfs2 -g twsvg -m /db2/TWS/log_arch -A yes -a size=1G

# mount /db2
# mkdir /db2/TWS
# mount /db2/TWS
# mkdir /db2/TWS/log_dir /db2/TWS/data /db2/TWS/log_arch
# mount /db2/TWS/log_dir
# mount /db2/TWS/data
# mkdir -p /db2/TWS/data/db2tws/NODE0000/SQL00001
# mount /db2/TWS/log_arch

# mkgroup -R compat id=500 twsgrp
# mkuser -R compat id=501 pgrp=twsgrp home=/db2/TWS rlogin=false cpu=-1 data=-1
stack=-1 cpu_hard=-1 data_hard=-1 stack_hard=-1 db2tws
# passwd db2tws

# chown -R db2tws.twsgrp /db2/TWS
```

Archive

26.3.2 Installing and upgrading DB2

Next, DB2 9.5 is installed in the /db2 file system on twspoc1 and twspoc2. After the installation of the GA release, DB2 9.5 is upgraded to FP 4 and the DB2 ESE restricted license is installed. The tasks must be performed as root on twspoc1 and twspoc2, as illustrated in Example 26-2.

Example 26-2 Instance owner and file system creation

```
# ./db2_install -b /db2/V9.5 -f NOTSAMP -p ESE -n
The execution completed successfully.

# ./installFixPack -b /db2/V9.5
DBI1017I  installFixPack is updating the DB2 product(s) installed in
      location /db2/V9.5.
...
The execution completed successfully.

# /db2/V9.5/adm/db2licm -a ./db2ese_o.lic
LIC1402I  License added successfully.
```

Note: The version of Tivoli System Automation for Multiplatforms that is included with DB2 9.5 is not be used. Tivoli System Automation for Multiplatforms 3.2 is installed separately.

26.3.3 Creating DB2 instances

Following the DB2 installation, the DB2 instance db2tws is created on both twspoc1 and twspoc2. This task must be performed as root on both twspoc1 and twspoc2. See Example 26-3.

Example 26-3 DB2 instance creation

```
# /db2/V9.5/instance/db2icrt -u db2tws db2tws
DBI1070I  Program db2icrt completed successfully.
```

Verify the DB2 level of the instance as db2tws. See Example 26-4.

Example 26-4 DB2 instance level verification

```
$ db2level
DB21085I  Instance "db2tws" uses "64" bits and DB2 code release "SQL09054" with
level identifier "06050107".
Informational tokens are "DB2 v9.5.0.4", "s090429", "U825478", and Fix Pack "4".
Product is installed at "/db2/V9.5".
```

26.3.4 Enabling TCP/IP communication

To communicate with the Tivoli Workload Scheduler database from the embedded WebSphere Application Server, TCP/IP communication must be enabled on the database manager. TCP port 50000 is used in this setup. Finally, the database manager is started.

Archived

Example 26-5 shows the commands to run as db2tws on twspoc1 and twspoc2.

Example 26-5 Enabling TCP/IP communication

```
$ db2 update dbm cfg using SVCENAME 50000
DB20000I  The UPDATE DATABASE MANAGER CONFIGURATION command completed
successfully.
$ db2set DB2COMM=tcpip
$ db2start
05/18/2010 11:50:48     0   0   SQL1063N  DB2START processing was successful.
SQL1063N  DB2START processing was successful.
```

Next, verify that the database manager is listening on TCP port 50000 by using the **netstat** or **lsof** command, as illustrated in Example 26-6.

Example 26-6 Verifying TCP/IP communication

```
# netstat -anfinet | grep 50000
tcp4   0   0   *.50000   *.*   LISTEN
# lsof –i 2>/dev/null | grep 50000
db2sysc   622648 db2tws    6u  IPv4 0xf10002000283abb0      0t0
TCP *:50000 (LISTEN)
```

26.4 Installing Tivoli Workload Scheduler

This section describes the installation procedure for Tivoli Workload Scheduler. The master domain manager is installed on twspoc1; the backup master domain manager is installed on twspoc2.

26.4.1 Creating Tivoli Workload Scheduler user and file systems

Prior to installing the master domain manager and backup master domain manager, the Tivoli Workload Scheduler user and file systems must be created. For demonstration purposes, the Tivoli Workload scheduler user *twsusr* is used. Figure 26-2 illustrates the file system structure that is created in the twsvg volume group on both nodes.

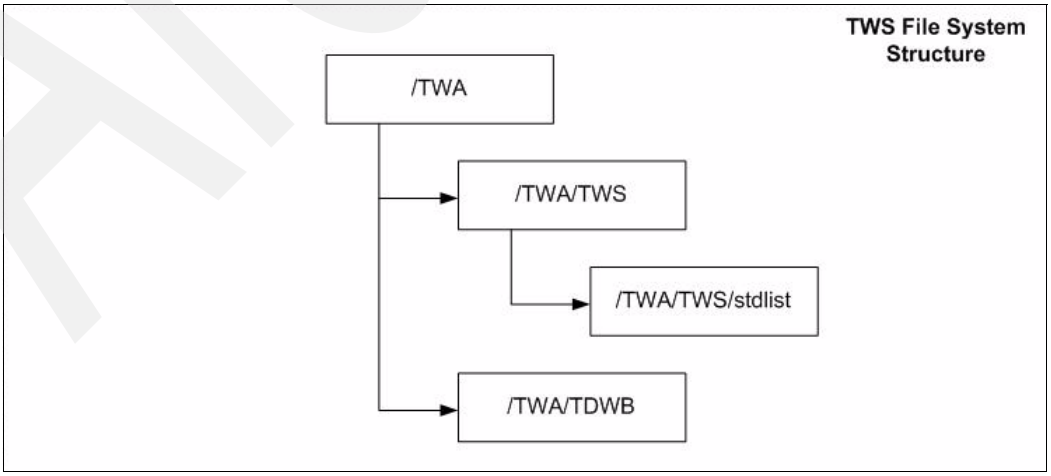


Figure 26-2 Tivoli Workload Scheduler file system structure

Archived

The /TWA file system is the main entry point for the Tivoli Workload Automation installation. As part of this Tivoli Workload Automation installation, Tivoli Workload Scheduler is installed in the /TWA/TWS file system; the Dynamic Workload Broker Component is installed in /TWA/TDWB. Because the embedded WebSphere Application Server instance is common to Tivoli Workload Scheduler and the Dynamic Workload Broker Component, it is installed in the /TWA file system. Finally, a separate file system /TWA/TWS/stdlist for Tivoli Workload Scheduler traces and job logs is created.

Example 26-7 outlines the steps to be performed as root on twspoc1 and twspoc2.

Example 26-7 Tivoli Workload Scheduler user and file system creation

```
# crfs -v jfs2 -g twsvg -m /TWA -A yes -a size=1G
# crfs -v jfs2 -g twsvg -m /TWA/TWS -A yes -a size=2G
# crfs -v jfs2 -g twsvg -m /TWA/TWS/stdlist -A yes -a size=512M
# crfs -v jfs2 -g twsvg -m /TWA/TDWB -A yes -a size=512M

# mount /TWA
# mkdir /TWA/TWS /TWA/TDWB
# mount /TWA/TWS
# mkdir /TWA/TWS/stdlist
# mount /TWA/TWS/stdlist
# mount /TWA/TDWB
# mkuser -R compat id=502 pgrp=twsgrp home=/TWA/TWS rlogin=false cpu=-1 data=-1
stack=-1 cpu_hard=-1 data_hard=-1 stack_hard=-1 twsusr
# passwd twsusr
# chown -R twsusr.twsgrp /TWA/TWS /TWA/TDWB
```

26.4.2 Installing the Master Domain Manager

The following steps must be performed on twspoc1 as root.:

1. Start the setup.
2. Select the language for the installer. Click **Next**.
3. Click **Next** on the welcome window.
4. Accept the license agreement. Click **Next**.
5. Select **Master domain manager**, as shown in Figure 26-3. Click **Next**.

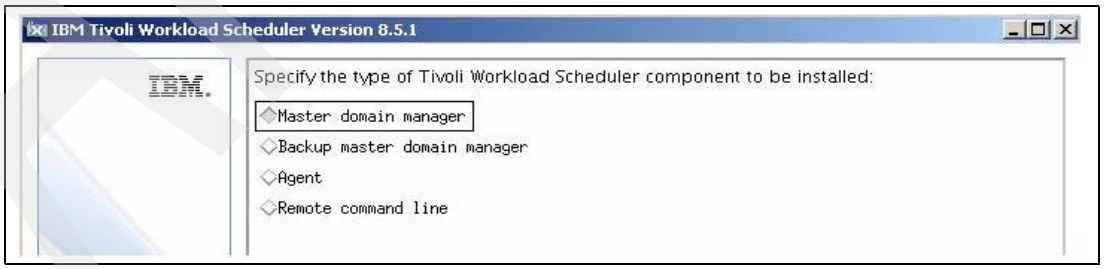


Figure 26-3 Tivoli Workload Scheduler setup, 1

6. Specify the Tivoli Workload Scheduler user and password. Click **Next**. See Figure 26-4 on page 769.

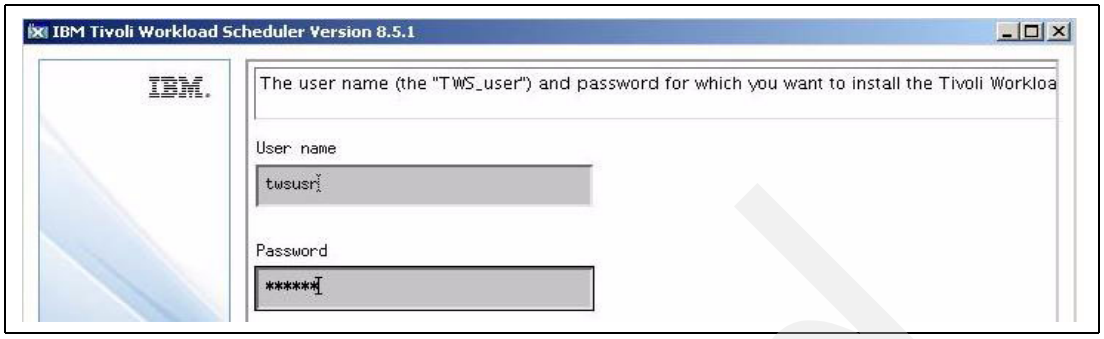


Figure 26-4 Tivoli Workload Scheduler setup, 2

7. Specify the company name and workstation name (`TWSPOC1`). Do *not* select **Add the FINAL job stream to the database to automate the production cycle** because it will be done manually. Click **Next**.
8. Specify the installation location (`/TWA`). Click **Next**. See Figure 26-5.

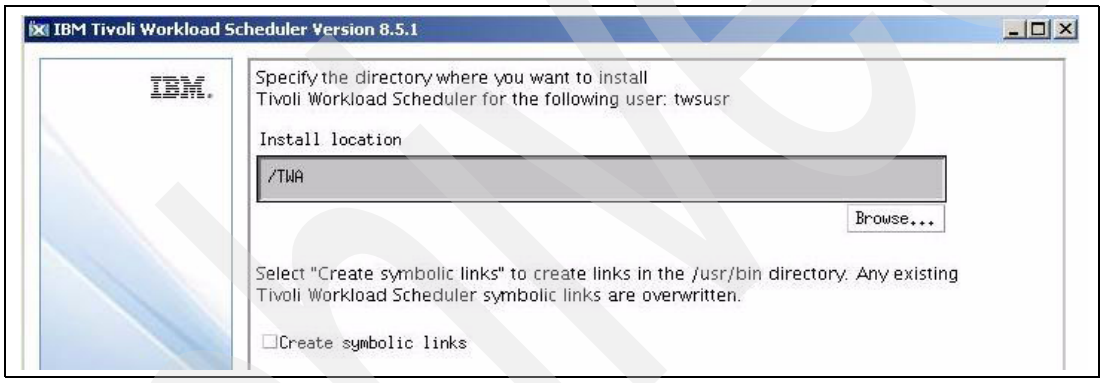


Figure 26-5 Tivoli Workload Scheduler setup, 3

9. Select **DB2 Database Enterprise Edition**. Click **Next**.
10. Specify the DB2 search path (`/db2/TWS/sqllib`), as shown in Figure 26-6. Click **Next**.

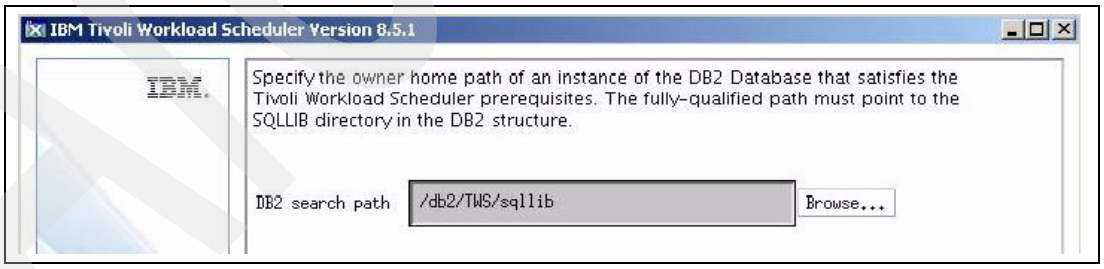


Figure 26-6 Tivoli Workload Scheduler setup, 4

11. Enter the DB2 instance details (name, port, instance owner, password). Click **Next**.
12. Enter the table space details. Select **Specify the advanced configuration parameters for the Tivoli Workload Scheduler database**. The following path is for the table space:

    `TWS_DATA is /db2/TWS/data/db2tws/NODE0000/SQL00001/TWS_DATA`

    The following path is for the TWS_LOG table space

    `/db2/TWS/data/db2tws/NODE0000/SQL00001/TWS_LOG`

Click **Next** to continue. See Figure 26-7.

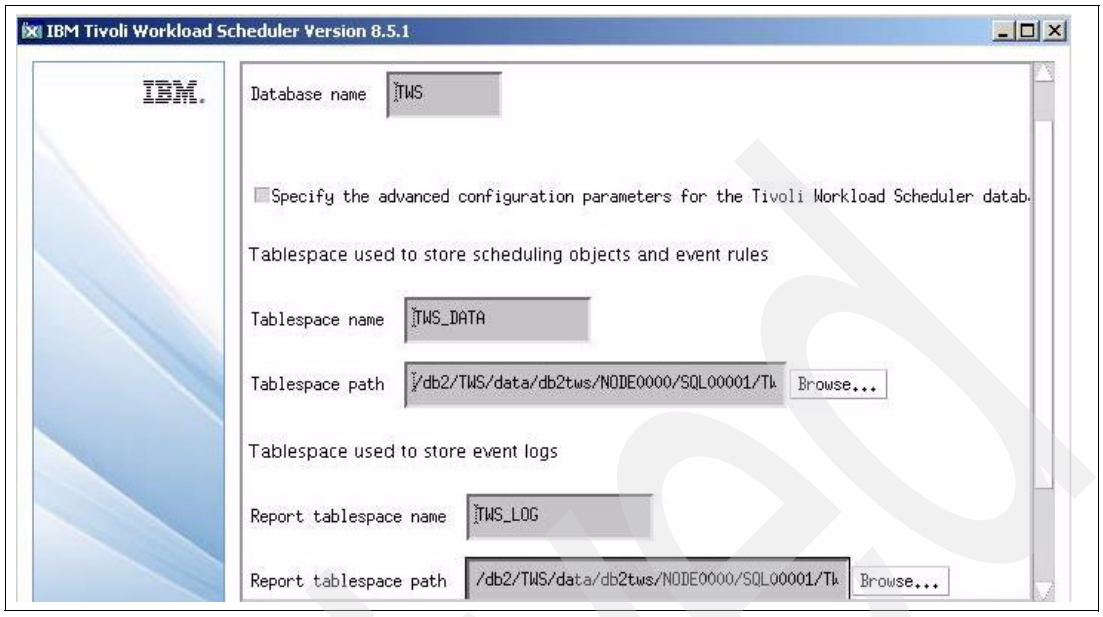


Figure 26-7 Tivoli Workload Scheduler setup, 5

13. Do *not* select the **Enable the dynamic scheduling capability** check box. See Figure 26-8. The Dynamic Workload Broker Component is configured separately. Click **Next**.

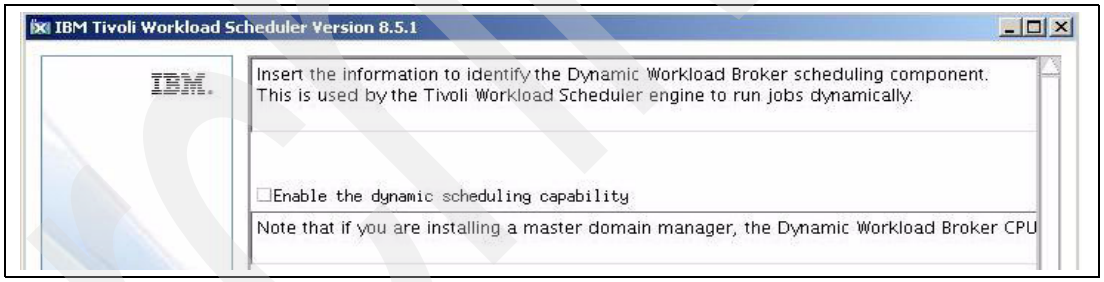


Figure 26-8 Tivoli Workload Scheduler setup, 6

14. Verify all settings. Click **Next**.
15. Click **Finish** when the installation is complete.
16. Upgrade the master domain manager to Fix Pack 1.
17. After the setup, stop the embedded WebSphere Application Server instance on twspoc1, as shown in Example 26-8.

Example 26-8 Stopping embedded WebSphere Application Server instance on twspoc1

