WebSphere software

IBM

Service Lifecycle Governance with IBM WebSphere Service Registry and Repository

- Discover how to implement service life cycle governance
- Examine how to build WSRR solutions
- Learn by example with practical scenarios

Nicole Hargrove
Ian Heritage
Prasad Imandi
Martin Keen
Wendy Neave
Laura Olson
Bhargav Perepa
Andrew White

Redbooks

ibm.com/redbooks

Archived

Archived

International Technical Support Organization

Service Lifecycle Governance with IBM WebSphere Service Registry and Repository

December 2009

Archived

SG24-7793-00

Note: Before using this information and the product it supports, read the information in "Notices" on page xvii.

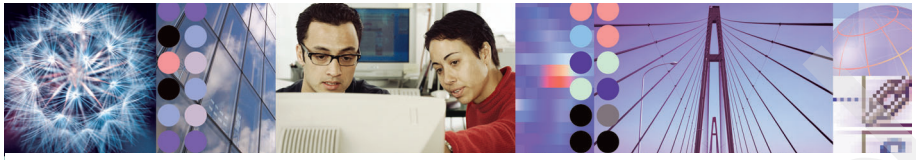
Archived

First Edition (December 2009)

This edition applies to WebSphere Service Registry and Repository V6.3 with WebSphere Application Server V7.0.

© Copyright International Business Machines Corporation 2009. All rights reserved.
Note to U.S. Government Users Restricted Rights -- Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contact an IBM Software Services Sales Specialist



Start SMALL, Start BIG, ... **JUST START**

architectural knowledge, skills, research and development . . .

that's IBM Software Services for WebSphere.

Our highly skilled consultants make it easy for you to design, build, test and deploy solutions, helping you build a smarter and more efficient business. **Our worldwide network of services specialists wants you to have it all!** Implementation, migration, architecture and design services: IBM Software Services has the right fit for you. We also deliver just-in-time, customized workshops and education tailored for your business needs. You have the knowledge, now reach out to the experts who can help you extend and realize the value.

For a WebSphere services solution that fits your needs, contact an IBM Software Services Sales Specialist:
ibm.com/developerworks/websphere/services/contacts.html

Archived

Archived

Contents

Contact an IBM Software Services Sales Specialist iii

Notices xvii
Trademarks xviii

Preface xix
The team who wrote this book xix
Become a published author xxii
Comments welcome xxiii

Part 1. Overview 1

Chapter 1. SOA and service governance overview 3
1.1 What is SOA governance? 4
1.2 SOA governance and service governance 6
1.3 The IBM SOA Governance and Management Method 7
1.3.1 Plan and organize 10
1.3.2 Program management controls 10
1.3.3 Service development 11
1.3.4 Service operations 11
1.4 Requirements to enable service governance 11
1.4.1 Govern both the service consumer and service provider 12
1.4.2 Manage service contracts and SLAs 12
1.4.3 Manage multiple service versions within life cycle states 12
1.4.4 Govern all types of services 13
1.4.5 Manage service metadata and artifacts 13
1.4.6 Govern and enforce service design and run time policies 14
1.4.7 Integrate with runtime environments 14
1.4.8 Find and publish services, metadata, and artifacts 15
1.4.9 Discover and identify services in a runtime environment 15
1.4.10 Communication, reporting, and management of change 16
1.4.11 Integrate with other products supporting SOA governance 16
1.5 WSRR as an enabler of service governance 17
1.5.1 Promoting service reuse 19
1.5.2 Enhancing connectivity 23
1.5.3 Optimizing service usage 26
1.5.4 Enable governance 35

Chapter 2. WebSphere Service Registry and Repository

© Copyright IBM Corp. 2009. All rights reserved.

technical overview 39
2.1 Architecture of WebSphere Service Registry and Repository 40
2.1.1 Registry and repository 41
2.1.2 User interfaces 42
2.1.3 Governance functions 42
2.1.4 Administration interfaces 43
2.1.5 Programming interfaces 43
2.2 Service life cycle support features 45
2.3 Document types 45
2.3.1 XML schema definition 47
2.3.2 Web Services Description Language service definitions 49
2.3.3 Service Component Architecture modules 52
2.3.4 Service Policy (WS-Policy) 54
2.3.5 XML metadata files 54
2.3.6 Derived objects 55
2.4 Content model 56
2.4.1 Service description entities 57
2.4.2 Service description metadata 62
2.4.3 User defined properties and relationships 66
2.4.4 Business models templates 66
2.5 WSRR deployment topologies 67
2.5.1 Recommended deployment topologies 67
2.5.2 Runtime deployment topology considerations 68
2.6 Deployment configurations 70
2.6.1 WebSphere Application Server configurations 71
2.6.2 Database configuration 74
2.7 Packaging 75

Chapter 3. Introduction to the governance enablement profile 77
3.1 Overview of the governance enablement profile 78
3.2 Models in the governance enablement profile 78
3.2.1 Asset entity 80
3.2.2 Business capability entity 82
3.2.3 Capability version entity 84
3.2.4 Document of understanding entity 88
3.2.5 Schema specification entity 89
3.2.6 Service interface specification entity 90
3.2.7 Service level agreement entity 92
3.2.8 Service level definition entity 95
3.2.9 Service entity 98
3.2.10 Service endpoint entity 100
3.2.11 Extension service endpoint entity 102
3.2.12 MQ service endpoint entity 102

Archived

3.2.13 SOAP service endpoint entity 102
3.3 Roles in the governance enablement profile 103
3.4 Life cycles in the governance enablement profile 105
3.4.1 Asset life cycle 106
3.5 Capability life cycle 108
3.6 SOA life cycle 109
3.6.1 SOA Life cycle: Assemble phase 112
3.6.2 SOA life cycle: Deploy phase 114
3.6.3 SOA life cycle: Manage phase 116
3.7 SLD life cycle. 117
3.7.1 SLA life cycle. 120
3.7.2 Endpoint life cycle 123
3.8 Policies in the governance enablement profile 124
3.8.1 Life cycle transition policies. 124
3.8.2 Life cycle detail policies. 129
3.8.3 Life cycle update policies 145

Part 2. Building WSRR solutions. 151

Chapter 4. JKHL Enterprises case study 153
4.1 Introduction to the case study 154
4.2 JKHLE SOA governance vision and requirements 154
4.3 JKHLE runtime environment 156
4.4 The JKHLE SOA governance solution 157
4.5 JKHLE service governance with the WSRR scenario 158
4.6 JKHLE organizational structure and personas 159

Chapter 5. Modeling in WebSphere Service Registry and Repository Studio 163
5.1 Overview of WebSphere Service Registry and Repository Studio 164
5.2 Using models with WebSphere Service Registry and Repository Studio 164
5.2.1 Configuration profiles 165
5.2.2 Profile templates 165
5.2.3 System models 165
5.2.4 Configuration profile files. 167
5.2.5 Business model systems. 167
5.2.6 Classification systems. 169
5.2.7 Life cycles 170
5.3 Extending templates versus editing a template. 170
5.4 Customizing the governance enablement profile for JKHL Enterprises . 171
5.4.1 Creating the configuration project for JKHLE 171
5.4.2 Applying JKHLE's customizations to the GEP63 business model . 174
5.4.3 Applying JKHLE's customizations to the life cycles 190
5.5 Applying JKHLE's customizations to the governance profile taxonomy . 209

Archived

5.6 Generating a profile 213
5.7 Transferring .emx models between clients 214
5.7.1 Exporting .emx files 214
5.7.2 Importing .emx files 215

Chapter 6. WebSphere Service Registry and Repository Web user interface 217
6.1 Overview of the Web UI 218
6.2 WSRR Web UI definition files 219
6.2.1 Perspective 220
6.2.2 Menu bar 221
6.2.3 Navigation tree 223
6.2.4 View query 224
6.2.5 Detail view 224
6.2.6 Collection view 225
6.2.7 Home page 226
6.3 Themes 229
6.4 Customizing the WSRR Web UI for the JKHLE business scenario 231
6.4.1 Removing the service interface specification from the WSRR Web UI 231
6.4.2 Removing the abstract asset links from the WSRR UI 237
6.4.3 Removing the life cycle states from the WSRR UI that were removed from the model 238
6.4.4 Removing the items from the tasks menu bar that reference the removed life cycle states 241
6.4.5 Updating the task names in the menu bar to reflect the changes in the runtime environments 244
6.4.6 Adding life cycle tasks in the UI 245
6.4.7 Replacing the term “DOU” with the term “Subscription Request” in the WSRR Web UI 246
6.4.8 Updating the SOALifecycleDecisionRights governance policy 247
6.4.9 Modifying default user preferences 249
6.4.10 Updating the configuration profile on the WSRR server 251
6.4.11 Creating the JKHLE theme 254
6.5 Troubleshooting 261

Chapter 7. Security 263
7.1 Overview of security 264
7.2 WebSphere Application Server security 264
7.2.1 J2EE security 264
7.2.2 Administrative security 265
7.2.3 Application security 265
7.2.4 Administrative roles 265

Archived

7.2.5 Special subjects 267
7.2.6 WSRR role mappings 267
7.2.7 WSRR administrator RunAs role 268
7.3 WSRR security 269
7.3.1 Levels of authentication 269
7.3.2 Configuration properties 270
7.3.3 Fine-grained access control 270
7.4 Implementing WebSphere Application Server security at JKHL Enterprises 275
7.4.1 Granting a user the WebSphere Application Server Operator role . 275
7.4.2 Mapping the LDAP group to the WSRR Administrator role 277
7.4.3 Configuring the WSRR Administrator RunAs role 278
7.5 Implementing WSRR fine-grained security at JKHLE 280
7.5.1 Mapping users and groups to roles in WSRR 280
7.5.2 Mapping permissions to roles in WSRR 284

Chapter 8. Promotion 289
8.1 Overview of the WSRR promotion feature 290
8.2 Promotion methods 291
8.2.1 Synchronous promotion 291
8.2.2 Asynchronous promotion 292
8.2.3 Manual promotion to a file 293
8.2.4 Promotion filtering by classification 293
8.3 Implementing promotion at JKHLE 295
8.4 Editing the sample configuration file 297
8.5 Troubleshooting 299

Chapter 9. Policies 301
9.1 Overview of policy management in WSRR 302
9.1.1 Supported policy frameworks and concepts in WSRR 302
9.1.2 Policy domains 304
9.1.3 Publishing and discovering policies 307
9.1.4 Authoring, editing, and attaching policies 310
9.1.5 Policy enforcement 311
9.1.6 Policy life cycle governance 314
9.2 Applying policy to the JKHLE business scenario 315
9.3 Implementing policy 316
9.3.1 Creating a WS-I compliance policy 316
9.3.2 Creating an updated property enforcement policy 329
9.3.3 Testing and deploying governance policies 341
9.4 References 347

Chapter 10. Reports 349
10.1 Customizing reports 350

Archived

10.2 Configuring a WSRR location in WSRR Studio 352
10.3 Creating a report project 358
10.4 Using sample reports in WSRR Studio 361
10.5 Configuring the sample reports 364
10.5.1 WSDL Port Availability Report sample report 364
10.5.2 List of WSDLs and their associated XSDs sample report 378
10.5.3 JKHLE provisioning report 382

Part 3. Scenarios 399

Chapter 11. Creating an organizational structure 401
11.1 Creating the JKHLE organizational structure 402
11.2 Creating an organization 403
11.3 Adding child organizations 404

Chapter 12. Governing a schema 407
12.1 Governing a schema specification 408
12.2 Creating the schema specification 408
12.3 Proposing the schema specification 412
12.4 Approving the schema specification 414

Chapter 13. Governing a service defined in a monolithic WSDL 417
13.1 Governing a new service at JKHLE 418
13.2 Creating a business capability 420
13.2.1 Creating a new business service 420
13.2.2 Attaching a charter 423
13.2.3 Proposing the business service 425
13.3 Reviewing and approving the business service 427
13.3.1 Reviewing the new business service 428
13.3.2 Assigning an owning organization to the business service 430
13.3.3 Approving the business service 430
13.4 Scoping a service version 432
13.4.1 Creating a new service version 433
13.4.2 Proposing the service version scope 436
13.4.3 Approving the service version scope 437
13.5 Planning a service version 439
13.5.1 Loading the WSDL 440
13.5.2 Completing service version details 442
13.5.3 Adding the interface specification relationship 443
13.5.4 Approving the service version 444
13.6 Creating a service level definition 446
13.6.1 Creating a service level definition 446
13.6.2 Adding the service interface 449
13.6.3 Approving the service level definition 450

Archived

13.7 Deploying a service version to staging 452
13.7.1 Adding relationship to provided Web service 453
13.7.2 Updating the service level definition 455
13.7.3 Classifying the endpoint 457
13.7.4 Classifying the WSDL 459
13.7.5 Approving staging deployment 460
13.7.6 Activating the endpoint 460
13.8 Deploying a service version to pre-production 462
13.8.1 Loading the pre-production endpoint WSDL 463
13.8.2 Updating the service level definition 464
13.8.3 Classifying the endpoint 465
13.8.4 Classifying the WSDL 465
13.8.5 Approving pre-production deployment 466
13.8.6 Activating the endpoint 466
13.9 Deploying a service version to production. 468
13.9.1 Loading the production endpoint WSDL 469
13.9.2 Updating the service level definition 469
13.9.3 Classifying the endpoint 471
13.9.4 Classifying the WSDL 471
13.9.5 Approving production deployment. 472
13.9.6 Activating the endpoint 472

Chapter 14. Governing a service that reuses an existing service 475
14.1 Governing a new service at JKHLE. 476
14.2 Creating a business capability. 478
14.2.1 Creating a new business service 478
14.2.2 Attaching a charter 480
14.2.3 Proposing the business service 483
14.3 Reviewing and approving business service 485
14.3.1 Reviewing the new business service 486
14.3.2 Assigning an owning organization to the business service 488
14.3.3 Approving the business service 488
14.4 Scoping a service version 490
14.4.1 Creating a new service version 491
14.4.2 Proposing the service version scope 495
14.4.3 Approving the service version scope 496
14.5 Creating and approving a subscription request. 498
14.5.1 Creating a subscription request 499
14.5.2 Approving the subscription request. 501
14.6 Planning a service version 503
14.6.1 Completing service version details 503
14.6.2 Loading the WSDL 505
14.6.3 Adding the interface specification relationship 506

Archived

14.6.4 Approving the service version 507
14.7 Creating a service level definition 509
14.7.1 Creating the service level definition 509
14.7.2 Adding the service interface 512
14.7.3 Approving the service level definition 513
14.8 Creating a service level agreement 515
14.8.1 Creating the service level agreement 516
14.8.2 Adding the agreed endpoints relationship 519
14.8.3 Approving the service level agreement 520
14.8.4 Activating the service level agreement 521
14.9 Deploying a service version to staging 522
14.9.1 Loading staging endpoint WSDL 524
14.9.2 Adding relationship to the provided Web service 524
14.9.3 Updating the service level definition 526
14.9.4 Classifying the endpoint 529
14.9.5 Approving staging deployment 530
14.9.6 Activating the endpoint 531
14.10 Deploying a service version to pre-production 533
14.10.1 Loading the pre-production endpoint WSDL 534
14.10.2 Updating the service level definition 535
14.10.3 Classifying the endpoint 536
14.10.4 Approving pre-production deployment 537
14.10.5 Activating the endpoint 537
14.11 Deploying a service version to production 539
14.11.1 Loading the production endpoint WSDL 540
14.11.2 Updating the service level definition 541
14.11.3 Classifying the endpoint 542
14.11.4 Approving production deployment 543
14.11.5 Activating the endpoint 543

Chapter 15. Governing a minor upgrade 545
15.1 Governing a minor upgrade to a service 546
15.2 Scoping a new service version 548
15.2.1 Creating a new service version 548
15.2.2 Approving the service version scope 549
15.3 Creating and approving a subscription request 551
15.3.1 Creating a subscription request 552
15.3.2 Approving the subscription request 553
15.4 Planning a service version 555
15.4.1 Completing service version details 555
15.4.2 Loading the WSDL 556
15.4.3 Adding the interface specification relationship 556
15.4.4 Approving the service version 557

Archived

15.5 Creating a service level definition 559
15.5.1 Creating the service level definition 559
15.5.2 Adding the service interface 560
15.5.3 Approving the service level definition 560
15.5.4 Creating a compatible service level definition relationship 561
15.6 Creating a service level agreement 563
15.6.1 Creating the SLA 564
15.6.2 Adding the agreed endpoints relationship 565
15.6.3 Approving the service level agreement 565
15.6.4 Activating the service level agreement 566
15.7 Deploying the upgrade version to staging 568
15.7.1 Loading the endpoint WSDL 568
15.7.2 Adding a relationship to the provided Web service 569
15.7.3 Updating the service level definition 569
15.7.4 Classifying the endpoint 570
15.7.5 Approving staging deployment 571
15.7.6 Activating the staging endpoint 571
15.8 Deploying the upgrade version to pre-production 573
15.8.1 Loading the pre-production endpoint WSDL 573
15.8.2 Updating the service level definition 574
15.8.3 Classifying the endpoint 575
15.8.4 Approving pre-production deployment 575
15.8.5 Activating the endpoint 576
15.9 Deploying the upgrade version to production 578
15.9.1 Loading the production endpoint WSDL 578
15.9.2 Updating the service level definition 579
15.9.3 Classifying the endpoint 580
15.9.4 Approving production deployment 580
15.9.5 Activating the endpoint 581
15.9.6 Superceding the v1.0 service version 582
15.9.7 Revoking the V1_0 staging endpoint 583
15.9.8 Revoking the V1_0 pre-production endpoint 585
15.9.9 Revoking the V1_0 production endpoint 586

Chapter 16. Governing a business process 589
16.1 Governing a new business process 590
16.2 Creating a business capability 592
16.2.1 Creating a new business capability 592
16.2.2 Attaching a charter 593
16.2.3 Proposing the business process 594
16.3 Reviewing and approving business process 595
16.3.1 Reviewing the new business process 596
16.3.2 Assigning an owning organization to the business process 597

Archived

16.3.3 Approving the business process 597
16.4 Scoping a process version 599
16.4.1 Creating a new capability version 599
16.4.2 Proposing the process version scope 600
16.4.3 Approving the process version scope 601
16.5 Creating and approving a subscription request 603
16.5.1 Creating a subscription request 603
16.5.2 Approving the subscription request 604
16.6 Planning a process version 607
16.6.1 Completing process version details 607
16.6.2 Loading the WSDL 608
16.6.3 Adding the interface specification relationship 609
16.6.4 Approving the process version 609

Part 4. Appendixes 611

Appendix A. IBM governance enabling tools 613
IBM software for build time SOA Governance automation 614
IBM Rational Method Composer 614
IBM WebSphere Business Glossary 614
IBM Rational System Architect 615
IBM Rational RequisitePro and RequisitePro Composer 615
IBM Rational Software Architect 615
IBM Rational Application Developer 616
IBM WebSphere Integration Developer 616
IBM Rational Asset Manager 616
IBM Rational Team Concert 616
IBM Rational ClearCase 616
IBM Rational Tester for SOA Quality and
IBM Rational Performance Tester Extension for SOA Quality 617
IBM Rational Clearcase and Buildforge 617
IBM WebSphere Service Registry and Repository 617
IBM software for runtime SOA Governance automation 617
IBM WebSphere Service Registry and Repository 618
IBM Tivoli Change and Configuration Management Database 618
IBM Tivoli Composite Application Manager for SOA 618
IBM Tivoli Security Policy Manager 618
IBM DataPower SOA Appliances 619
IBM WebSphere Message Broker and IBM WebSphere Enterprise Service
Bus 619

Appendix B. Additional material 621
Locating the Web material 621

Archived

Abbreviations and acronyms 623

Related publications 625
IBM Redbooks publications 625
How to get IBM Redbooks 625
Help from IBM 626

Archived

Archived

Notices

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

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

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:
IBM Director of Licensing, IBM Corporation, North Castle Drive Armonk, NY 10504-1785 U.S.A.

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

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

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

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

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

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

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

© Copyright IBM Corp. 2009. 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:

ClearCase®	IBM®	RequisitePro®
ClearQuest®	Rational Team Concert™	Tivoli®
DataPower®	Rational®	VisualAge®
DB2®	Redbooks®	WebSphere®
developerWorks®	Redpapers™	z/OS®
Global Business Services®	Redbooks (logo) ®	

The following terms are trademarks of other companies:

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

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.

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.

Archived

Preface

IBM® WebSphere® Service Registry and Repository (WSRR) provides service registry and repository functions for service-oriented architecture (SOA) enterprise applications. This IBM Redbooks® publication uses business scenarios to illustrate SOA governance using WSRR as the authoritative registry and repository.

We divided this book into the following parts:

- Part one of this book introduces SOA and service governance, provides a technical overview of WSRR, and describes the WSRR governance enablement profile.
- Part two of this book provides step-by-step guidance to building WSRR solutions. This part addresses topics such as modeling in WSRR Studio, creating a WSRR user interface, security, promotion, policies, and reports.
- Part three describes a series of common usage scenarios and describes step-by-step how to implement them in WSRR. These scenarios describe how to govern schemas, existing services, new services, upgrades to services, and business processes.

This book is based on WebSphere Service Registry and Repository V6.3 and is intended for IT architects and IT specialists looking to get a better understanding of how to achieve service life cycle governance.

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), Raleigh Center.

Nicole Hargrove is a Master Certified Senior IT Specialist on the East WebSphere Extended Technical team based in Raleigh, N.C. She has 14 years of experience in infrastructure and application design and deployment and has worked at IBM for 12 years. Her areas of expertise include WebSphere Service Registry and Repository, WebSphere Portal, and WebSphere Application Server, on distributed as well as mainframe platforms, and Rational® Application Developer. She holds a BS with a double major in Accounting and Information Systems from Old Dominion University and an MS in Advanced Technology and eCommerce from Johns Hopkins University. She co-authored the IBM Redbooks

© Copyright IBM Corp. 2009. All rights reserved.

publications regarding WebSphere Version 5 Application Development Handbook and WebSphere Portal v6 on z/OS®.

Ian Heritage is the Solution Test lead for the WebSphere Service Registry and Repository development team based at the IBM Hursley Development Lab in the U.K. Prior to this position, Ian led the Level 3 service team for WSRR and the WSRR Functional Test team. Ian spent the first 5 years of his IBM career working on WebSphere Voice Response and Unified Messaging for WebSphere Voice Response. Ian has worked closely with customers to implement a WSRR solution that fits into their business processes. Ian began his career with IBM in 2000 after graduating from the University of Hertfordshire with an MSc in Computer Science and a BSc in Psychology. He is an ISEB Certified Test Practitioner.

Prasad Imandi is a Senior Software Engineer working in Level 2 service of IBM AIM organization in RTP, N.C. He has 14 years of experience working in multiple WebSphere products at IBM. He is currently focused on WebSphere Message Broker, WebSphere Business Events, and WSRR products and leads product service teams. Prasad works with customers worldwide and focuses on product integration and troubleshooting issues. He holds a Masters degree in Computer Science and Engineering from Jadavpur University, India. Prasad contributed to the IBM Redbooks publication *Getting Started with WebSphere Enterprise Service Bus V6*, SG24-7212.

Martin Keen is a Consulting IT Specialist at the ITSO, Raleigh Center. He writes extensively about WebSphere products and SOA. He also teaches IBM classes worldwide about WebSphere, SOA, and ESB. Before joining the ITSO, Martin worked in the EMEA WebSphere Lab Services team in Hursley, U.K. Martin holds a bachelor's degree in Computer Studies from Southampton Institute of Higher Education.

Wendy Neave works as an IT Architect in IBM Global Services in Australia. She has 20 years of experience in application design and development and is an IBM Certified IT Specialist. She holds a Bachelor of Education (Environmental Science) and an Associate Diploma in Computing. Over the last 9 years, Wendy has worked primarily on Web and SOA-based applications. Wendy contributed to the IBM Redbooks publications *Patterns: Direct Connections for Intra- and Inter-enterprise*, SG24-6933, *BPEL4WS Business Processes with WebSphere Business Integration: Understanding, Modeling, Migrating*, SG24-6381, and *Patterns: Extended Enterprise SOA and Web Services*, SG24-7135.

Laura Olson is an IBM Certified Consulting IT Specialist. Laura specializes in SOA Governance and has developed numerous SOA solutions involving federation of registries and repositories in heterogeneous environments. She has published multiple articles about SOA on developerWorks®. Laura contributed to

the IBM Redbooks publication *Building SOA Solutions Using the Rational SDP*, SG24-7356.

Bhargav Perepa is a WebSphere IT Specialist in IBM Federal Software Group in the Washington, D.C. area, U.S. He has 15 years of experience in application and system software development at IBM. Bhargav holds a Masters degree in Computer Sciences from IIT, Chicago, and an MBA degree from the University of Texas, Texas. His areas of expertise include solution building, development, implementation, and skill transfer of WebSphere Application Server, Business Activity Monitoring (BAM), SOA, Web Services, and Service Governance technologies. Prior to his current IBM role, Bhargav worked on development of C++ middleware product Component Broker (CB), Java™ middleware product WebSphere Application Server, Fault Tolerant CORBA (FT CORBA) Object Request Broker (ORB), and ORB interoperability efforts at IBM Austin Research Laboratory. He has worked on IBM VisualAge®, ENFIN Synchronicity, Digitalk Smalltalk, and IBM VisualAge C++ efforts at IBM Chicago, previously. Bhargav is an IBM developerWorks contributing author and has written several IBM developerWorks articles in his areas of expertise.

Andrew White is a software developer in the U.K. He has 3 years of experience in middleware development at IBM. Andrew holds a degree in Computer Science from The University of Nottingham. His areas of expertise include WebSphere Application Server, WebSphere Service Registry and Repository, WebSphere DataPower®, WebSphere Process Server, Tivoli® Composite Application Manager for Service-Oriented Architecture (ITCAM4SOA), Tivoli Security Policy Manager), SOA, and Microsoft® .net.

Figure 1 The team (from left to right): Ian, Andrew, Wendy, Martin, Bhargav, Prasad, Laura, and Nicole

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

- Nicola Hills
- Martin Smithson
- Arnauld Desprets
- Michael Ellis
- John Falkl
- Greg Flurry
- Subhash Kumar
- Robert Laird
- Matthew Oberlin
- Martin Rowe
- Andre Tost
- Ewan Withers
- Mark Allman
- Andrew Borley
- Gary Thornton
- Phil Rowley

Become a published author

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

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

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

ibm.com/redbooks/residencies.html

Archived

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 e-mail to:

  redbooks@us.ibm.com

- Mail your comments to:

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

Archived

Archived

Part 1

Overview

Archived

© Copyright IBM Corp. 2009. All rights reserved.

Archived

1

SOA and service governance overview

This chapter provides an overview of service-oriented architecture (SOA) and service governance. It includes the following sections:

- What is SOA governance?
- SOA governance and service governance
- The IBM SOA Governance and Management Method
- Requirements to enable service governance
- WSRR as an enabler of service governance

© Copyright IBM Corp. 2009. All rights reserved.

1.1 What is SOA governance?

Service-oriented architecture (SOA) is a compelling technique for developing software applications that best align with business models. However, SOA increases the level of cooperation and coordination that is required between business and information technology (IT), as well as among IT departments and teams. This cooperation and coordination is provided by *SOA governance*, which covers the tasks and processes for specifying and managing how services and SOA applications are supported.

Governance is the means of establishing and enforcing how a group agrees to work together. Specifically, there are two aspects to governance:

- Establishing chains of responsibility, authority, and communication to empower people by determining who has the rights to make what decisions.
- Establishing measurement, policy, and control mechanisms to enable people to carry out their roles and responsibilities

While governance determines who has the authority and responsibility for making the decisions, *management* is the process of making and implementing the decisions. To put it another way, governance says what should be done, while management makes sure that it gets done.

IT governance is about who is responsible for what in an IT department and how the department knows that the responsibilities are being preformed. Specifically, IT governance establishes:

- Decision making rights that are associated with IT
- Mechanisms and policies that are used to measure and control the way IT decisions are made and carried out

SOA adds the following unique aspects to governance:

- Acts as an extension of IT governance that focuses on the life cycle of services to ensure the business value of SOA.
- Determines who should monitor, define, and authorize changes to existing services within an enterprise.

Archived

SOA governance is the intersection of business and IT governance. It focuses on the life cycle of services to ensure the business value of SOA. SOA governance is the effective management of this life cycle, which is the key goal to SOA governance. Figure 1-1 illustrates this concept.

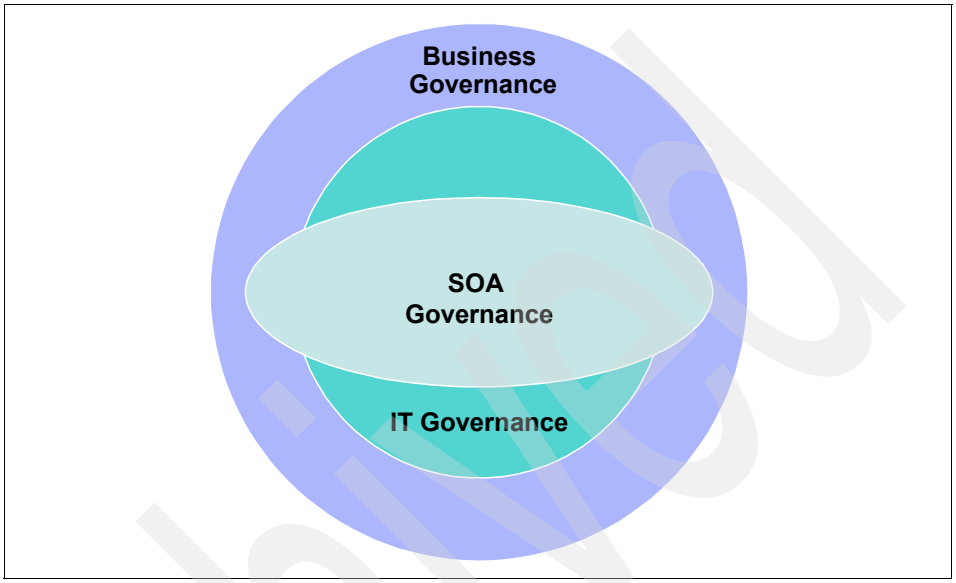


Figure 1-1 SOA governance in relation to business and IT governance

Governance becomes more important in SOA than in general IT. In an SOA, consumer and provider services can be developed, run, and managed by different departments. Working together successfully requires complex coordination. For SOA to succeed, multiple applications need to share common services, which means they need to coordinate on making those services common and reusable. Governance issues are more complex than in the days of monolithic applications or even in the days of reusable code and components.