```
$ conman 'stopappserver;wait' 2>/dev/null
%stopappserver;wait
AWSBHU622I A stop command was issued for the application server on workstation
"TWSPOC1".
```

Archived

26.4.3 Configuring HADR and ACR

Perform the following steps as db2tws on twspoc1:

1. Update the location of the log files and archived log files, as shown in Example 26-9.

Example 26-9 Updating location of log files and archived log files

```
$ db2 update db cfg for TWS using NEWLOGPATH /db2/TWS/log_dir
DB20000I  The UPDATE DATABASE CONFIGURATION command completed successfully.
$ db2 update db cfg for TWS using LOGARCHMETH1 DISK:/db2/TWS/log_arch
DB20000I  The UPDATE DATABASE CONFIGURATION command completed successfully.
```

2. Set the LOGINDEXBUILD parameter so that index creation, re-creation, or reorganization operations are logged. See Example 26-10.

Example 26-10 Enabling LOGINDEXBUILD

```
$ db2 update db cfg for TWS using LOGINDEXBUILD ON
DB20000I  The UPDATE DATABASE CONFIGURATION command completed successfully.
```

3. Back up the Tivoli Workload Scheduler database, as illustrated in Example 26-11.

Example 26-11 Backing up DB2 database

```
$ db2 backup db TWS to /tmp
Backup successful. The timestamp for this backup image is : 20100519203148
```

4. Copy the backup image to twspoc2, for example using secure copy (SCP).

On twspoc2, perform the following steps as db2tws.

1. Restore the Tivoli Workload Scheduler database, as outlined in Example 26-12.

Example 26-12 Restoring DB2 database

```
$ db2 restore db TWS from /tmp replace history file
DB20000I  The RESTORE DATABASE command completed successfully.
```

2. Configure HADR, as shown in Example 26-13.

Example 26-13 Configuring HADR on twspoc2

```
$ db2 update db cfg for TWS using HADR_LOCAL_HOST twspoc2
DB20000I  The UPDATE DATABASE CONFIGURATION command completed successfully.
$ db2 update db cfg for TWS using HADR_LOCAL_SVC 55002
DB20000I  The UPDATE DATABASE CONFIGURATION command completed successfully.
$ db2 update db cfg for TWS using HADR_REMOTE_HOST twspoc1
DB20000I  The UPDATE DATABASE CONFIGURATION command completed successfully.
$ db2 update db cfg for TWS using HADR_REMOTE_SVC 55001
DB20000I  The UPDATE DATABASE CONFIGURATION command completed successfully.
$ db2 update db cfg for TWS using HADR_REMOTE_INST db2tws
DB20000I  The UPDATE DATABASE CONFIGURATION command completed successfully.
$ db2 update db cfg for TWS using HADR_TIMEOUT 10
DB20000I  The UPDATE DATABASE CONFIGURATION command completed successfully.
$ db2 update db cfg for TWS using HADR_PEER_WINDOW 120
DB20000I  The UPDATE DATABASE CONFIGURATION command completed successfully.
```

Archive

3. Configure ACR, as illustrated in Example 26-14.

Example 26-14 Configuring ACR on twspoc2

```
$ db2 update alternate server for db TWS using hostname twspoc1 port 50000
DB20000I  The UPDATE ALTERNATE SERVER FOR DATABASE command completed
successfully.
```

4. Restart DB2, using db2stop and db2start.
5. Start HADR on twspoc2 as standby, as shown in Example 26-15.

Example 26-15 Starting HADR on twspoc2

```
$ db2 start hadr on db TWS as standby
DB20000I  The START HADR ON DATABASE command completed successfully.
```

On twspoc1, perform the following steps as db2tws:

1. Configure HADR, as illustrated in Example 26-16.

Example 26-16 Configuring HADR on twspoc1

```
$ db2 update db cfg for TWS using HADR_LOCAL_HOST twspoc1
DB20000I  The UPDATE DATABASE CONFIGURATION command completed successfully.
$ db2 update db cfg for TWS using HADR_LOCAL_SVC 55001
DB20000I  The UPDATE DATABASE CONFIGURATION command completed successfully.
$ db2 update db cfg for TWS using HADR_REMOTE_HOST twspoc2
DB20000I  The UPDATE DATABASE CONFIGURATION command completed successfully.
$ db2 update db cfg for TWS using HADR_REMOTE_SVC 55002
DB20000I  The UPDATE DATABASE CONFIGURATION command completed successfully.
$ db2 update db cfg for TWS using HADR_REMOTE_INST db2tws
DB20000I  The UPDATE DATABASE CONFIGURATION command completed successfully.
$ db2 update db cfg for TWS using HADR_TIMEOUT 10
DB20000I  The UPDATE DATABASE CONFIGURATION command completed successfully.
$ db2 update db cfg for TWS using HADR_PEER_WINDOW 120
DB20000I  The UPDATE DATABASE CONFIGURATION command completed successfully.
```

2. Configure ACR, as shown in Example 26-17.

Example 26-17 Configuring ACR on twspoc1

```
$ db2 update alternate server for db TWS using hostname twspoc2 port 50000
DB20000I  The UPDATE ALTERNATE SERVER FOR DATABASE command completed
successfully.
```

3. Restart DB2, using db2stop and db2start.
4. Start HADR on twspoc1 as primary, as shown in Example 26-18.

Example 26-18 Starting HADR on twspoc1

```
$ db2 start hadr on db TWS as primary
DB20000I  The START HADR ON DATABASE command completed successfully.
```

5. As db2tws, verify that the status of the HADR pair is Peer, as illustrated in Example 26-19 on page 773.

Example 26-19 Verifying HADR status

```
$ db2pd -d TWS –hadr | grep –p 'HADR Information'
HADR Information:
Role      State      SyncMode HeartBeatsMissed   LogGapRunAvg (bytes)
Primary   Peer       Nearsync 0                  0
```

On twspoc2, perform a takeover on the Tivoli Workload Scheduler database as db2tws, as outlined in Example 26-20.

Example 26-20 Performing HADR takeover on twspoc2

```
$ db2 takeover hadr on db TWS
DB20000I  The TAKEOVER HADR ON DATABASE command completed successfully.
```

26.4.4 Installing the Backup Master Domain Manager

Perform the following steps on twspoc2 as root.:

1. Start the setup.
2. Select the language for the installer. Click **Next**.
3. Click **Next** on the welcome window.
4. Accept the license agreement. Click **Next**.
5. Select **Backup master domain manager**, as shown in Figure 26-9. Click **Next**.

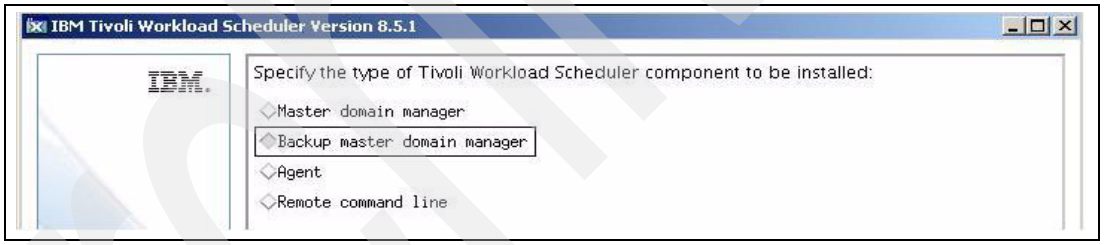


Figure 26-9 Tivoli Workload Scheduler setup, 7

6. Specify the Tivoli Workload Scheduler user and password. Click **Next**.
7. Specify the company name, workstation name (TWSPOC2), and Master Domain Manager (TWSPOC1).
8. Specify the install location (`/TWA`). Click **Next**.
9. Select **DB2 Database Enterprise Edition**. Click **Next**.
10. Specify the DB2 search path (`/db2/TWS/sqllib`). Click **Next**.
11. Enter the DB2 instance details (name, port, instance owner, password). Click **Next**.
12. The table space details cannot be altered during the installation of the backup master domain manager. Click **Next** to continue.
13. Do not select **Enable the dynamic scheduling capability,** as shown in Figure 26-10 on page 774. The Dynamic Workload Broker Component server is configured separately. Click **Next**.

Archived

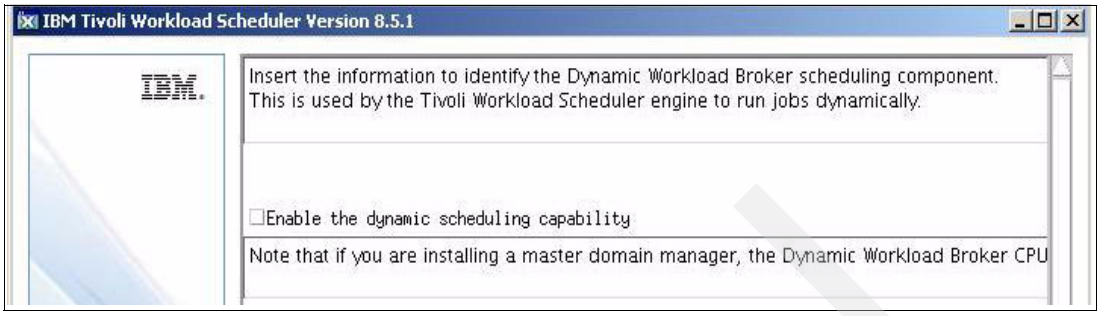


Figure 26-10 Tivoli Workload Scheduler setup, 8

14. Verify all settings. Click **Next**.
15. Click **Finish** when the installation is complete.
16. Upgrade the backup master domain manager to Fix Pack 1.
17. After the setup, stop the embedded WebSphere Application Server instance on twspoc2, as shown in Example 26-21.

Example 26-21 Stopping embedded WebSphere Application Server instance on twspoc2

```
$ conman 'stopappserver;wait' 2>/dev/null
%stopappserver;wait
AWSBHU622I A stop command was issued for the application server on workstation
"TWSPOC2".
```

18. Finally, as db2tws, perform a takeover on the Tivoli Workload Scheduler database on twspoc1, as illustrated in Example 26-22.

Example 26-22 Performing HADR takeover on twspoc1

```
$ db2 takeover hadr on db TWS
DB20000I  The TAKEOVER HADR ON DATABASE command completed successfully.
```

26.5 Installing and configuring Tivoli System Automation for Multiplatforms

This section describes the steps to install and configure the Tivoli System Automation for Multiplatforms cluster. Perform the following steps on twspoc1 and twspoc2 as root.:

1. Install the Tivoli System Automation for Multiplatforms software and prepare node security, as shown in Example 26-23.

Example 26-23 Tivoli System Automation for Multiplatforms installation

```
# ./prereqSAM
prereqSAM: All prerequisites for the ITSAMP installation are met on operating
system:  AIX 6100-04
# ./installSAM
SAM is currently not installed.
. . .
installSAM: All packages were installed successfully.
# preprpnode twspoc1 twspoc2
```

2. Add `CT_MANAGEMENT_SCOPE=2` to `/etc/environment` and start a new shell. This environment variable determines the scope of the cluster. If set to 2, all nodes in the cluster are in scope.
3. Install the HADR scripts for Tivoli System Automation for Multiplatforms, as shown in Example 26-24.

Example 26-24 Installing HADR scripts for Tivoli System Automation for Multiplatforms

```
# /db2/V9.5/install/tsamp/db2cptsa
DBI1099I  The DB2 HA scripts for the SA MP Base Component were
      successfully installed in /usr/sbin/rsct/sapolicies/db2.
...
The DB2 installer successfully installed the DB2 HA scripts.
```

On twspoc1, run the following tasks as root:

1. Create and start the Tivoli System Automation for Multiplatforms cluster, as illustrated in Example 26-25.

Example 26-25 Creating and starting the Tivoli System Automation for Multiplatforms cluster

```
# mkrpdomain TWS twspoc1 twspoc2
# startrpdomain TWS
```

2. Verify the state of the cluster, as illustrated in Example 26-26.

Example 26-26 Verifying the state of the Tivoli System Automation for Multiplatforms cluster

```
# lsrpdomain
Name OpState RSCTActiveVersion MixedVersions TSPort GSPort
TWS  Online  2.5.5.1          No            12347  12348
# lsrpnode
Name    OpState RSCTVersion
twspoc1 Online  2.5.5.1
twspoc2 Online  2.5.5.1
```

3. Create a network tie breaker for the cluster, as shown in Example 26-27.

Example 26-27 Creating network tie breaker for the cluster

```
# mkequ VLAN1 IBM.NetworkInterface:en0:twspoc1,en0:twspoc2
# mkrsrc IBM.TieBreaker Type="EXEC" Name="NETWORKTB"
DeviceInfo='PATHNAME=/usr/sbin/rsct/bin/samtb_net Address=192.168.1.1 Log=1'
PostReserveWaitTime=30
# chrsrc -c IBM.PeerNode OpQuorumTieBreaker="NETWORKTB"
```

26.6 Integrating DB2 HADR and Tivoli System Automation for Multiplatforms

This section integrates the existing DB2 HADR pair in the Tivoli System Automation for Multiplatforms cluster using *db2haicu* (DB2 High Availability Instance Configuration Utility). This tool automatically creates the required DB2 HADR resources in the cluster based on an XML input file, as shown in Example 26-28 on page 776.

Example 26-28 XML input file for db2haicu

```
<?xml version="1.0" encoding="UTF-8"?>
<DB2Cluster xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="db2ha.xsd" clusterManagerName="TSA" version="1.0">
   <ClusterDomain domainName="TWS">
      <ClusterNode clusterNodeName="twspoc1"/>
      <ClusterNode clusterNodeName="twspoc2"/>
   </ClusterDomain>
   <FailoverPolicy>
      <HADRFailover/>
   </FailoverPolicy>
   <DB2PartitionSet>
      <DB2Partition dbpartitionnum="0" instanceName="db2tws"/>
   </DB2PartitionSet>
   <HADRDBSet>
      <HADRDB databaseName="TWS" localInstance="db2tws"
remoteInstance="db2tws" localHost="twspoc1" remoteHost="twspoc2"/>
      <VirtualIPAddress baseAddress="192.168.1.202"
subnetMask="255.255.255.0" networkName="VLAN1"/>
   </HADRDBSet>
</DB2Cluster>
```

Run the following commands on twspoc2 as db2tws:

1. Verify that twspoc2 is the standby node in the HADR pair, as shown in Example 26-29.

   Example 26-29 Verifying HADR standby node

   ```
   $ /db2/TWS/sqllib/bin/db2 get snapshot for database on TWS | grep Role
     Role                    = Standby
   ```

2. Run the db2haicu setup, as illustrated in Example 26-30.

   Example 26-30 The db2haicu setup

   ```
   $ /db2/V9.5/bin/db2haicu -f ./db2haicu.xml
   Welcome to the DB2 High Availability Instance Configuration Utility (db2haicu).
   . . .
   Adding DB2 database partition 0 to the cluster ...
   Adding DB2 database partition 0 to the cluster was successful.
   . . .
   All cluster configurations have been completed successfully.
   db2haicu exiting ...
   ```

Run the following commands on twspoc1 as db2tws.

1. Verify that twspoc1 is the primary node in the HADR pair, as shown in Example 26-31.

   Example 26-31 Verifying HADR primary node

   ```
   $ /db2/TWS/sqllib/bin/db2 get snapshot for database on TWS | grep Role
     Role                    = Primary
   ```

2. Run the **db2haicu** setup, as illustrated in Example 26-32 on page 777.

Example 26-32 The db2haicu setup

```
$ /db2/V9.5/bin/db2haicu -f ./db2haicu.xml
Welcome to the DB2 High Availability Instance Configuration Utility (db2haicu).
. . .
Adding DB2 database partition 0 to the cluster ...
Adding DB2 database partition 0 to the cluster was successful.
Adding HADR database TWS to the domain ...
Adding HADR database TWS to the domain was successful.

All cluster configurations have been completed successfully.
db2haicu exiting ...
```

3. As root, verify the state of the resource groups, as shown in the **lssam** output of Example 26-33.

Example 26-33 Output of lssam command

```
Online IBM.ResourceGroup:db2_db2tws_db2tws_TWS-rg Nominal=Online
   |- Online IBM.Application:db2_db2tws_db2tws_TWS-rs
      |- Online IBM.Application:db2_db2tws_db2tws_TWS-rs:twspoc1
      '- Offline IBM.Application:db2_db2tws_db2tws_TWS-rs:twspoc2
   '- Online IBM.ServiceIP:db2ip_192_168_1_202-rs
      |- Online IBM.ServiceIP:db2ip_192_168_1_202-rs:twspoc1
      '- Offline IBM.ServiceIP:db2ip_192_168_1_202-rs:twspoc2
Online IBM.ResourceGroup:db2_db2tws_twspoc1_0-rg Nominal=Online
   '- Online IBM.Application:db2_db2tws_twspoc1_0-rs
      '- Online IBM.Application:db2_db2tws_twspoc1_0-rs:twspoc1
Online IBM.ResourceGroup:db2_db2tws_twspoc2_0-rg Nominal=Online
   '- Online IBM.Application:db2_db2tws_twspoc2_0-rs
      '- Online IBM.Application:db2_db2tws_twspoc2_0-rs:twspoc2
Online IBM.Equivalency:db2_db2tws_db2tws_TWS-rg_group-equ
   |- Online IBM.PeerNode:twspoc1:twspoc1
   '- Online IBM.PeerNode:twspoc2:twspoc2
Online IBM.Equivalency:db2_db2tws_twspoc1_0-rg_group-equ
   '- Online IBM.PeerNode:twspoc1:twspoc1
Online IBM.Equivalency:db2_db2tws_twspoc2_0-rg_group-equ
   '- Online IBM.PeerNode:twspoc2:twspoc2
```

As a result, Tivoli System Automation for Multiplatforms fully control the Tivoli Workload Scheduler database. If a problem occurs with the DB2 primary node, Tivoli System Automation for Multiplatforms automatically performs a failover to the DB2 standby node.

Note: The virtual IP address is collocated with the DB2 primary node.

26.7 Integrating Tivoli Workload Scheduler and Tivoli System Automation for Multiplatforms for high availability

This section integrates Tivoli Workload Scheduler and Tivoli System Automation for Multiplatforms. Each Tivoli Workload Scheduler component is described separately.

26.7.1 The embedded WebSphere Application Server

On twspoc1 and twspoc2, perform the following steps as root.

1. Create a file (`/tmp/ds.txt`) that contains the entries shown in Example 26-34. DB2Type4ServerName will contain the virtual IP address of the DB2 primary node.

Example 26-34 Updating DataSourceProperties, 1

```
DB2Type4ServerName=192.168.1.202
DB2Type4DbFailOverEnabled=true
```

2. Update the data source properties, as illustrated in Example 26-35.

Example 26-35 Updating DataSourceProperties, 2

```
# /TWA/wastools/changeDataSourceProperties.sh /tmp/ds.txt
...
Configuring JDBC Path Variables...
Configuring DB2 Type2 DataSource...
Configuring DB2 Type4 DataSource...
Configuring Oracle Type2 DataSource...
Configuring Oracle Type4 DataSource...
Configuring JNDI Names...
Saving data...
```

Note: Enabling the `DB2Type4DbFailOverEnabled` custom property instructs the JDBC driver of the embedded WebSphere Application Server to retry the connection to the database `DB2Type4ConnRetriesDuringDBFailover` times (default 100). The interval is regulated by the `DB2Type4ConnRetryIntervalDuringDBFailover` option (default is3000 ms).

As twsusr on twspoc1 and twspoc2, disable the auto restart option of the embedded WebSphere Application Server in the localopts file (`Appserver auto restart = no`). The reason is because Tivoli System Automation for Multiplatforms will be responsible for monitoring and managing the embedded WebSphere Application Server instance.

Next, the Tivoli System Automation for Multiplatforms resources, resource groups and equivalencies for the embedded WebSphere Application Server instances are created.

Run the following commands on twspoc1 as root:

1. Create resources and resource groups for the embedded WebSphere Application Server instances, as shown in Example 26-36.

Example 26-36 Creating resources and resource groups for the eWAS instances

```
# mkrsrc -f ./eWAS_twspoc1-rs.def IBM.Application
# mkrsrc -f ./eWAS_twspoc2-rs.def IBM.Application
# mkrg -n twspoc1 eWAS_twspoc1-rg
# mkrg -n twspoc2 eWAS_twspoc2-rg
# addrgmbr -g eWAS_twspoc1-rg IBM.Application:eWAS_twspoc1-rs:twspoc1
# addrgmbr -g eWAS_twspoc2-rg IBM.Application:eWAS_twspoc2-rs:twspoc2
# resetrsrc -s 'Name == "eWAS_twspoc1-rs"' IBM.Application
# resetrsrc -s 'Name == "eWAS_twspoc2-rs"' IBM.Application
```

2. Create managed relations for the embedded WebSphere Application Server instances, as illustrated in Example 26-37.

Example 26-37 Creating managed relations for the eWAS instances

```
# mkrel -p StartAfter -S IBM.ResourceGroup:eWAS_twspoc1-rg -G
IBM.ResourceGroup:db2_db2tws_db2tws_TWS-rg
# mkrel -p StartAfter -S IBM.ResourceGroup:eWAS_twspoc2-rg -G
IBM.ResourceGroup:db2_db2tws_db2tws_TWS-rg
```

3. Create shadow resources and a shadow equivalency for the embedded WebSphere Application Server instances, as shown in Example 26-38.

Example 26-38 Creating shadow resources and shadow equivalency for the eWAS instances

```
# mkrsrc -f ./eWAS_twspoc1_shadow-rs.def IBM.Application
# mkrsrc -f ./eWAS_twspoc2_shadow-rs.def IBM.Application
# mkequ -p A,NoControl eWAS_shadow-equ
IBM.Application:eWAS_twspoc1_shadow-rs:twspoc1,eWAS_twspoc2_shadow-rs:twspoc2
```

4. Start the resource groups for the embedded WebSphere Application Server instances, as illustrated in Example 26-39.

Example 26-39 Starting eWAS resource groups

```
# chrg -o online eWAS_twspoc1-rg
# chrg -o online eWAS_twspoc2-rg
```

The creation of the Tivoli System Automation for Multiplatforms objects for the embedded WebSphere Application Server instances results in the output of the **lssam** command, as shown in Example 26-40.

Example 26-40 Output of lssam command

```
Online IBM.ResourceGroup:eWAS_twspoc1-rg Nominal=Online
        '- Online IBM.Application:eWAS_twspoc1-rs:twspoc1
Online IBM.ResourceGroup:eWAS_twspoc2-rg Nominal=Online
        '- Online IBM.Application:eWAS_twspoc2-rs:twspoc2
Online IBM.Equivalency:eWAS_shadow-equ
        |- Online IBM.Application:eWAS_twspoc1_shadow-rs:twspoc1
        '- Online IBM.Application:eWAS_twspoc2_shadow-rs:twspoc2
```

Archived

Figure 26-11 shows the setup of the embedded WebSphere Application Server and DB2 instances.

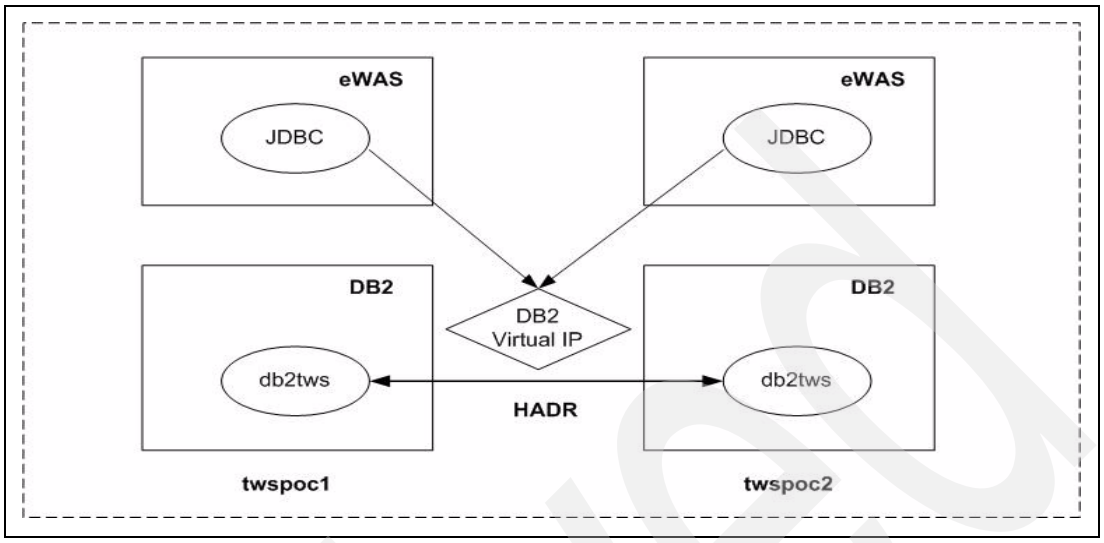


Figure 26-11 DB2 and embedded WebSphere Application Server setup

26.7.2 Engine

This section focuses on the Tivoli Workload Scheduler engine. Perform the following steps on twspoc1 as twsusr:

1. Enable Switch Fault Tolerance, as shown in Example 26-41.

Example 26-41 Enabling Switch Fault Tolerance

```
$ optman chg enSwFaultTol=YES
AWSJCL050I Command "chg" completed successfully.
```

2. Create the virtual master workstation by using Composer command line, as shown in Example 26-42.

Example 26-42 Creating the virtual master workstation

```
CPUNAME XA_VMDM
  OS UNIX
  NODE NotApplicable TCPADDR 31111
  FOR MAESTRO HOST $MASTER ACCESS "unixlocl"
    TYPE X-AGENT
    AUTOLINK OFF
    BEHINDFIREWALL OFF
    FULLSTATUS OFF
END
```

3. Create the CPU definition for the backup master domain manager by using composer, as illustrated in Example 26-43 on page 781. Note that the FULLSTATUS flag is enabled for the backup master domain manager workstation.

Archive

Example 26-43 Creating CPU definition for the backup master domain manager

```
CPUNAME TWSPOC2
  OS UNIX
  NODE twspoc2 TCPADDR 31111
  DOMAIN MASTERDM
  FOR MAESTRO
    TYPE FTA
    AUTOLINK ON
    BEHINDFIREWALL OFF
    FULLSTATUS ON
END
```

4. Add the FINAL stream to the database by using composer, as shown in Example 26-44.

Example 26-44 Adding the FINAL stream to the database

```
SCHEDULE XA_VMDM#FINAL
ON RUNCYCLE RULE1 "FREQ=DAILY;INTERVAL=1"
AT XXXX
CARRYFORWARD
FOLLOWS XA_VMDM#FINAL.@ PREVIOUS
:
XA_VMDM#MAKEPLAN
 SCRIPTNAME "/TWA/TWS/MakePlan"
 STREAMLOGON twsusr
 TASKTYPE UNIX
 RCCONDSUCC "(RC=0) OR (RC=4)"
 RECOVERY STOP

XA_VMDM#SWITCHPLAN
 SCRIPTNAME "/TWA/TWS/SwitchPlan"
 STREAMLOGON twsusr
 TASKTYPE UNIX
 RECOVERY STOP
 FOLLOWS LOCKRESOURCEGROUPS

XA_VMDM#CREATEPOSTREPORTS
 SCRIPTNAME "/TWA/TWS/CreatePostReports"
 STREAMLOGON twsusr
 TASKTYPE UNIX
 RECOVERY CONTINUE
 FOLLOWS SWITCHPLAN

XA_VMDM#UPDATESTATS
 SCRIPTNAME "/TWA/TWS/UpdateStats"
 STREAMLOGON twsusr
 TASKTYPE UNIX
 RECOVERY CONTINUE
 FOLLOWS SWITCHPLAN

XA_VMDM#LOCKRESOURCEGROUPS
 SCRIPTNAME "/usr/sbin/rsct/sapolicies/tws/lockResourceGroups.ksh"
 STREAMLOGON twsusr
 TASKTYPE UNIX
 RECOVERY STOP
 FOLLOWS MAKEPLAN

XA_VMDM#UNLOCKRESOURCEGROUPS
 SCRIPTNAME "/usr/sbin/rsct/sapolicies/tws/unlockResourceGroups.ksh"
 STREAMLOGON twsusr
```

Archived

```
 TASKTYPE UNIX
 RECOVERY STOP
 FOLLOWS SWITCHPLAN
END
```

On twspoc1 and twspoc2, perform the following steps as twsusr:

1. Update the JnextPlan script, as shown in Example 26-45.

Example 26-45 Updating JnextPlan script

```
. . .
#conman “startappserver; wait”
MakePlan $*
/usr/sbin/rsct/sapolicies/tws/lockResourceGroups.ksh
SwitchPlan
/usr/sbin/rsct/sapolicies/tws/unlockResourceGroups.ksh &
. . .
```

2. Disable *mm resolve master* in the `localopts` file.

On twspoc1, perform the following steps as twsusr.

1. Create the initial plan, using the JnextPlan script.
2. Set the limit for the workstations, as shown in Example 26-46.

Example 26-46 Setting the limit for the workstations

```
$ conman ‘lc @;limit 10;noask’
%lc @;limit 10;noask
Command forwarded to batchman for TWSPOC1
Command forwarded to batchman for TWSPOC2
Command forwarded to batchman for XA_VMDM
```

Finally, stop the Tivoli Workload Scheduler engine on twspoc1 and twspoc2 by running the **conman stop** and **conman shut** commands.

Next, the Tivoli System Automation for Multiplatforms resources, resource groups and equivalencies for the engine are created. Perform the following steps on twspoc1 as root:

1. Create resources and resource groups for the engine processes, as shown in Example 26-47.

Example 26-47 Creating resources and resource groups for the engine processes

```
# mkrsrc -f ./netman_twspoc1-rs.def IBM.Application
# mkrsrc -f ./netman_twspoc2-rs.def IBM.Application
# mkrsrc -f ./mailman_twspoc1-rs.def IBM.Application
# mkrsrc -f ./mailman_twspoc2-rs.def IBM.Application
# mkrg -n twspoc1 fta_twspoc1-rg
# mkrg -n twspoc2 fta_twspoc2-rg
# addrgmbr -g fta_twspoc1-rg IBM.Application:netman_twspoc1-rs:twspoc1
# addrgmbr -g fta_twspoc2-rg IBM.Application:netman_twspoc2-rs:twspoc2
# addrgmbr -g fta_twspoc1-rg IBM.Application:mailman_twspoc1-rs:twspoc1
# addrgmbr -g fta_twspoc2-rg IBM.Application:mailman_twspoc2-rs:twspoc2
# resetrsrc -s 'Name == "netman_twspoc1-rs"' IBM.Application
# resetrsrc -s 'Name == "mailman_twspoc1-rs"' IBM.Application
# resetrsrc -s 'Name == "netman_twspoc2-rs"' IBM.Application
# resetrsrc -s 'Name == "mailman_twspoc2-rs"' IBM.Application
```

Archived

2. Create managed relations for the engine processes, as shown in Example 26-48.

Example 26-48 Creating managed relations for the engine processes

```
# mkrel -p StartAfter -S IBM.Application:mailman_twspoc1-rs:twspoc1 -G
IBM.Application:netman_twspoc1-rs:twspoc1
# mkrel -p StartAfter -S IBM.Application:mailman_twspoc2-rs:twspoc2 -G
IBM.Application:netman_twspoc2-rs:twspoc2
```

3. Create shadow resources and a shadow equivalency for the mailman processes, as illustrated in Example 26-49.

Example 26-49 Creating shadow resources and shadow equivalency for mailman processes

```
# mkrsrc -f ./mailman_twspoc1_shadow-rs.def IBM.Application
# mkrsrc -f ./mailman_twspoc2_shadow-rs.def IBM.Application
# mkequ -p A,NoControl mailman_shadow-equ
IBM.Application:mailman_twspoc1_shadow-rs:twspoc1,mailman_twspoc2_shadow-rs:tws
poc2
```

4. Start the resource groups for the engine processes, as shown in Example 26-50.

Example 26-50 Starting resource groups for engine processes

```
# chrg -o online fta_twspoc1-rg
# chrg -o online fta_twspoc2-rg
```

Important: The batchman and jobman processes are not monitored because mailman, batchman, and jobman act as one logical unit. If one of them terminates unexpectedly, they all stop.

Finally, create the following Tivoli System Automation for Multiplatforms objects for the master domain manager. Perform the following steps as root on twspoc1:

1. Create a floating resource and a resource group for the master domain manager, as illustrated in Example 26-51.

Example 26-51 Creating resources and resource groups for the master domain manager

```
# mkrsrc -f ./mdm-rs.def IBM.Application
# mkrg mdm-rg
# addrgmbr -g mdm-rg IBM.Application:mdm-rs
```

2. Create a virtual IP address which will be collocated with the master domain manager, as shown in Example 26-52. This virtual IP address will be used by the CLI tools on the FTAs.

Example 26-52 Creating virtual IP address for the master domain manager

```
# mkrsrc IBM.ServiceIP Name="mdm_192_168_1_203-rs"
IPAddress="192.168.1.203" NetMask="255.255.255.0"
NodeNameList="{'twspoc1','twspoc2'}"
# addrgmbr -g mdm-rg IBM.ServiceIP:mdm_192_168_1_203-rs
```

3. Create the managed relations for the master domain manager, as shown in Example 26-53.

Example 26-53 Creating managed relations for the master domain manager

```
# mkrel -p DependsOn -S IBM.ServiceIP:mdm_192_168_1_203-rs -G
IBM.Equivalency:VLAN1
# mkrel -p DependsOn -S IBM.ServiceIP:mdm_192_168_1_203-rs -G
IBM.Equivalency:eWAS_shadow-equ
# mkrel -p DependsOn -S IBM.Application:mdm-rs -G
IBM.Equivalency:mailman_shadow-equ
```

On twspoc1 and twspoc2 as root, create a least-privilege resource that will allow twsusr to run the **rgreq** command, as shown in Example 26-54.

Example 26-54 Creating least-privilege resource

```
# mklpcmd RGREQ /usr/sbin/rsct/bin/rgreq twsusr@LOCALHOST rx
```

On twspoc1 as root, start the resource group for the master domain manager, as shown in Example 26-55.

Example 26-55 Starting resource group for the master domain manager

```
# chrg –o online mdm-rg
```

The creation of the Tivoli System Automation for Multiplatforms objects for the Tivoli Workload Scheduler engine results in the output of the **lssam** command, as shown in Example 26-56.

Example 26-56 lssam: Tivoli Workload Scheduler engine