As companies use SOA to better align IT with the business, they can also ideally improve overall IT governance. Employing SOA governance is key for companies to realize the benefits of SOA. For SOA to be successful, SOA business and technical governance is not optional, it is essential.

Automation is critical for successful governance. No matter what level of regulation the enterprise decides to have, automation will make it easier to deliver high-quality services on time and on budget. Those who are subject to governance will welcome a governance process that gives near real-time feedback on what must change. It saves time and effort and helps build in quality.

IBM has a wide range of products to enable and automate SOA governance throughout the SOA life cycle. For example, IBM WebSphere Service Registry and Repository focuses on service governance. For full list of products and the life cycle that they support, see Appendix A, “IBM governance enabling tools” on page 613.

Getting to an acceptable governance maturity level does not happen by accident. An effective and evolving governance framework must be intentional and focused. It requires leadership. It must define clear roles and responsibilities. It must enable well-thought-out and consistently implemented policies and procedures.

Along with the products to automate SOA governance, IBM developed the SOA Governance and Management Method (SGMM), which is a proven practice for implementing SOA governance. You can find more information about this practice in 1.3, “The IBM SOA Governance and Management Method” on page 7.

Note: This book focuses on how WebSphere Service Registry and Repository enables service governance. For an in depth view of SOA governance, see *SOA Governance: Achieving and Sustaining Business and IT Agility*, ISBN 0137147465.

1.2 SOA governance and service governance

SOA governance itself is a very broad subject. It covers everything from establishing the organizations and policies, to governing the life cycle of the services and tracking measurements to ensure the success of the SOA. Due to this broad definition of SOA governance, SOA governance and service governance tend to be used interchangeably. However, there is a fundamental difference between the two concepts, as illustrated in Figure 1-2.

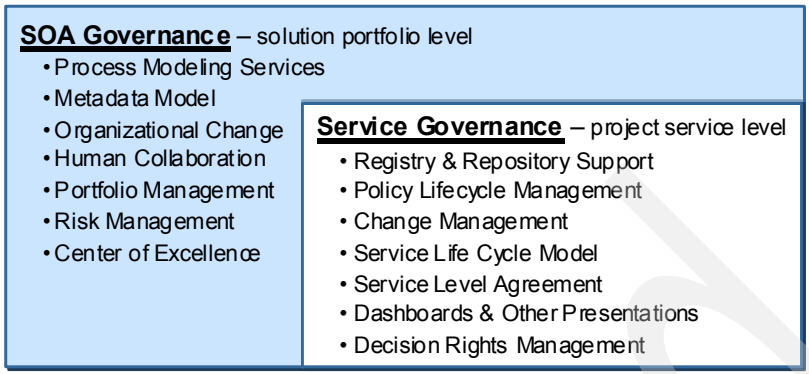


Figure 1-2 Service governance is a subset of SOA governance

SOA governance is the catalyst, or *intersection*, between business governance and IT governance that ensures the business value of SOA. It also focuses on managing the portfolio of service needs and identifying where to make investments in service orientation, as well as the ability to track those investments. Within SOA governance, the business has the ability to establish and measure KPIs that track the success of the SOA investment, such as return on investment and business value.

Service governance defines a set of processes for identifying, building, deploying, and managing services. Service governance stays at the project level to provide architectural standards, guidelines, and processes that manage the service life cycle. Service governance is a subset of SOA governance, where SOA governance is more concerned with governing the overall solutions to align with the business objectives. Service governance is more closely aligned with IT governance in how SOA based services are constructed and maintained but is done in the bigger context of managing the overall business requirements

1.3 The IBM SOA Governance and Management Method

The IBM SOA Governance and Management Method (SGMM) describes, in detail, leading practices for implementing SOA governance and its supporting mechanism and processes. SGMM is available as a plug-in with IBM Rational Method Composer and is also a service offering from IBM Global Business Services®.

SGMM denotes specific SOA Governance domains and capabilities that enable IBM to assess an enterprise's current governance maturity and then create an SOA governance heat map. The heat map in turn guides the usage of SGMM

assets, including checklists, guidance, leading practices, processes, and procedures for the definition of a transition plan that creates good SOA governance for the enterprise. IBM with SGMM clearly guides the enterprise on the SOA journey, with SOA governance at the correct level given the current maturity of the organization.

Figure 1-3 illustrates the basic methology of the SGMM process.

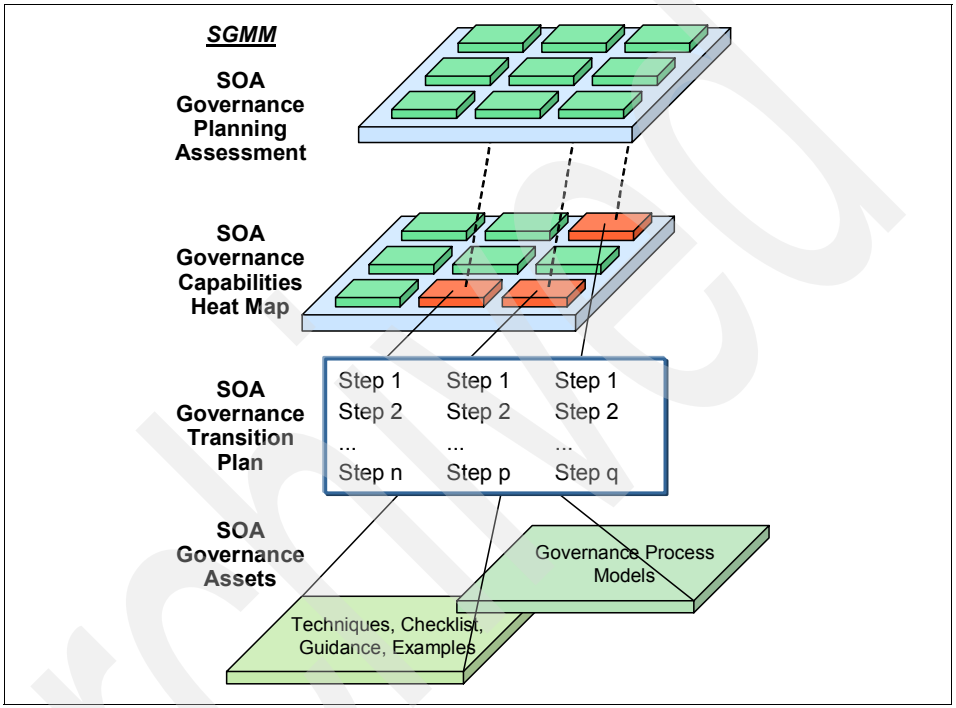


Figure 1-3 Basic SGMM process

The SGMM process shown in Figure 1-3 consists of:

- SOA Governance Planning Assessment

  Create a governance baseline by assessing the governance maturity levels. Identify based on current maturity, desired maturity, and prioritization discussions, which SOA Governance capabilities need to be addressed next.

- SOA Governance Capabilities Heat Map

  Based on the planning assessment, create an SOA Governance heat map using the 26 governance capabilities that are required to effectively control SOA.

Archived

- SOA Governance Transition Plan

  Access each SOA Governance Capability and place the steps for that capability from SGMM in the transition plan.

- SOA Governance Assets

  Guidance, checklist, techniques and examples documents give detailed instructions on how to perform a specific governance task. Governance Process Models are leading practice models for a particular governance task that should be used as a starting point and then modified to fit a particular enterprise.

The SGMM domains and capabilities are an extension of the IT governance Control Objectives for information and Related Technology (COBIT) domains and capabilities. For more information about COBIT, see:

http://www.isaca.org/cobit

The COBIT domains and capabilities are modified for SOA as informed by the Open Group Service Integration Maturity Model (OSIMM). IBM contributed the OSIMM to The Open Group Architecture Forum (TOGAF) as an industry standard for SOA maturity.

Figure 1-4 shows an SGMM capability heat map with the four domains and their capabilities. This book focuses mostly on the capabilities in the service development domain.

Archived

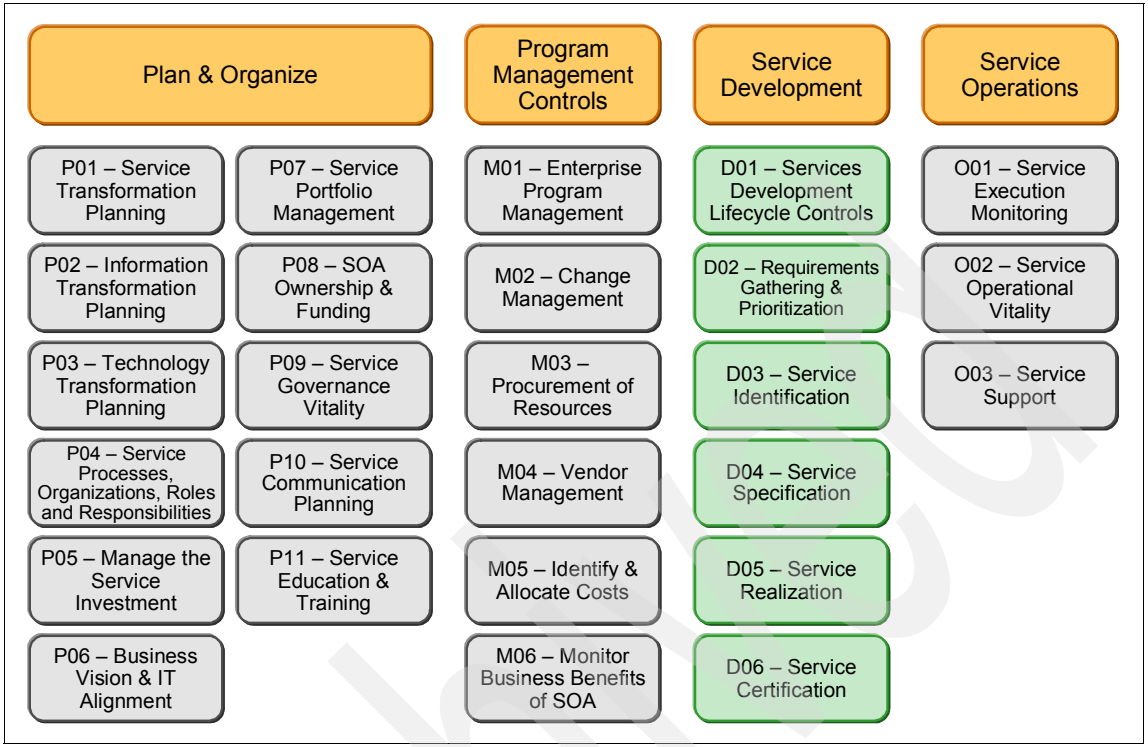


Figure 1-4 SGMM governance capabilities map

1.3.1 Plan and organize

This domain covers the strategy and tactics for planning how to govern an SOA. It is concerned with aligning IT and business objectives and focuses on architectural improvements to the application portfolio by creating the correct set of functional and information services that help optimize business processes and reusability. This domain identifies SOA standards and policies and enforces those standards using the correct set of organizations, roles and responsibilities, and governance processes. This domain requires that an organization plan the strategic vision with 2-way communication between those who own the SOA program and the business and IT stakeholders.

1.3.2 Program management controls

This domain covers the ability of the enterprise to consistently manage for quality of service throughout the enterprise. This domain requires a true enterprise program management capability (as opposed to just project management).

Program management controls can identify and manage the results of SOA program change, whether services are developed internally or externally or acquired. These controls manage the financial aspects of the SOA journey properly, which includes identifying and allocating shared costs, monitoring the actual benefits, and working with centralized control points such as procurement.

1.3.3 Service development

This domain covers the ability to create or modify services in a quality manner by governing the service development life cycle. The activities in this domain govern the design, development, testing, and certification of individual services and automated business processes. Governance of these activities is critical to ensuring the functional and technical quality of all services and automated business processes that the enterprise produces. The key result of this domain is to find the correct level of governance control points for the service development life cycle that matches the enterprise culture yet results in quality services.

1.3.4 Service operations

This domain covers the disciplines that are necessary to protect the integrity of the production environment for services. IT operations professionals need to ensure that the services are deployed and managed as efficiently as possible and that the integrity of the operational environment is maintained. These processes must be assessed regularly for quality and compliance with control requirements. This domain requires addressing performance management issues, which include whether IT can detect and address issues in a timely manner, maintain service level agreements (SLAs), as well as ensure that governance controls are effective and efficient and that adequate security controls are in place and enforced.

1.4 Requirements to enable service governance

In this section, we discuss the following capabilities that a registry and repository must have in order to enable service governance:

- Govern both the service consumer and service provider
- Manage service contracts and SLAs
- Manage multiple service versions within life cycle states
- Govern all types of services
- Manage service metadata and artifacts
- Govern and enforce service design and run time policies
- Integrate with runtime environments

- Find and publish services, metadata, and artifacts
- Discover and identify services in a runtime environment
- Communication, reporting, and management of change
- Integrate with other products supporting SOA governance

1.4.1 Govern both the service consumer and service provider

The service consumer and service provider typically have their own development life cycles and iterate at different paces. Governance of both the consumer and provider is an imperative. For example, what happens if the service consumer misses a deadline? How will that affect the provisioning of the provider service? How will that affect which version of the provider service should be used? What happens if the service provider misses a deadline? If either the service provider or the service consumer missing a deadline can impact the enterprise, from development to operations. How do you manage consumer versions? When a new version of the consumer is developed how do you determine which version of the provider service the new version should use?

1.4.2 Manage service contracts and SLAs

A *service level agreement* (SLA) is the negotiated agreement between the service consumer and service provider. The SLA can be a legally binding formal contract or an informal contract. An SLA will capture information such as a service's expected availability and performance. The service consumer's expected usage of the service. An SLA will also records things such as roles and responsibilities between the consumer and provider. For example, capturing who will take responsibility when service becomes unavailable, what is the provider's course of action to bring the service back online and the notification procedure.

1.4.3 Manage multiple service versions within life cycle states

A *service* is considered an evolutionary continuum of service specifications, both compatible and incompatible and the implementations of those service specifications. Many service versions can exist over the lifetime of a service.

The lifetime of a service version starts when a business need is identified for the characteristics that are defined by that version and ends when that business need ends for that version. For business and technical reasons, multiple service versions can exist at any given point in a service lifetime. Thus, a service version has a life cycle that is independent from, but related to, the life cycle of a service and other service versions.

Figure 1-5 illustrates the relationship between the service lifetime and the service versions lifetime.

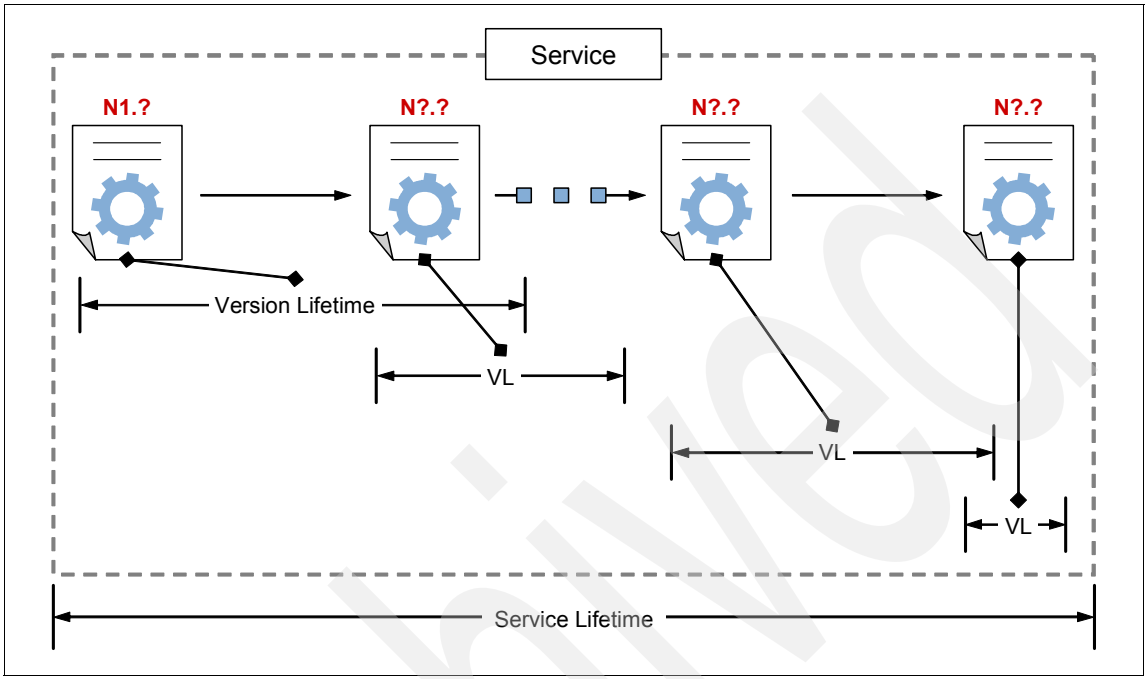


Figure 1-5 Service and service version lifetime

1.4.4 Govern all types of services

SOA does not equal Web services. Services in SOA come in many different types, such as traditional Web services, REST, EJB, JMS, MQ, and COBOL services. Typically, an SOA consists of a variety of these service types. In fact, the two most common SOA services in the field today are Web services and MQ-based services.

1.4.5 Manage service metadata and artifacts

To provide the governance for a service's life cycle, you need a registry that manages metadata about the service artifacts and that socializes the service, as well as a repository that owns the artifacts of the service. A registry and repository in which both the registry and repository capabilities are provided in one component is the ultimate solution. This capability gives you a trusted source for SOA artifacts, because the artifacts cannot be changed or moved without the registry's knowledge.

1.4.6 Govern and enforce service design and run time policies

At its most basic form, a *policy* is a requirement or a capability. A *policy-driven approach* to an SOA offers a way to represent a variety of requirements, including business, technical, and operational requirements that might have been captured previously into expressions that can be interpreted and acted upon throughout the life cycle of a service.

Design-time policies

You can transform policies that are authored when you design an SOA governance process into executable policies where the registry and repository ensure that the enterprise's SOA standards are met. Examples of these types of policies include:

- Funding must be determined before a service can be approved.
- All WSDLs must conform to the WS-Interoperability standard.
- Before a service becomes operational, it must have the authorization token WS-Security policy attached.

Runtime policies

It is important to manage the life cycle of a policy along with the services to which they are attached. Lack of visibility about policies and their impact if the policies changed can lead to system outages and poor or no policy enforcement.

1.4.7 Integrate with runtime environments

Inflexibility in your SOA environment can cause missed business opportunities and higher costs due to redevelopment, retesting and redeployment every time a service, service location, or runtime policies changes. You need to optimize the registry and repository for the demanding runtime environment. This optimization allows run times, such as an Enterprise Service Bus (ESB), to access the registry for tasks such as dynamic endpoint selection, policy retrieval and enforcement, and contract validation.

Archived

Figure 1-6 illustrates a dynamic endpoint selection from WSRR.

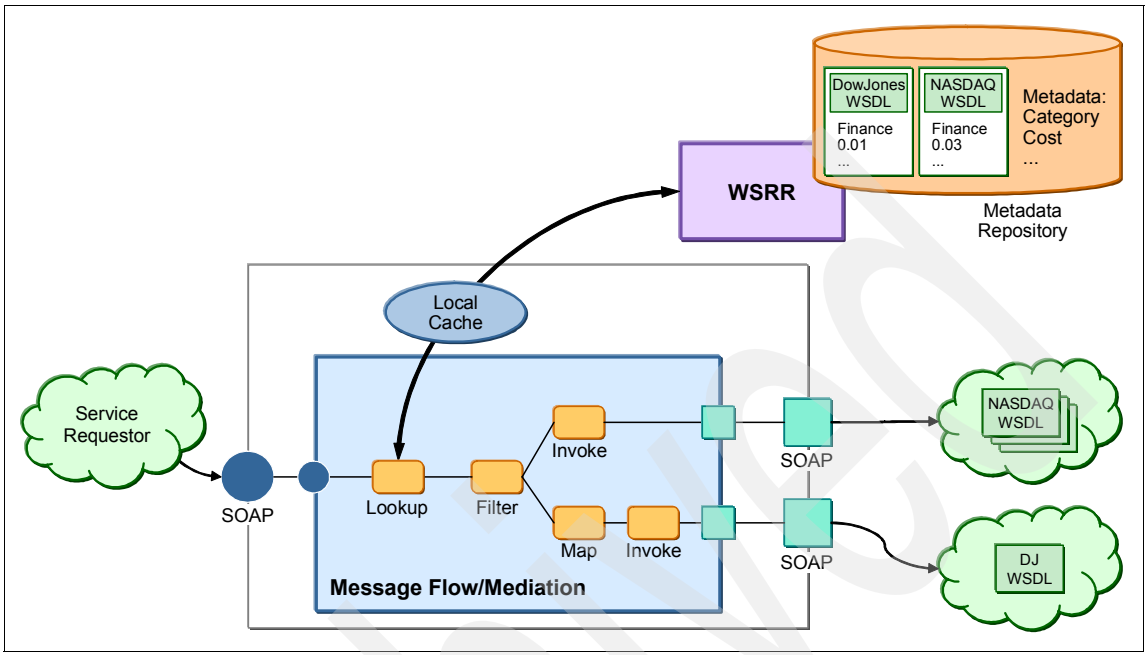


Figure 1-6 Dynamic endpoint selection and transformation from a runtime environment

1.4.8 Find and publish services, metadata, and artifacts

The ability to find and publish service artifacts and metadata using multiple avenues is a must in heterogeneous SOA environments. A registry and repository needs plug-ins for the different developer environments (such as Microsoft Visual Studio and Eclipse), a Web user interface for administration, and multiple exposed APIs, such as REST, SOAP, Java, and UDDI for ease of integration with other design and runtime environments.

1.4.9 Discover and identify services in a runtime environment

To gain control of the services an enterprise, an enterprise can use an automatic discovery capability to discover services that are deployed on the enterprise's various runtime environments and can publish the services with the registry and repository. You can also use this capability to identify rogue services, which are services that are deployed in an environment without going through the proper governance processes.

1.4.10 Communication, reporting, and management of change

For an SOA to be successful, you need executive sponsorship and support. An important part of maintaining this sponsorship is the communication of the health and the value of the SOA. A registry and repository solution must have the ability to create robust reports for championing SOA to executives. Reports are also important for service planning and operational aspects.

Change will happen, services will be versioned, policies will be changed, and SLAs will expire. When change happens, the enterprise needs to assess the impact of these changes and then communicate these events to all parties involved. These change events might need to kick off other processes. A registry and repository needs to have impact analysis capabilities and a notification system that you can configure to support the required types of communications.

1.4.11 Integrate with other products supporting SOA governance

A service registry and repository provides a place to organize, manage, find, and govern the service descriptions. In heterogeneous deployment environments that are typical in an SOA, the service registry and repository provides a standard, interoperable means to access, query, and manipulate the service descriptions.

The service registry and repository plays a central role throughout the SOA life cycle. Therefore, it is important to note that it be integrated with applications from the service development life cycle, change and release management, service run times, service management (operational efficiency and resilience), and other service registries and repositories. Figure 1-7 shows these integration points.

Archived

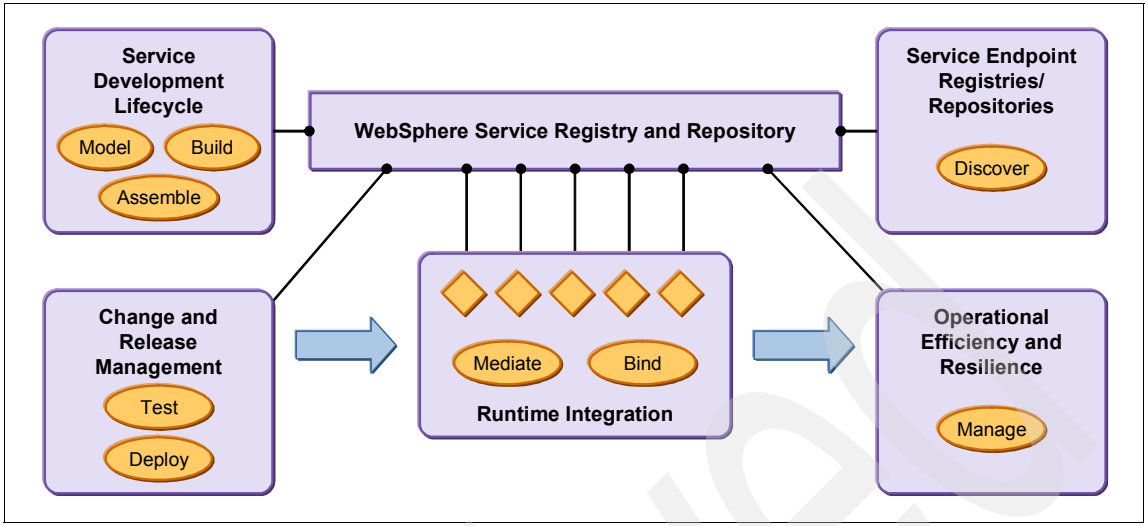


Figure 1-7 WebSphere Service Registry and Repository integration throughout the SOA life cycle

1.5 WSRR as an enabler of service governance

WSRR is an enabler of service governance. It serves as the authoritative source for SOA services and artifacts throughout the life cycle, from model to assemble, deploy, manage, and eventually retirement. WSRR is an *integrated registry and repository*, which means that it provides both registry and repository capabilities in one component. WSRR is optimized for the runtime environment to enable dynamic endpoint selection and retrieval policies and to ensure that the proper subscriptions are in place to invoke a service.

To fulfill service governance needs, a registry and repository must support five capability areas. Figure 1-8 maps these five capability areas with the service governance capabilities, which we discussed in 1.4, “Requirements to enable service governance” on page 11.

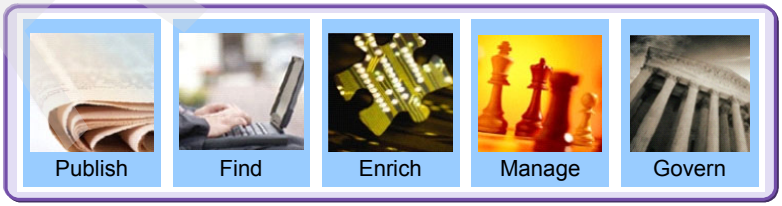


Figure 1-8 Registry and repository capability areas

Archived

The capabilities can be mapped to multiple capabilities areas.

- Publish and find
  - Discover and identify services in a runtime environment
  - Find and publish services, metadata, and artifacts
  - Manage service metadata and artifacts
  - Integrate with other products supporting SOA governance
- Enrich
  - Integrate with runtime environments
  - Govern and enforce service design and run time policies
  - Integrate with other products that support SOA governance
- Manage
  - Communication, reporting, and management of change
  - Manage service contracts and SLAs
  - Manage multiple service versions within life cycle states
  - Integrate with other products that support SOA governance
- Govern
  - Govern all types of services
  - Govern both the service consumer and service provider
  - Govern and enforce service design and run time policies
  - Manage multiple service versions within life cycle states
  - Integrate with other products that support SOA governance

Figure 1-9 shows how the capability areas map to the SOA life cycle stages and the value propositions that they support. The next sections outline the value proposition that WebSphere Service Registry and Repository provides and how WebSphere Service Registry and Repository supports service governance needs.

Archived

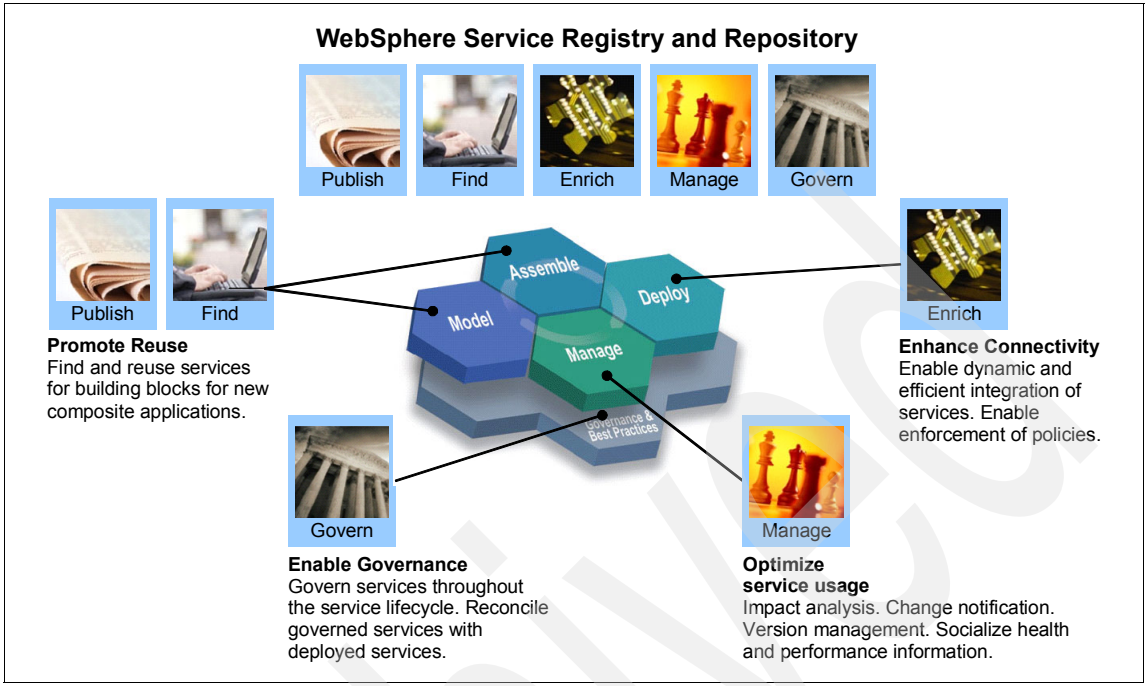
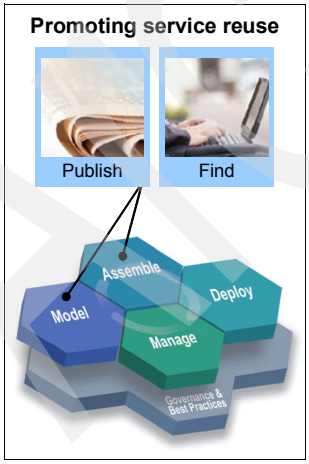


Figure 1-9 WSRR provides value throughout the SOA life cycle

1.5.1 Promoting service reuse



A common issue within an SOA environment is achieving the proper level of service reuse. Most often this issue is tied to an inability to find the available services, either because of a lack of a registry and repository or the inability to find the service in an easily defined way in an existing registry.