```
Online IBM.ResourceGroup:fta_twspoc1-rg Nominal=Online
        |- Online IBM.Application:mailman_twspoc1-rs:twspoc1
        '- Online IBM.Application:netman_twspoc1-rs:twspoc1
Online IBM.ResourceGroup:fta_twspoc2-rg Nominal=Online
        |- Online IBM.Application:mailman_twspoc2-rs:twspoc2
        '- Online IBM.Application:netman_twspoc2-rs:twspoc2
Online IBM.ResourceGroup:mdm-rg Nominal=Online
        '- Online IBM.Application:mdm-rs
                |- Online IBM.Application:mdm-rs:twspoc1
                '- Offline IBM.Application:mdm-rs:twspoc2
        '- Online IBM.ServiceIP:mdm_192_168_1_203-rs
                |- Online IBM.ServiceIP:mdm_192_168_1_203-rs:twspoc1
                '- Offline IBM.ServiceIP:mdm_192_168_1_203-rs:twspoc2
```

Archived

Figure 26-12 illustrates the setup.

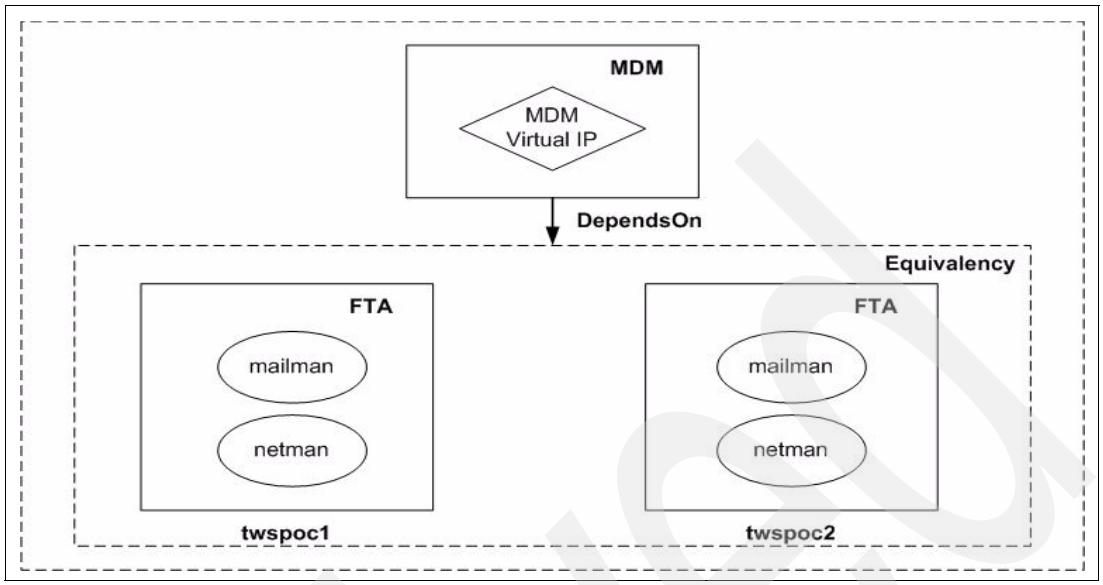


Figure 26-12 FTA and MDM setup

26.7.3 Event Processor

The following subsection will focus on the Event Processor, which can be hosted by either the master domain manager or the backup master domain manager.

On twspoc1, perform the following steps as root.

1. Create a floating resource and a resource group for the Event Processor, as shown in Example 26-57.

Example 26-57 Creating floating resource and resource group for the Event Processor

```
# mkrsrc -f ./evtproc-rs.def IBM.Application
# mkrg evtproc-rg
# addrgmbr -g evtproc-rg IBM.Application:evtproc-rs
# resetrsrc -s 'Name == "evtproc-rs"' IBM.Application
# mkrel -p DependsOn -S IBM.Application:evtproc-rs -G IBM.Equivalency:eWAS_shadow-equ
```

2. Start the resource group for the Event Processor, as shown in Example 26-58.

Example 26-58 Starting resource group for the Event Processor

```
# chrg –o online evtproc-rg
```

Archived

The creation of the Tivoli System Automation for Multiplatforms objects results in the output of the **lssam** command, as shown in Example 26-59.

Example 26-59 The lssam command output

```
Online IBM.ResourceGroup:eWAS_twspoc1-rg Nominal=Online
        '- Online IBM.Application:eWAS_twspoc1-rs:twspoc1
Online IBM.ResourceGroup:eWAS_twspoc2-rg Nominal=Online
        '- Online IBM.Application:eWAS_twspoc2-rs:twspoc2
Online IBM.ResourceGroup:evtproc-rg Nominal=Online
        '- Online IBM.Application:evtproc-rs
                |- Online IBM.Application:evtproc-rs:twspoc1
                '- Offline IBM.Application:evtproc-rs:twspoc2
```

Figure 26-13 illustrates the dependency.

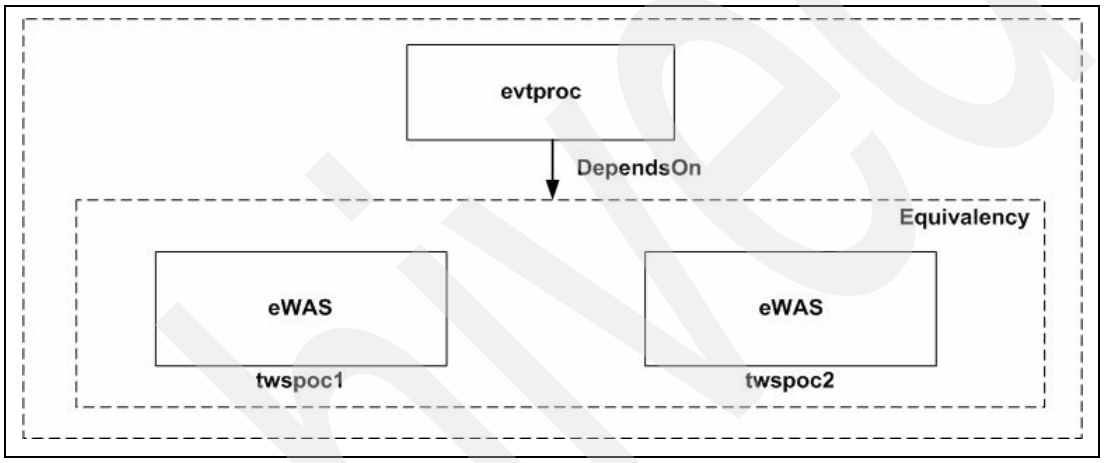


Figure 26-13 Event Processor setup

26.7.4 Tivoli Dynamic Workload Console

Prepare the environment for Tivoli Workload Dynamic Console. On twspoc1, a standard volume group (tdwcvg) was created on an hdisk shared between twspoc1 and twspoc2. After, a file system for the Tivoli Dynamic Workload Console was created (`/TDWC`), with automount disabled. Next, the volume group was imported on twspoc2. Finally, the automatic varyon of the volume group was disabled on both nodes.

To install Tivoli Dynamic Workoad Console for high availability, perform the following steps on twspoc1 as root:

1. Install the Tivoli Dynamic Workload Console in a separate Tivoli Workload Automation instance (`/TDWC`).
2. Stop the Tivoli Dynamic Workload Console by using the **stopWas.sh** script.
3. In the `serverindex.xml` file, which is in the following location, locate the entries that refer to twspoc1:

   `/TDWC/eWAS/profiles/twaprofile/config/cells/DefaultNode/nodes/DefaultNode`

   Update the entries by changing them to the virtual address that will be associated with the Tivoli Dynamic Workload Console.
4. Unmount the file system and varyoff the volume group (also on twspoc2).

5. Create floating resources and a resource group for the Tivoli Dynamic Workload Console, as shown in Example 26-60.

Example 26-60 Creating resources for the Tivoli Dynamic Workload Console

```
# mkrg tdwc-rg
# mkrsrc -f ./tdwc-rs.def IBM.Application
# mkrsrc -f ./tdwcvg-rs.def IBM.Application
# addrgmbr -g tdwc-rg IBM.Application:tdwc-rs
# addrgmbr -g tdwc-rg IBM.Application:tdwcvg-rs
# mkrsrc IBM.AgFileSystem Name=tdwcfs-rs Vfs=JFS2 Force=1 ResourceType=1
NodeNameList='{"twspoc1","twspoc2"}' DeviceName=/dev/xxx MountPoint=/TDWC
# addrgmbr -g tdwc-rg IBM.AgFileSystem:tdwcfs-rs
# mkrsrc IBM.ServiceIP Name="tdwc_192_168_1_204-rs"
IPAddress="192.168.1.204" NetMask="255.255.255.0"
NodeNameList="{'twspoc1','twspoc2'}"
# addrgmbr -g tdwc-rg IBM.ServiceIP:tdwc_192_168_1_204-rs
```

6. Create the managed relations for the Tivoli Dynamic Workload Console, as illustrated in Example 26-61.

Example 26-61 Creating managed relations for the Tivoli Dynamic Workload Console

```
# mkrel -p DependsOn -S IBM.Application:tdwc-rs -G IBM.AgFileSystem:tdwcfs-rs
# mkrel -p DependsOn -S IBM.Application:tdwc-rs -G
IBM.ServiceIP:tdwc_192_168_1_204-rs
# mkrel -p DependsOn -S IBM.AgFileSystem:tdwcfs-rs -G IBM.Application:tdwcvg-rs
```

7. Start the resource group for the Tivoli Dynamic Workload Console, as shown in Example 26-62.

Example 26-62 Starting resource group for the Tivoli Dynamic Workload Console

```
# chrg –o online tdwc-rg
```

The creation of the Tivoli System Automation objects for the Tivoli Dynamic Workload Console results in the output of the **lssam** command, as shown in Example 26-63.

Example 26-63 lssam: Tivoli Dynamic Workload Console

```
Online IBM.ResourceGroup:tdwc-rg Nominal=Online
       |- Online IBM.AgFileSystem:tdwcfs-rs
               |- Online IBM.AgFileSystem:tdwcfs-rs:twspoc1
               '- Offline IBM.AgFileSystem:tdwcfs-rs:twspoc2
       |- Online IBM.Application:tdwc-rs
               |- Online IBM.Application:tdwc-rs:twspoc1
               '- Offline IBM.Application:tdwc-rs:twspoc2
       |- Online IBM.Application:tdwcvg-rs
               |- Online IBM.Application:tdwcvg-rs:twspoc1
               '- Offline IBM.Application:tdwcvg-rs:twspoc2
       '- Online IBM.ServiceIP:tdwc_192_168_1_204-rs
               |- Online IBM.ServiceIP:tdwc_192_168_1_204-rs:twspoc1
               '- Offline IBM.ServiceIP:tdwc_192_168_1_204-rs:twspoc2
```

Archived

26.7.5 Dynamic Workload Broker Component

To customize the Tivoli Workload Broker Component for high availability, perform the following commands as twsusr:

1. (twspoc1) Edit the configuration file BrokerWorkstation.properties in the /TWA/TDWB/config folder, as shown in Example 26-64.

Example 26-64 BrokerWorkstation.properties on twspoc1

```
Broker.Workstation.Name=BROKER
Broker.Workstation.RetryLink=60
MasterDomainManager.Name=TWSPOC1
```

2. (twspoc2) Edit the configuration file BrokerWorkstation.properties in the /TWA/TDWB/config folder, as illustrated in Example 26-65.

Example 26-65 BrokerWorkstation.properties on twspoc2

```
Broker.Workstation.Name=BROKER
Broker.Workstation.RetryLink=60
MasterDomainManager.Name=TWSPOC2
```

3. (twspoc1) Create a workstation for the Dynamic Workload Broker Component using Composer, as shown in Example 26-66.

Example 26-66 Defining the broker workstation

```
CPUNAME BROKER
  OS OTHER
  NODE localhost TCPADDR 41114
  DOMAIN MASTERDM
  FOR MAESTRO
    TYPE BROKER
    AUTOLINK ON
    BEHINDFIREWALL OFF
    FULLSTATUS OFF
END
```

4. (twspoc1) Run **JnextPlan -for 0000** command to make the broker workstation available in the plan.
5. (twspoc1) Increase the CPU limit of the broker workstation.

Archived

Perform the following steps on twspoc1 as root:

1. Create floating resources and a resource group for the Dynamic Workload Broker Component, as illustrated in Example 26-67.

Example 26-67 Creating resources for the Dynamic Workload Broker Component

```
# mkrg tdwb-rg
# mkrsrc -f ./tdwb-rs.def IBM.Application
# addrgmbr -g tdwb-rg IBM.Application:tdwb-rs
# mkrsrc IBM.ServiceIP Name="tdwb_192_168_1_205-rs"
IPAddress="192.168.1.205" NetMask="255.255.255.0"
NodeNameList="{'twspoc1','twspoc2'}"
# addrgmbr -g tdwb-rg IBM.ServiceIP:tdwb_192_168_1_205-rs
# resetrsrc -s 'Name == "tdwb-rs"' IBM.Application
# mkrel -p Collocated -S IBM.Application:tdwb-rs -G    IBM.Application:mdm-rs
# mkrel -p DependsOn -S IBM.ResourceGroup:tdwb-rg -G IBM.ResourceGroup:mdm-rg
```

2. Start the resource group for the Dynamic Workload Broker Component, as shown in Example 26-68.

Example 26-68 Starting resource group for the Dynamic Workload Broker Component

```
# chrg –o online tdwb-rg
```

The creation of the Tivoli System Automation for Multiplatforms objects for the Dynamic Workload Broker Component results in the output of the **lssam** command., as shown in Example 26-69.

Example 26-69 lssam: Dynamic Workload Broker

```
Online IBM.ResourceGroup:tdwb-rg Nominal=Online
        |- Online IBM.Application:tdwb-rs
                |- Online IBM.Application:tdwb-rs:twspoc1
                '- Offline IBM.Application:tdwb-rs:twspoc2
        '- Online IBM.ServiceIP:tdwb_192_168_1_205-rs
                |- Online IBM.ServiceIP:tdwb_192_168_1_205-rs:twspoc1
                '- Offline IBM.ServiceIP:tdwb_192_168_1_205-rs:twspoc2
```

Figure 26-14 illustrates the dependency between the Dynamic Workload Broker Component and the master domain manager.

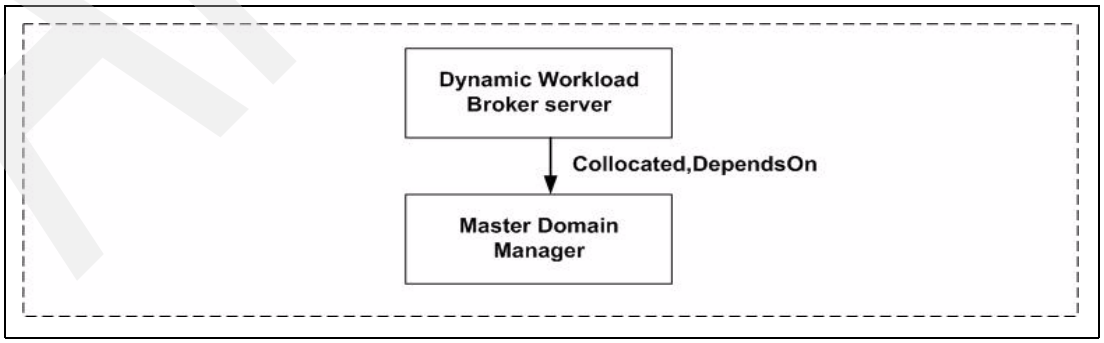


Figure 26-14 Dynamic Workload Broker Component dependency

Finally, update the Server Connection for the Dynamic Workload Broker Component in the Tivoli Dynamic Workload Console, as shown in Figure 26-15 on page 790.



Figure 26-15 Dynamic Workload Broker Component connection in Tivoli Dynamic Workload Console

26.8 Testing the policies

This section explains how the policies react to specific scenarios that can happen, planned or unplanned, on the master domain manager.

26.8.1 Plan extension

Both the JnextPlan script and the FINAL stream have been modified to inform Tivoli System Automation for Multiplatforms that a plan extension is occurring. As the entire network is restarted, this operation is interpreted as normal and not a failure. Therefore, during the switching of the plan, the resource groups for the master domain manager and backup master domain manager are locked, as shown in the **lssam** output of Example 26-70.

Example 26-70 lssam: JnextPlan, 1

```
Stuck online IBM.ResourceGroup:fta_twspoc1-rg Request=Lock Nominal=Online
    |- Failed offline IBM.Application:mailman_twspoc1-rs:twspoc1
       '- Online IBM.Application:netman_twspoc1-rs:twspoc1
Stuck online IBM.ResourceGroup:fta_twspoc2-rg Request=Lock Nominal=Online
       |- Failed offline IBM.Application:mailman_twspoc2-rs:twspoc2
       '- Online IBM.Application:netman_twspoc2-rs:twspoc2
Online IBM.ResourceGroup:mdm-rg Request=Lock Nominal=Online
       |- Online IBM.Application:mdm-rs
                |- Online IBM.Application:mdm-rs:twspoc1
                '- Offline IBM.Application:mdm-rs:twspoc2
       '- Online IBM.ServiceIP:mdm_192_168_1_203-rs
                |- Online IBM.ServiceIP:mdm_192_168_1_203-rs:twspoc1
                '- Offline IBM.ServiceIP:mdm_192_168_1_203-rs:twspoc2
```

If the Tivoli Workload Scheduler process tree (batchman, mailman and jobman) is not active on the master domain manager 60 seconds after SwitchPlan, Tivoli System Automation for Multiplatforms intervenes and performs a failover to the backup master domain manager as shown in Example 26-71 on page 791.

Example 26-71 Issam: JnextPlan, 2

```
Failed offline IBM.ResourceGroup:fta_twspoc1-rg Request=Lock Nominal=Online
        |- Failed offline IBM.Application:mailman_twspoc1-rs:twspoc1
        '- Offline IBM.Application:netman_twspoc1-rs:twspoc1
Online IBM.ResourceGroup:fta_twspoc2-rg Request=Lock Nominal=Online
        |- Online IBM.Application:mailman_twspoc2-rs:twspoc2
        '- Online IBM.Application:netman_twspoc2-rs:twspoc2
Pending online IBM.ResourceGroup:mdm-rg Request=Lock Nominal=Online
        |- Pending online IBM.Application:mdm-rs
                |- Offline IBM.Application:mdm-rs:twspoc1
                '- Pending online IBM.Application:mdm-rs:twspoc2
        '- Online IBM.ServiceIP:mdm_192_168_1_203-rs
                |- Offline IBM.ServiceIP:mdm_192_168_1_203-rs:twspoc1
                '- Online IBM.ServiceIP:mdm_192_168_1_203-rs:twspoc2
```

26.8.2 Corrupt Symphony / message files

If the Symphony file or the message files become corrupt on the master domain manager (most likely because of a full file system), batchman, mailman and jobman terminate unexpectedly. As a result, Tivoli System Automation for Multiplatforms performs a failover to the backup master domain manager.

Example 26-72 illustrates the commands that were run on the master domain manager to simulate this situation.

Example 26-72 Filling the Tivoli Workload Scheduler file system

```
# dd if=/dev/zero of=/TWA/TWS/bigfile bs=512 count=4194304
dd: 0511-053 The write failed.
: There is not enough space in the file system.
2526355+0 records in.
2526352+0 records out.
twspoc1:[/usr/sbin/rsct/sapolicies/tws]
# df -m /TWA/TWS
Filesystem    MB blocks      Free %Used    Iused %Iused Mounted on
/dev/fslv06     2048.00      0.00  100%     5402    96% /TWA/TWS
```

The policies react to the situation as shown in Example 26-73.

Example 26-73 Issam: Corrupt Symphony/message files

```
Failed offline IBM.ResourceGroup:fta_twspoc1-rg Request=Lock Nominal=Online
        |- Failed offline IBM.Application:mailman_twspoc1-rs:twspoc1
        '- Offline IBM.Application:netman_twspoc1-rs:twspoc1
Online IBM.ResourceGroup:fta_twspoc2-rg Request=Lock Nominal=Online
        |- Online IBM.Application:mailman_twspoc2-rs:twspoc2
        '- Online IBM.Application:netman_twspoc2-rs:twspoc2
Pending online IBM.ResourceGroup:mdm-rg Request=Lock Nominal=Online
        |- Pending online IBM.Application:mdm-rs
                |- Offline IBM.Application:mdm-rs:twspoc1
                '- Pending online IBM.Application:mdm-rs:twspoc2
        '- Online IBM.ServiceIP:mdm_192_168_1_203-rs
                |- Offline IBM.ServiceIP:mdm_192_168_1_203-rs:twspoc1
                '- Online IBM.ServiceIP:mdm_192_168_1_203-rs:twspoc2
```

Archived

26.8.3 The embedded WebSphere Application Server crash situation

Because of the severe nature of the crash situation, Tivoli System Automation for Multiplatforms does not attempt to restart the embedded WebSphere Application Server instance on the master domain manager. Instead, the following actions are performed:

- A failover of the engine to the backup master domain manager is triggered.
- The Event Processor is switched to the embedded WebSphere Application Server on the backup master domain manager.
- The Dynamic Workload Broker Component is restarted on the backup master domain manager.

Example 26-74 shows the **lssam** command output after an embedded WebSphere Application Server crash on the master.

Example 26-74 lssam: eWAS crash

```
Failed offline IBM.ResourceGroup:eWAS_twspoc1-rg Nominal=Online
        '- Failed offline IBM.Application:eWAS_twspoc1-rs:twspoc1
Online IBM.ResourceGroup:eWAS_twspoc2-rg Nominal=Online
        '- Online IBM.Application:eWAS_twspoc2-rs:twspoc2
Online IBM.ResourceGroup:evtproc-rg Nominal=Online
        '- Online IBM.Application:evtproc-rs
                |- Failed offline IBM.Application:evtproc-rs:twspoc1
                '- Online IBM.Application:evtproc-rs:twspoc2
Online IBM.ResourceGroup:tdwb-rg Nominal=Online
        |- Online IBM.Application:tdwb-rs
                |- Failed offline IBM.Application:tdwb-rs:twspoc1
                '- Online IBM.Application:tdwb-rs:twspoc2
        '- Online IBM.ServiceIP:tdwb_192_168_1_205-rs
                |- Offline IBM.ServiceIP:tdwb_192_168_1_205-rs:twspoc1
                '- Online IBM.ServiceIP:tdwb_192_168_1_205-rs:twspoc2
Online IBM.ResourceGroup:mdm-rg Nominal=Online
        |- Online IBM.Application:mdm-rs
                |- Offline IBM.Application:mdm-rs:twspoc1
                '- Online IBM.Application:mdm-rs:twspoc2
'- Online IBM.ServiceIP:mdm_192_168_1_203-rs
                |- Offline IBM.ServiceIP:mdm_192_168_1_203-rs:twspoc1
                '- Online IBM.ServiceIP:mdm_192_168_1_203-rs:twspoc2
```

26.8.4 Node failure

If a hardware failure of the master domain manager occurs, Tivoli System Automation for Multiplatforms performs a failover of the following resource groups to the backup master domain manager:

- “DB2 Primary” resource group: db2_db2tws_db2tws_TWS-rg
- “Event Processor” resource group (if it was running on the master): evtproc-rg
- “Master Domain Manager” resource group: mdm-rg
- “Dynamic Workload Broker Component server” resource group: tdwb-rg
- “Tivoli Dynamic Workload Console” resource group: tdwc-rg

27

Implementing high availability for Tivoli Workload Scheduler dynamic workload

This chapter describes implementing high availability for Tivoli Workload Scheduler dynamic agent workload using Tivoli System Automation for Multiplatforms. The design of the automation policies is based on concepts described in Chapter 24, “Tivoli Workload Scheduler: Architecture and concepts” on page 719 and Chapter 25, “Tivoli System Automation for Multiplatforms: Concepts” on page 745.

The policies used in this chapter can be downloaded from the ITSO FTP site. For download instructions, see Appendix A, “Additional material” on page 803.

This chapter contains the following topics:

- 27.1, “Overview” on page 794
- 27.2, “Setup” on page 794
- 27.3, “Benefits” on page 795
- 27.4, “Installing Tivoli Workload Scheduler” on page 796
- 27.5, “Installing and configuring Tivoli System Automation for Multiplatforms” on page 796
- 27.6, “Virtualizing dynamic workload” on page 797
- 27.7, “Testing the policies” on page 801

© Copyright IBM Corp. 2011. All rights reserved.

27.1 Overview

This chapter concentrates on implementing high availability for dynamic agent workload. The goal is to reduce the amount of time that is needed to recover from an outage by virtualizing the workload. As a result, the availability of job execution can improve significantly. Consider the scenario illustrated in Figure 27-1.

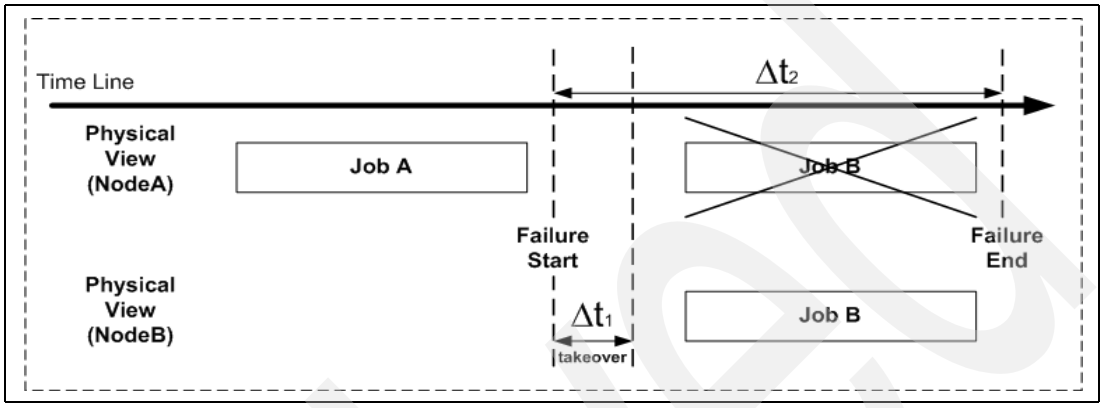


Figure 27-1 Job execution: physical view

Under normal circumstances, both Job A and Job B run on NodeA. By using virtualizing techniques, Job B can be switched to NodeB in case of a failure. As a result, the time to recover logically from the physical outage of NodeA is reduced, as shown in Figure 27-2.

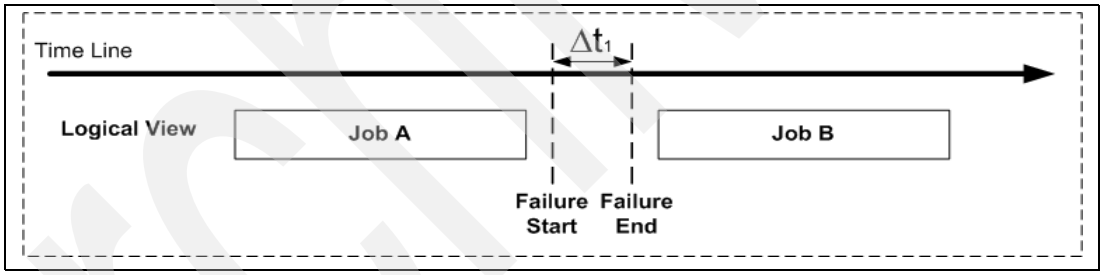


Figure 27-2 Job execution: logical view

Important: Neither Tivoli Workload Scheduler nor Tivoli System Automation for Multiplatforms has built-in features to automatically recover a job if the failure occurred during its execution.

27.2 Setup

The setup used in this chapter consists of two AIX 6.1 nodes fta1 and fta2, each located in a separate computer room. As illustrated in Figure 27-3 on page 795, 2 Tivoli Workload Automation instances are installed:

- Tivoli Workload Automation instance A on fta1 will be used for dynamic scheduling only.
- Tivoli Workload Automation instance B on fta2 will be used for dynamic scheduling only.

Archived

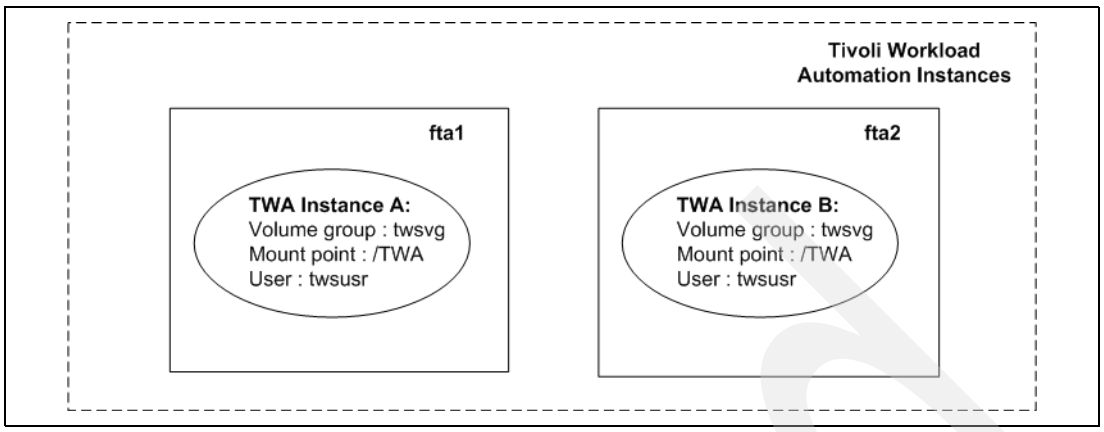


Figure 27-3 Tivoli Workload Automation instances

The following software is used:

- AIX 6.1 TL 4 SP 4
- Tivoli System Automation for Multiplatforms 3.2
- Tivoli Workload Scheduler 8.5.1 FP 1

27.3 Benefits

When making a dynamic job highly available, the use of Tivoli System Automation for Multiplatforms can be beneficial in the following scenarios:

- If the application that depends on the dynamic job is already managed by Tivoli System Automation for Multiplatforms, the failover takes place at the same time and as one logical unit, honoring the managed relations between all components.
- The amount of time that is needed for the Dynamic Workload Broker Component to become aware of an unavailability of a computer is global for all computers. The use of Tivoli System Automation for Multiplatforms can overcome this limitation by dynamically altering a Dynamic Workload Broker Component resource. See 27.6, “Virtualizing dynamic workload” on page 797 for more details about the implementation.
- Unlike Tivoli System Automation for Multiplatforms, Dynamic Workload Broker Component is unable to detect split-brain situations, as illustrated in Example 27-4.

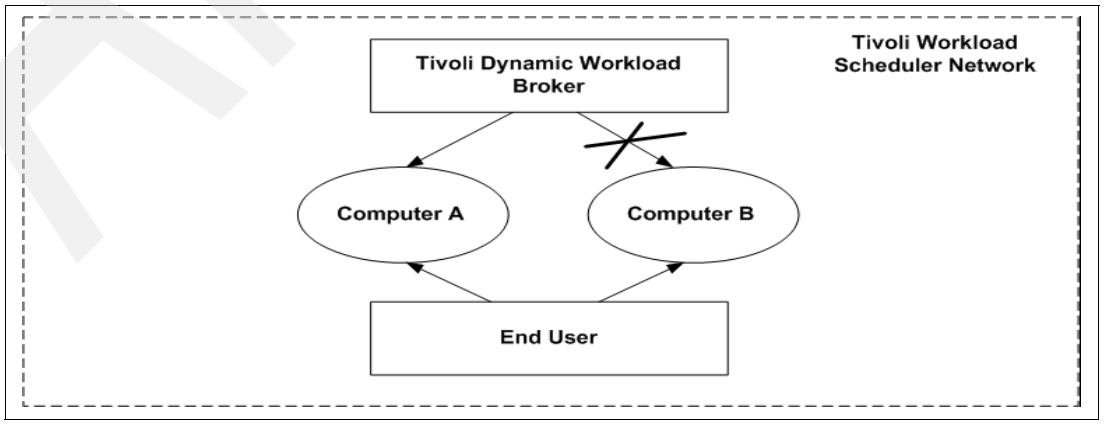


Figure 27-4 Split-brain

Archived

27.4 Installing Tivoli Workload Scheduler

This section focuses on the installation of the separate Tivoli Workload Scheduler agents by using command-line tools. Perform the following steps on fta1 and fta2 as root:

1. Install and upgrade the Tivoli Workload Scheduler agent, as shown in Example 27-1.

Example 27-1 Installing and upgrading the Tivoli Workload Scheduler agent

```
# ./twsinst -new -uname twsusr –hostname FQDN_of_FTA -addjruntime true
-tdwbhostname virtual_IP_of_TDWB -tdwbport 31116
# ./twspatch -install -uname twsusr
```

Note: To be able to use the command-line interface for the Dynamic Workload Broker Component, the agent must have been installed with the runtime environment for Java jobs (-addjruntime true).

2. Stop the Tivoli Workload Scheduler agent, by running the commands (shown in Example 27-2) on fta1 and fta2 as twsusr.

Example 27-2 Shutting down Tivoli Workload Scheduler agent

```
$ ShutDownLwa
$ conman shut
```

27.5 Installing and configuring Tivoli System Automation for Multiplatforms

This section describes the steps required to install and configure the Tivoli System Automation for Multiplatforms cluster.

Perform the following steps on fta1 and fta2 as root:

1. Install the Tivoli System Automation for Multiplatforms software and prepare node security, as shown in Example 27-3.

Example 27-3 Tivoli System Automation for Multiplatforms installation

```
# ./prereqSAM
prereqSAM: All prerequisites for the ITSAMP installation are met on operating
system:  AIX 6100-04
# ./installSAM
SAM is currently not installed.
installSAM: All packages were installed successfully.
# preprpnode fta1 fta2
```

2. Add `CT_MANAGEMENT_SCOPE=2` variable to `/etc/environment` and start a new shell. This environment variable determines the scope of the cluster. If set to 2, all nodes in the cluster are in scope.

Archive

On fta1, perform the following tasks as root:

1. Create and start the Tivoli System Automation for Multiplatforms cluster, as illustrated in Example 27-4.

Example 27-4 Creating and starting the Tivoli System Automation for Multiplatforms cluster

```
# mkrpdomain FTA fta1 fta2
# startrpdomain FTA
```

2. Verify the state of the cluster, as illustrated in Example 27-5.

Example 27-5 Verifying the state of the Tivoli System Automation for Multiplatforms cluster

```
# lsrpdomain
Name OpState RSCTActiveVersion MixedVersions TSPort GSPort
FTA Online  2.5.5.1           No            12347  12348
# lsrpnode
Name    OpState RSCTVersion
fta1 Online  2.5.5.1
fta2 Online  2.5.5.1
```

3. Create a network tie breaker for the cluster, as shown in Example 27-6.

Example 27-6 Creating network tie breaker for the cluster