WSRR provides multiple ways to publish and find services and metadata in the service registry. WSRR can manage all types of services, from Web services to REST, EJB, JMS, MQ, and COBOL services. WSRR has a Web user interface in which you can publish services, metadata, and supporting documentation. Business services are proposed and approved using a wizard that enforces the metadata model constraints that are defined by an enterprise to represent a business service and other entities in the registry, as shown in Figure 1-10.

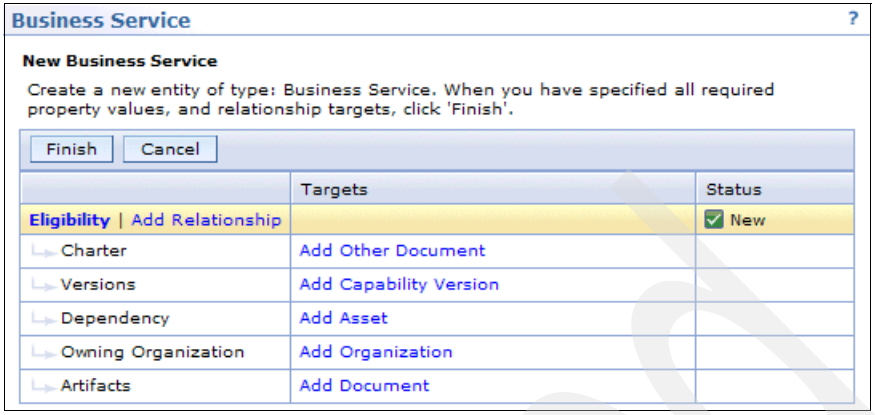


Figure 1-10 Business services

Figure 1-11 shows how you can load various types of documents (such as WSDL, XSD, XML, Policy, binary documents, SCA integration modules, and compressed or JAR files) into WSRR and then annotate these files with a namespace, description, and document version.

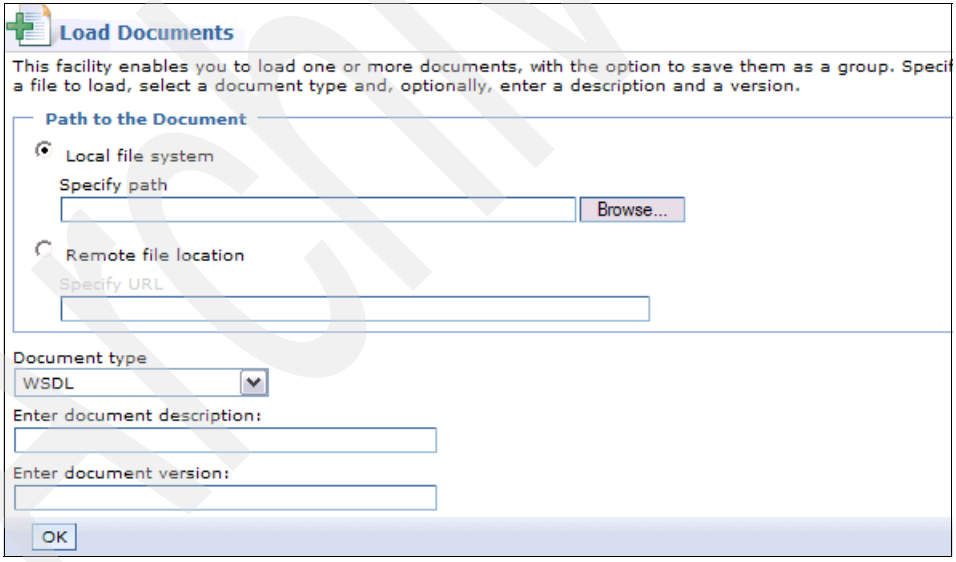


Figure 1-11 Publish documents to the repository

Archived

The WSRR Web user interface provides robust search and filtering capabilities, which enables users to find services in their own unique ways, leading to faster and more accurate findings. Figure 1-12 shows the WSRR automatic suggestion search capability in which the search suggests content based on the type in the search box.

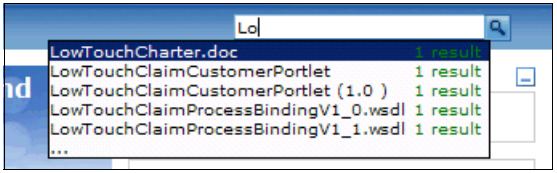


Figure 1-12 Auto complete search

WSRR provides filtering capabilities in which users can drill down using classifications that were applied to the items in the registry. Users can apply and remove filters to adjust search results and can use different filter routes to reach a particular service or content based on their roles. For example, a business analyst can use the business classifications, such as finance or human resources, to reach a service whereas a developer can use more technical classification, such as SOAP or MQ. The user can then save the filtered search to use again later.

Figure 1-13 shows filtering to retrieve a service endpoint.

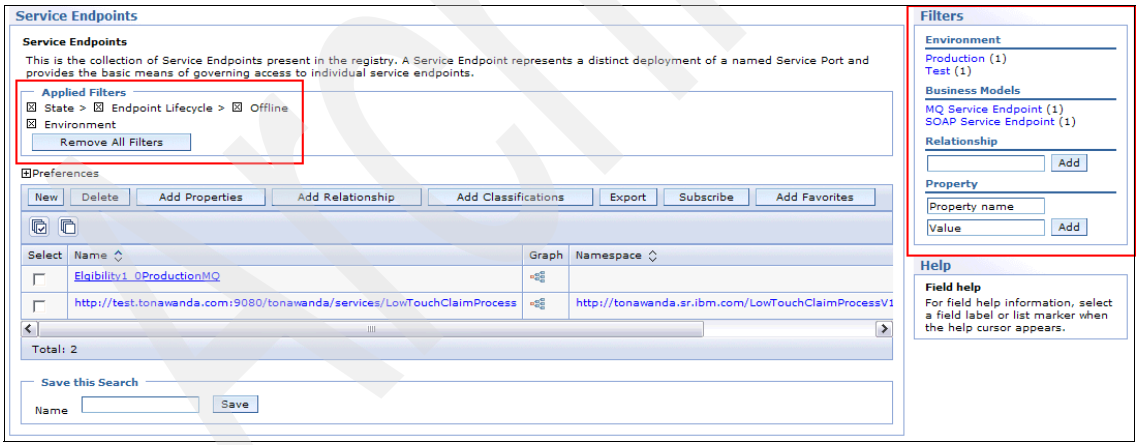


Figure 1-13 Filtered search of an endpoint

WSRR provides a rich query wizard to build and save queries for future use, as shown in Figure 1-14.

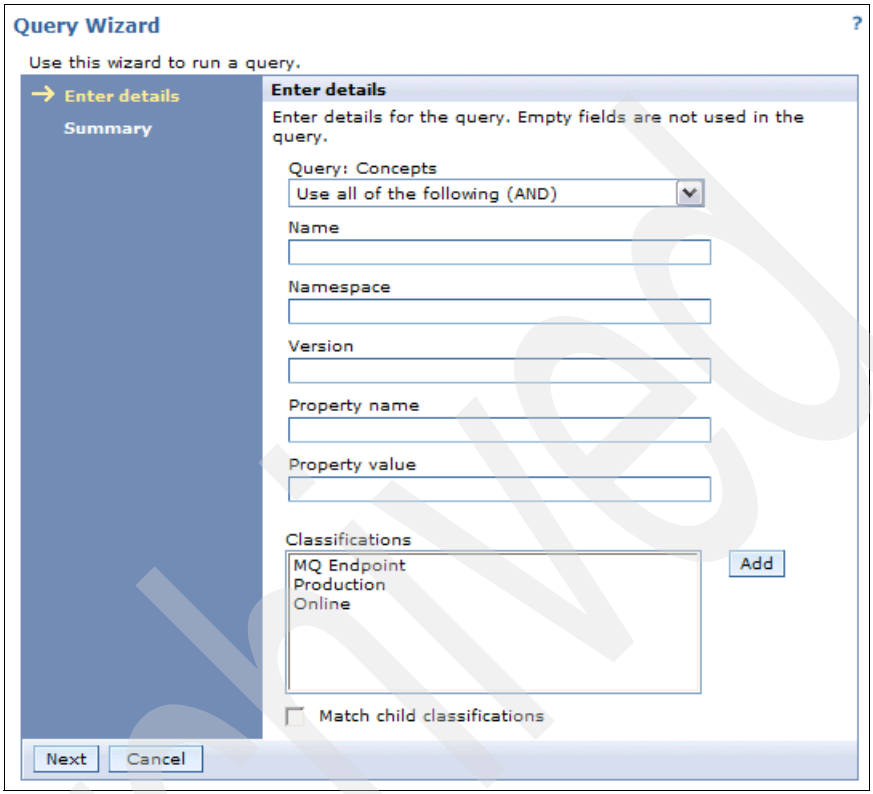


Figure 1-14 WSRR Query Wizard

Archived

WSRR Studio enables the discovery and publishing of services, metadata, and supporting documents, as shown in Figure 1-15.

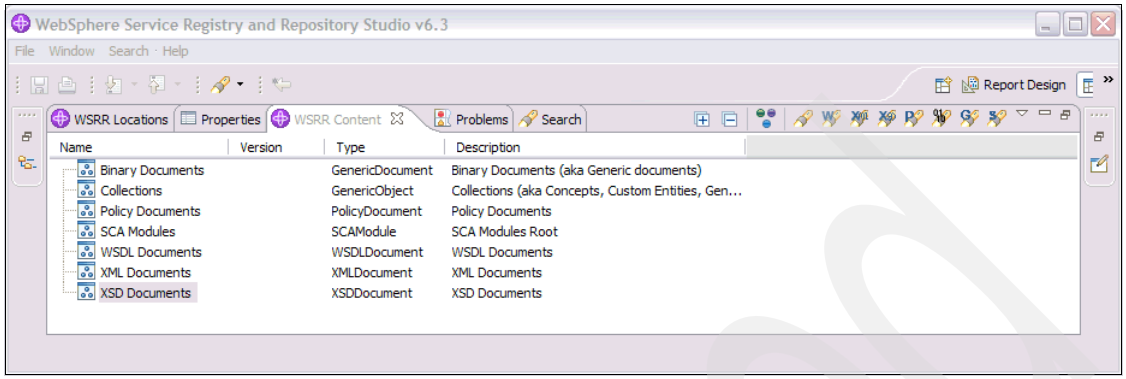


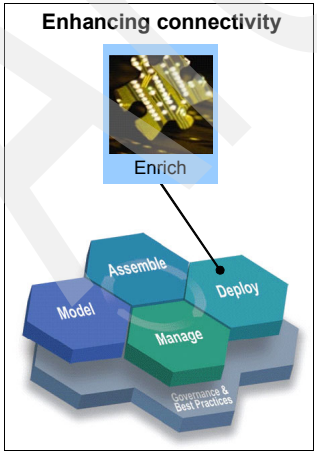
Figure 1-15 WSRR Studio

In addition, WSRR has a service discovery mechanism, which we describe in 1.5.4, “Enable governance” on page 35.

WSRR provides multiple APIs from which you can publish and find services, metadata, and supporting documents. Regardless of the environment, you can use one of the following APIs to work with content in WSRR:

- Java
- SOAP
- REST
- UDDI v3

1.5.2 Enhancing connectivity



IT flexibility provides an enterprise with the ability to avoid missed business opportunities and to lower cost.

IT flexibility is the ability to influence the runtime environment without having to redeploy an application or a mediation, which can be time consuming and costly. WSRR provides enhanced runtime connectivity using dynamic endpoint selection and policy resolution and enforcement, as illustrated in Figure 1-16.

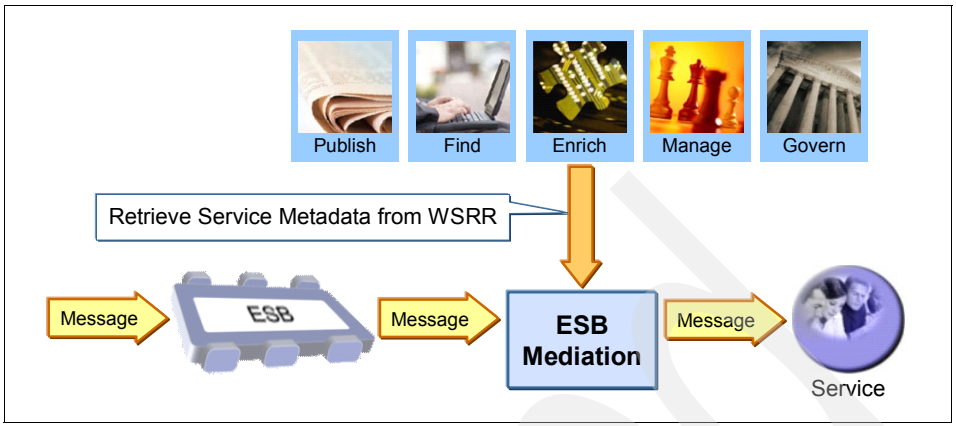


Figure 1-16 Using WSRR increases IT flexibility

WSRR provides multiple APIs (such as Java, SOAP, REST, and UDDI) to access the registries content from the various runtime environments that are available in the marketplace today.

IBM has enhanced its ESB products with primitive mediation and nodes that allow mediation flow developers to drag, drop, and configure queries to retrieve service endpoints and metadata from WSRR. The ESBs also provide caching for WSRR for performance.

For more information about IBM ESB use cases and their support for WSRR to achieve a flexible IT, consult the following resources:

- *Integrating WebSphere Service Registry and Repository with WebSphere Process Server and WebSphere ESB*, REDP-4557
- *Integrating WebSphere Service Registry and Repository with WebSphere MQ and WebSphere Message Broker*, REDP-4558

A key goal of SOA is to provide policy driven interactions between services, which leads to business agility and faster time to value because behavior can be changed by modifying the applicable policies. However, to maintain a compliant solution, the policies themselves must be managed as closely as the service implementations. Therefore, the policies are governed along side the services in which they are applied, allowing you to analyze the impact of the policies if they are changed.

For example, changing a policy can result in a new version of a service. With the new version of the service, all consumers of that service must be notified. WSRR provides support for loading, viewing, editing, attaching, and governing policies for a number of policy domains based on the WS-Policy standards. The WSRR

policy wizard (Figure 1-17), along with its policy domain libraries, make it easy to author, view, edit, and attach policies.

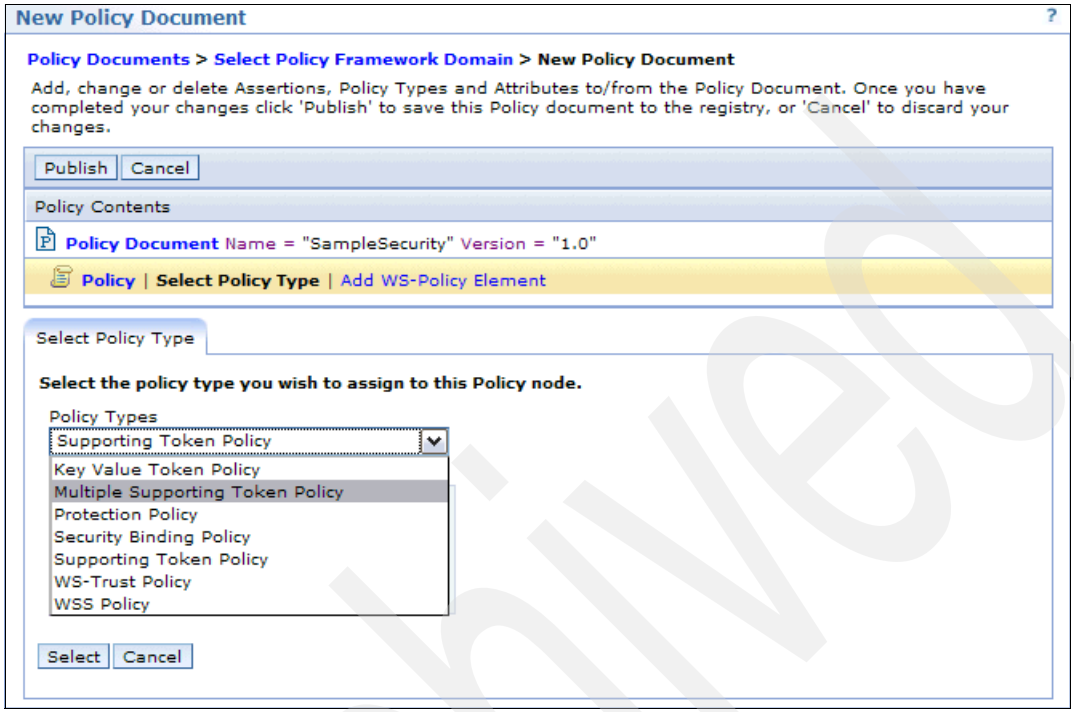


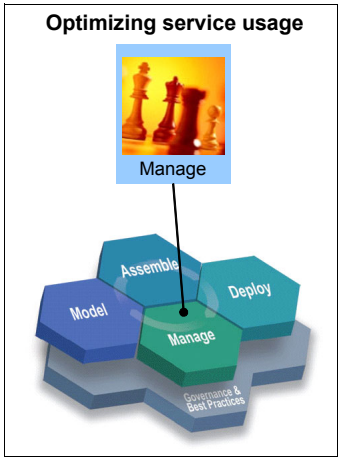
Figure 1-17 WSRR New Policy Document

WSRR and IBM Tivoli Security Policy Manager have built-in integration that allows users to author policies in Tivoli Security Policy Manager and attach these policies to services that are stored in WSRR. Tivoli Security Policy Manager V7.0 is a standards-based application security solution. It provides centralized application entitlement management, SOA security policy management, and security as runtime services to strengthen access control for new applications and services, to help improve compliance, and to drive operational governance throughout the enterprise.

For more information about WSRR and Tivoli Security Policy Manager integration and use case scenarios, see *Integrating WebSphere Service Registry and Repository with IBM Tivoli Security Policy Manager*, REDP-4561.

Archived

1.5.3 Optimizing service usage



After services are deployed, their life cycle does not end. You need to monitor the services and collect metrics to ensure that the services are doing the correct job and doing that job properly. When changes to the services occur, a new service version is implemented, and you need to assess the impact of those changes. You need to govern the new service versions as well as plan for the retirement of the previous versions. Proper communication channels must exist to notify various stake holders regarding the health of the service and of any changes to the service.

Graphical navigation

WSRR provides *graphical navigation* of entities that are managed in the registry and repository. Graphical navigation allows for easy visualization and navigation of an asset's relationships. To view the graphical navigation, click the graph icon in the graph column when on a collection view, as shown in Figure 1-18.

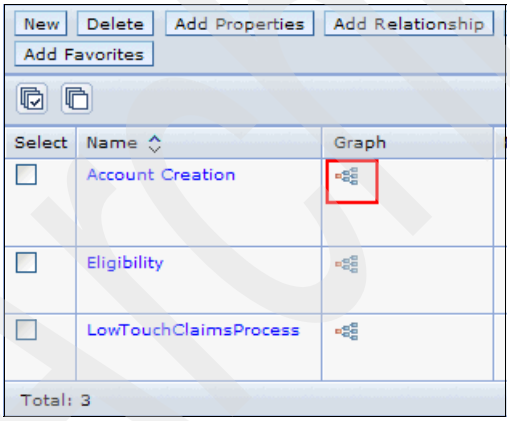


Figure 1-18 WSRR collection view graphical navigation icon

When viewing the Details pages of an entity, click **Graphical View**, as shown in Figure 1-19.

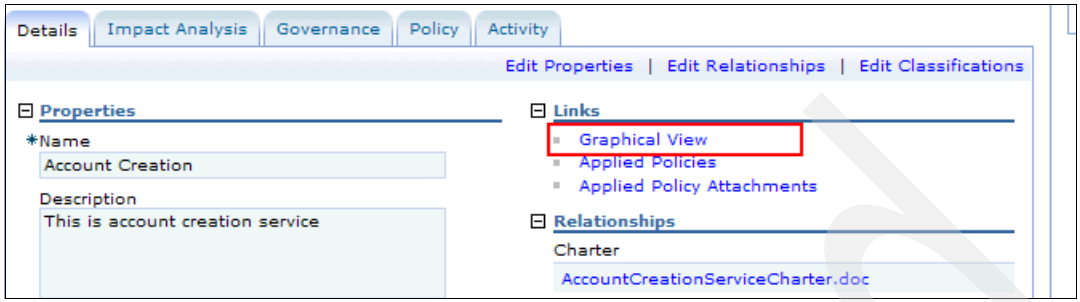


Figure 1-19 WSRR Details Graphical View navigation link

The graphical navigation lets you visually browse and navigate the relationships of an entity. The graphical navigation view allows you to act on multiple entities at the same time. For example, you can select multiple entities by holding down the Ctrl key and then selecting to add a property, add a relationship, add a classification, or add to favorites all the entities that were selected, as shown in Figure 1-20.

Archived

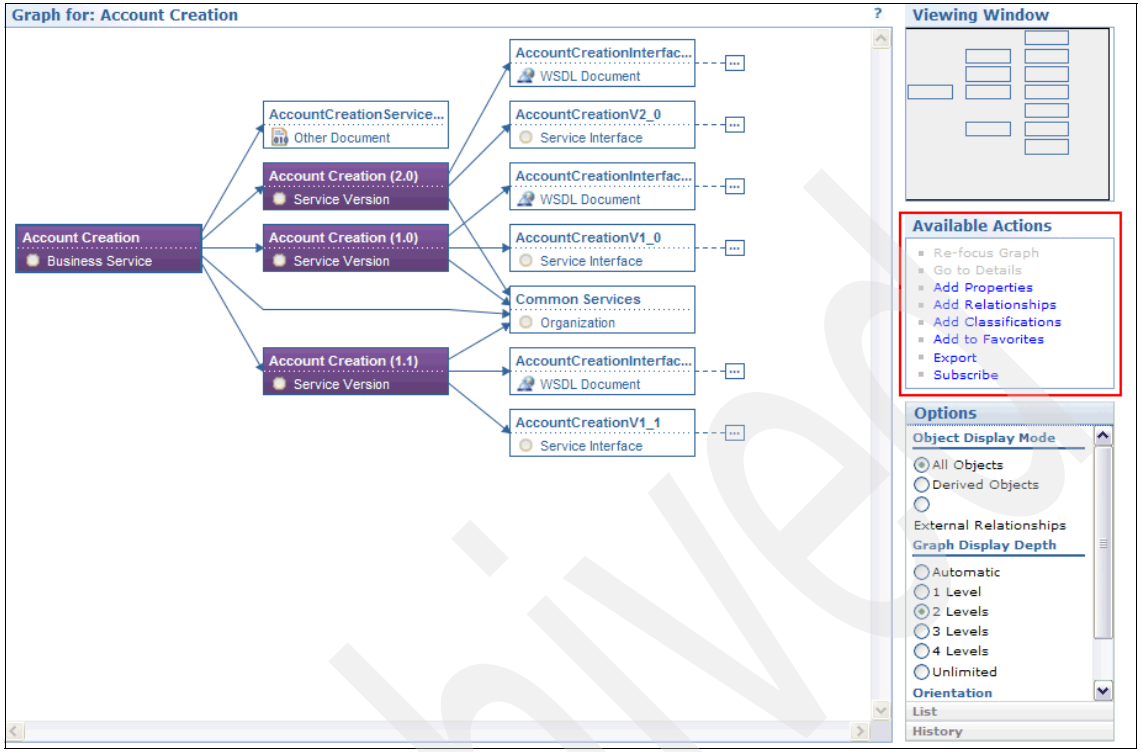


Figure 1-20 Graphical navigation, selecting multiple entities

Impact analysis

WSRR also provides *impact analysis* of entities that are managed in the registry and repository. Impact analysis enables you to assess change before the change is implemented. For example, in the case of a service version retirement, it is vital to identify all consumers of the service version who might be affected by the retirement. These parties must be consulted, and their approval must be provided before the retirement of the service occurs.

Archived

WSRR provides an impact analysis wizard that you can use to configure the impact analysis query that is run, as shown in Figure 1-21.

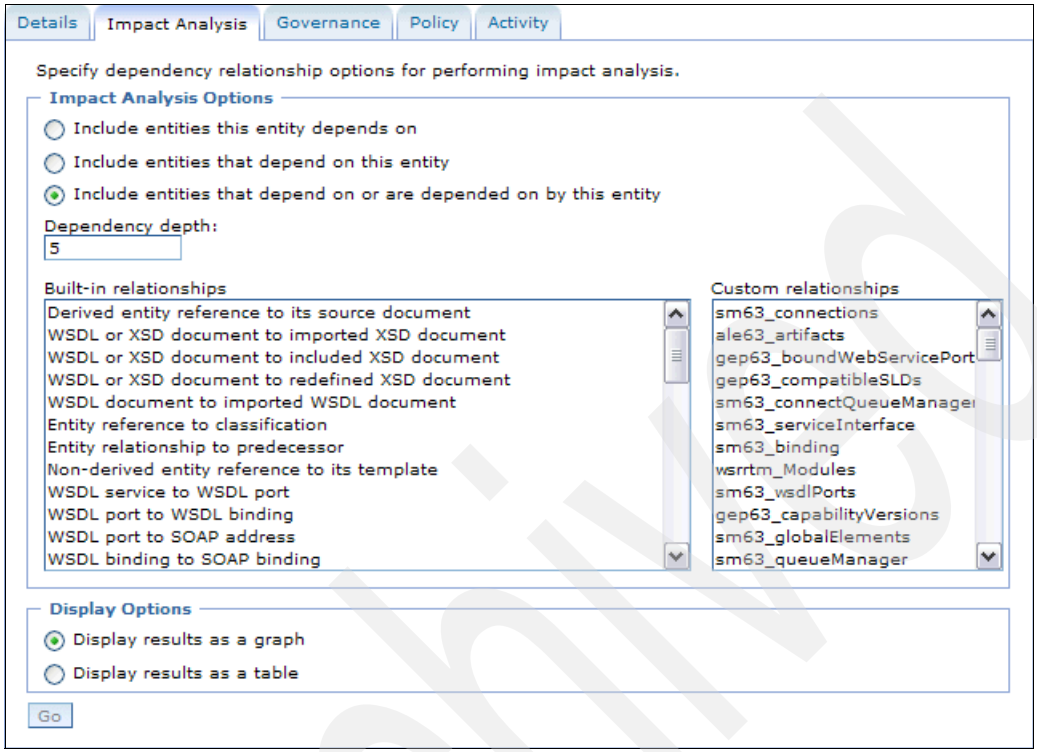


Figure 1-21 Impact Analysis wizard

Archived

Figure 1-22 shows the impact analysis of a service level definition (SLD) for the Eligibility 1.0 service version.

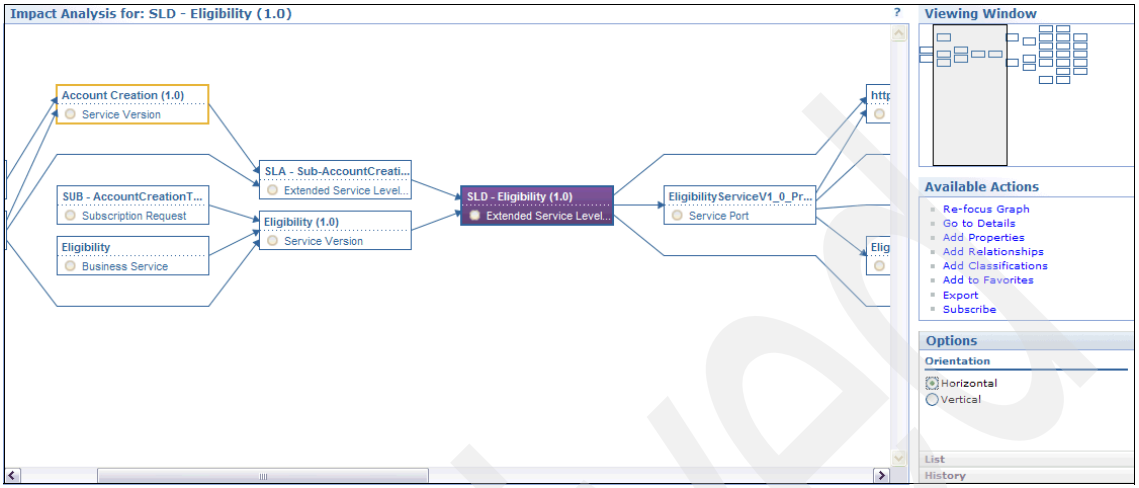


Figure 1-22 Impact Analysis of the SLD for Eligibility 1.0

Service versioning

WSRR provides support for service versioning. You can customize the service version life cycle to fit the enterprise's specific service versioning processes. Figure 1-23 show a graphical navigation view of the Account Creation business services and its supporting service versions.