```
# mkequ VLAN1 IBM.NetworkInterface:en0:fta1,en0:fta2
# mkrsrc IBM.TieBreaker Type="EXEC" Name="NETWORKTB"
DeviceInfo='PATHNAME=/usr/sbin/rsct/bin/samtb_net Address=192.168.1.1 Log=1'
PostReserveWaitTime=30
# chrsrc -c IBM.PeerNode OpQuorumTieBreaker="NETWORKTB"
```

27.6 Virtualizing dynamic workload

This section describes the steps to virtualize a dynamic job and to make it highly available using Tivoli System Automation for Multiplatforms. Assume that the following job requirements must be fulfilled:

- Dynamic job XYZ must run on fta1, if the computer is available.
- Only if fta1 is not available, dynamic job XYZ must run on computer fta2.

To implement the requirements, a Dynamic Workload Broker Component logical resource is needed. By default, fta1 is the only workstation in the *Computer List* that is associated with the logical resource. If fta1 becomes unavailable, Tivoli System Automation for Multiplatforms dynamically updates the *Computer List* to include fta2 and remove fta1.

As twsusr on fta1 and fta2, modify the `CLIConfig.properties` file in the `/TWA/TWS/TDWB_CLI/config` folder to include the virtual IP address of the Dynamic Workload Broker Component server. See Example 27-7.

Example 27-7 Updating CLI connection properties

```
ITDWBServerHost = 192.168.1.205
ITDWBServerPort = 31115
ITDWBServerSecurePort = 31116
```

Currently, the Dynamic Workload Broker Component CLI does not support non-blocking TCP sockets. Therefore, as root on fta1 and fta2, raise the general OS TCP connection timeout interval to 300 seconds for the automation policies to react properly during a failover of the server that hosts the Dynamic Workload Broker Component. See Example 27-8.

Example 27-8 Raising TCP connection timeout interval

```
# no -p -o tcp_keepinit=600
Setting tcp_keepinit to 600
Setting tcp_keepinit to 600 in nextboot file
```

Finally, perform the following commands on fta1 as twsusr:

1. Create a Dynamic Workload Broker Component logical resource, as shown in Example 27-9.

   Example 27-9 Creating Dynamic Workload Broker Component logical resource

   ```
   $ /TWA/TWS/TDWB_CLI/bin/resource.sh -usr twsusr -pwd xxx -create
   -logical Redbook -type HA -quantity 1
   Setting CLI environment variables....
   AWKCLI153I Logical Resource "Redbook" created
   ```

2. As shown in Example 27-10, create a generic JSDL on twspoc1 that defines the Dynamic Workload Broker Component job.

   Example 27-10 Creating generic JSDL job definition

   ```
   $ cat > /tmp/jsdl.txt << 'EOF'
   <?xml version="1.0" encoding="UTF-8"?>
   <jsdl:jobDefinition
   xmlns:jsdl="http://www.ibm.com/xmlns/prod/scheduling/1.0/jsdl"
   xmlns:jsdle="http://www.ibm.com/xmlns/prod/scheduling/1.0/jsdle" name="HA">
     <jsdl:application name="executable">
       <jsdle:executable interactive="false">
         <jsdle:script>${tws.job.taskstring}</jsdle:script>
         <jsdle:credential>
           <jsdle:userName>${tws.job.logon}</jsdle:userName>
         </jsdle:credential>
       </jsdle:executable>
     </jsdl:application>
     <jsdl:resources>
       <jsdl:logicalResource name="Redbook" quantity="1" subType="HA"/>
     </jsdl:resources>
   </jsdl:jobDefinition>
   EOF
   $ /TWA/TDWB/bin/jobstore.sh -usr twsusr -pwd xxx -create /tmp/jsdl.txt
   Setting CLI environment variables....
   AWKCLI111I Calling the Job Dispatcher to save the job definition.
   AWKCLI114I The job definition was successfully saved in the Job Repository.
   ```

3. Create the associated Tivoli Workload Scheduler pool workstation using composer, as illustrated in Example 27-11 on page 799.

Archived

Example 27-11 Creating Tivoli Workload Scheduler pool workstation

```
CPUNAME REDBOOKPOOL
  OS UNIX
  NODE NotApplicable TCPADDR 31111
  FOR MAESTRO HOST BROKER ACCESS "/jsdl/HA"
    TYPE X-AGENT
    AUTOLINK OFF
    BEHINDFIREWALL OFF
    FULLSTATUS OFF
END
```

4. Run **JnextPlan -for 0000** command to make the pool workstation available in the plan.
5. Raise the CPU limit of the pool workstation.

Next, the Tivoli System Automation for Multiplatforms resources, resource groups, relationships, and equivalencies for the Dynamic Workload Broker Component clients are created.

Perform the following steps on fta1 as root:

1. Create the resources and resource groups for the Dynamic Workload Broker Component clients, as shown in Example 27-12.

Example 27-12 Creating resources for the Dynamic Workload Broker Component clients

```
# mkrsrc -f ./jobmanager_fta1-rs.def IBM.Application
# mkrsrc -f ./jobmanager_fta2-rs.def IBM.Application
# mkrg -n fta1 jobmanager_fta1-rg
# mkrg -n fta2 jobmanager_fta2-rg
# addrgmbr -g jobmanager_fta1-rg IBM.Application:jobmanager_fta1-rs:fta1
# addrgmbr -g jobmanager_fta2-rg IBM.Application:jobmanager_fta2-rs:fta2
# resetrsrc -s 'Name == "jobmanager_fta1-rs"' IBM.Application
# resetrsrc -s 'Name == "jobmanager_fta2-rs"' IBM.Application
```

2. Start the resource groups for the Dynamic Workload Broker Component clients, as illustrated in Example 27-13.

Example 27-13 Starting resource groups for Dynamic Workload Broker Component clients

```
# chrg -o online jobmanager_fta1-rg
# chrg -o online jobmanager_fta2-rg
```

3. Create the required shadow resources and shadow equivalency for the Dynamic Workload Broker Component clients, as shown in Example 27-14.

Example 27-14 Creating shadow resources for Dynamic Workload Broker Component clients

```
# mkrsrc -f ./jobmanager_fta1_shadow-rs.def IBM.Application
# mkrsrc -f ./jobmanager_fta2_shadow-rs.def IBM.Application
# mkequ -p A,NoControl jobmanager_shadow-equ
IBM.Application:jobmanager_fta1_shadow-rs:fta1,jobmanager_fta2_shadow-rs:fta2
```

Archived

4. Create a floating resource that controls the Dynamic Workload Broker Component logical resource, as illustrated in Example 27-15.

Example 27-15 Creating resource for Dynamic Workload Broker Component logical resource

```
# mkrg logicalresource-rg
# mkrsrc -f ./logicalresource-rs.def IBM.Application
# addrgmbr -g logicalresource-rg IBM.Application:logicalresource-rs
# mkrel -p DependsOn -S IBM.Application:logicalresource-rs -G
IBM.Equivalency:jobmanager_shadow-equ
```

5. Start the resource group for the Dynamic Workload Broker Component logical resource, as shown in Example 27-16.

Example 27-16 Starting the resource group for the Dynamic Workload Broker Component logical resource

```
# chrg -o online logicalresource-rg
```

The creation of the Tivoli System Automation for Multiplatforms objects for the Dynamic Workload Broker Component clients results in the output of the **lssam** command, as shown in Example 27-17.

Example 27-17 lssam: Dynamic Workload Broker Component clients

```
Online IBM.ResourceGroup:jobmanager_fta1-rg Nominal=Online
        '- Online IBM.Application:jobmanager_fta1-rs:fta1
Online IBM.ResourceGroup:jobmanager_fta2-rg Nominal=Online
        '- Online IBM.Application:jobmanager_fta2-rs:fta2
Online IBM.ResourceGroup:logicalresource-rg Nominal=Online
        '- Online IBM.Application:logicalresource-rs
                |- Online IBM.Application:logicalresource:fta1
                '- Offline IBM.Application:logicalresource:fta2
```

Figure 27-5 illustrates the dependency.

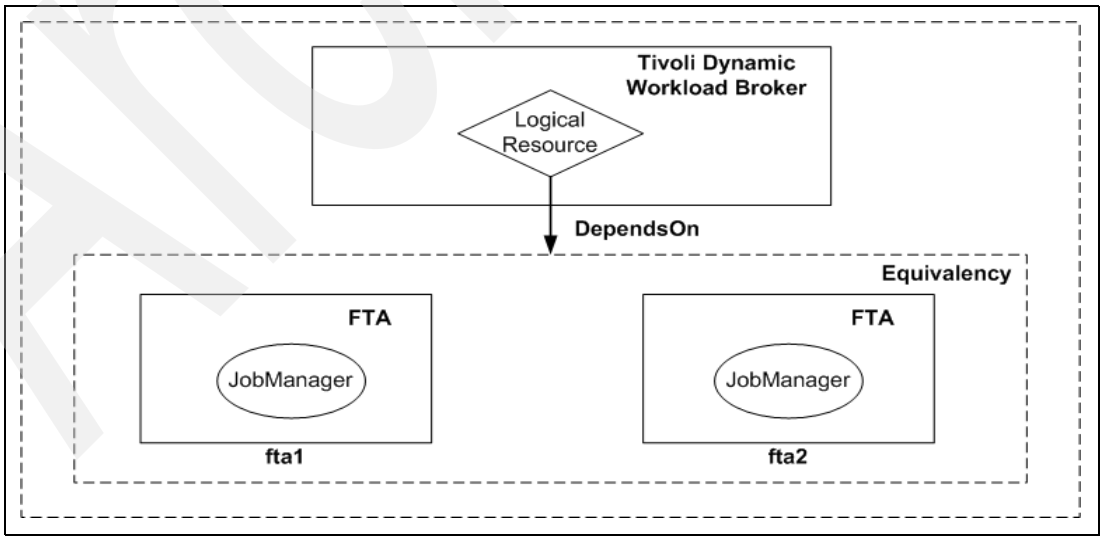


Figure 27-5 Logical resource dependency

Important: The use of Tivoli System Automation for Multiplatforms is not mandatory for making a dynamic job highly available on its own. The same can be accomplished by creating a JSDL that optimizes on Logical Resource quantity. Both computers can be put in a *Logical Resource Group*, in which the primary computer is given a larger number of units. As a result, the primary always gets the job unless it is unavailable.

The use of Tivoli System Automation for Multiplatforms can be more important or beneficial in the following cases:

- The application that is dependent of the dynamic job is already managed by Tivoli System Automation for Multiplatforms. As a result, the failover takes place at the same time and as one logical unit.
- The amount of time that is needed for the Dynamic Workload Broker Component to be aware of an unavailability of a computer, is global for all computers.

27.7 Testing the policies

To test the policies, use the Tivoli Workload Scheduler job. See Example 27-18.

Example 27-18 HA job definition

```
REDBOOKPOOL#HAJOB
 DOCOMMAND "hostname"
 STREAMLOGON root
 TASKTYPE UNIX
 RECOVERY STOP
```

As described previously, the job must run on computer fta1, if it is available. Only if fta1 is not available, run the job on the fta2 computer. Example 27-19 shows the job run during normal operations (both fta1 and fta2 are available).

Example 27-19 HA job run

```
$ conman 'sbj REDBOOKPOOL#HAJOB;alias=TEST1' 2>/dev/null
$ conman 'sj REDBOOKPOOL#JOBS.TEST1;stdlist' 2>/dev/null
%sj REDBOOKPOOL#JOBS.TEST1;stdlist
...
fta1
...
```

Next, a failure of fta1 is simulated by shutting down the host by force. The resulting reaction of the automation policies is shown in Example 27-20.

Example 27-20 lssam: failure of fta1

```
Failed offline IBM.ResourceGroup:jobmanager_fta1-rg Nominal=Online
        '- Failed offline IBM.Application:jobmanager_fta1-rs:fta1
Online IBM.ResourceGroup:jobmanager_fta2-rg Nominal=Online
        '- Online IBM.Application:jobmanager_fta2-rs:fta2
Online IBM.ResourceGroup:logicalresource-rg Nominal=Online
        '- Online IBM.Application:logicalresource-rs
                |- Failed offline IBM.Application:logicalresource-rs:fta1
                '- Online IBM.Application:logicalresource-rs:fta2
```

Finally, a new submission of the job is performed. Example 27-21 shows the job run.

Example 27-21 HA job run

```
$ conman 'sbj REDBOOKPOOL#HAJOB;alias=TEST1' 2>/dev/null
$ conman 'sj REDBOOKPOOL#JOBS.TEST2;stdlist' 2>/dev/null
%sj REDBOOKPOOL#JOBS.TEST2;stdlist
. . .
fta2
. . .
```

Archived

A

Additional material

This book refers to additional material that can be downloaded from the Internet as described here.

Locating the web material

The web material that is associated with this book is available in softcopy on the Internet from the IBM Redbooks web server. Point your web browser at:

ftp://www.redbooks.ibm.com/redbooks/SG247878

Alternatively, you can go to the IBM Redbooks website at:

ibm.com/redbooks

Select the **Additional materials** and open the directory that corresponds with the IBM Redbooks form number, SG247878.

Archived

© Copyright IBM Corp. 2011. All rights reserved.

Using the web material

The additional web material that accompanies this book includes the files in Table 27-1.

Table 27-1 Files and chapter where used

File name	Description
`Cognos_Integration_Package_1.zip`	Additional material for Chapter 8, “Tivoli reporting integration: IBM Tivoli Monitoring and TADDM” on page 199
`Cognos_Integration_Package_2.zip`	Additional material for Chapter 9, “Tivoli reporting integration: IBM Tivoli Monitoring and IBM Tivoli Service Request Manager” on page 245
`TPM_TIM_Integration_Package.zip`	Additional material for Chapter 15, “Tivoli Provisioning Manager and Tivoli Identity Manager integration” on page 405
`TWSMasterHA-policies.tar`	Tivoli System Automation for Multiplatforms policies used in Chapter 26, “Implementing high availability for the Tivoli Workload Scheduler master domain manager” on page 763
`TWSVirtualAgentHA-policies.tar`	Tivoli System Automation for Multiplatforms policies used in Chapter 27, “Implementing high availability for Tivoli Workload Scheduler dynamic workload” on page 793

System requirements for downloading the web material

Use the following system configuration:

Hard disk space: 30 MB minimum
Operating System: Windows or Linux

How to use the web material

Create a subdirectory (folder) on your workstation, and extract the contents of the web material `.zip` file into this folder.

Archive

Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this book.

IBM Redbooks

For information about ordering these publications, see “How to get Redbooks” on page 806. Note that some of the documents referenced here may be available in softcopy only.

- *Integrating Tivoli Products*, SG24-7757
- *Automated Storage Provisioning for Windows using TotalStorage Productivity Center and Tivoli Provisioning Manager*, REDP-4242
- *Exploring Storage Management Efficiencies and Provisioning - Understanding IBM TotalStorage Productivity Center and IBM TotalStorage Productivity Center with Advanced Provisioning*, SG24-6373
- *An Introduction to Storage Provisioning with Tivoli Provisioning Manager and TotalStorage Productivity Center*, REDP-3900
- *IBM Tivoli Provisioning Manager V7.1.1: Deployment and IBM Service Management Integration Guide*, SG24-7773

Online resources

These Web sites are also relevant as further information sources:

- Tivoli Documentation Central:

  https://www.ibm.com/developerworks/wikis/display/tivolidoccentral/Home
- Important Technote about Tivoli Monitoring for Energy Management AEM-plug-in:

  http://www.ibm.com/support/docview.wss?uid=swg21432288
- Integrated Service Management Library:

  http://www.ibm.com/software/brandcatalog/ismlibrary/
- Tivoli Business Service Manager:

  http://publib.boulder.ibm.com/infocenter/tivihelp/v3r1/index.jsp?topic=/com.ibm.tivoli.itbsm.doc/welcome.htm
- IBM Tivoli Monitoring for Energy Management:

  http://publib.boulder.ibm.com/infocenter/tivihelp/v15r1/index.jsp?topic=/com.ibm.itmem.doc_621if1/EM_welcome.htm
- IBM Systems Director:

  http://www.ibm.com/systems/software/director/resources.html
- Tivoli Application Dependency Discovery Manager:

  http://publib.boulder.ibm.com/infocenter/tivihelp/v10r1/index.jsp?topic=/com.ibm.taddm.doc_7.2/welcome_page/welcome.html

© Copyright IBM Corp. 2011. All rights reserved.

- Tivoli Usage and Accounting Manager:

  http://publib.boulder.ibm.com/infocenter/tivihelp/v3r1/index.jsp?topic=/com.ibm.ituam.doc_7.1.2/welcome.htm

- BIRT Reporting Designer v2.2.1 for Tivoli Common Reporting:

  http://www.ibm.com/software/brandcatalog/ismlibrary/details?catalog.label=1TW10OT02

- IBM Maximo Asset Management for Energy Optimization information:
  - http://www.ibm.com/software/tivoli/products/maximo-asset-mgmt-energy-optimize/requirements.html?S_CMP=rnav
  - https://www.ibm.com/developerworks/wikis/display/tivolidoccentral/IBM+Maximo+Asset+Management+for+Energy+Optimization

- IBM Fix Central:

  http://www.ibm.com/support/fixcentral

- IBM Tivoli Monitoring 6.2.2 Fix Pack 2 Agent Reports package:

  http://www.ibm.com/support/docview.wss?uid=swg24025630

- The Open Group ARM information:

  http://www.opengroup.org/management/arm/

- How to integrate IBM Rational Asset Manager with your IBM Tivoli database white paper:

  http://www.ibm.com/developerworks/rational/library/08/1223_sanaie/index.html

- IBM Tivoli Monitoring Installation and Setup Guide:

  http://publib.boulder.ibm.com/infocenter/tivihelp/v15r1/topic/com.ibm.itm.doc_6.2.2/itm_install.htm

- Installation and configuration of Tivoli Netcool/OMNIbus 7.3:

  http://publib.boulder.ibm.com/infocenter/tivihelp/v8r1/topic/com.ibm.netcool_OMNIbus.doc_7.3.0/omnibus/wip/install/concept/omn_ins_installingupgradinguninstalling.html

- Installation and configuration of Tivoli Netcool/OMNIbus Web GUI 7.3:

  http://publib.boulder.ibm.com/infocenter/tivihelp/v8r1/topic/com.ibm.netcool_OMNIbus.doc_7.3.0/webtop/wip/task/web_ins_installing.html

- Installation and configuration of Tivoli Netcool/OMNIbus EIF Probe:

  http://publib.boulder.ibm.com/infocenter/tivihelp/v8r1/topic/com.ibm.netcool_OMNIbus.doc/probes/tivoli_eif/common/topicref/pro_install_post73.html

- IBM Tivoli Netcool/Impact developer works wiki to the online resources:

  http://www.ibm.com/developerworks/wikis/display/tivolibsm/Tivoli+Netcool+Impact

How to get Redbooks

You can search for, view, or download Redbooks, Redpapers, Technotes, draft publications and Additional materials, as well as order hardcopy Redbooks publications, at this Web site:

ibm.com/redbooks

Archived

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services

Archived

Archived

Index

Symbols

.baroc files 373
$CANDLEHOME directory 644
$COLLATION_HOME directory 644
$OMNIHOME/gates/tsrm/tsrm.map file, configuring 449

A

access list
  configuring 653
  creating 653
Action Helper in Event Processor 722
-action parameter, CMS Registry Loader 708
actions
  automated remediation actions 326
  creating 441
  list of 442
  triggering 156
activating an escalation 499
Active Energy Manager 22
  configuring 26
  installing 25
  installing the plug-in 27
  selecting a query 52
  viewing reports 54
Active Event List 474
Actual CI Table (ACTCI) 493
Affinity relationship 751
agent management integration 5
Air Handle Unit
  APOGEE device 23
  data collection 23
alerts.status table 418
  adding additional fields 445
alerts.status, update 449
Allocation Repository
  Dynamic Workload Broker Component 723
APC InfraStruXure 23
APIs
  Java 409
  Tivoli Identity Manager 409
Application Designer Control Palette 438
Application Management Configuration Editor 358
Application Management console 191
application signature
  options 693
  VACTCINUM condition 694
-applid parameter, CMS Registry Loader 709
ARM metrics configuring 368
aspects of integration 5
asset class, top-level-IT 517
asset data
  authorized 515
  deployed 515
  loading 110
  reports 122
Asset Management for Energy Optimization 105, 110
asset power consumption 122
assets
  controlling 480
  definition of 478
  implementing 480
  life cycle 478
  linking to CIs 482
  linking to deployed assets 483
  searching for approved assets 561
  synchronizing 591
  usage 478
assets and CIs
  differences 478
  implementing 510
  integration 480
Assigned EIF Receivers 372
ASYNC synchronization mode 735
authorized asset data 515
Authorized Classification in CCMDB 667
automated remediation actions 326
automatic client reroute (ACR)
  configuring 772
  DB2 733–734

B

Baseline Power Metrics, viewing reports 54
bidirectional integration 133
BIRT reports
  asset power consumption 122
  exporting 88
  importing 84
  testing 88
BladeCenter
  data collection 22
  discovering with TMS DLA 71
blades, discovering with TMS DLA 71
branches, Cross product model 238
BrokerWorkstation.properties file 788
-bundle parameter, CMS Registry Loader 709
Business Applications, discovering 402
Business Service Management dashboards 65

C

CAM_Performance_Trend chart 187
canvasDynamicSubMenuActions.xml file 688
canvasOpenURLActions.xml file 689
Carbon Footprint 20
Catalog, requesting 334, 343, 349
CCMDB
  activating CI types 678
  controlling assets 480
  creating a connection 560

© Copyright IBM Corp. 2011. All rights reserved.

creating Request for Change 418
creating request for change 416
creating the OMP 566
creating web services 458
defining an endpoint 584
importing data 510
installing the integration module 551
integrating with Rational Asset Manager 549
launching from Tivoli Business Service Manager 687
populating data 425
server, backing up the maxdb71 database 551
setting up an integration module 583
setting up web services 580
test Connection 560
using to create an enrichment policy 456
CCMDB and TADDM
deployment considerations 705
download URL 667
integration 695
Change and Configuration Management Database
See CCMDB
Change Window Calendars, creating 425
ChangeClassification 441
CHANGECLASSIFICATION attribute 439
Changeclassification attribute 429
changePassword extension, configuring 612
chart type, selecting 80
Charting portlet, selecting 78
chiller
data collection 23
description 20
CI
activating in CCMDB 678
activating types 505
column expressions 509
configuring adapter mapping 673
controlling 480
creating
creating a type schema 499
creating new type mappings 502, 506
description 478
discovering 400
exporting from TADDM to CCMDB 668
implementing 481
importing 499
importing from TADDM 678
life cycle 479
linking to assets 482, 494
numbers and names 428
relationships with deployed assets 492
usage 478
CI actual adapter 679
mapping execution 679
CI Applications CMS Integration PSI Package, install 705
CI types 505
activating 573, 678
creating mappings 568, 574
executing the type mapping 571
importing 706–707
mapping execution for an adapter 677
running the type mapping 577
selecting a connection method 567
setting up an adapter 566, 574
testing the connection 569
verifying that data has been imported 579
cinfo command 648
CISPEC data type 456
ClassificationID 441
CLASSIFICATIONID attribute 439
Classificationid attribute 435
CLI
tools 783
updating connections 797
view status 648
CLIConfig.properties file 797
clusters 755
cmdbdiscovery.sh command 163
CMS Registry Loader
-action parameter 708
functions 708
remove launch entries 710
syntax 708
validating launch entries 710
CMS tables creation 705
Cognos BI Modeling tool - Framework Manager 229
Cognos BI Reporting engine
defining data sources 222
collocated relationship 751
column expressions, CIs 509
commandLine.bat command 677
commandLine.bat file 679
commandline.bat/sh command 571
commands
cinfo 648
cmdbdiscovery.sh 163
commandLine.bat 677
commandLine.bat/sh 571
conman shut 782
conman stop 782
db2 CATALOG DATABASE 208
db2 CATALOG TCPIP NODE 208
db2 CONNECT TO 212, 214
find 648
lssam 786
jar update 410
KfwTmsDla.exe 160
loadidml.bat 564
loadidml.sh 160, 564
lssam 789
MonitorCommand 760
rgreq 784
samctrl 760
StartCommand 760
startFusion.bat 668
startFusion.sh 499, 668
StopCommand 761
tacmd 645
tacmd createnode 159
tacmd getDeployStatus 648
Common Data Model (CDM)

creating new classes 71
definition 665
common elements 12
common reporting 99
comparison rule
reconciliation task 487
Computer Collections, discovering 402
Computer Room Air Conditioner (CRAC)
data collection 22
description 20
config.properties file
connecting to databases 221
contains a sample 221
configuration 14–15
lab 12
configuration items
See CI
conman sbj command 720
conman sbs command 720
conman showcpus command 722
conman shut command 782
conman stop command 782
conman switchevtproc command 722
connection, verification 73
Content Manager Datasource 251
Context Menu Service (CMS)
See also CMS 701
cooling device, metric collection 23
core collectors 38
createTADDM72ActualCIDataSchema.db2 script 673
creating resources and resource groups 779
credentials
configuring for database access 265
credential-less discovery 642
CritRsrcProtMethod persistent attribute 756
cron task
activating 533
instance name 491
LDAPSync 627
parameters 628
PLUSEITMMETERS 111
PLUSEITMMETRICS 113
reconciliation task 490
saving 533
VMMSync 627
Cross product model, branches 238
custom dashboard
creating for transaction management 185
log in to console 185
custom workspace creation 44
custom_canvas_actions.properties file 689

D

dashboard
creating a custom dashboard 176
energy 73
using for Green IT 65
Data Analyzer 37
Data Center
defining parameters 37
data center
configuring 114
configuring asset hierarchies 115
Heat view 117
Layout view 116
power capacity 92
power capacity, viewing a chart 92
surveying 115
viewing 116
Data Center Infrastructure Efficiency (DCIE) 92
description 20
viewing a chart 92
viewing reports 54
data collection
Air Handle Unit 23
chiller 23
Computer Room Air Conditioner (CRAC) 22
core collectors 38
creating a collector data source 39
energy metrics 38
generator 23
IBM BladeCenter 22
meter 23
power and thermal data 24
power distribution unit (PDU) 22
rack servers 22
racks 22
sensors 22
system z10 22
uninterruptible power supply (UPS) 22
data integration initiatives 4
Data Integration Services (DIS) 701
data model classes 83
data schema
analysis 671, 675
defining a new schema 673
importing 670, 674
data source
configuration, IBM_TRAM 252
connecting 106
creating 73, 263
creating for Tivoli Service Request Manager 264
data structures, importing 262
data synchronization utility 221
data types
CISPEC 456
creating 454
names 455
data.jar file, updating 410
databases
connecting source and target 109
extraction for reports 4
replication, DB2 HADR 734
used by Tivoli products 200
DB2
automatic client reroute (ACR) 733–734
backing up the database 771
cresting instance owner 764
discovering 402
entering instance details 773

Archived

high availability disaster recovery (HADR) 733–734
synchronization modes 735
installing instances 764
JDBC driver connection 734
restoring the database 771
table spaces 769
upgrading 766
db2 CATALOG DATABASE command 208
db2 CATALOG TCPIP NODE command 208
db2 CONNECT TO command 212, 214
DB2INST1 schema 229
-dbpwd parameter
CMS Registry Loader 709
-dbuser parameter
CMS Registry Loader 709
Definitive Media Library (DML)
creating 587
description 546
delta discoveries 667
DependsOn relationship 751
deployed asset data 515
Deployer's Workbench 666
deployment, checking status 648
DIS table creation 705
discovery bundle, deploying 655
Discovery Library Adapter (DLA) 70
creating an output file 563
generating an output file 558
importing into TADDM 558
Discovery Library Toolkit
installing 81
installing the service templates 176
discovery profiles
configuring 649
TADDM 642
disk volumes, provisioning SAN attached disks 391
DL_TADDM_HostName parameter 686
dynamic links, creation of 456
dynamic resource group 749
Dynamic Workload Broker Component
creating a logical resource 798
creating floating resources 789
creating resources 799
creating shadow resources 799
starting a resource group 800
virtual IP address 797
virtualizing workload 797

E

Eaton breaker, viewing reports 54
Eaton Power Xpert 22
EIF 729
probe rules 369
updating configuration 369
EIF probe 306, 418
configuring 308
connecting to Tivoli Netcool/OMNIbus 418
installing 307
processing incoming alarms 309
restarting 424
starting 451
updating alert.status table 419
EIF Receiver
assigned 372
forwarding events 372
EIF server host name configuration 425
EIF Slot Customization 370
embedded WebSphere Application Server 720
connection properties 721
crash 792
creating managed relations 779
creating resources and resource groups 779
creating shadow resources 779
DB2 JDBC driver 733
Dynamic Workload Broker Component 723
event processor 722
JDBC driver connection 734
starting resource groups 779
stopping instances 774
endpoints
defining for CCMDB 584
discovering 649
Energy Dashboard 91
configuring 73
configuring a service tree template 77
creating the page 78
customizing 91
launching TADDM 97
launching Tivoli Enterprise Portal 97
Service Tree 95
energy metrics
collecting 23
configuring 38
viewing 43
energy monitoring agents 22
enrichment policy, creating 456
enRole.properties file, modifying 611
Enterprise Asset Manager 637
entry report 279
environment variables 173, 175
environmental monitoring device, metric collection 23
escalation
activating 499
changing the status of a record 522
changing the status opf a record 494
configuring for linking CIs to assets 494, 497, 522, 538
creating a new escalation 497
saving 499
Escalations
creating 442
escalation point creation 522
eTAI
accepting requests 299
parameters 300
plug-in 292
Event Correlation Engine in Event Processor 722
event forwarding 135
Event Integration Facility
See EIF

event list editing 473
Event Management and Monitoring (EMM) 351
event mapping 472, 474
  list of mappings 416
Event Processor
  Action Helper 722
  description 722
  Event Correlation Engine 722
  Rule Builder 722
event synchronization
  sending error events to the object server 424
event updates 308

F

Fault Management 133
find command 648
FIPS Compliance 174
FixPack deployment 595
floating resource
  creating 789, 800
  creating for the Event Processor 785
  creating for the master domain manager 783
  creating for Tivoli Dynamic Workload Console 787
fully qualified domain name (FQDN) 220
functional integration initiatives 4

G

GATE_TSRM.props file
  configuring 447
  example 447
generator
  agent 23
  data collection 23
globally unique identifier (GUID)
  generating for assets 482
  identifying components 665
  linking CIs to assets 483
  links between CIs and assets 510

H

HADR
  installing scripts 775
  verifying primary node 776
  verifying standby node 776
Hardware Discovery 401
heat map 117, 120–122
high availability disaster recovery (HADR)
  configuring 771
  DB2 733–734
Historical Power Trends reports 54
hourly metrics, combining 253
HTTP monitor element, creating 353
HUB TEMS
  configuring 425
  restarting 425

I

IBM Tivoli Monitoring DLA 161
IBM_TRAM
  data source configuration 252
  schema 216–217, 238
IBM_TRAM.ComputerSystem table
  column mappings 218
IBM.Application resource class 747
IBM.GblResRM resource manager 747, 749
IBM.ManagedResource class 749
IBM.ServiceIP resource class 747
IdML book, importing 82
IdML file
  copying 72
  loading 72
image list, obtaining 332
Impact Application, creating 437
Impact column heading 441
Impact mapping classification IDs 441
Impact table, creating 429
IMPACTMAP object 429
imram_x.x.zip file 550
Incident tickets
  creating 418
incident tickets
  creating 416, 418
IncidentClassification 441
INCIDENTCLASSIFICATION attribute 440
incidentclassification attribute 431
incidents, opening 446
initiatives, integration 4
Installed Software Components, discovering 402
instance name, cron task 491
integration
  aspects 5
  elements 12
  launch 5
integration endpoint, configuring 111
Integration Global Directory, configuring 446
integration initiatives 4
  agent management 5
  data 4
  functional 4
  manageability 5
  navigation 4
  reporting 5
  security 4
integration module
  installing for CCMDB 551
  setting up for CCMDB 583
Intelligent PDU 20
interceptors 296
Internet Service Monitoring (ISM)
  configuring 351
  service level classification 355
IT equipment power usage 92
  viewing a chart 92
ITM61 data source 106
ITMFEMRO_Windows.exe 28
ITMScopeSensor 652
ITMUSER schema 229

Archived

J

jar update command 410
Java APIs 409
Java wrapper module 409
JDBC connections 221
JDBC driver