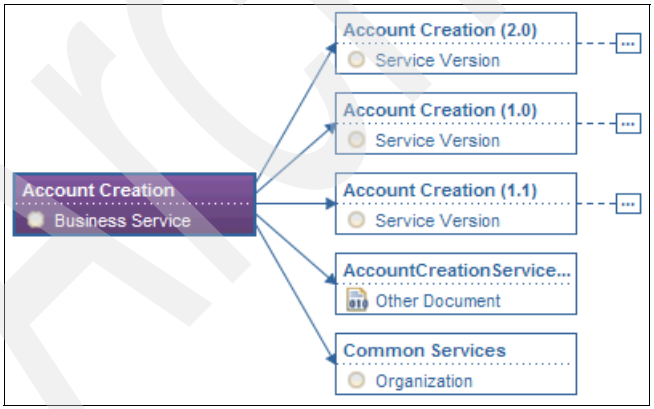


Figure 1-23 Account Creation service versions

Archived

Monitoring, metrics, and reports

Having accurate visibility into an SOA solution can increase the success of the SOA. Visibility includes the following different:

- Design time considerations:
  - Which services are being built?
  - What is the services' current life cycle?
  - Which consumers have subscribed to use the service?
- Operational aspects:
  - What are the available SLDs for this service?
  - What endpoints are available?
  - What are the active SLAs for this service?
- Run time observed aspect gathered from a monitoring system:
  - How well is this service actually performing?

For all these aspects, metadata is stored in with WSRR, and you can create reports to provide the visibility you need to manage the SOA.

WSRR provides reporting capabilities using WSRR Studio. WSRR Studio uses the open source, Eclipse-based, Business Intelligence and Reporting Tools (BIRT) to build customized reports. Using BIRT reports, you can run WSRR queries and then use the information that BIRT returns to generate detailed reporting charts in a number of formats, including HTML, PDF, and Microsoft Excel.

Archived

Figure 1-24 shows an example of a provisioning report created in WSRR Studio. The provisioning report displays information about a service's available SLD and the SLAs. The report alerts in red if the total of the SLA *Maximum Message Per Day* exceeds the *Certified Maximum Messages Per Day*. This alert in turn notifies the operations team that more resources are needed for this service.

Provisioning Report - Business Capabilities by Consumer Subscription

Business Capability name	Capability Versions		Service Level Definitions			Service Level Agreements				
	Name	Version	Name	Version	Certified Peak Message Rate	Certified Maximum Messages Per Day	name	version	Expected Production Date	Peak Message Rate	Maximum Messages Per Day
Eligibility	Eligibility (1.0)	1.0	SLD - Eligibility (1.0)		500	4000	SLA - Sub-AccountCreationToEligibility		2009-07-31	550	5000
										TOTAL	5000.0
LowTouchClaimsProcess	LowTouchClaimsProcess (1.0)	1.0	SLD - LowTouchClaimsProcess (1.0)		400	600	SLA - SUB-ClaimsProcessToLowTouch		2009-07-22	400	
							SLA -		2009-06-28	300	

Figure 1-24 Service provisioning report

Run time observed aspects can be added to the above report using IBM Tivoli Composite Application Manager for SOA (ITCAM for SOA). WSRR and ITCAM for SOA have built-in integration that allows ITCAM for SOA to discover and manage services that are registered in WSRR. ITCAM for SOA can then compare the services that are registered in WSRR with the services that ITCAM for SOA observes to determine rogue services. This function is also used to determine the services that are registered in WSRR but that are not being used. ITCAM for SOA can publish metrics from the observed services to WSRR, where the metrics can be used by a mediation in a runtime environment and used to build robust reports to provide to management.

For more information about building reports with WSRR Studio, see Chapter 10, "Reports" on page 349.

Archived

Change notification

When change occurs to a service or another item in the registry and repository, all interested parties must be notified. WSRR provides an extensible notifications framework that allows you to notify user and client applications when events occur.

Although you can configure client applications to filter the events that they require, users often want to subscribe to certain events on certain types of artifacts. WSRR provides a basic subscription mechanism for a customer-supplied user notification plug-in. Users can subscribe to any or all the following operations:

- `create`
- `update`
- `delete`
- `transition`
- `validate`
- `make_governable`
- `remove_governance`

Archived

Figure 1-25 shows the e-mail subscription wizard from the WSRR Web user interface.

Subscriptions ?

Service Level Definitions > Create a Subscription

Creates a new subscription. You must specify a name, a destination address and at least one operation. Optionally, you can specify target entity matching criteria to limit the entities to which this subscription applies.

Details

*Name

Description

Owner
admin

*Destination address

Entity list
SLD - Eligibility (1.0) (concept)

Operations

- Create
- Update
- Delete
- Validate
- Make governable
- Remove governance
- Transition

Transitions
Activate SLA
Approve
Approve Asset
Approve Asset Specification
Approve Certification

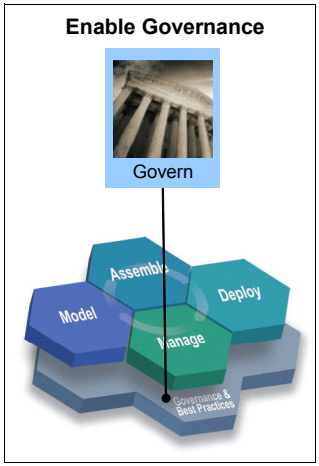
Preferred notification message language
English

Apply OK Cancel

Figure 1-25 WSRR e-mail subscription wizard

Archived

1.5.4 Enable governance



To fully realize the value that we have discussed thus far, an enterprise needs governance throughout the life cycle of the services as well as governance of the service metadata. In other words, in order to eliminate rouge services, promote reuse, and increase flexibility the services need to be taken through a well-defined life cycle and the quality of the service metadata needs to be ensured.

Metadata model and life cycles

To ensure the quality of the service metadata, WSRR includes a customizable metadata model that is based on the Web Ontology Language (OWL). The OWL is a W3C-endorsed format that can be used to define an ontology. It can define relatively rich semantics, including relations between classes of entities (for example disjointness), cardinality (for example “exactly one”), equality, properties, characteristics of properties (for example symmetry), and enumerated classes. OWL allows WSRR to maintain and enforce the rich metadata model that is needed to manage service metadata, such as service versioning as well as consumer and provider relationships.

WSRR enables the definition of custom life cycles for SOA artifacts. Different SOA artifacts require different life cycles. The life cycle is a state machine that is enforced by a validation framework, allowing you to assign policies that validate the artifacts and to determine if the transition to the next life cycle state should occur. It also allows you to specify notifications when the transition occurs. For example, this notification can be an e-mail, or it can invoke an external process. Figure 1-26 shows an example of a service (capability) version life cycle.

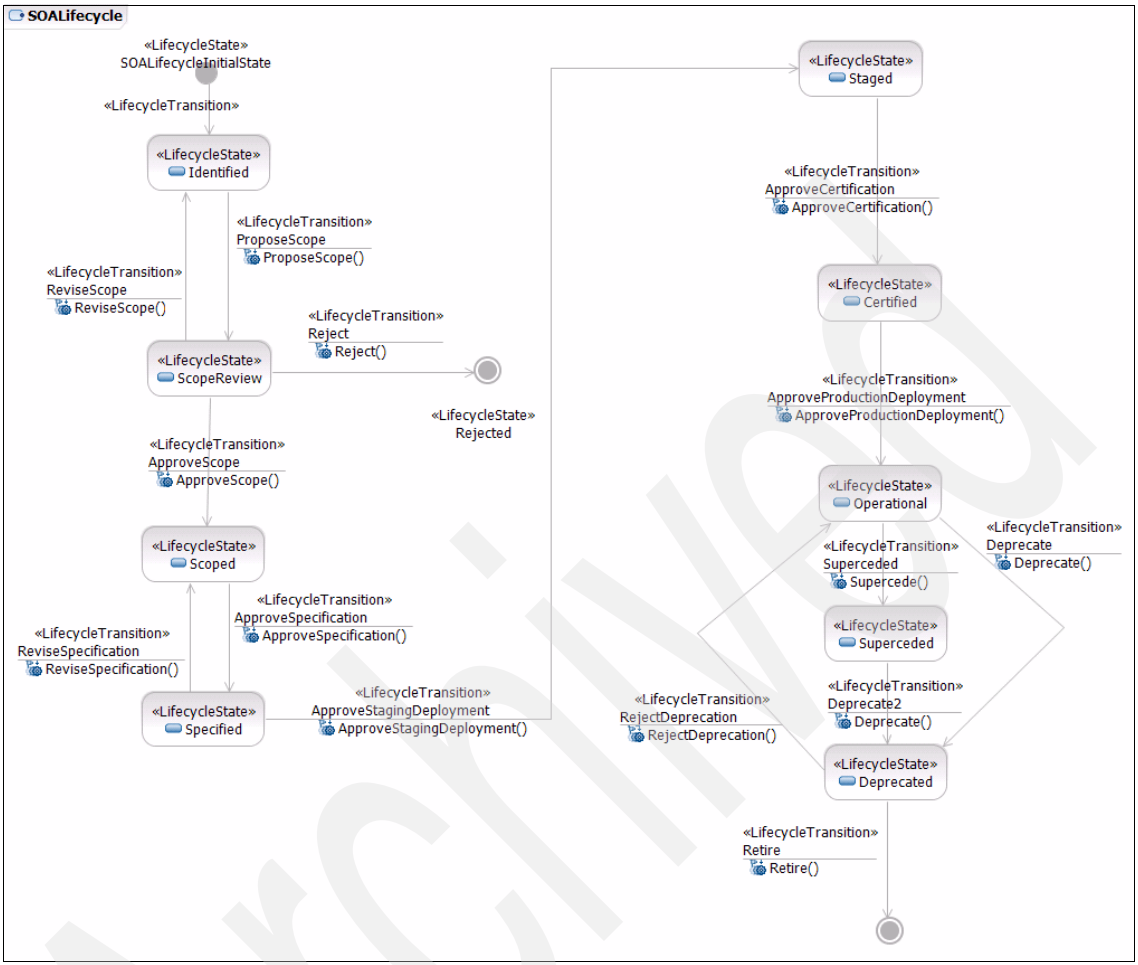


Figure 1-26 Example service version life cycle

For more information about how to customize the WSRR metadata model and life cycles, see Chapter 5, “Modeling in WebSphere Service Registry and Repository Studio” on page 163.

Governance policies

The WSRR validation framework enforces policies to which artifacts and metadata must conform before transitioning to the next state in the life cycle. WSRR provides a Governance Validator plug-in, that is built on top of the validation framework, which enforces customizable governance WS-Policies. The WSRR Policy Wizard has Governance Validator template libraries that you can use to configure the WS-Policies that are enforced by the Governance Validator

plug-in. WSRR also exposes validation framework interfaces to allow you to create your own validator plug-ins that can be applied to all CRUD operations and governance operations. The customizable governance life cycle, along with the validation framework, enables you to have enforceable governance that ensures the quality of services and service metadata.

For more information about using the WSRR Policy Wizard to configure Governance Validator policies, see Chapter 9, “Policies” on page 301.

Environment promotion

WSRR provides a promotion capability to extend governance to multiple environment topologies. Promoting allows you to manage metadata that relates to multiple deployment environments in a central WSRR instance, while also allowing the appropriate subset of information to be copied to an environment specific WSRR instance in response to the life cycle transitions.

For more information about promotion, see Chapter 8, “Promotion” on page 289.

Service discovery

The WSRR service discovery mechanism enables the discovery of deployed services in target application environments. You can use service discovery to understand the services that an enterprise has deployed when you first attempt to use service governance. Later, you can use it to control the target environments by discovering services that are deployed but that are not governed in WSRR.

For more information about service discovery, see the WSRR Information Center:

http://publib.boulder.ibm.com/infocenter/sr/v6r3/topic/com.ibm.sr.doc/cwsr_service_discovery.html

Governance enablement profile

The WSRR governance enablement profile can jump start an service governance implementation. In encapsulates the requirements that we discuss throughout this chapter. The governance enablement profile is based on the experience of field professionals when implementing service governance. The governance enablement profile provides component models, life cycles, governance policies, security controls, and Web user interface commands to define services in an SOA environment and to manage those services from initial specification through to deployment in production.

For more information about the governance enablement profile, see Chapter 3, “Introduction to the governance enablement profile” on page 77.

Archived

Throughout this book, we describe how to customize different elements of the governance enablement profile. In Part 3, “Scenarios” on page 399, we provide a variety of scenarios that demonstrate how to take advantage of a customized governance enablement profile.

Archived

2

WebSphere Service Registry and Repository technical overview

This chapter provides an overview of the architecture and the key components in WebSphere Service Registry and Repository (WSRR). It includes the following topics:

- Architecture of WebSphere Service Registry and Repository
- Service life cycle support features
- Document types
- Content model
- WSRR deployment topologies
- Deployment configurations
- Packaging

© Copyright IBM Corp. 2009. All rights reserved.

2.1 Architecture of WebSphere Service Registry and Repository

WebSphere Service Registry and Repository (WSRR) provides service registry and repository functions for service-oriented architecture (SOA) enterprise middleware and applications. WSRR provides registry functions that support publication of metadata about the function, requirements, and semantics of services that enable service consumers to find services or to analyze relationships between services. It also provides repository functions to store, manage, and version service metadata. WSRR is a J2EE application that is based on IBM WebSphere Application Server and that uses a relational database as a backing store for service metadata.

The registry and repository component provides basic service metadata storage, update, and retrieval functions that support create, retrieve, update, delete, and query capability for service metadata that is stored in the database according to the WSRR content model.

Figure 2-1 illustrates the key components in the WSRR architecture.

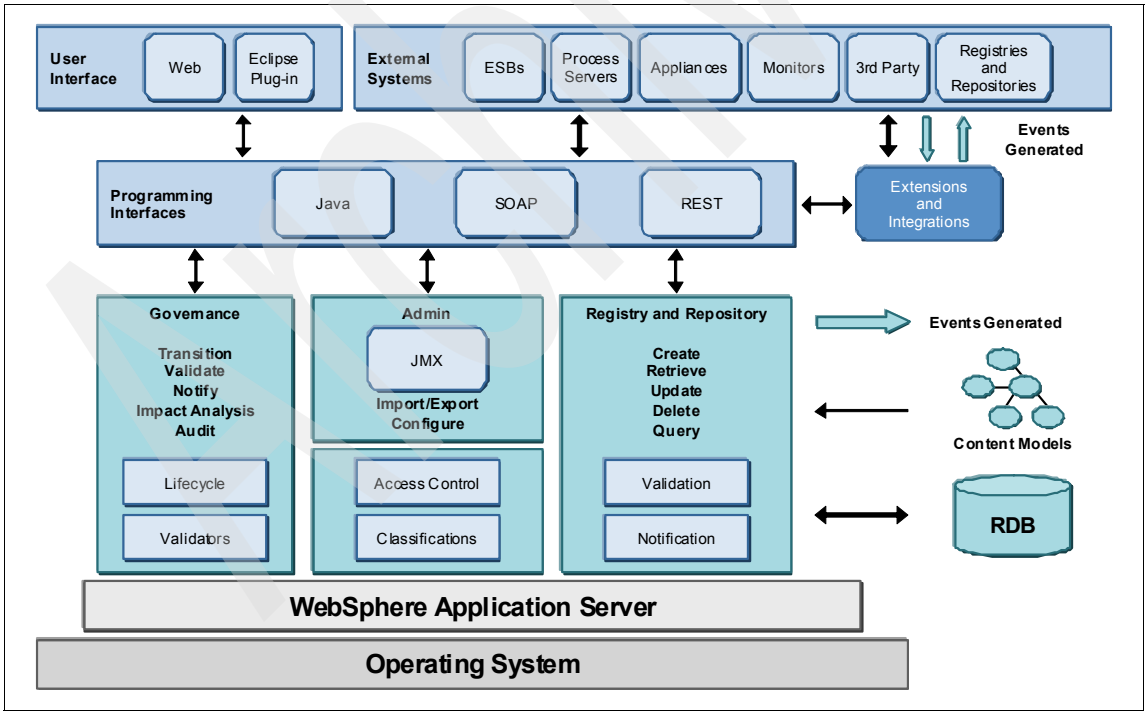


Figure 2-1 Overview of the WSRR architecture

WSRR uses a relational database as a backend store for service information and metadata persistence. WSRR supports DB2®, DB2 for z/OS Oracle, Microsoft SQL server, and IBM Cloudscape (for test environments only) relational databases.

WSRR includes service information and metadata artifacts that are in the form of XML documents. You can put any XML document into WSRR, but some types of XML documents, such as WSDL and XSD, are modeled. When one of these types of XML documents is loaded into WSRR, it is parsed into a finer-grained information model. WSRR can also store binary documents, For example, a service definition document written in Microsoft Word, that aid in the governance of services.

2.1.1 Registry and repository

WSRR functions as both a *registry* and a *repository*:

- The repository allows users to store, manage, and query content of documents that include service metadata descriptions (such as WSDL, XSD, WS-Policy, SCDL, or XML documents). WSRR stores documents that contain service metadata and also provides a fine-grained representation of the content of those documents (for example ports or portTypes in WSDL documents).
- WSRR also provides registry functions for populating registered service declarations and elements of the derived content models with user-defined properties, relationships, and classifiers. A rich query interface makes use of these user-defined attributed when users want to find a service endpoint, interface description, or other metadata about a service.

WSRR allows users to plug in validation and modification functions that run when changes are made to the repository content (for example, checks for completeness of a service definition). It also provides notifications of any changes to the content of the repository and allows users to register interest in consuming those notifications.

WSRR includes a default notification handler that publishes change events on a JMS topic. The event specifies the type of event (create, update, delete, or transform), the artifact impacted (identified by its URI), and other information about the artifact. To avoid access control problems, the content of the artifact is not shipped with the event but is retrieved separately.

Archived

2.1.2 User interfaces

WSRR includes two user interfaces:

- A servlet-based Web user interface (UI)

  This interface is deployed with the WSRR run time and is the main interface for users in different roles to interact with WSRR. You can use the Web-based UI to manage and govern service metadata. The Web UI supports scenarios to complete the following tasks:

  – Search for and publish service information

  – Analyze and manage metadata

  – Import and export data

  – Perform impact analysis

  – Customize components that control the configuration of the WSRR system, manage access control, and create and modify classification systems using administrative functions

  The Web UI allows you to customize views of the WSRR content that are represented to a user. A set of UI *definition files* describes the content and layout of the various components that make up the WSRR Web interface. In addition, user- and role-specific *perspectives* are supported. WSRR ships with a set of predefined perspectives for the most common user roles; however, you can customize the predefined perspectives or introduce new, role-specific perspectives as needed.

- Plug-in interfaces

  A subset of the Web UI is offered as an Eclipse and Microsoft Visual Studio plug-in to meet the needs of developer and analyst roles that use Eclipse-based and Microsoft Visual tools. The plug-in is provided to support lookup, retrieval, and publishing of service metadata in the context of the development tools or management consoles.

2.1.3 Governance functions

WSRR includes the following governance functions:

- Control access to service metadata.
- Promotion of services through phases of their life cycle in various deployment environments. A life cycle model for governed entities uses a state machine that describes the life cycle states and the valid transitions between them.
- Validation, modification, and notification plug-ins to guard the transition and the actions to be taken as result of the transition.

- Logging of basic audit information about WSRR content updates.
- Impact analysis of changes to specific artifacts. A set of predefined impact queries supports patterns of navigation through the WSRR content, such as which WSDL files import or use this XSD.
- E-mail notifications for users who are interested in specific WSRR content changes.

2.1.4 Administration interfaces

The WSRR administration interfaces support the import and export of WSRR content for exchange with other metadata repositories (for example other WSRR installations) and provide a JMX-based application programming interface (API) for WSRR configuration and basic administration. With WSRR you can use a fine-grained access control model to define which user roles can perform what kind of actions on which artifacts. You can also define and import classification systems from basic classifications sets to taxonomies and classification hierarchies.

You can also use the WSRR JMX-based administration API to complete basic configuration as well as to load and manage metadata in support of WSRR content classification and life cycle management. With the API, you can load definitions of state machines that are used to model the life cycle of governed entities, and you can load OWL-described classifier systems.

In addition, you can use the administration API to register plug-ins for validation and modification functions and to notify providers of modifications. You can use validation functions to control basic create, retrieve, update, and delete operations as well as in the context of life cycle state transitions for governed entities.

2.1.5 Programming interfaces

WSRR supports the following APIs that you can use to interact with WSRR:

- A Java based API that uses Service Data Objects (SDOs) data graphs
- A SOAP based API that uses XML data structures
- A Representational State Transfer (REST) based API that uses XML or JSON data
- UDDI V3

All APIs support publishing (creating and updating) service metadata artifacts and the metadata that is associated with those artifacts, retrieving service

Archived

metadata artifacts, deleting the artifacts and their metadata, and querying the content of WSRR. Basic create, retrieve, update, and delete operations, as well as governance operations and a flexible query facility based on XPath, are provided through these APIs.

Clients are provided for the Java and SOAP APIs. SDOs capture the data graphs inherent in the content model, allowing access to physical documents, logical parts of the physical documents, and concepts.

Java, SOAP, and REST use a query API that allows the use of XPath expressions to perform unanticipated coarse- and fine-grained queries. Queries can be performed using semantic annotations, properties, and all or parts of physical service metadata artifacts. Fragments of metadata can be returned (such as properties for name or endpoint address), all metadata can be returned (data graph), and metadata and documents can be returned. In addition to free-form XPath-based queries, a set of predefined queries are offered to address common paths through the WSRR content model.

To take advantage of the UDDI V3 APIs, WSRR must be configured to synchronize with a UDDI V3 registry. IBM UDDI registry is provided along with WebSphere Application Server, which comes with WSRR.

Command-line interface

Interactive and scripted administration of WSRR is possible with the Jython-based command-line interface. The command-line interface provides full support for all the WSRR programmatic APIs, including all administrative operations. It can be used from a stand-alone Jython or JACL interpreter, or it can be run inside wsadmin and used with the facilities available there.

Archived

2.2 Service life cycle support features

WSRR provides a number of features to support the management of service metadata throughout the service life cycle. Figure 2-2 illustrates the main phases of the service life cycle and how WSRR provides features to support each phase.

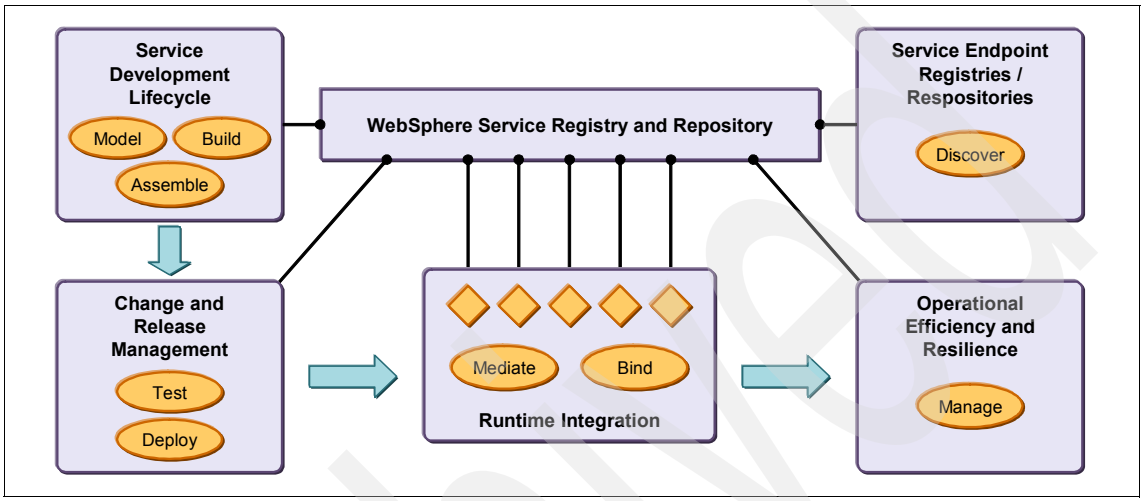


Figure 2-2 Service life cycle phases

We describe the service life cycles that are supported by the governance enabled profile in Chapter 3, “Introduction to the governance enablement profile” on page 77.

2.3 Document types

Any service metadata artifact type can be stored in WSRR and receive the benefits of broader visibility and reuse, management, and governance. WSRR supports loading the following document types:

- Web Services Description Language (WSDL)
- XML Schema Definition (XSD)
- WS-Policy

Archived

- Service Component Architecture (SCA) integration modules

  The components of an SCA integration module that can be loaded are:

  - SCA module definitions
  - SCA import definitions
  - SCA export definitions
- XML documents
- Other documents

Figure 2-3 shows the various categories of service-related documents that can be viewed in WSRR.

Figure 2-3 Service documents

Figure 2-4 illustrates the document logical objects model.

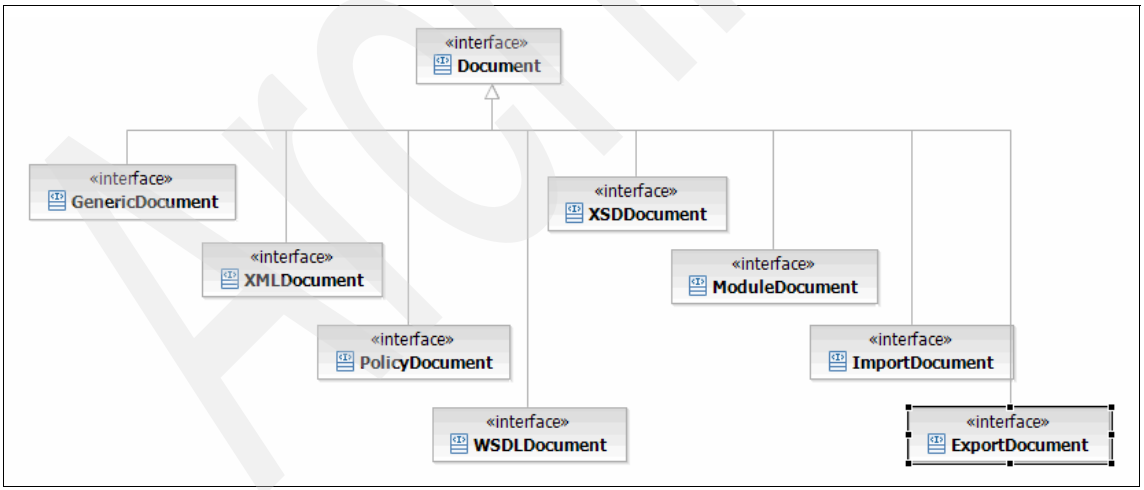


Figure 2-4 Document objects

Archived

2.3.1 XML schema definition

In an SOA environment, services exchange information in an XML format. To achieve interoperability, different services have to agree on the schemas for this information exchange. The industry standard for defining the XML schema is XML Schema Definition (XSD).

In an SOA environment, the following artifacts use XML schemas to define the service implementation:

- Business objects

  Businesses often adopt standardized representations of information that is key to their business. Some of these business objects are standardized throughout companies to support particular market sectors.

- Business messages

  Message-driven architectures rely primarily on XML schema definitions to define the formats that are used to exchange information in business messages.

- Business events

  As with messages, business event structures are often defined in XML schemas to assist in reporting and systems management.

- Service operation signatures

  Even when services are described using WSDL, the parameters of the service operations are often expressed using XML schemas that are expressed in XSD documents. This method allows different services to share a common information model.

With WSRR, you can register XSD documents and identify the important XML schema constructs within those documents that will affect the interoperability or common dependencies between managed services. The following service metadata types can be identified when an XSD document is loaded:

- Elements
- Attributes
- Simple types
- Complex types

Archived

Figure 2-5 shows the details that can be viewed when an XSD document is loaded.

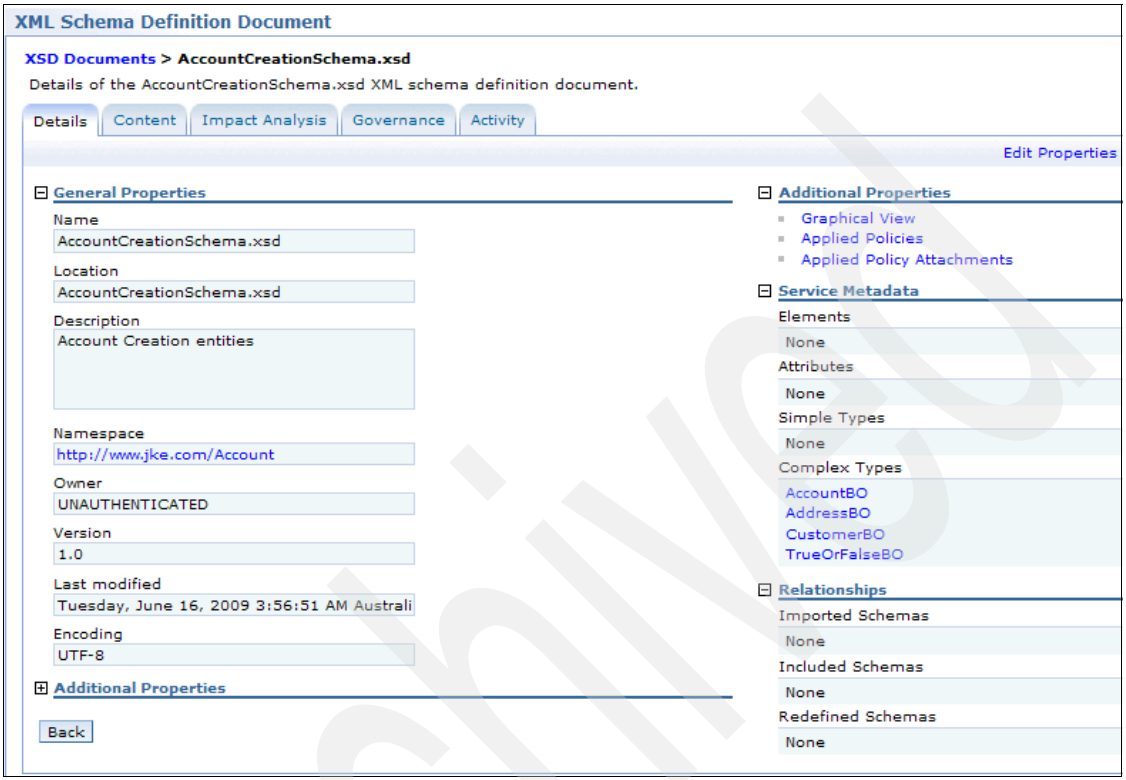


Figure 2-5 XSD document details

Each of these constructs results in an object that is stored in the registry and that is related to the schema document that declared it. These *derived* or *logical* objects cannot be created or deleted by the user. They are managed entirely by the registering or deregistering of the XSD document that defines them.

Archived

Figure 2-6 illustrates the XSD logical objects model.

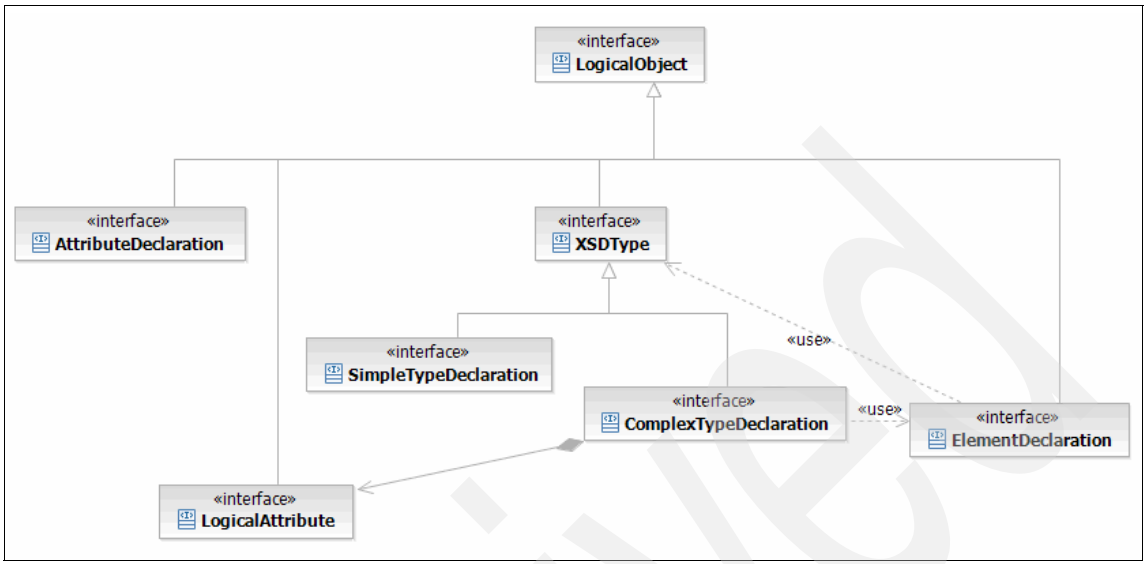


Figure 2-6 XSD logical objects

2.3.2 Web Services Description Language service definitions

In an SOA, the loose coupling between services means that organizations have the flexibility to change which services depend on which other services. Before any change can occur, the organization needs to be confident that the consumer service is compatible with the provider. Thus, all deployed or about to be deployed services need to have a clear service definition that can be published for consumers to discover. The industry standard for Web service definitions is *Web Services Description Language* (WSDL).

Web service definitions are often provided at the following levels, all of which use WSDL to describe them, as illustrated in Figure 2-7:

- *Service interface definitions* describe the interfaces in terms of operations and signatures that are provided by a service. These types of definitions will often reference XML schemas to define common message formats or operation parameters.
- *Service binding definitions* describe how the interfaces are represented in the infrastructure and over the wire. These definitions define the transport protocols that are supported (such as SOAP, JMS, and so on) and reference the service interface definition to indicate exactly which operations are actually supported over this transport protocol.

Archived

- *Service endpoint definitions* describe each individual deployed service and describe how a consumer would actually find and connect to a service. They define the endpoints and indicate which bindings and, thus, interfaces are supported on each endpoint.

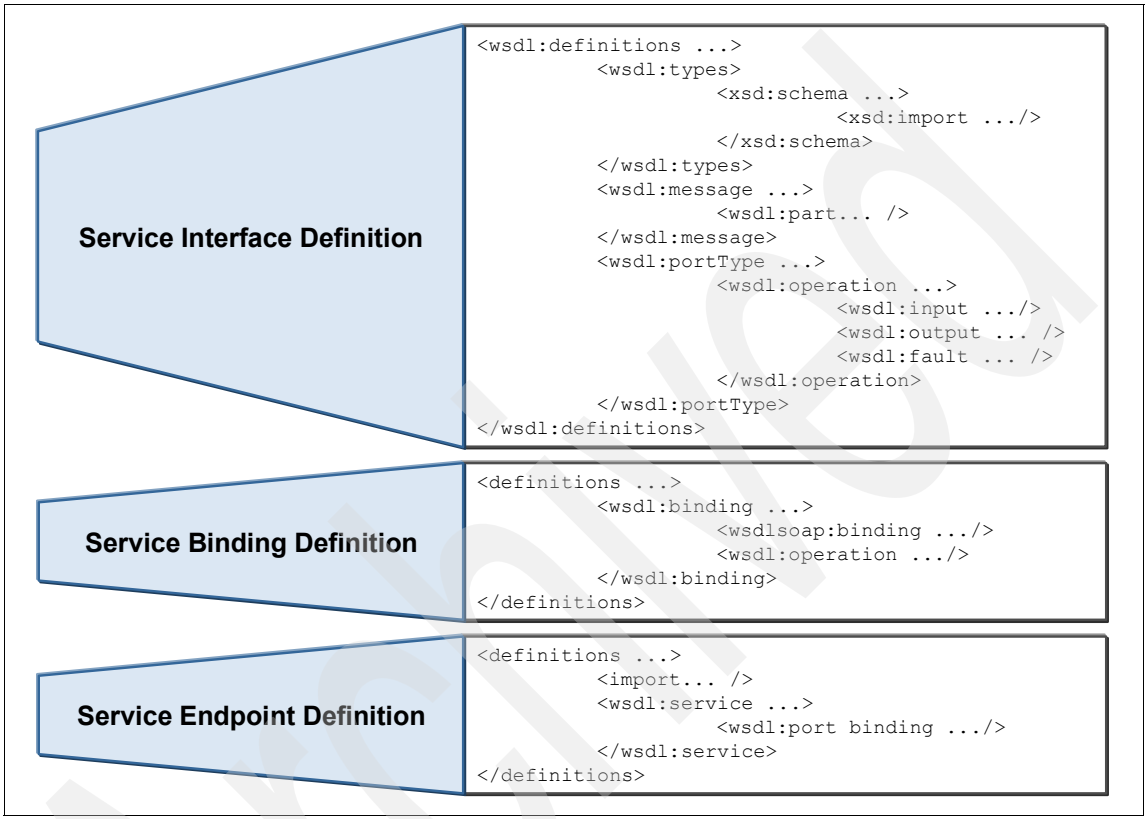


Figure 2-7 Service definition levels

It is good practice to make sure that each of these levels of service definition are defined in different WSDL documents. This practice is not enforced by WSRR, but the split of definitions across multiple documents is supported.

Archived

With WSRR, you can register WSDL documents and identify the important WSDL constructs within those documents that will affect the interoperability or common dependencies between managed services. For each WSDL document, WSRR identifies the following WSDL constructs:

- *Port Types* identify the service interfaces that are defined in the document, including the operations and their signatures. These interfaces often reference XML schema constructs that define operation parameters.
- *Bindings* identify the bindings that are defined in the document, including the SOAP binding and the portType that is supported.
- *Services* identify the service, ports, SOAP address, and the bindings to which they relate.

Figure 2-8 shows the details that can be viewed when a WSDL document is loaded.

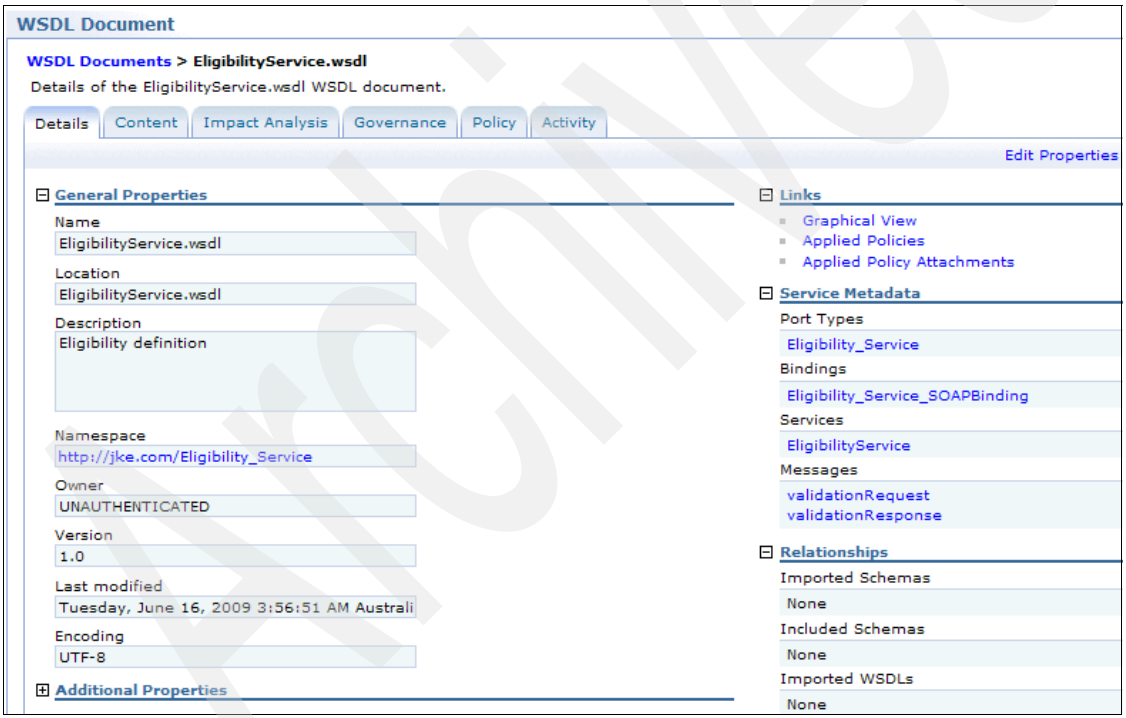


Figure 2-8 WSDL document detail

Each of these constructs results in a related collection of objects that is stored in the registry and that are related to the WSDL document that defined them. The user cannot create or delete these *derived* or *logical* objects. They are managed entirely by the loading or deleting of the WSDL document that defines them. This

Archived

has the implication that if any document is deleted or updated (in terms of content), any derived objects are deleted and the attached metadata is lost.

These derived objects allow the named construct to be annotated with metadata to indicate how it is used throughout the SOA enterprise. This annotation allows a particular construct that is declared within a document to be managed differently from other constructs that are declared within the same document. For example, different policies might be attached to different operations of a service interface although all the operations are defined in the same WSDL document.

Figure 2-9 illustrates the WSDL logical objects model.

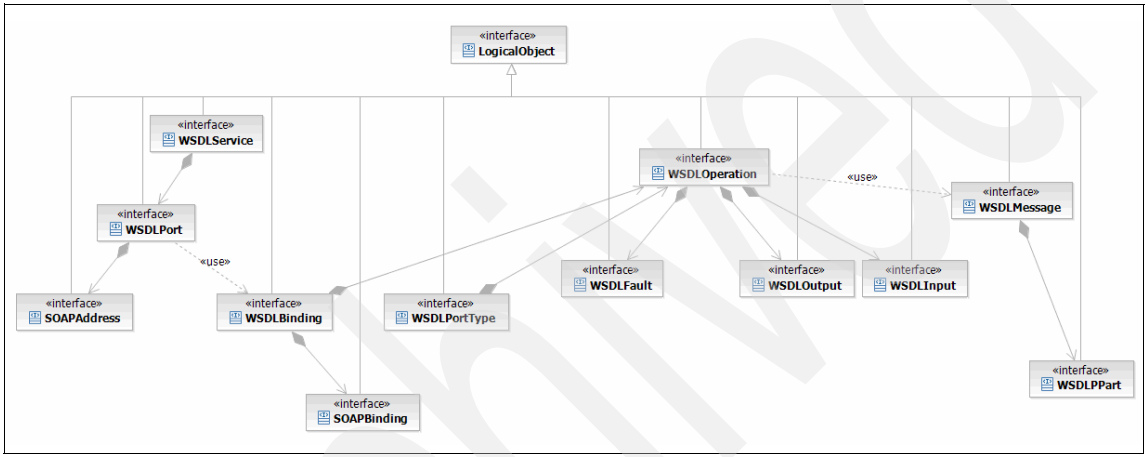


Figure 2-9 WSDL logical objects

2.3.3 Service Component Architecture modules

WSRR provides support for conventional service and schema definitions through the use of WSDL and XSD documents. However, these document standards are not enough to describe the dependencies between services that are necessary to support SOAs. An SOA offers the ability to assemble components from existing services either at development time or at run time.

The emerging standard that allows these assemblies to be described is the *Service Component Architecture* (SCA). SCA modules are components that can be deployed into an SCA environment and that represent processes or other compound services. Modules are assembled from development time components into a single deployable artifact, but these modules can still be related to (can still depend on) other external deployed services.

SCA provides a number of important constructs that are defined as part of an SCA module deployment description:

- SCA libraries provide a number of XSD and WSDL documents that describe the services artifacts that are reused or referenced within the module.
- SCA module files provide the definition of the module in terms of the internal components that are used within the module. WSRR does not interpret the internal content information but does identify any externalized dependencies as imports and exports.
- SCA *imports* provide the definitions of the external services on which the module depends. These import files define the interfaces, bindings, and endpoints that the module needs to resolve when it is deployed.
- SCA *exports* provide the endpoint descriptions that the module exposes in terms of interfaces, bindings, and endpoints.

One of the advantages of SCA over WSDL is that it allows a wider range of bindings to be expressed. Thus, SCA modules can work with JMS and other message based paradigms as well as the Web services that WSDL supports.

Figure 2-10 illustrates the SCA logical objects model.

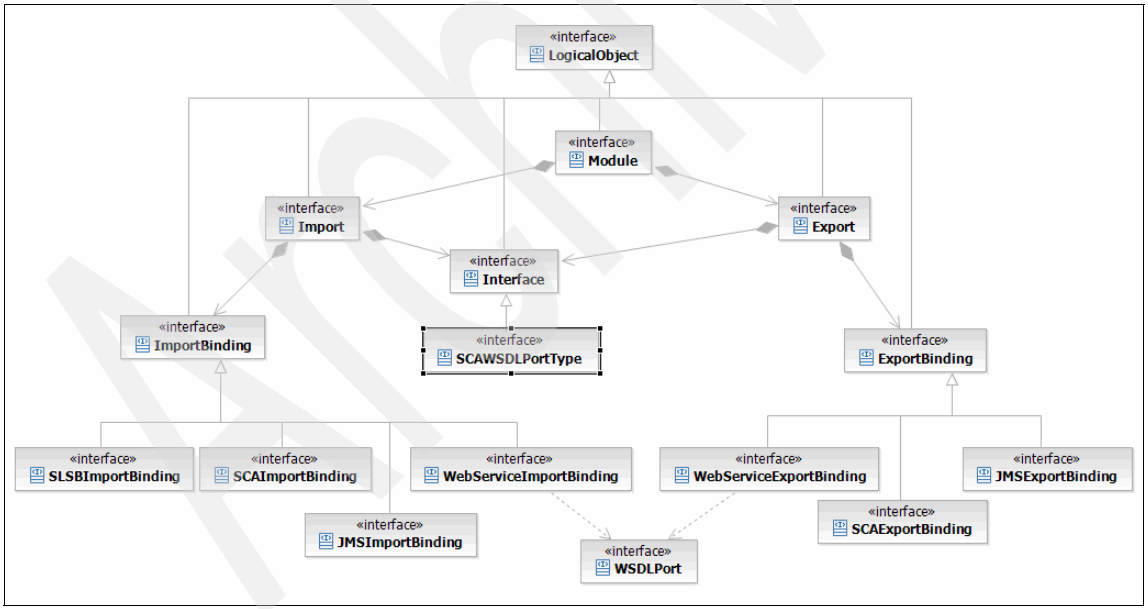


Figure 2-10 SCA logical objects

2.3.4 Service Policy (WS-Policy)

One of the key goals of SOA is to provide policy-driven interactions between services that results in business agility and faster time to value because behavior can be changed by modifying the applicable policies. However, the policies themselves then need to be managed as closely as the service implementations in order to maintain a compliant solution.

WSRR aims to provide a copy of record for all policy documents. Using the metadata facilities within WSRR, these policies can be related to the service artifacts to which they apply.

WSRR provides a very basic interpretation of the content of the WS-Policy documents so that no derived objects are created to which metadata can be applied. It is, therefore, the client application's responsibility to interpret and enforce any policies that are deemed applicable by the metadata that is contained within WSRR.

Figure 2-11 shows the Policy logical objects model.

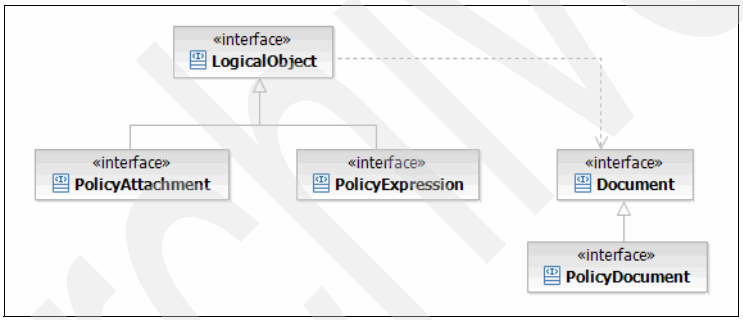


Figure 2-11 Policy logical objects

2.3.5 XML metadata files

WSRR aims to provide a single copy of record for all service metadata. Many SOA artifacts can be described by well known standards, such as XSD and WSDL. Many customers, however, have their own service descriptions that are stored in some other XML format, such as business rules, business policies, Business Process Execution Language (BPEL) files, and so on. WSRR provides a means of storing these XML documents and, thus, relating them to the other service artifacts in the registry. WSRR does not interpret the content of these XML documents, so no derived objects are created.

Archived

Figure 2-12 shows the details that can be viewed when a XML document is loaded.

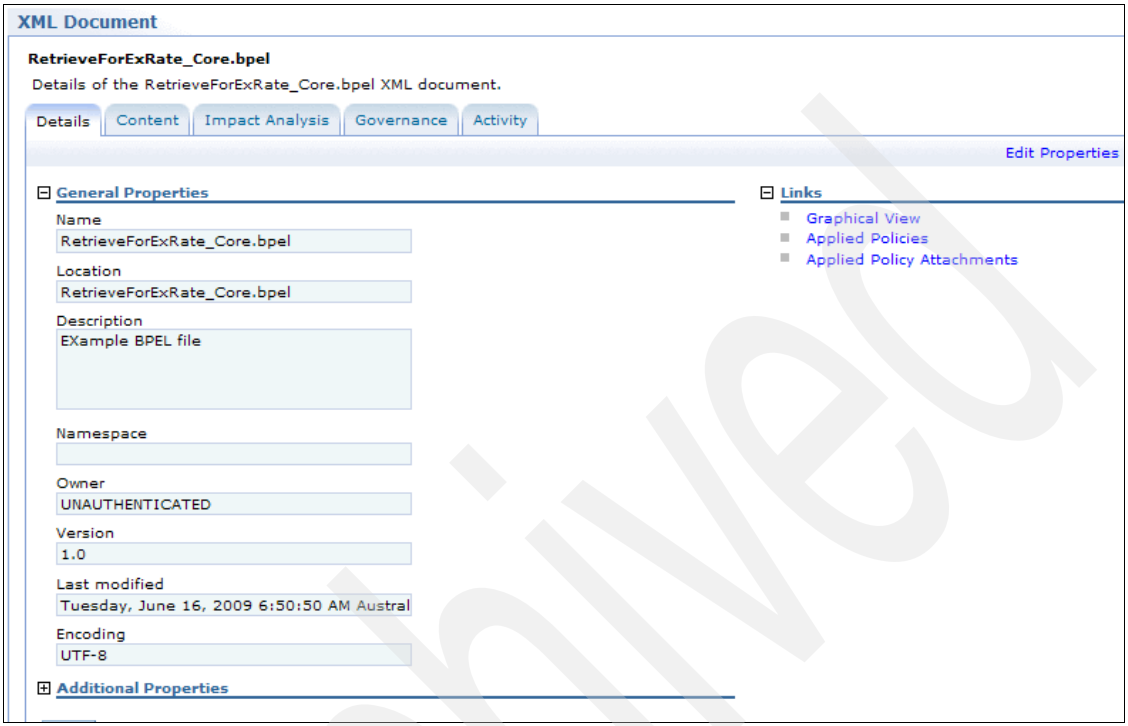


Figure 2-12 XML document detail

2.3.6 Derived objects

Derived objects allow the named construct to be annotated with metadata to indicate how it is used throughout the SOA enterprise. This annotation allows a particular construct that is declared within a document to be managed differently from other constructs declared within the same document. This has the implication that if any document is deleted or even updated (in terms of content), any derived objects are deleted, and the attached metadata is lost.

Archived

In addition to this parsing of XSD and WSDL documents, WSRR automatically detects and maintains dependencies between documents. When WSRR loads or creates the dependent document, the following dependencies occur:

- If the document is already available within WSRR, the `xsd:include`, `xsd:import`, and `xsd:redefine` statements do not include enough information to resolve between different versions of the same schema (same name, location, and namespace). Thus, special handling is required.
- If the document is provided with the dependent document as part of the load, the document on which that dependent document depends is also loaded and parsed.
- If the document is available from the URI included in the import or location hint, then the document on which that dependent document depends is also loaded and parsed.

For each of these dependencies encountered and resolved, WSRR ensures that a built-in relationship is established between the dependent document and the document on which the dependent document depends.

2.4 Content model

The content model has the following entities that represents service description artifacts and service description metadata:

- Documents
- Logical models
- User-defined metadata
- Generic objects
- Queries

Archived

Figure 2-13 shows an overview of the different types of metadata stored in WSRR.

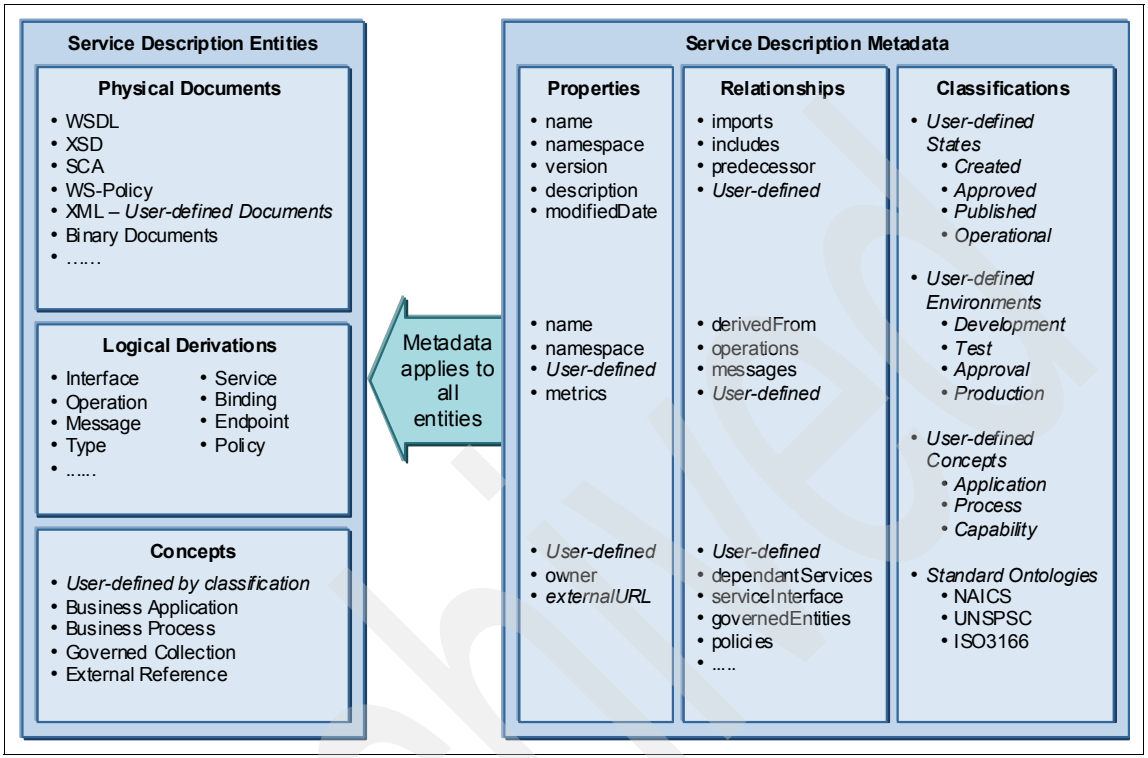


Figure 2-13 Content model overview

All WSRR content elements have a WSRR-assigned URI, a name, and a description.

2.4.1 Service description entities

WSRR stores and manages the following types of service description entities:

- Physical documents
- Logical derivations
- Concepts

You can interact with all types of service description entities, using them in queries, applying service annotations, and establishing relationships from and to them.

Physical documents

The most elemental building blocks for the WSRR content model are service metadata artifact documents (*physical documents*), such as XSD or WSDL files. These service metadata documents are stored and managed in WSRR. The coarse-grained model made from registry objects that represent those documents is referred to as the *physical model*.

Documents are versionable objects in the WSRR content model. Thus, in addition to a URI, name, and description, documents also have a *version property* as shown in Figure 2-14.

Figure 2-14 Version property

The key WSRR metadata document types are:

- XML Schema Definition (XSD)
- Web Services Description Language (WSDL)
- WS-Policy
- Service Component Description Language (SCDL) which is used to define SCA integration modules.

For these document types, WSRR provides special services, including "shredding" of the documents upon receipt into logical derivations.

Other types of XML service metadata can be stored using a generic content type of *XMLDocument*. Documents of XMLDocument type are not decomposed into logical derivations. Other documents in binary data format can also be stored by using a document of type *GenericDocument*. For more details refer to 2.3, "Document types" on page 45.

Archived

Logical derivations

Logical derivations or *logical objects* enable users to explore WSRR content beyond the boundaries of the files that are stored. Logical objects cannot be versioned individually because they are derived from a physical document (which can be versioned). Thus, these objects cannot be manipulated individually. However, logical objects can have additional service description metadata allocated to them, such as properties, relationships, and classifications.

For the key document types, WSRR also defines a few predefined properties, makes an effort to detect relationships to other key documents, and where available, records those relationships in the information model. An XSDDocument, for example, has a targetNamespace property and the relationships importedXSDs, redefinedXSDs, and includedXSDs. When an entry for a key-type document is created in WSRR, it is inspected for relationships to other key-type artifacts. If these related artifacts are not already in represented in WSRR, they are added and, in either case, the relationship between the artifacts is recorded.

The set of logical derivations comprises the logical model of WSRR. The logical model has entities such as portType, port, and message, which are related to WSDL files, and complexType or simpleType, which are related to XSD documents. Elements of the logical model have properties and relationships that reflect a subset of the characteristics as defined in the underlying document. For example, a WSDLService element has a namespace property and a relationship to the ports that it contains.

Note: All the individual results of document parsing are aggregated into one logical model that represents the content of individual documents as well as the relationships between the content in the different documents.

Archived

Figure 2-15 illustrates an example of logical objects derived from a WSDL file.

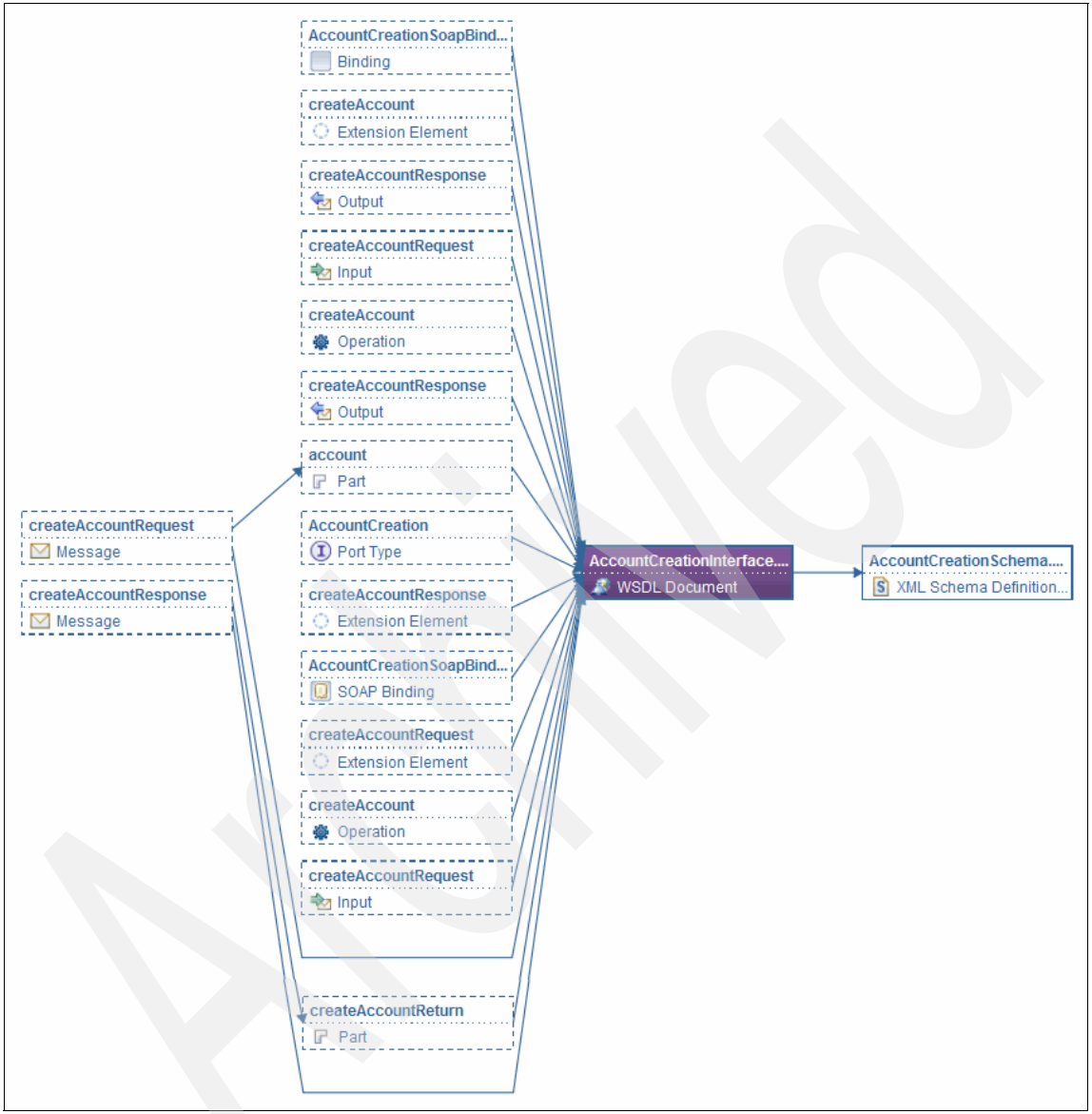


Figure 2-15 Logical derivations example

WSRR stores other types of service metadata using the XMLDocument, a generic document type. Documents of XMLDocument type are not decomposed into the logical model.