  connecting to DB2 734
JnextPlan script 782
Job Dispatcher
  Dynamic Workload Broker Component 723
Job execution details 727
Johnson Controls Metasys 23
join creation 271
JRExec server, running 471
jssecacerts.cert file 672

K

KfwTmsDla.exe command 160
kilo watt hour (kWh) description 20
kis.baroc file 373
kis.rules file 374
kto.baroc file 373

L

L1 discovery
  data for a sample server 658
  supported operating systems 644
L2 discovery
  data using Tivoli Monitoring Agents 659
  supported operating systems 644
lab configuration 12
Language Support Feature installation for Rational Asset Manager 555
launch entry
  CMS 709
  creating and updating 710
  listing currently registered entries 710
  removing currently registered entries 710
  XML file validation 710
launch integration 5
launch points
  registering 711
launch-in-context
  See LIC
LDAP realm name
  configuring 386
LDAPSync cron task 627
LIC 12
  components 701
  integration 687
  navigation integration 5
life cycle, definition of 478
Lightweight Directory Access Protocol (LDAP) 131
  authorizing data 626
  managing Tivoli Service Request Manager users 597
Lightweight Third Party Authentication (LTPA) 130
link clause
  adding 484, 525
  configuring 486
link rule
  adding a clause 484, 525, 528
  assigning 484
  creating for linking assets 483, 523
  linking assets 515
  reconciliation task 487
  saving 487, 529
  selecting 489, 531
Linux Agent 210
Linux_IP_Address_DV view 219
Linux_Machine_Information_DV view 219
Live Refresh 600
loadidml.bat command 564
loadidml.sh command 160, 564
login tracking 602
login.jsp file 603
LOGINDEXBUILD parameter 771
LOGINTRACKING parameter 614
LOGINTRACKING variable 601
long term switch 740
Lotus Notes password change 630
lssam command 786, 789–790, 800

M

Machine_Information_DV view 220
manageability integration initiatives 5
managed relations, creating for master domain manager 784
MANAGEDSYSTEM table 210, 238
ManagedSystem, create a relationship 257
manual temperature sensors 118
mapping
  between TADDM and CCMDB 673
  creating a new mapping 676
  importing between TADDM and CCMDB 676
  source and target 672
mapping table, creating 425
MAXDB connection settings 452
maxdb71 database
  backing up the database 551
  backing up the registry 551
maximo.ear file
  copying 604
  deploying 604
  rebuilding 604
maximo.jar file
  copying 610
  extracting 598
maximoserviceprofile.jar file
  copying 613
  extracting 598
MemberLocation persistent attribute 748
Memory Hourly
  creating a query subject 259
MEO71 data source 106
Metadata Wizard 263
Metadata wizard 267–269
meter, data collection 23
metrics
  collecting 22–23

combining 253
viewing and reporting 43, 279
missing libraries 173
installing 173
Model Query Language (MQL) 159
MonitorCommand command 757, 760
monitoring agents
Active Energy Manager 22
APC InfraStruXure 23
Eaton Power Xpert 22
Johnson Controls Metasys 23
Siemens Apogee 23
monitoring events, sending 418
monitoring profiles
deploying 357
distributing 356

N

namespace, creating 258
Naming and Reconciliation Service 702
navigation in integration 5
navigator
creating 44
notepad view 47
opening 46
web browser view 47
NCHOME environment variable 173
nco_config
creating a tool 448
creating a trigger 448
nco_event 473
NEARSYNC synchronization mode 735
Netcool/OMNIbus event list 473
network devices, adding 35
New data source wizard 264
NominalState persistent attribute 748
NT_Computer_Information_DV view 218
NT_System_DV view 218

O

Object Server
alerts, opening incidents 446
connection settings 454
installing 447
Object Structure Service Definition 601
object structures
request creation object 329
using 329
ObjectServer
database, modifying 372
schema, updating 175
Offering Catalog application, accessing 629
OMNIbus EventReader 472
omnibusUpdater 184
OMNIHOME environment variable 173
Operational Management Products (OMP) 589
creating 566
description 701
operational quorum 755
OpState
Failed Offline 758–759
offline 757, 759
online 757
Oracle, discovering 402
Ownergroup 441
Ownergroup attribute 434

P

password
changing 615
Lotus Notes 630
resetting 621
password link, adding 603
patch utility script, Object Server 447
PATH environment variable 173
persistent resource attribute
CritRsrcProtMethod 756
ProtectionMode 756
persistent resource group 748
planman deploy command 728
plot chart view, adding 47
PLUSEITMMETERS 111
PLUSEITMMETRICS 113
PMRDPBCAPI web service 330
pmrdpbcapigetAvailablePoolList, returning resource pool information 330
PMSCMR Object Instance 334
PMSCMR object instance 334
PMSCSRU security group 629
PMZHBR1_PMSCMRCREAT object structure 334
power capping 20
power consumption monitoring 98
power distribution unit (PDU)
data collection 22
description 20
discovering with TMS DLA 71
power metrics
modifying default values 96
viewing status thresholds 95
Power Saver mode 20
Power Savings Recommendations 55
viewing reports 55
Power Usage Comparison
viewing reports 55
Power Usage Effectiveness (PUE) 20
probe rules
Event Integration Facility 369
ProbeEIF, starting 451
Process agent
configuring 450
Process Agent, starting 451
Process Solution Installer (PSI)
installing Rational Asset Manager 552
Promotion Scope
CCMDB 667
ProtectionMode persistent attribute 756
provisioning workflows
parameters 388
running 387

Archived

Publish Package wizard 277

Q

qualifierCCMDB72Classification.db2 script 671
query sets
  creating relationships with query subjects 255
Query Studio 237
query subjects
  creating relationships with query sets 255
  defining 258–259
  testing 253
Quorum State 755
quorum, operational 755

R

Rack Air Removal Unit
  metric collection 23
Rack Servers
  discovering with TMS DLA 71
rack servers
  data collection 22
racks
  data collection 22
RAMIM 583
Rational Asset Manager
  .zip files for CCMDB integration 550
  description 546
  integrating with CCMDB 549
  logging in 549
  OMP 589
recommendations report 57
recon.engine.dataset.map.ASSE system property 521
reconciliation
  for linking assets to deployed assets 483
  process 483
  properties 519
Reconciliation module applications 483
reconciliation task
  components 487
  creating 487, 529
  creating for linking assets 483
  defining a schedule 488
  relationship 492
  scheduling an execution 532
Redbooks Web site 806
  Contact us xlvi
registryLoader.bat file 708
registryLoader.sh file 708
relationship
  Affinity 751
  Collocated 751
  creating 255–257, 270, 273, 535
  DependsOn 751
  managed 750
  reconciliation task 492
  start 750
  StartAfter 750
  stop 750
  testing 272, 275
Relationship Rules, CCMDB 667
repeaters, adding 35
report sets
  importing 89
  viewing 54
Report Studio
  Creating a new report 279
  launching 278
Reporting and Optimization
  collection settings 32
  configuring 29
  configuring preferences 33
  data analysis 37
  Data Center Infrastructure Efficiency (DCiE) metrics 24
  description 23
  filtering report data 55
  installing 28
  viewing data 42
  viewing report sets 54
Reporting integration 5
Reporting Package selection 279
reports
  Active Energy Manager 54
  Baseline Power Metrics 54
  common reporting 99
  CPU utilization 24, 54
  creating 276, 278
  Data Center Efficiency 54
  Eaton breaker 54
  energy 24
  energy usage metrics 24
  entry report 279
  Historical Power Trends 54
  Invoice linked to a specific account 100
  Power Savings Recommendations 55
  Power Usage Comparison 55
  recommendations report 57
  resource types 59
  running 60
  server incident list report 285
  summary list report, number of incidents per server 286
  Tivoli Monitoring for Energy Management 93
  Tivoli Usage and Accounting Manager 89
  top accounts in terms of costs 99
request creation object 329
request for change 418
  creating 416, 418
request XML
  accessing web service information 331
requests, viewing 621
Resource Advisor
  Dynamic Workload Broker Component 723
resource classes
  definition of 746–747
  IBM.Application 747
  IBM.ServiceIP 747
resource groups 783
  creating for the Event Processor 785

creating for the master domain manager 783
creating for Tivoli Dynamic Workload Console 787
dynamic 749
persistent 748
resource pool name 330
resource pools
images list 332
returning information 330
resource, definition of 746
Response Plans
creating 443
response XML
accessing web service information 331
REST interface
creating a catalog request 334
description 328
obtaining images list 332
URL attributes 335
Resync Agent 357
rgreq command 784
routers, adding 35
Rule Builder in Event Processor 722
run.bat script
updating for Java Virtual machine 222

S

SALLBS signature option 693
samctrl command 760
SAN attached disk volumes, provisioning 391
schema name, properties 234
schema, CI type 668
Scope Sensor 652
scriptframework.properties file, modifying 612
security integration initiatives 4
sendevent command 732
sensors
data collection 22
discovering with TMS DLA 71
manual temperature 118
server incident list report 285
server pools, configuring 35
Service Catalog
creating a service request 632
launching Lotus Notes 630
Offering Catalog 629
requesting a new ID 629
viewing a catalog order 633
Service Component Repository 162
service definition 662
service hierarchy definition 662
service instance, requesting 369
service level agreement (SLA) 481
service level classification, configuring 355
service level failure, checking for 369
service templates, assigning 83
service tree 94
scorecard 95
template, configuring for Energy Dashboard 77
set up IBM_TRAM schema 213
shadow equivalency 753, 779, 783, 799
shadow resource 753, 779, 783
shadow resources 799
shared console 5
short term switch 739
shortcut, creating 276
Siemens Apogee 23
SIMPBS signature option 693
single sign-on (SSO) 12, 130
configuring for Tivoli Enterprise Portal console 318
configuring for WebGUI console 319
situation status, updating 418
Situation Update Forwarder (SUF)
installing 307
running 424
running SQL files 310
updating situation status 418
situations
checking for a service level failure 369
checking for a transaction slowdown 369
distributing 356
SOAP Client, configuring 470
soapcli.sh file 470
soapcli.sh script, running 471
Software Discovery 401
source mapping 672
SRM interfaces, activating 599
SSL, using 672
Start Product Console 137
start relationship 750
StartAfter relationship 750
StartCommand command 760
startFusion.bat command 668
startFusion.sh command 499, 668
status thresholds
power metrics 95
thermal metrics 95
stop relationship 750
StopCommand command 761
storage devices
adding 35
grouping 35
linking 35
subclusters 755
summary list report
number of incidents per server 286
SVIEWER signature option 693
Switch Fault Tolerance, enabling 780
Switch Manager
long term switch 740
short term switch 739
switches, adding 35
SYNC synchronization mode 735
System z10, data collection 22
System_DV view 220

T

table spaces
TWS_DATA 769
TWS_LOG 769
tacmd command 645

tacmd createnode command 159
tacmd getDeployStatus command 648
tacmd.exe command 645
TADDM 68, 135, 400
  assigning service templates 83
  checking for updates 686
  configuration details 98
  configuring a CI type schema 668
  creating a data source 499, 566
  creating a service 136
  database configurations 684
  default discovery profiles 642
  deploying discovery bundle 655
  discovering Tivoli Monitoring endpoints 649
  Discovery Configuration 401
  integrating with Tivoli Business Service Manager 81
  integrating with Tivoli Monitoring Energy Management 70
  IT resources topology and details 72
  launching 97
  logging in 649
  monitoring agent bundle 656
  restarting the server 645
  selecting a connection method 501
  Start Product Console 137
  supporting new devices 71
TADDM DIS/CMS Integration Package
  populate CMS Registry database 706
TADDM Integration Package
  registering launch points 711
  utilities 711
TADDM72CIType 566
TAI method 293
TAL API automations
  installing 447
TAMETai class name 296
target mapping 672
task filter, reconciliation task 487
TBSMChartService 186
TCR Framework Manager 248
tedgen utility 370
TESTVM virtual machine 162
thermal metrics
  collecting 23
  modifying default values 96
  viewing status thresholds 95
thermal metrics viewing 43
tie breaker
  network 756
Time Dimension
  create a relationship 256
TIME_DIMENSION table 215
TIPHOME environment variable 175
Tivoli Application Dependency Discovery Manager
  See TADDM
Tivoli Asset Management for IT
  Reconciliation module applications 483
Tivoli Asset Manager
  controlling assets 480
Tivoli Business Service Manager 68, 81, 183
  download URL 667
  installing 136
  logging in 148
Tivoli Common Reporting 89
  installation path 70
Tivoli Common Reporting for Asset and Performance Management 206
Tivoli Composite Application Manager
  transaction monitoring 358
Tivoli Composite Application Manager for Transactions 183
Tivoli Data Warehouse 38
  configuring 29
  views 218
Tivoli Directory Integrator description 407
Tivoli Dynamic Workload Console
  floating resource 787
  resource group 787
Tivoli Enterprise Monitoring Agents 356
Tivoli Enterprise Monitoring Server
  configuring with Tivoli Enterprise Portal Server 313
  logging in 645
Tivoli Enterprise Portal
  launching 97
  multiple servers 314
  single server 313
Tivoli Enterprise Portal Server
  configuring with Tivoli Enterprise Monitoring Server 313
Tivoli Event Synchronization
  configuring 423
  installing 420
  running from SQL 423
  testing connectivity 424
Tivoli Identity Manager 627
  APIs 409
  description 406
  discovering 409
  downloading 410
  integrating with Tivoli Provisioning Manager 406
Tivoli Identity Manager Self Service console 619
Tivoli Integrated Portal 5, 25
  default port 173
  logging in 383
Tivoli Integrated Portal, dashboard host name 685
Tivoli Integration Composer 566, 666
  connecting data sources 106
  mapping 110
Tivoli Management System Discovery Library Adapter (TMS DLA) 68
Tivoli Monitoring
  bidirectional integration 133
  event forwarding 135
  forwarding events to Tivoli Netcool/OMNIbus Probe 425
Tivoli Monitoring endpoints
  discovery 649
Tivoli Monitoring Energy Management
  integrating with TADDM 70
Tivoli Monitoring for Energy Management 43, 70, 110,

113
launching reports 93
objects 112–113
Reporting and Optimization feature 23
reporting on power consumption 98
Tivoli Monitoring for Virtual Servers 128
Tivoli Monitoring Sensor
configuring 644
Tivoli Monitoring/Tivoli Composite Application Manager
updating situation status 418
Tivoli Netcool/Impact
creating a new project 452
creating Request for Change 418
creating request for change 416
event mapping table 416
reading the ObjectServer table 418
running a workflow 418
running workflows 416
Tivoli Netcool/OMNIbus 183, 306, 375
bidirectional integration 133
context integration 311
creating incident tickets 416
EIF probe 306
EIF probe installation 418
EIF probes 418
event integration 307
event mapping 472
event updates 308
installing 172
ObjectServer 372
opening incidents 446
process agent 450
reading the ObjectServer table 418
receiving alerts from other products 418
updating the ObjectServer schema 175
Tivoli Network Manager
fix packs 175
installing 174
Tivoli Provisioning Manager 393, 400
automation package 407
configuring security 385
input parameters 408
integrating with Tivoli Identity Manager 406
running a workflow 418
running workflows 416
Tivoli Provisioning Manager Data Center Model (DCM) 400
Tivoli Reporting Package, importing 235
Tivoli Service Automation Manager 407
object structures 329
Tivoli Service Request Manager
adding password link 603
creating a data source 264
creating incident tickets 416, 418
creating service requests 597
enabling user deletion 601
managing users 597
opening incidents 446
Tivoli Service Request Manager Gateway
creating incident tickets 418
Tivoli Service Request Manager Service Catalog 627
Tivoli Service Request Manager Service Desk 627
Tivoli Storage Productivity Center
importing security keys 387
Tivoli System Automation for Multiplatforms 753, 755, 796
Tivoli Usage and Accounting Manager 38, 69, 84
report engine 98
reports 89
Tivoli Workload Scheduler
backing up 771
creating traces and job logs 768
engine 780
installing 767, 796
restoring 771
store-and-forward mechanism 732
tivoli_eif.rules file 373–374
TMS DLA 161
discovering devices 71
TMS DLA book
fields 82
TMS IdML book
generating 71
tool creation
in navigation menu 312
in nco_config 448
top-level IT asset class 517
TPC Discovery 389
TPC_DefineTPCServer.wkf workflow 387
TPM_TIM_Create_Service workflow 409
TPM_TIM.tcdriver
downloading 408
Transaction Collector 368
Transaction Collector agents 365
transaction management 185, 188
Transaction Reporter 371
Transaction Reporter (KTO) agent 372
transaction slowdown
checking for 369
Transaction Tracking
configuring 360
creating a new transaction 361
distributing 365
treeTemplates.xml file
configuring for Energy Dashboard 77
trust associations
enabling 292, 295
interceptors 296
tsamService.sh script 376
TWS_DATA table space 769
TWS_LOG table space 769
type mappings, CI 502
type schema, CI 499

U

uninterruptible power supply (UPS)
data collection 22
description 20
UNIX Agent 210
unixlocl method 742

Archived

updatedb script 598
updatedb.bat, running 599
Urgency 441
Urgency attribute 433
user account
  adding 621
  requesting 619
user deletion, enabling 601

V

VACTCINUM condition 694
verifying a connection 73
views
  Linux 219
  of data center 116
  UNIX 220
  Windows 218
virtual IP address, creating 783
VMMSync cron task 627
VMWare ESX
  discovering 402
VMware ESX servers 141
VMware Infrastructure 141, 147
VMware Virtual Center 135
VMware Virtual Center server 141

W

Web Application URL
  configuring 446
Web Server
  discovering 402
Web Service End Point 466
Web Service Method Parameters 463
web services
  creating 580
  creating in CCMDB 458
  deploying 582, 599
  setting up in CCMDB 580
Web Services Policy
  creating 460
WebLogic Server
  discovering 402
WebSEAL
  authenticating users 292
  definition 293
  integration flow 292
  specifying a password 300
Webserving Plugin 363
WebSphere Application Server
  discovering 402
WebSphere plug-in 363
wimconfig.xml 132
Windows Agent 210
Windows-Linux CPU Hourly
  create a relationship 256
wires
  creating 181
workflows
  creating 444
  running 416, 418
WSDL file
  compiling 459

X

XML file
  launch entry 709
XML Toolkit 149, 667
  checking for TADDM updates 686
  installing 136, 681
  TESTVM virtual machine 162
-xmlfile parameter
  CMS Registry Loader 709

Archived





Redbooks

Tivoli Integration Scenarios

(1.5” spine)
1.5”<-> 1.998”
789 <->1051 pages

REDBOOKS

Archived

Archived

Tivoli Integration Scenarios



Describes security, data, navigation, reporting, and task integration

Provides extensive samples with a scenario-based approach

Discusses Tivoli product interoperability

This IBM Redbooks publication provides a broad view of how Tivoli system management products work together in several common scenarios. You must achieve seamless integration for operations personnel to work with the solution. This integration is necessary to ensure that the product can be used easily by the users.

Product integration contains multiple dimensions, such as security, navigation, data and task integrations. Within the context of the scenarios in this book, you see examples of these integrations.

The scenarios implemented in this book are largely based on the input from the integration team, and several clients using IBM products. We based these scenarios on common real-life examples that IT operations often have to deal with. Of course, these scenarios are only a small subset of the possible integration scenarios that can be accomplished by the Tivoli products, but they were chosen to be representative of the integration possibilities using the Tivoli products.

We discuss these implementations and benefits that are realized by these integrations, and also provide sample scenarios of how these integrations work.

This book is a reference guide for IT architects and IT specialists working on integrating Tivoli products in real-life environments.

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

SG24-7878-01 ISBN 0738435139