Concepts

Concept is the abstract term for the WSRR technical entity called *Generic Object*. It represents an entity held in WSRR that does not have a physical document associated with it. However, it can be augmented with metadata (properties, relationships, and classifications) to describe whatever is required.

> **Note:** *GenericObject* is the API name for the objects that appear in the Web UI as concepts.

Concepts can be used to represent a reference to content in some other metadata repository, such as a portlet in a portlet catalog, an asset in an asset repository, service implementation artifacts kept in a source code library, or information about SOA infrastructure topologies in a configuration management database.

Figure 2-16 shows concepts using the Web UI.

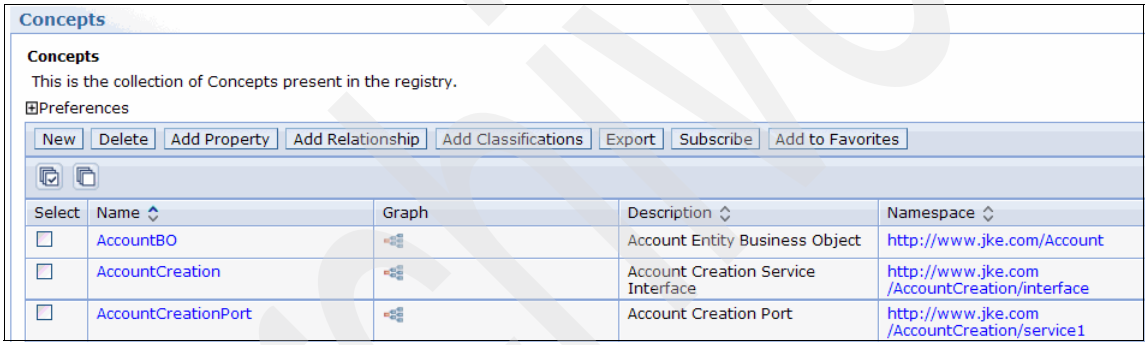


Figure 2-16 Concepts in Web UI

You can use a concept to group physical artifacts together for ease of retrieval. Also, you can use concepts to represent a business-level view on the SOA metadata that is managed in WSRR, including concepts such as Business Process, Business Service, Business Object, and Business Policy.

You can use a generic object in API calls and XPATH expressions to refer to WSRR concepts (or custom entities or collections). For example, the XPATH expression `/WSRR/GenericObject` retrieves all concepts from WSRR.

A concept is simply a container for metadata. There are no constraints as to what a concept can represent. Thus, you must provide additional metadata, such as classifications or templates, to distinguish between different “types” of concepts. It is up to client applications to enforce the interpretation of these types of

concepts. WSRR provides support through templates and validator plug-ins to allow you to extend WSRR to perform this interpretation.

Concepts provide the following predefined properties:

Name	A mandatory name identifying the concept. This name must be unique throughout all objects in the repository when combined with the namespace and version properties.
Namespace	An optional namespace that is used to distinguish similarly named objects.
Version	An optional textual string that identifies different versions of the same name or namespace identified object.
Description	An optional textual description of the concept.

2.4.2 Service description metadata

In addition to content that is directly related to service metadata documents, WSRR supports a number of user-defined metadata types that are used to enhance the service metadata to explain their semantics. These user-defined metadata types are called *service description metadata*.

WSRR supports the following types of service description metadata:

- Properties
- Relationships
- Classifications

You can use all three types to enhance entities in the physical or logical model, as well as concepts.

To allow semantic queries to target individual elements of the service metadata and to perform meaningful dependency analyses before making changes, you can:

- Associate a property businessValue with a physical model entity that represents a WSDL file.
- Define a new relationship makesUseOf between an entity in the logical model that represents a portType and an entity in the physical model that represents an XML document.
- Can create a classifier importantThings and associate it with a port entity in the logical model and with an entity in the physical model that represents a Policy document.

Properties

Properties are simple name and value pairs that are associated with any of the service description entities. Some properties are assigned by the system, such as the unique ID, the owner, and the last time the service entity was changed. These system-assigned properties cannot be changed. Other properties are derived through the parsing of a key-type service description document into its logical model. Properties of this type include name and namespace.

Sometimes, you can change these system-assigned values. You can also create your own properties, which are referred to as *user-defined properties*. You can use user-defined properties such as a simple, unstructured, and untyped extension mechanism.

Note: User-defined properties are referred to as *custom properties* in the Web UI.

WSRR treats general properties and custom properties in the same way and all property values are treated as strings. Properties can be used in queries, and can be used to establish fine-grained access control.

Relationships

In an SOA environment, organizations expect to see reuse and loose flexible coupling between services. Thus, it is essential to able to determine which services depend on what other services (or other service artifacts) at any time in the life cycle of the business applications.

Even at the IT level, services are described using standards-based documents, such as WSDLs and XSDs, which have dependencies between them. As business services progress through their life cycles and evolve to meet market needs, the targets of these dependencies change. It is crucial to keep this information up to date.

Any changes to services in the deployed environment can have an impact on other business applications. You need to track changes in interface versions and compatibility and evaluate the impact of any change.

With WSRR, you can define these relationships and the dependencies between objects as *relationship metadata*. WSRR defines and associates a number of built-in relationships that are defined against the classes of objects that are exposed in the WSRR information model. These relationships are associated primarily with documents and imports as well as the includes between them. These built-in relationships are often created automatically when a document is published and, therefore, are not modifiable by a user.

Relationships tie together one source service description entity to one or more target service description entities. Every relationship is given a name. A source is only allowed to have a single relationship with a given name.

Some relationships are assigned by WSRR during the parsing of key types of documents. One example of a system-assigned relationship is the relationship established between XSD documents based on the importing of one into the other.

WSRR also allows the following examples of custom relationships:

- Relate a GenericObject that represents an external object to a service using a user defined relationship.
- Relate all of the service description documents that will be governed as a unit to a governable entity.
- Relate a monitoring policy to a service endpoint.

These custom relationships can be added or deleted from any object on an instance by instance basis. For custom relationships, the user can define both the name and target objects.

WSRR treats built-in relationships and custom relationships in similar ways for the purposes of manipulation and query.

Classifications

WSRR provides a means of classifying service artifact descriptions represented in WSRR. These *classifications* play a major role in many interactions with WSRR. They allow you to annotate service descriptions and parts of service definitions with a corporate vocabulary and extend the service information model with additional information that is relevant to your particular context. For example, you can use classifications to segregate service endpoints that are deployed in different environments. WSRR also uses classifications to capture the governance state.

Each classification system represents a different way of organizing services and is represented as a hierarchy of classifications similar to a library indexing system. WSRR supports any number of classification systems that can be predefined to organize your service artifacts in the best way.

User-defined category systems are imported and shared through the use of documents encoded by using the *Web Ontology Language* (OWL). Although any valid OWL document can be used as a classifier system, at present, WSRR exploits only a subset of the expressiveness of OWL. It represents OWL classes as classifiers and interprets the `subClassOf` relationship between those classes as establishing a classifier hierarchy. Other OWL concepts, such as data or

relationships that represent properties or other built-in OWL relationships, are ignored.

> **Note:** Using the ampersand (&) character in the URI for a classification system or a class is not supported.

You can modify the structure of a classification system and create a new classification system directly from the Web UI. Any class in the underlying ontology can be used as a classifier. The same classifier can be used to classify multiple entities and an entity can be associated with multiple classifiers.

Ontologies, OWL, and taxonomies can be defined as:

- Ontologies

  An *ontology* is a vocabulary used to describe some domain, which includes describing the entities in the domain as well as their relationships.

- OWL

  OWL is a W3C-endorsed format that can be used to define an ontology. It defines relatively rich semantics, including relations between classes of entities (for example disjointedness), cardinality (for example “exactly one”), equality, properties, characteristics of properties (for example symmetry), and enumerated classes.

  WSRR does not provide any facilities for editing or composing a new OWL file. You can use external tools to produce an OWL file and then load the resulting file into WSRR.

  For more details about OWL, visit:

  `http://www.w3.org/2004/OWL/`

- Taxonomies

  At a more simplistic level, OWL can also describe *taxonomies*. A taxonomy is a system of hierarchical types that describe entities. The types are expressed in a class and subclass system.

  So for example, a simple taxonomy can consist of a class called *Transport*, which can have subclasses of *Air Transport* and *Land Transport*. Then, *Land Transport* can in turn have the subclasses *Bus* and *Car*. This hierarchy taxonomy means that a *Car* is a type of *Land Transport* and is also a type of *Transport*.

Archived

2.4.3 User defined properties and relationships

You can use user-defined properties and relationships to customize the set of predefined properties and relationships that are provided in the WSRR meta-model. You can add properties to a WSDL document, or you can configure a concept with properties and relationships to represent its structure.

2.4.4 Business models templates

Business modelling allows you to represent the objects that are relevant to your organization inside WSRR. The business modelling feature of WSRR:

- Extends the typing function of templates
- Provides more flexibility when modeling your business objects
- Performs additional validation when creating instances of these business objects
- Provides extended typing functions to allow properties and relationships to be validated at the point of create and update

Business model templates provide a basis for creating business model objects. A business model object is a concept that you create from a business model template. The business model template defines property and relationship names that must be included in the concept.

Figure 2-17 shows the collection of Business Model Templates provided with WSRR.

Create New Instance

Select	Name	URI	Description
☐	Enterprise Application	http://www.ibm.com/xmlns/prod/serviceregistry/6/1/TechnicalModel#EnterpriseApplication	Discovered Enterprise Application
☐	Enterprise Module	http://www.ibm.com/xmlns/prod/serviceregistry/6/1/TechnicalModel#EnterpriseModule	Enterprise Application Module
☐	Manual Client Channel Table MQ Endpoint	http://www.ibm.com/xmlns/prod/serviceregistry/6/1/TechnicalModel#ManualClientChannelTableMQEndpoint	Manual Client Channel Table MQ Endpoint
☐	Manual Client MQ Endpoint	http://www.ibm.com/xmlns/prod/serviceregistry/6/1/TechnicalModel#ManualClientMQEndpoint	Manual Client MQ Endpoint
☐	Manual MQ Endpoint	http://www.ibm.com/xmlns/prod/serviceregistry/6/1/TechnicalModel#ManualMQEndpoint	Manual MQ Endpoint

Figure 2-17 Business model templates

2.5 WSRR deployment topologies

WSRR includes two types of deployment topologies:

- Governance registry and repository

  This registry is where all the design-time discovery, development and governance operations are performed. Different roles access this registry to perform their respective development and governance tasks. Metadata for all runtime environments exist in this registry/repository. As the life cycles of the service changes the content is promoted to the respective runtime environments such as the test, pre-production, and production environments.

- Runtime registry

  This registry is used by runtime environments for dynamic endpoint selection, policy resolution and other metadata queries. A limited set of content from the governance registry is promoted to this registry as and when services move through their life cycle, using the WSRR promotion feature. Typically, users do not work directly with this registry, and content is mostly read-only, although additional metadata might be added to registry content, Instead, users work in the governance registry, and content is promoted automatically to the runtime registry at the appropriate point in the governance life cycle.

The number of registries that are deployed and the ways in which they are configured depends on the nature of the SOA environment and the goals of the WSRR deployment project. For example, initially there might be need only for a governance registry repository as an enterprise begins to develop services and establish an SOA Governance process. Eventually, services can be promoted from governance registry and repositories to runtime registry environments.

This section discusses the deployment topologies of the governance registry and repository and the runtime registries.

2.5.1 Recommended deployment topologies

There are two basic recommended deployment topologies, as illustrated in Figure 2-18:

- Pilot deployment topology

  The pilot deployment can be used as a “sandbox” or development environment where the governance and promotion processes are tested. This pilot topology contains two separate governance registries and one runtime registry. The pilot is used for customization of WSRR governance life cycles and profiles. There are two governance instances because one is used for developing the governance processes and the other is used to test the

governance processes before deploying to the production governance instances.

After the new governance and promotion processes are tested, you export and then import the profiles to the respective WSRR instances in the production environment.

The pilot environment does not contain “real” service metadata. Service metadata does not move between the pilot environment and the full production environment.

- Full production deployment topology

  In this topology, information from the single governance registry is promoted to a series of staging environments where different focussed testing can be performed before final deployment to the production runtime registry.

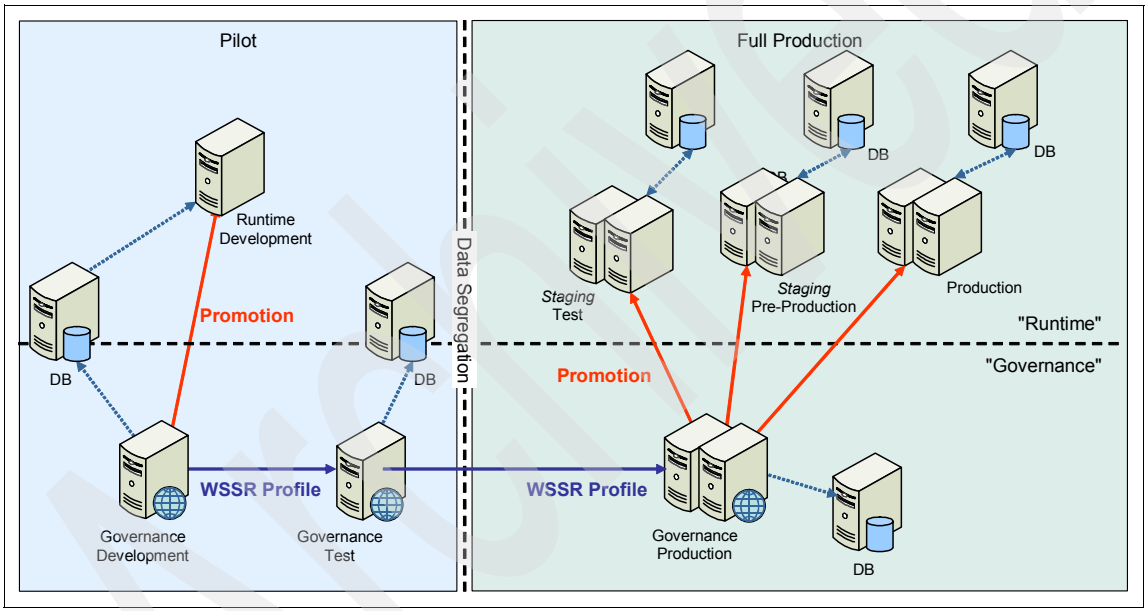


Figure 2-18 Recommended topology

These topologies are applicable even if only a governance registry and repository is required. In this example, the runtime registries illustrated in the figure can be removed.

2.5.2 Runtime deployment topology considerations

This section describes the consideration that must be taken for a runtime deployment topology.

Runtime registries and environments

A service can pass through a series of runtime environments between its initial development and its release into production. For example a development, test, staging and production environment. In a full production deployment, there is one registry for each runtime environment that you want to test in an isolated manner.

The number and types of runtime environments depend on the nature of your development process.

Service promotion

Conceptually, a service moves from one environment to the next when it reaches the appropriate life cycle state as a result of governance operations. In practice, all governance operations are performed on the service in the Governance Registry, and the WSRR promotion feature is used to promote a copy of the service to the next runtime registry in the sequence and, ultimately, the runtime production registry.

Figure 2-19 illustrates the promotion sequence of a service to the various environments.

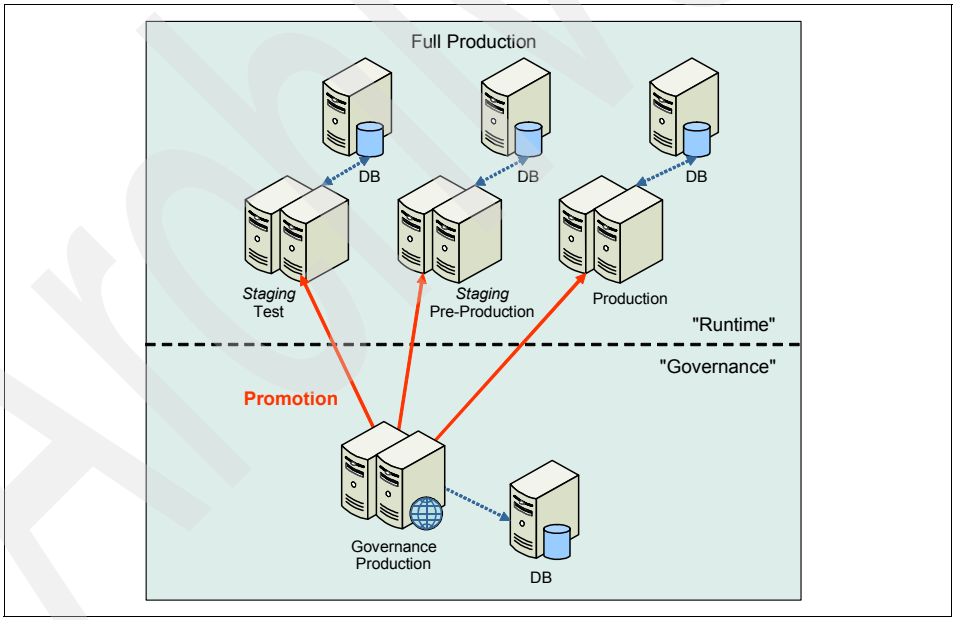


Figure 2-19 Example full production deployment topology

Archived

Synchronization of life cycle states and metadata

Changes to life cycle states and metadata must be performed only in the governance registry. This ensures that the life cycle states and metadata remain synchronized for the separate copies of the same service that exist in the two registries and prevents the possibility of a negative impact on business runtime operations that might result from a change to a service in the runtime registry.

After changing the life cycle state or metadata of a service in the governance registry, re-promote the service to the runtime registry to update the service on the target run time.

Deployment profile configuration

Consideration must be given to the profiles used in each registry in the deployment. Typically, they support different sets of operations (for example, the runtime registry might not support life cycle changes or changes to service metadata). This implies different roles and permissions to be assigned in the runtime registries. The policies, auto-actions and notifications are also different in the runtime registry. For example, the modification plug-ins should be turned off or configured to not run in the runtime registry.

Runtime registry “cleanup”

After a service has reached the end of its useful life and after you have updated the life cycle state in the governance registry and re-promoted the service accordingly, you eventually delete the service from the runtime registry as part of general “cleanup” operations. Thus, the sequence of actions involved in governing a service can be summarized as follows:

1. Publish the service to the governance registry, or discover the service from another application environment (for example WebSphere Application Server) using the WSRR service discovery mechanism.
2. Govern the service through the development life cycle.
3. Deploy the service to production. The service is promoted automatically to the runtime registry.
4. Retire the service. The service is re-promoted to the runtime registry to update its life cycle state. You can then delete the service from the runtime registry using a scheduled task for cleanup.

2.6 Deployment configurations

When installing WSRR the deployment configuration of both the database and of WebSphere Application Server must be considered, as discussed in this section.

2.6.1 WebSphere Application Server configurations

There are different configurations of WebSphere Application Server and WSRR to consider. You can install WSRR in the following deployment configurations:

- Stand-alone
- Federated node
- Cluster

> **Note:** A remote database is mandatory for a cluster.

Stand-alone deployment configuration

This simple configuration the WSRR installer expects that the application server and database are installed on a single node (a single computer) in a stand-alone application server, as illustrated in Figure 2-20. A stand-alone node contains a database in addition to WebSphere Application Server, which includes an application server that contains WSRR.

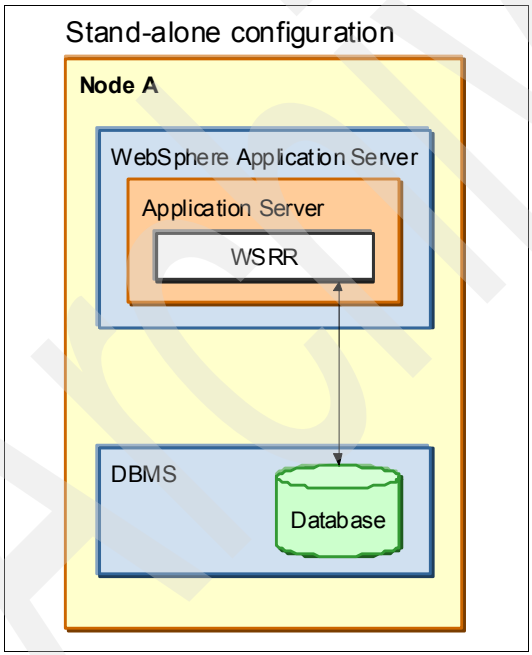


Figure 2-20 Stand-alone configuration

Archived

Note: You can also configure the stand-alone environment with a remote database. In this case, use the deployment scripts instead of the deployment wizard. For more information, see:

http://publib.boulder.ibm.com/infocenter/sr/v6r3/topic/com.ibm.sr.doc/twsr_installn15.html

Federated node deployment configuration

In this configuration, the WebSphere Application Server is federated into a WebSphere Application Server cell that is managed by the WebSphere Application Server Deployment Manager. Figure 2-21 shows two separate WSRR systems each within its own cell. The WSRR configuration can be a stand-alone or a remote database.

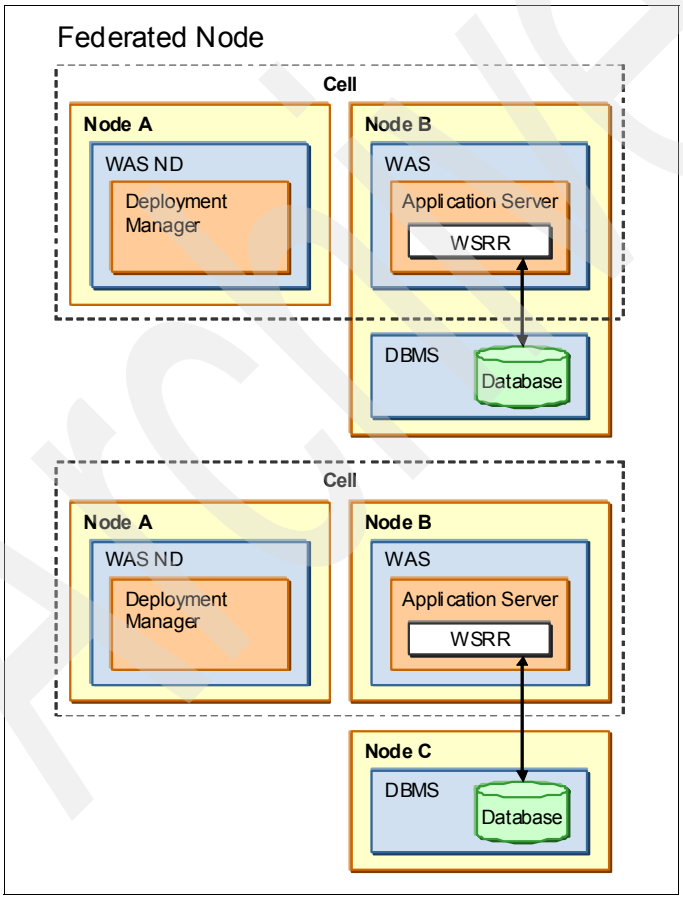


Figure 2-21 Federated node configuration

Cluster deployment configuration

In this configuration, WSRR is installed as a WebSphere Application Server cluster, as illustrated in Figure 2-22. The configuration must use a remote database, not a stand-alone database.

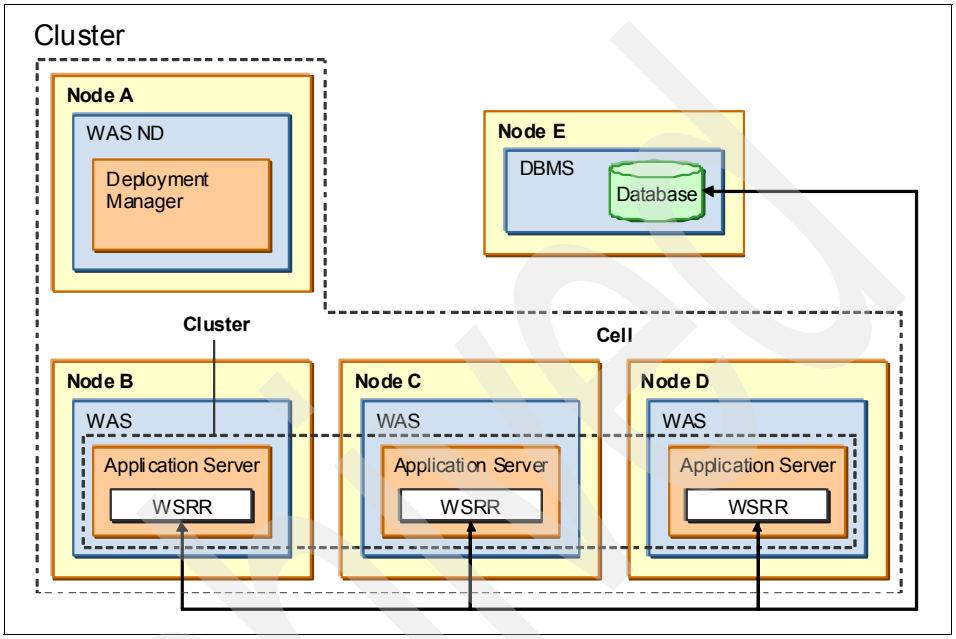


Figure 2-22 Clustered configuration

Archived

2.6.2 Database configuration

This section describes the possible database configurations.

Local database configuration

In this configuration the database is installed on the same node as the application server, as illustrated in Figure 2-23.

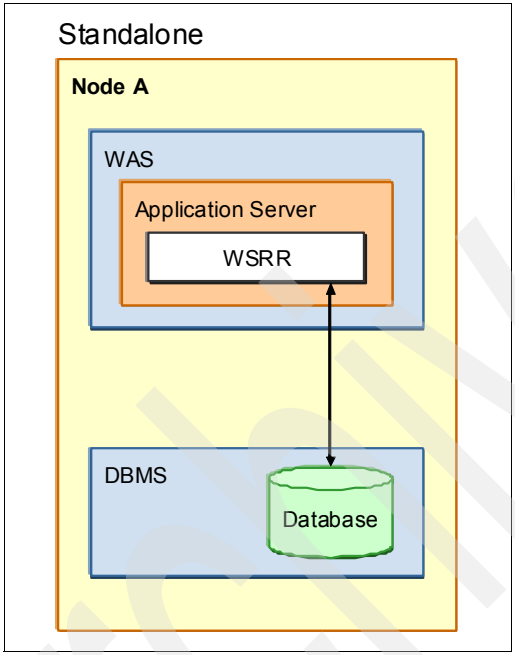


Figure 2-23 Local database configuration

Archived

Remote database configuration

In this configuration, the database is installed on a separate node (usually a separate physical computer) to the application server, as illustrated in Figure 2-24.

> **Note:** To verify which platforms can be the targets for remote databases, see the system requirements at:
>
> http://www.ibm.com/software/integration/wsrr/sysreqs/index.html

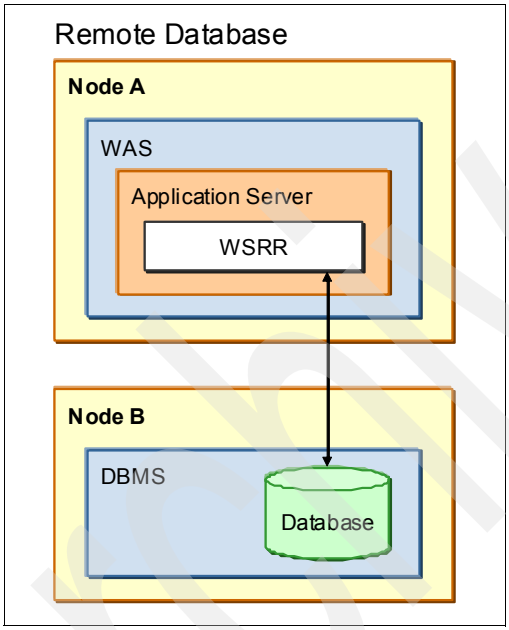


Figure 2-24 Remote database configuration

2.7 Packaging

WebSphere Service Registry and Repository V6.3 ships with the following major installation components, which are provided in separate installation files:

- Server component
- Client component
- WebSphere Service Registry and Repository Studio

WebSphere Service Registry and Repository Studio software is installed using IBM Installation Manager.

Archived

You can find more detailed requirements about the supported operating systems and hardware requirements on the WSRR support Web site:

http://www.ibm.com/software/integration/wsrr/sysreqs/index.html

Archived

3

Introduction to the governance enablement profile

This chapter provides an overview of the governance enablement profile.

Archived

© Copyright IBM Corp. 2009. All rights reserved.

3.1 Overview of the governance enablement profile

The governance enablement profile is a WebSphere Service Registry and Repository profile that provides a starting point to manage services from the initial specification through the deployment in production in a service-oriented architecture (SOA) environment. This profile is installed and activated by default during the deployment of WebSphere Service Registry and Repository.

The governance enablement profile consists of the following components:

- Models

  Entities that represent the objects in an organization's SOA environment

- Roles

  A predefined set of roles into which users can be assigned according to their SOA tasks

- Life cycles

  A predefined set of states and transitions that are applied to artifacts to indicate their progress in the development process

In this chapter, we describe these components.

3.2 Models in the governance enablement profile

The governance enablement profile provides entities using *models* that can be used to represent the service description artifacts and metadata of an organization's SOA environment. The governance enablement profile provides the following models:

- Governance enablement model

  This model provides entities that can be used by an organization to represent SOA entities and their properties and relationships.

- Service model

  When Web Service Definition Language (WSDL), XML Schema Definition (XSD), and Service Component Architecture (SCA) modules and MQ artifacts are imported into WebSphere Service Registry and Repository, entities are created. These derived (logical) entities are represented as entities in the service model.

Archived

- Governance enablement profile extensions model

  This model provides entities that are used as examples for the recommended approach for extending the default service level agreement (SLA) and service level definition (SLD) entities.

- Advanced Lifecycle Edition model

  This model provides entities that represent the core entities in Rational Asset Manager to facilitate synchronization between Rational Asset Manager and WebSphere Service Registry and Repository.

For more information, go to the information center:

http://publib.boulder.ibm.com/infocenter/sr/v6r3/topic/com.ibm.sr.doc/rwsr_gep_models.html

Figure 3-1 depicts the entities in respect to the various of layers of governance needed throughout the enterprise.

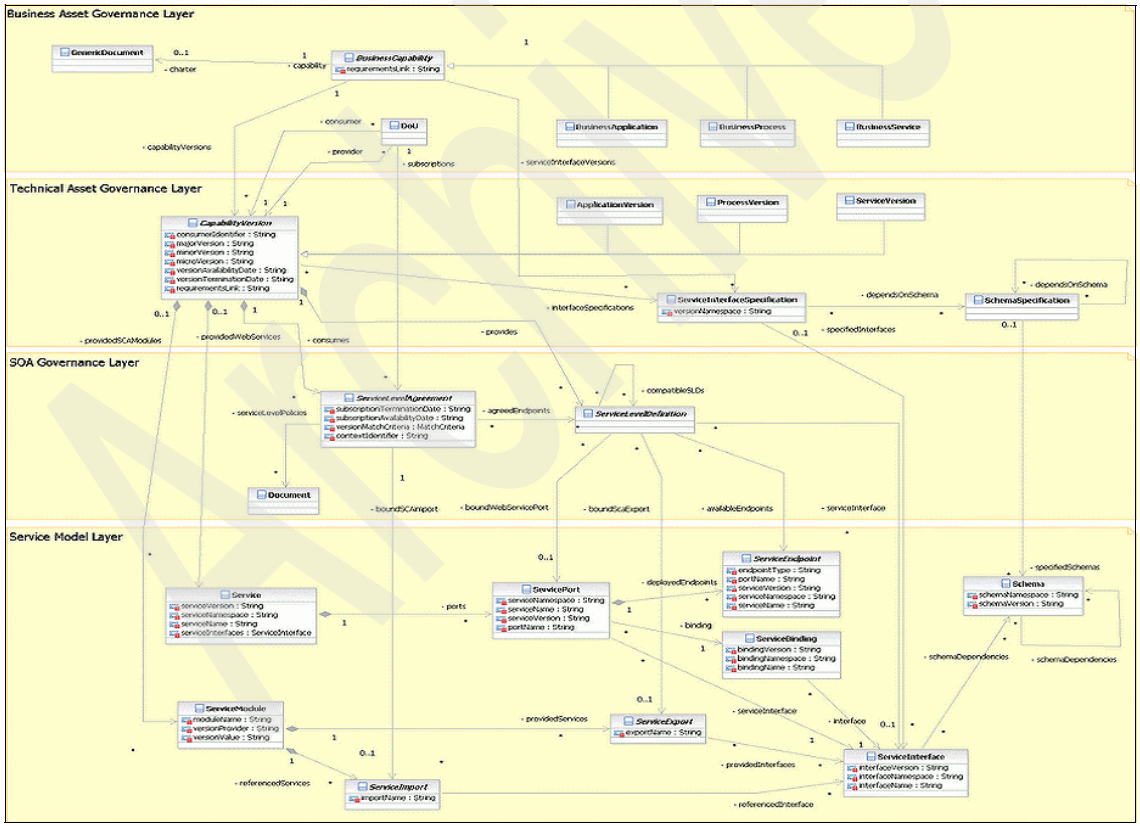


Figure 3-1 Governance model entities

Archived

In this section, we provide details about the properties and relationships for each entity in the governance enablement model and specific entities in the service model and Advanced Lifecycle Edition model.

3.2.1 Asset entity

An *asset* entity in the governance enablement profile represents any object type in WebSphere Service Registry and Repository that might correspond to an asset in Rational Asset Manager or other federated repository. If a WebSphere Service Registry and Repository entity is defined as an asset, that entity can also use Rational Asset Manager or WebSphere Service Registry and Repository Advanced Lifecycle Edition capabilities to govern that asset.

An asset entity is found in the Advanced Lifecycle Edition model. Figure 3-2 depicts the owning and reverse relationships that an asset entity has with other entities in the governance enablement profile.

Note: The dependency relationship is used only if the profile does not provide a more relevant relationship to represent the dependency.

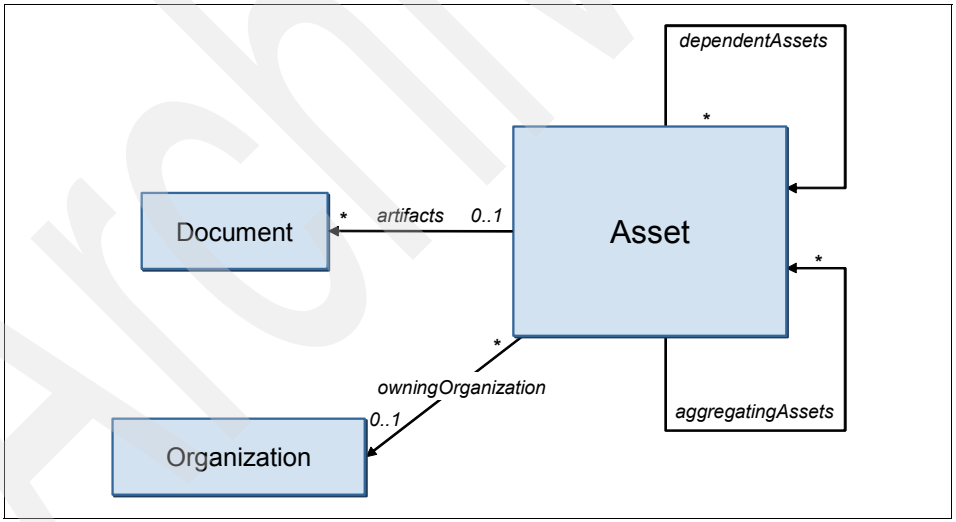


Figure 3-2 Asset entity relationship

Table 3-1 describes the properties that are defined for an asset entity from the Advanced Lifecycle Edition model.

Table 3-1 Asset entity properties

Displayed property name	Actual property name	Description	Type
Name	name	A required descriptive name for the entity	String
Description	description	A textual description of the entity	String
Namespace	namespace	The XML schema namespace for the entity	String
Version	version	The version of the entity	String
GUID	ale63_guid	The globally unique identifier used to identify the asset in Rational Asset Manager	String
Full description	ale63_fullDescription	A full textual description of the entity held in Rational Asset Manager	String
Asset type	ale63_assetType	The specific asset type used in Rational Asset Manager to determine the type, and therefore the properties and relationships of the entity	String
Asset Web link	ale63_assetWebLink	A URL that, if followed, opens a browser window in Rational Asset Manager showing the details of this particular entity	String
Remote state	ale63_remoteState	The state of the asset in Rational Asset Manager	String
Asset owners	ale63_assetOwners	Those users that have ownership permissions or roles in Rational Asset Manager	String
Owner e-mail	ale63_ownerEmail	A default e-mail address to use for all inquiries or notifications about this Asset entity	String
Community name	ale63_communityName	The community that this asset belongs to in Rational Asset Manager	String

Archived

Table 3-2 describes the owned relationships that are defined for an asset entity in the Advanced Lifecycle Edition model.

Table 3-2 Asset entity owned relationships

Owned relationship	Relationship name	Description	Type	Cardinality
Dependency	ale63_dependency	A non-specific dependency of this asset on another asset	Asset	0,*
Artifacts	ale63_artifacts	Those documents that form part of the asset	Document	0,*
Owning organization	ale63_owningOrganization	The organization to which this asset belongs	Organization	0,1

Table 3-3 describes the reverse relationships that are defined for an asset entity as defined in the Advanced Lifecycle Edition model.

Table 3-3 Asset entity reverse relationships

Reverse relationship	Relationship name	Description	Type	Cardinality
Dependent assets	ale63_dependency	Assets that depend on this asset	Asset	0,*
Aggregating assets	ale63_aggregation	Assets that aggregate this asset	Asset	0,*

The business capability, capability version, document of understanding (DOU), schema specification and service interface specification entities inherit properties and relationships from the asset entity.

3.2.2 Business capability entity

A *business capability* entity in the governance enablement profile represents a construct that expresses a generalized (unrealized) feature within the service-oriented architecture (SOA) environment. This entity is used as the starting point for linking business requirements with technical functionality in the SOA environment.

The following model types inherit their properties and relationships from this entity:

- Business application: An existing application or a particular channel to market, such as a Web or Portal application
- Business process: A collection of related activities or tasks in an organization
- Business service: An unrealized service in the organization

Figure 3-3 depicts the owning and reverse relationships that a business capability entity has with other entities in the governance enablement profile model.

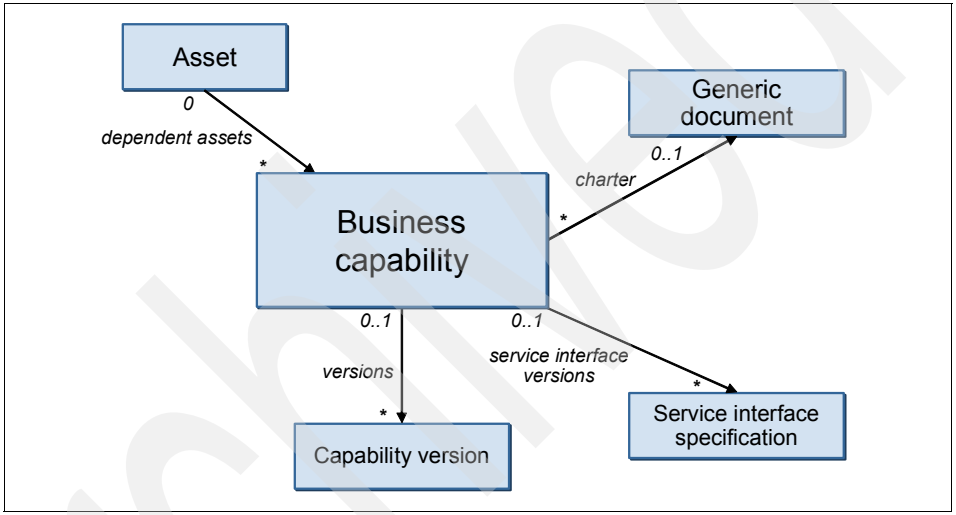


Figure 3-3 Business capability model

This entity inherits properties and relationships from the asset entity plus the additional properties (shown in Table 3-4) as defined in the governance enablement profile model.

Table 3-4 Business capability properties

Displayed property name	Actual property name	Description	Type
Business requirements link	ale63_requirementsLink	Follow this link to view business requirements allocated to this capability	String

Table 3-5 describes the owning relationships that are defined for the business capability entity by the governance enablement profile model.

Table 3-5 Business capability entity owned relationships

Owned relationship	Relationship name	Description	Type	Cardinality
Charter	gep63_charter	A human readable document that defines a particular business capability	Binary document	0,1
Service interface versions	gep63_serviceInterfaceVersions	All service interface specifications that can be used to gain access to this capability	Service interface specification	0,*
Versions	gep63_capabilityVersions	All versions of this business capability that have been, or are being, developed and deployed	Capability version	0,*

Table 3-6 describes the reverse relationships that are defined for the business capability entity in the governance enablement profile model.

Table 3-6 Business capability entity reverse relationships

Reverse relationship	Relationship name	Description	Type	Cardinality
Dependent assets	ale63_dependency	Assets that depend on this asset	Asset	0,*

3.2.3 Capability version entity

A *capability version* entity in the governance enablement profile defines a specific version of a feature to realize specific versions of the business capability entities (application, process, and service). This entity also specifies the following main characteristics for a particular version:

- A set of SLDs that are the formal specification for any provided services, including both functional and quality of service (QoS) characteristics
- A set of SLAs that are the agreed specification for any consumed services and reference the particular SLD that must be used for any interactions
- A definition of the SCA modules and Web services that deliver the capability in this particular version.

Archived

The following model types inherit properties and relationships from this entity:

- Application version

  A specific version or release of a Web application that only consumes services

- Process version

  A specific version or release of an SOA process that can expose some of its capabilities as services with appropriate SLDs

- Service version

  A specific version or release of a business service that provides a range of functional and non-functional specifications

Figure 3-4 depicts the owning and reverse relationships that a capability version entity has with other entities in the governance enablement profile model.

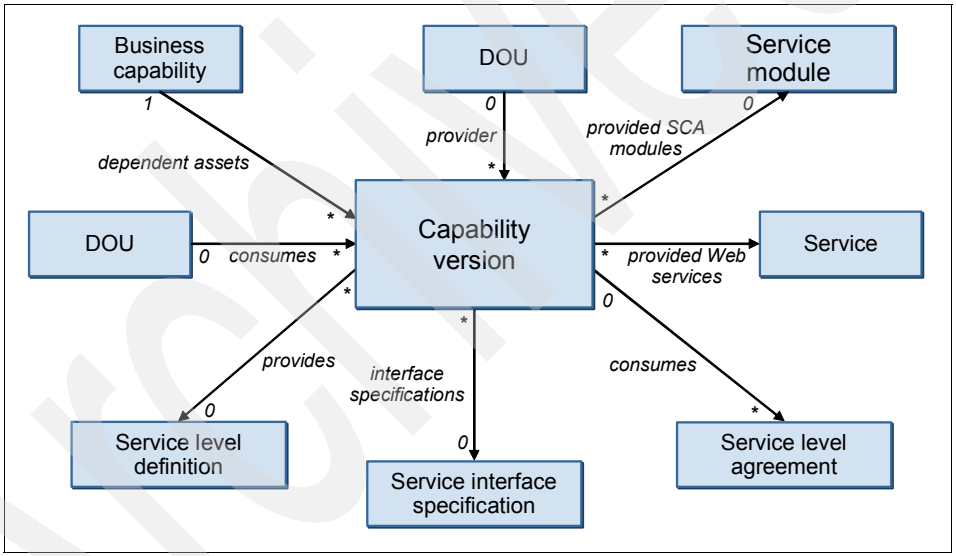


Figure 3-4 Capability version entity

Archived

This entity inherits properties and relationships from the business capability entity plus the additional properties (shown in Table 3-7) as defined in the governance enablement profile model.

Table 3-7 Capability version properties

Displayed property name	Actual property name	Description	Type
Consumer identifier	gep63_consumerIdentifier	Provides a key that is used to identify the consuming capability version in any interaction where the version has subscribed to another provider or capability	String
Version availability date	gep63_versionAvailabilityDate	The expected date that this capability is will be operational	Date
Version termination date	gep63_versionTerminationDate	The expected date that this version of the capability will be retired from operational use	Date
Version requirements link	ale63_requirementsLink	Follow this link to view requirements allocated to this version	URI

During any interactions between the consumer and provider, if the consumer identifier is included in the interaction header, the provider enterprise service bus (ESB) can identify the particular capability version that is invoking the endpoint.

Table 3-8 describes the owning relationships that are defined for a capability version in the governance enablement profile model.

Table 3-8 Capability version entity owned relationships

Owned relationship	Relationship name	Description	Type	Cardinality
Interface specifications	gep63_interfaceSpecifications	All service interface specifications that can be used to gain access to this capability	Service interface specification	0,*
Provided SCA modules	gep63_providedSCAModules	The SCA modules that are provided by this capability	Service module	0,*
Consumes	gep63_consumes	All the dependencies that this capability version has on other providers	SLA	0,*

Owned relationship	Relationship name	Description	Type	Cardinality
Provides	gep63_provides	All SLDs that this version support	SLD	0,*
Provided Web services	gep63_providedWebServices	All Web services provided by this capability	Service	0,*

SCA modules declare the endpoints that they import (*imports*) and the endpoints that they provide (*exports*). This relationship allows the capability version entity to be mapped to the provided and consumed endpoints from the SCA module. The SCA module forms part of the SCA description of the service or process and is expressed in Service Component Definition Language (SCDL) during the development of the capability.

The *consumes* relationship also defines the endpoints as well as the QoS for each SLA entity that is used in any interaction with the provider.

The *provides* relationship also defines the formal definition of the interaction and non-functional characteristics for any interaction, including the physical communication mechanisms that are used to deliver the messages for interaction with the provided services, as well as the QoS characteristics that are related to the interaction, such as security and identity.

The *provided Web services* relationship corresponds to the specific versions of services that are declared in WSDL as part of the capability version (service version, process version and application version) entity. From the service, consumers can navigate to the bindings and interfaces that are supported for this particular Web service.

Archived

Table 3-9 describes the reverse relationships that are defined for a capability version entity in the governance enablement profile model.

Table 3-9 Capability version entity reverse relationships

Reverse relationship	Relationship name	Description	Type	Cardinality
Chartered Business Capability	gep63_capabilityVersions	The business capability that this version realizes	Business capability	1,*
Consumer DOUs	gep63_consumer	The DOUs for which this version of the capability is a consumer	DOU	0,*
Provider DOUs	gep63_provider	The DOUs for which this version of the capability acts as a provider	DOU	0,*

3.2.4 Document of understanding entity

A *document of understanding* (DOU) entity in the governance enablement profile represents an agreement between organizations that details how the consuming organization's realization of their capability (capability version entity) uses the providing organization's realized capability. Figure 3-5 depicts the owning relationships that a DOU entity has with other entities in the governance enablement profile model.

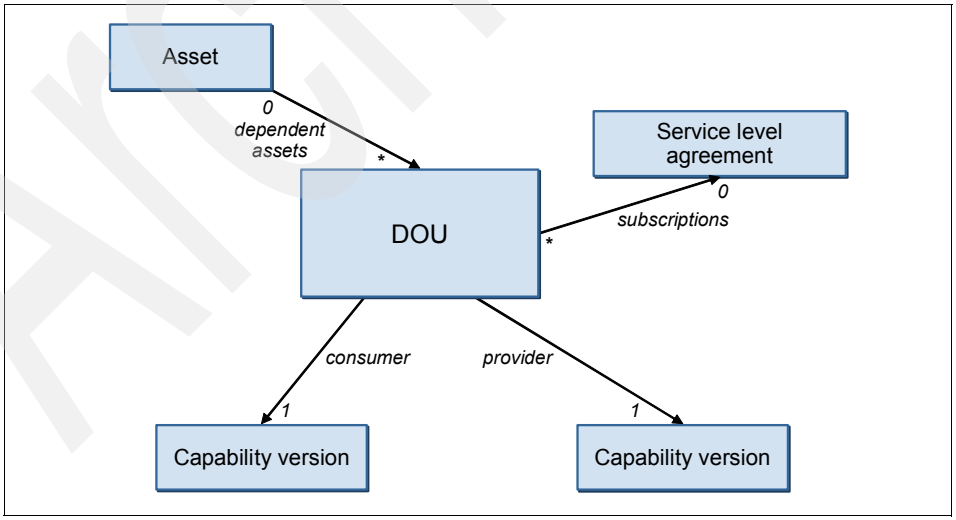


Figure 3-5 DOU model

A DOU entity inherits properties and relationships from the asset entity as well as the additional relationships shown in Table 3-10.

Table 3-10 DOU entity owned relationships

Owned relationship	Relationship name	Description	Type	Cardinality
Subscriptions	gep63_subscriptions	The SLAs that realize this DOU	SLA	0,*
Consumer	gep63_consumer	The capability version that consumes services from the provider capability version under the terms of the DOU	Capability version	1
Provider	gep63_provider	The capability version that provides services to the consumer capability version under the terms of the DOU	Capability version	1

3.2.5 Schema specification entity

A *schema specification* entity in the governance enablement profile defines shared schema documents, which allow the schema specification entities to be governed independently. Figure 3-6 depicts the owning relationships that a schema specification entity has with other entities in the governance enablement profile model.

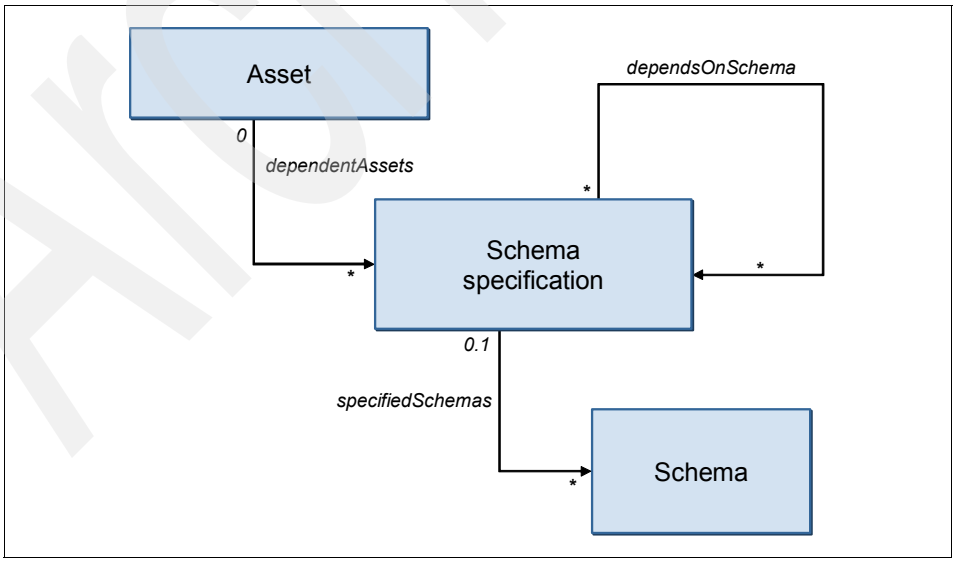


Figure 3-6 Schema specification model

Archived

A schema specification entity has the same properties and relationships as an asset entity as well as the additional owned relationships shown in Table 3-11.

Table 3-11 Schema specification entity owned relationships

Owned relationship	Relationship name	Description	Type	Cardinality
Specified schemas	gep63_specifiedSchemas	The schemas and the associated namespaces that this schema specification entity defines	Schema	0,*
Depends on schema	gep63_dependsOnSchema	A dependency of this asset on another schema specification entity	Schema specificatio n	0,*

3.2.6 Service interface specification entity

A *service interface specification* entity in the governance enablement profile defines a particular interaction pattern and message structure that is supported throughout the realized versions of business capabilities. This entity is independent of any transport or QoS. The interface specification usually refers to a set of WSDL port types that are shared throughout the capability versions.

Interface version numbers have the format *major.minor*. A single major interface version represents a backward compatible set of minor interface versions, where each minor version extends only the existing interface and makes no changes to interfaces or operations that are declared in earlier minor versions.

Archived

Figure 3-7 depicts the owning and reverse relationships that a service interface specification entity has with other entities in the governance enablement model.

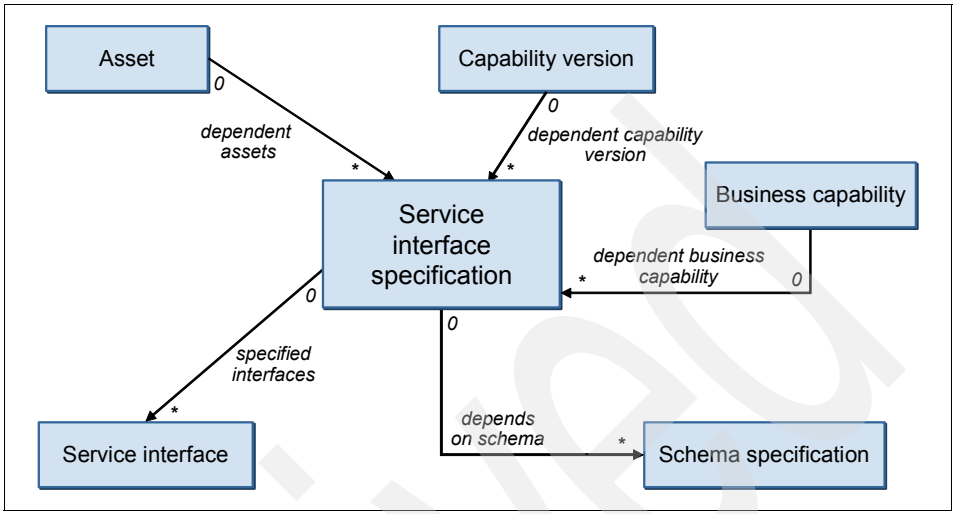


Figure 3-7 Service interface specification model

A service interface specification entity has the same properties and relationships as an asset entity as well as the additional owned relationships shown in Table 3-12 as defined by the governance enablement profile model.

Table 3-12 Service interface specification entity owned relationships

Owned relationship	Relationship name	Description	Type	Cardinality
Depends on schema	gep63_dependsOnSchema	A dependency of this service interface specification entity on a schema specification entity	Schema specification	0,*
Specified interfaces	gep63_specifiedInterfaces	The service interfaces (WSDL port types) that this service interface specification entity defines	Service interface	0,*

Archived

Table 3-13 describes the defined reverse relationships for a service interface specification entity for the governance enablement profile model.

Table 3-13 Service interface specification entity reverse relationships

Reverse relationship	Relationship name	Description	Type	Cardinality
Dependent Business Capabilities	gep63_serviceInterfaceVersions	Business capabilities that depend on this service interface specification entity	Business capability	0,*
Dependent Capability Versions	gep63_interfaceSpecifications	Capability versions that depend on this service interface specification entity	Capability version	0,*

3.2.7 Service level agreement entity

A *service level agreement* (SLA) entity in the governance enablement profile defines a specific dependency that a capability version entity (for example a service version, process version, or application version) has on a particular SLD that is provided by another service version. Although the SLA specifies a particular SLD, the implication is that backward compatible SLDs can be substituted and will support the same SLAs.

Figure 3-8 depicts the owned and reverse relationships that an SLA entity has with other entities in the governance enablement profile model.

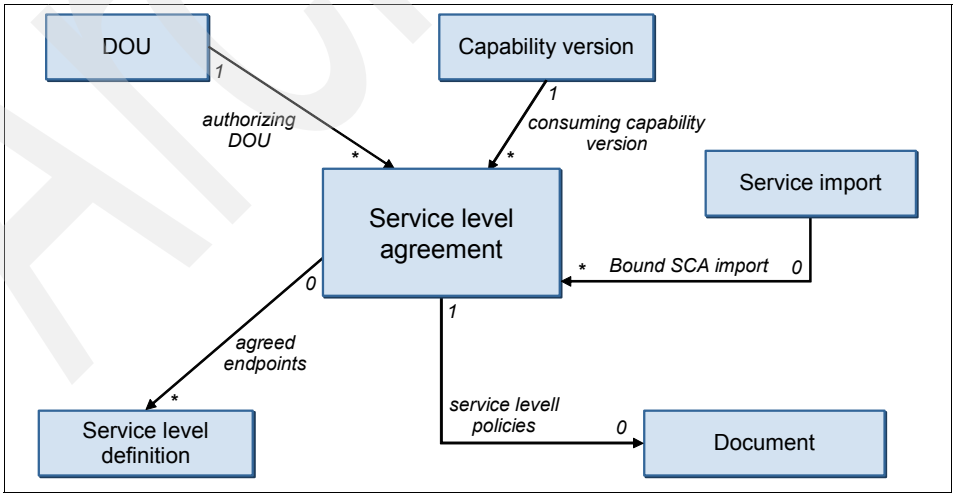


Figure 3-8 SLA model

Table 3-14 describes the properties that are defined for an SLA entity in the governance enablement model.

Table 3-14 SLA entity properties

Displayed property name	Actual property name	Description	Type
Name	name	Descriptive name of the SLA entity	String
Description	description	Textual description of the SLA entity	String
Context identifier	gep63_contextIdentifier	Alternate SLA entities that must be used when invoking the same provided service or endpoint (SLD)	String
SLA availability date	gep63_subscriptionAvailabilityDate	The date from which subscribed service providers will be available	Date
SLA termination date	gept63_subscriptionTerminationDate	The date after which subscribed service providers will no longer be available	Date
Version match criteria	gep63_versionMatchCriteria	The possible values are: ► LatestCompatibleVersion, which means that the subscription must use the latest available version that supports the required SLA ► ExactVersion, which means that the subscription uses only the associated specification or SLD	String enumeration list

Archived

Table 3-15 describes the owned relationships that are defined for the SLA entity by the governance enablement model.

Table 3-15 SLA entity owned relationships

Owned relationship	Relationship name	Description	Type	Cardinality
Service level policies	gep63_serviceLevelPolicies	Documents (or policies) that form part of the SLA when the consuming capability uses the endpoints that are agreed in this subscription	Document	0,*
Bound SCA import	gep63_boundSCAimport	Ties the subscription to a specific SCA import that is defined in this version of the capability	Service import	0,1
Agreed endpoints	gep63_agreedEndpoints	The specific SLD in the providing capability that this SLA is intended to use; the SLD allows any endpoint addresses that realize that SLD to be located	SLD	0,*

Table 3-16 describes the reverse relationships that are defined for an SLA entity in the governance enablement model.

Table 3-16 SLA entity reverse relationships

Reverse relationship	Relationship name	Description	Type	Cardinality
Consuming Capability Version	gep63_consumes	Capability version that consumes this SLA	Capability version	1
Authorizing DOU	gep63_subscriptions	The DOU that defines the underlying agreement for this SLA	DOU	1

Extended SLA entity

The *Extended SLA* entity in the governance enablement profile represents the details of a specific service subscription and defines the QoS properties that are used in any interactions between the consuming capability version and the

agreed provider endpoints. An Extended SLA entity has the same properties and relationships as an SLA entity as well as the additional properties listed in Table 3-17 as defined by the governance enablement profile model.

Table 3-17 Extended SLA entity properties

Displayed property name	Actual property name	Description	Type
Average messages per day	gpx63_averageMessagesPerDay	The average number of messages or transactions over a one day period	int
Maximum messages per day	gpx63_maximumMessagesPerDay	The maximum number of messages or transactions in any one day period	int
Minimum messages per day	gpx63_minimumMessagesPerDay	The minimum number of messages or transactions in any one day period	int

3.2.8 Service level definition entity

The *service level definition* (SLD) entity in the governance enablement profile provides a formal specification of the physical communication mechanisms that are used to deliver the messages for interaction with a provided service and includes non-functional characteristics that are related to the interaction, such as security and identity. Examples of an SLD include:

- For a Web service, the WSDL port (with some policy assertions using WS-Policy syntax), the binding (and policies), and the relationship to the service interface (port type).
- For an SCA export, the export (with some policy assertions using WS-Policy syntax), the SCA binding that is defined in the export, and the relationship to the service interface.

In either case, a list of callable and interchangeable endpoints, which support the SLD entity, is provided.

Archived

Figure 3-9 depicts the owned and reverse relationships that an SLD entity has with other entities in the governance enablement profile model.

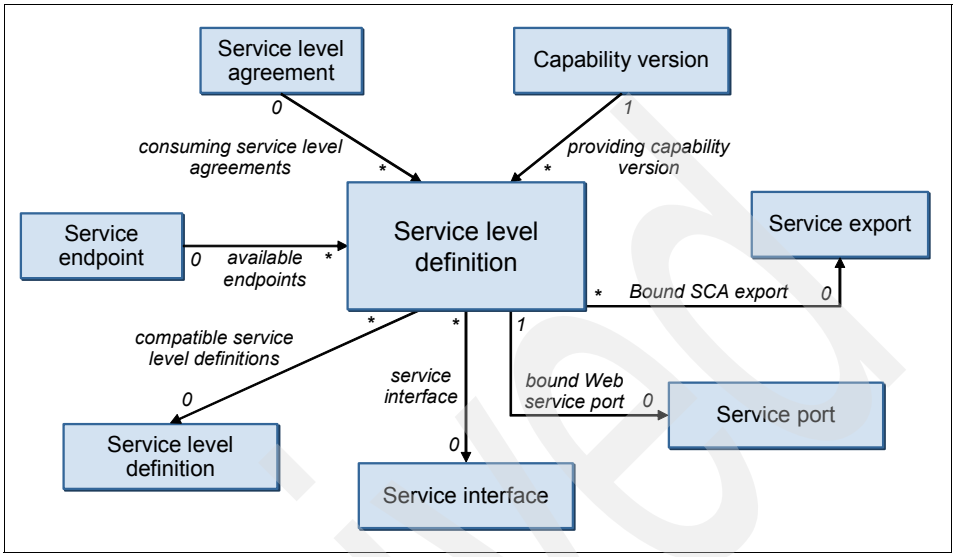


Figure 3-9 SLD model

Table 3-18 describes the properties that are defined for an SLD entity in the governance enablement model.

Table 3-18 SLD entity properties

Displayed property name	Actual property name	Description	Type
Name	name	Descriptive name of the SLD	String
Description	description	Textual description of the SLD	String

Archived

Table 3-19 describes the owned relationships that are defined for an SLD entity in the governance enablement model.

Table 3-19 SLD entity owned relationships

Owned relationship	Relationship name	Description	Type	Cardinality
Available endpoints	gep63_availableEndpoints	All deployed, callable, endpoints that realize the SLD and its associated SLAs	Service endpoint	0,*
Compatible SLDs	gep63_compatibleSLDs	Other interaction specifications whose SLAs can be directly supported by this SLD	SLD	0,*
Bound web service port	gep63_boundWebServicePort	The specific web service port in the deployed capability to which this SLD corresponds to	Service port	0,1
Service interface	gep63_serviceInterface	The specific service interface that this SLD supports	Service interface	0,*
Bound SCA export	gep63_boundScaExport	The specific export in the deployed capability to which this SLD corresponds	Service export	0,1

Table 3-20 describes the reverse relationships that are defined for an SLD entity in the governance enablement profile model.

Table 3-20 SLD entity reverse relationships

Reverse relationship	Relationship name	Description	Type	Cardinality
Consuming SLAs	gep63_agreedEndpoints	SLAs that reference this SLD	SLA	0,*
Providing Capability Version	gep63_provides	The capability version that provides this SLD	Capability version	1

Extended SLD entity

The *Extended SLD* entity in the governance enablement profile provides additional QoS information for a subscribable endpoint. The Extended SLD entity

can be used to define particular metrics that are associated with endpoints. An Extended SLD entity has the same properties and relationships as an SLD entity as well as the additional properties listed in Table 3-21.

Table 3-21 Extended SLD entity properties

Displayed Property name	Actual property name	Description	Type
Average response time	gpx63_averageResponseTime	The typical response time, in milliseconds, in an interaction with this endpoint (or SLD)	int
Availability	gpx63_availability	The level of availability that consumers can expect from this endpoint	String enumeration list. The possible values are: ► 24/7 High Availability ► Working Hours Only ► As Available (no guarantee)

3.2.9 Service entity

A *service* entity in the governance enablement profile represents a specific WSDL service that is defined by the service namespace, service name, and version. This entity identifies and collects all service ports, and thus endpoints, that are available with that particular service version. A service entity is defined in the service model.

Archived

Figure 3-9 depicts the owned relationships that a service entity has with other entities in the governance enablement model.

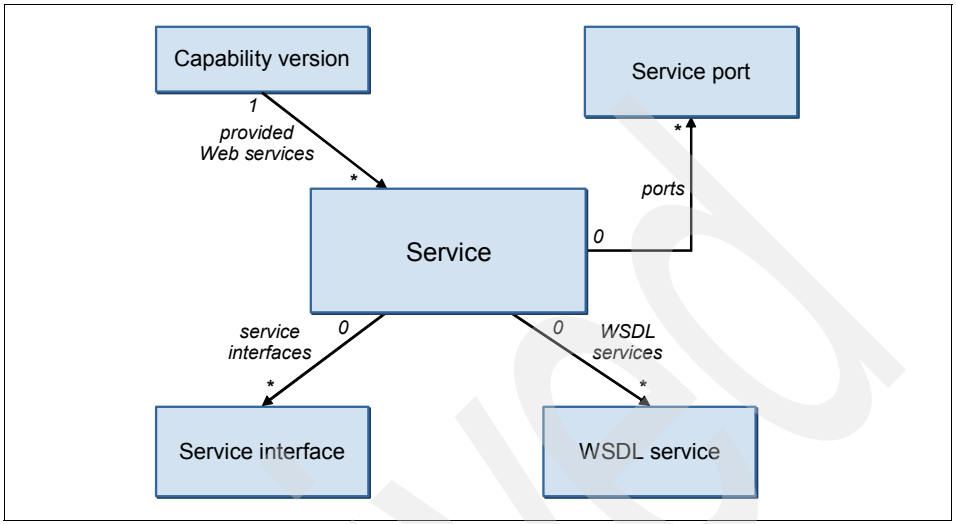


Figure 3-10 Service model

Table 3-22 describes the service entity properties that are defined in the service model.

Table 3-22 Service entity properties

Displayed property name	Actual property name	Description	Type
Name	name	A descriptive name for the service	String
Description	description	A textual description of the service	String
Namespace	namespace	The XML schema namespace for the service	String
Version	version	The version of the service	String
Service name	sm63_serviceName	The name of the derived WSDL service entity that is represented by this service	String
Service namespace	sm63_serviceNamespace	The namespace of the derived WSDL service entity that is represented by this service	String
Service version	sm63_serviceVersion	The version of the derived WSDL service entity that is represented by this service	String

Archived

Table 3-23 describes the owned relationships for a Service entity as defined in the Service model.

Table 3-23 Service entity owned relationships

Owned relationship	Relationship name	Description	Type	Cardinality
Service interfaces	sm63_serviceInterfaces	The correlated WSDL port type entities that are used by this service	Service interface	0,*
WSDL services	sm63_wsdlServices	The derived WSDL Service entities that are represented by this service	WSDL service	0,*
Ports	sm63_ports	The correlated WSDL port entities that are contained by this service	Service port	0,*

The derived WSDL service targets are created when a WSDL document is loaded.

3.2.10 Service endpoint entity

A *service endpoint* entity in the governance enablement profile represents a distinct deployment of a named service port and provides the basic means of governing access to individual service endpoints. The following model types inherit their properties and relationships from this entity:

- SOAP service endpoint, which represents the governed service endpoint for any services that use SOAP addressing to define their endpoints
- MQ service endpoint, which represents the governed service endpoint for all types of MQ service
- Extension service endpoint, which represents the governed service endpoint for any services that use WSDL extensions to define their endpoints

A service endpoint entity is defined in the service model. Table 3-24 describes properties that are defined for a service endpoint entity in the service model.

Table 3-24 Service endpoint entity properties

Displayed property name	Actual property name	Description	Type
Name	name	A descriptive name for the service endpoint	String

Displayed property name	Actual property name	Description	Type
Description	description	A textual description of the service endpoint	String
Namespace	namespace	The XML schema namespace for the service endpoint	String
Version	version	The version of the service endpoint	String
Endpoint type	sm63_endpointType	The type of this correlated endpoint entity	String
Port name	sm63_portName	The name of the containing correlated service port entity	String
Service version	sm63_serviceVersion	The version of the containing correlated service entity	String
Service namespace	sm63_serviceNamespace	The namespace of the containing correlated service entity	String
Service name	sm63_serviceName	The name of the containing correlated service entity	String

Table 3-25 describes the reverse relationships that are defined for a service endpoint entity in the service model.

Table 3-25 Service endpoint reverse relationships

Reverse relationship	Relationship name	Description	Type	Cardinality
Dependent service ports	sm63_deployedEndpoints	The service ports that reference this service endpoint	Service port	0,*

Archived

3.2.11 Extension service endpoint entity

An *extension service endpoint* entity in the governance enablement profile represents the governed service endpoint for any services that use WSDL extensions to define endpoints. The extension service endpoint entity has the same properties as a service endpoint entity as well as the additional relationships shown in Table 3-26 as defined in the service model.

Table 3-26 Extension service endpoint entity owned relationships

Owned relationship	Relationship name	Description	Type	Cardinality
Extension logical object	sm63_extensionLogicalObject	The extension service endpoint entity	Extension logical object	0,1

3.2.12 MQ service endpoint entity

An *MQ service endpoint* entity in the governance enablement profile represents the governed service endpoint for all types of MQ services. An MQ service endpoint entity has the same properties as a service endpoint entity as well as the additional relationships listed in Table 3-27 as defined in the service model.

Table 3-27 MQ service endpoint entity owned relationships

Owned relationship	Relationship name	Description	Type	Cardinality
MQ endpoint	sm63_mqEndpoint	The MQ service endpoint of this correlated endpoint entity	MQ endpoint	0,1

3.2.13 SOAP service endpoint entity

A *SOAP service endpoint* entity represents the governed service endpoint for any services that use SOAP addressing to define their endpoints. A SOAP entity has the same properties as a Service endpoint as well as the additional relationships listed in Table 3-28.

Table 3-28 SOAP service endpoint owned relationships

Owned relationship	Relationship name	Description	Type	Cardinality
SOAP address	sm63_soapAddress	The SOAP address of this correlated endpoint entity	Manual SOAP endpoint	0,1

3.3 Roles in the governance enablement profile

The governance enablement profile provides the following roles:

- The *Business* role defines the governance processes, policies, and standards that are shared across the enterprise to ensure effective interoperability, agility, and robustness of the SOA solutions. In an organization, the Business role represents business managers, analysts, and subject matter experts who are interested in how the SOA services and processes contribute to the business.
- The *SOA Governance* role represents architects, architecture boards, and SOA centers of excellence. This role also includes individuals from other roles (business, development, and operations) who define the governance processes, policies, and standards that are shared throughout the organization to ensure effective interoperability, agility, and robustness of SOA solutions.
- The *Development* role represents software development practitioners, including architects, release managers, software developers, testers, assembly developers (who use tools such as WebSphere Integration Developer), integrators, and asset librarians. They develop the software specifications and implementations to realize the requirements that are provided by the business and ensure that the implementation meets the business needs and adheres to the governance standards. Development roles usually are associated with a specific organization or department, with responsibility for delivering implementations to support a particular area of the business.
- The *Operations* role represents operations managers, operations architects, system administrators, integration testers, and IT resource managers. They manage the IT infrastructure, and deploy, configure, and test the implementations produced by development. They are responsible for operational QoS and capacity planning, and although their main activities are later in the life cycle, they are involved in all specification activities and reviews to ensure that the planned capabilities can be successfully delivered. Operations roles can be centralized or arranged by line of business or organization, according to individual company preferences.

Archived

Figure 3-11 provides a top-down example of model usage according to role.

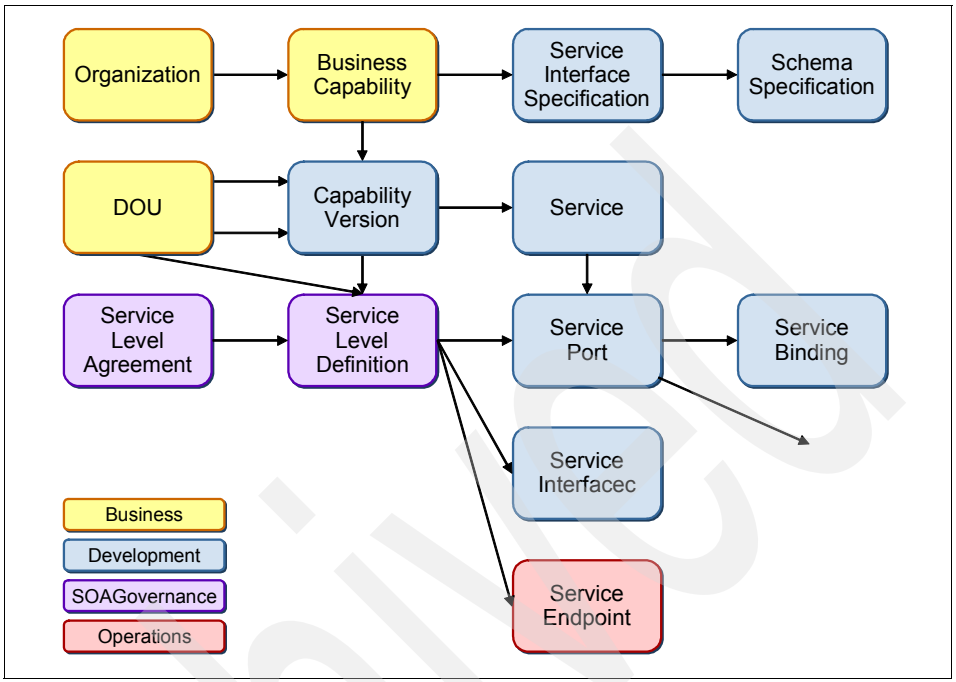


Figure 3-11 Top down model usage according to role

Business users can identify capabilities that are required by the organization. Development users, in conjunction with SOA governance and operations users, can collaborate and define the contracts between departments by defining DOUs, SLAs, and SLDs. These defined entities can eventually be hooked in to implementation entities when the original business capabilities are realized in to runnable services and represented in the profile.

3.4 Life cycles in the governance enablement profile

Entities from the various models in the governance enablement profile are governed by moving them through states in a *life cycle*. The life cycle state of an entity indicates its progress in the development process. Table 3-29 describes the life cycles that are defined in the governance enablement profile and the entities that pass through them.

Table 3-29 Life cycles and their associated entities

Name	Usage	Governed entity
Asset life cycle	Govern an entity from initial identification to retired	DOU, service interface specification, and schema specification
Capability life cycle	Govern a capability from initial identification to retired	Business capability, business application, business process, and business service
SOA life cycle	Govern a capability version from initial identification to retired	Capability version, application version, process version, and service version
SLD life cycle	Govern an SLD from initial identification to retired	SLD
SLA life cycle	Govern an SLA from initial identification to retired	SLA
Endpoint life cycle	Govern an endpoint from being approved for use to retired	Service endpoint, MQ service endpoint, SOAP service endpoint, and extension service endpoint

The next sections describe each of the governance enablement profile life cycles.

Archived

3.4.1 Asset life cycle

The *asset* life cycle in the governance enablement profile is used to govern an asset entity from being initially identified, through to being approved for reuse by consumers, and eventually to being withdrawn from use. Figure 3-12 depicts the transitions and states of an asset entity as defined in the asset life cycle.

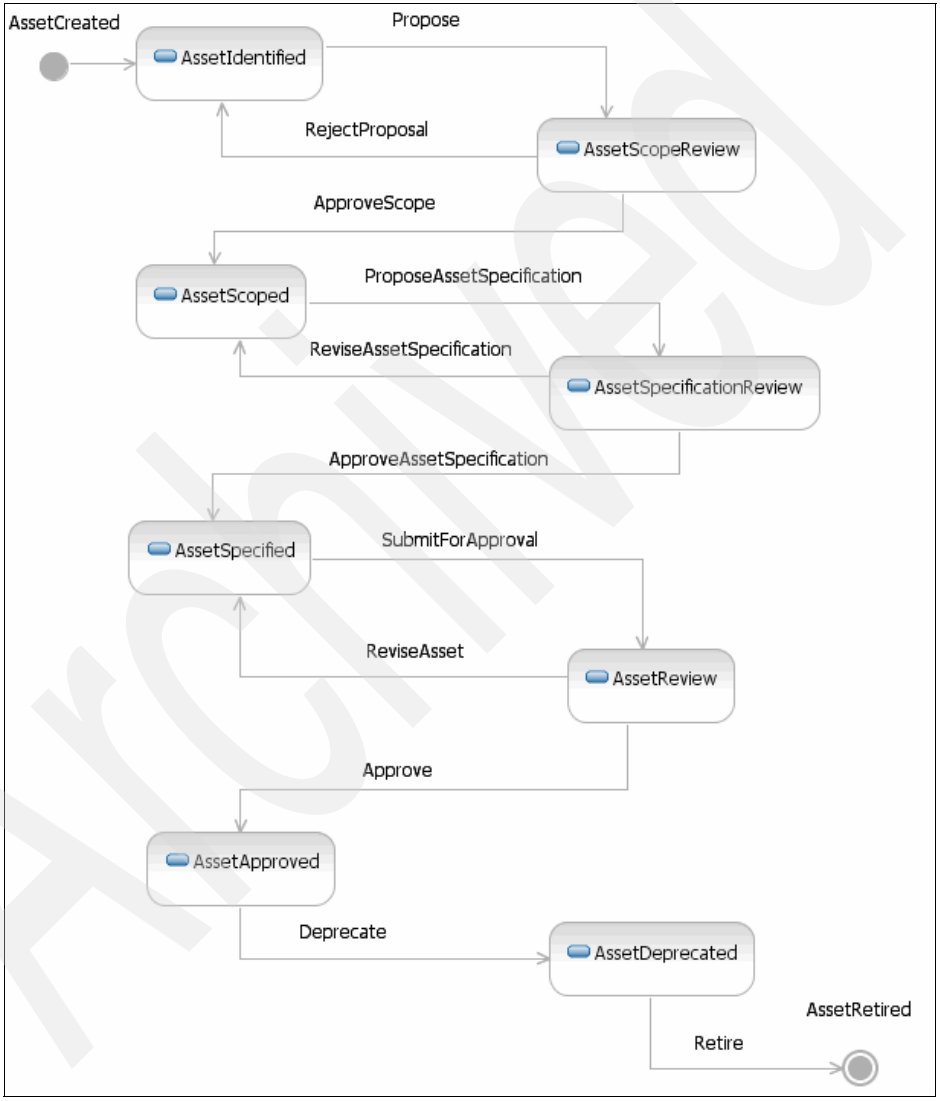


Figure 3-12 Asset life cycle

Archived

Table 3-30 describes the states of the asset life cycle and, for each state, names the transition that moves an asset entity forward to that state.

> **Note:** Where there is a transition that moves an asset entity from one state back to a previous state, that transition is not listed in Table 3-30. Refer to Figure 3-12 to see all transitions.

Table 3-30 Asset entity life cycle explanation

Transition	State	Description
Initial state	Asset identified	The requirements for an asset are defined. The scope for its use, and the other SOA artifacts that will depend on this asset, are also being made clear so that the purpose of the asset can be readily identified.
Propose	Asset scope review	The requirements proposed for the asset are considered and a decision is made as to whether further development of the asset should be undertaken. If an asset proposal is rejected, it can be proposed again after modification.
Approve proposal	Asset scoped	The main development work on the asset starts, with a specification being produced according to the agreed scope and requirements. When the development team decide that the specification is complete, it is submitted for review.
Propose asset specification	Asset specification review	The specification for the asset is reviewed. Specifications that are rejected can be proposed again after modification.
Approve asset specification	Asset specified	The main asset development work is done according to the specification.
Submit asset for approval	Asset review	The asset is reviewed by the stakeholders for conformance with the requirements that were approved. If these requirements are deemed not to be met, the asset is rejected and goes back to development for rework.
Approve	Asset approved	The asset is visible to potential consumers for reuse.
Deprecate asset	Asset deprecated	The asset is not available to new users but is still be visible to existing subscribers and consumers.
Retire asset	Asset retired	The asset has been withdrawn from use and there are no active consumers using this asset. It ceases to be visible to general users.

Archived

3.5 Capability life cycle

The *capability* life cycle in the governance enablement profile is used to govern a business capability entity from being initially identified, through to being approved when all the governance requirements are satisfied, and eventually to being deprecated and retired when it is no longer necessary. Figure 3-13 depicts the transitions and states of a business capability entity as defined in the capability life cycle.

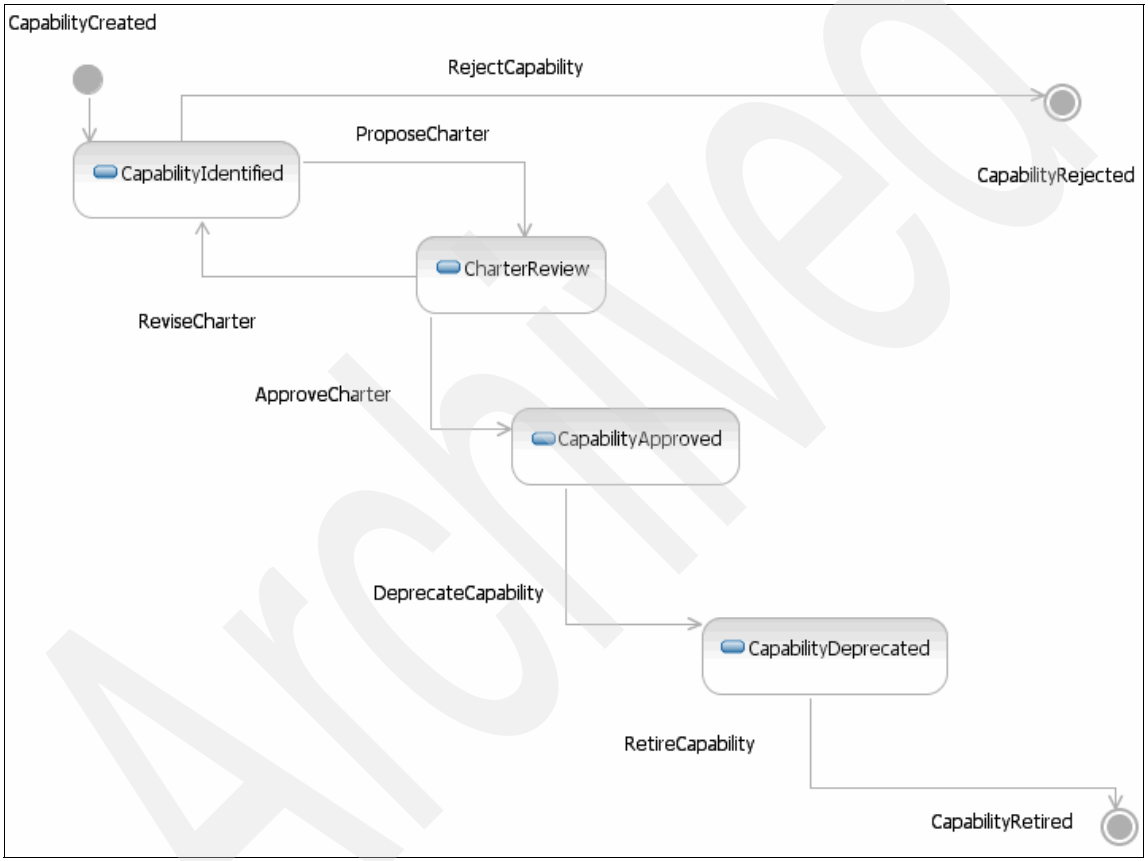


Figure 3-13 Capability life cycle

Archived

Table 3-31 describes the states of the capability life cycle, and, for each state, names the transition that moves a business capability entity forward to that state.

> **Note:** Where there is a transition that moves a business capability entity from one state back to a previous state, that transition is not listed in Table 3-31. Refer to Figure 3-13 to see all transitions.

Table 3-31 Capability life cycle explanation

Transition	State	Description
Initial state	Capability identified	This state is entered when a business capability is first identified. During this state, a charter is produced to scope the capability that is required and agree where in the organization the responsibility for delivering this capability is to be assigned.
Reject capability	Capability rejected	Charter review has determined the capability is not needed.
Propose charter	Charter review	Either approved or sent back for revision.
Approve charter	Capability approved	Governance requirements for the business capability are met.
Deprecate capability	Capability deprecated	Business capability is still available for existing users but has been superceded.
Retire capability	Capability retired	Business capability versions are not longer in use.

3.6 SOA life cycle

The *SOA* life cycle in the governance enablement profile is used to govern a capability version entity from being initially identified, through to being deployed in production, and eventually deprecated when it is no longer required. The SOA life cycle is separated into multiple phases, which we describe in the following sections.

Archived

SOA life cycle: Model phase

The *model phase* of the SOA life cycle in the governance enablement profile is used to govern a capability version entity from being initially identified, through to its specification being proposed for review. Figure 3-14 depicts the transitions and states of the model phase of an entity as defined in the SOA life cycle.

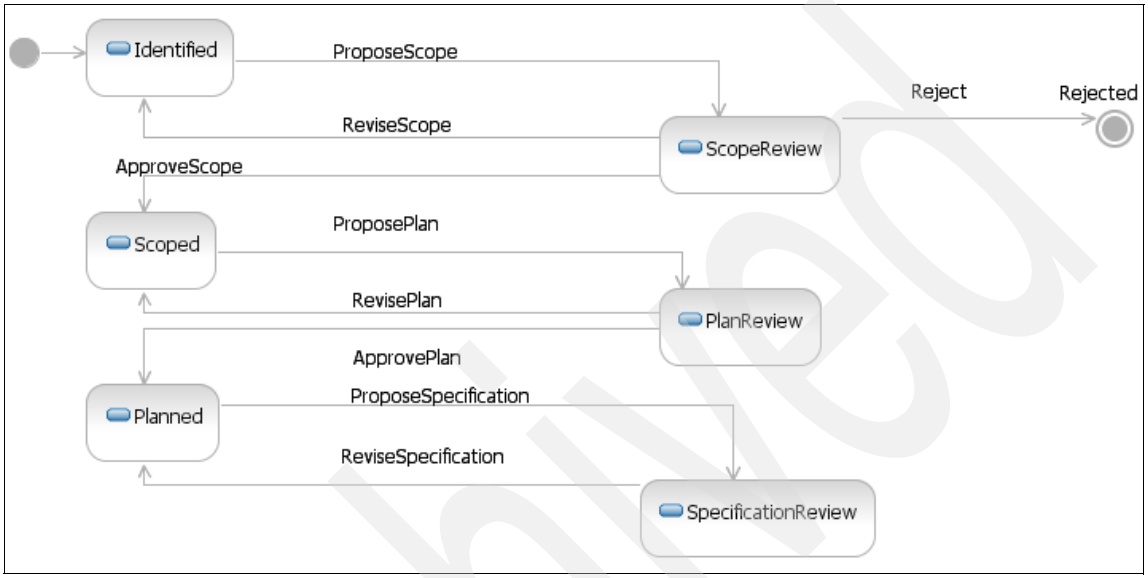


Figure 3-14 SOA life cycle, model phase

Archived

Table 3-32 describes the states of the model phase of the SOA life cycle and, for each state, names the transition that moves an entity forward to that state.

Note: Where there is a transition that moves an entity from one state back to a previous state, that transition is not listed in Table 3-32. Refer to Figure 3-14 to see all transitions.

Table 3-32 SOA life cycle model phase explanation

Transition	State	Description
Initial state	Identified	A new version of a capability has been requested or identified. The stakeholders identify the requirements for this version of the capability.
Propose scope	Scope review	The stakeholders review the requirements and intended ownership of the version, and agree the scope. New stakeholders and requirements might be identified, requiring the scope to be revised and proposed again. Any new version of a capability must be mapped to an approved business capability to ensure that its role in the organization is clear.
Approve scope	Scoped	The development and owning organizations must work together to define the funding and time frames for the project.
Propose plan	Plan review	The various lines of business involved in the provision and consumption of the capability now have the opportunity to review the details of the implementation of the capability version.
Approve plan	Planned	Development of the capability begins. The first activities are to do with agreeing the specification for the version. This includes the complete definition of all exposed interfaces, and any provided SLDs that other capabilities can use, together with any SLAs for capabilities that this version will consume. Development of these specifications is likely to require some development of the actual implementations to ensure that the specification can be realized.
Propose specification	Specification review	The specification review must ensure that the specifications that are provided for this version of the capability will meet the stakeholder requirements. Any use of this version by other consumers, as defined by the SLDs, will be based entirely on the specification and costed accordingly. Any dependencies on other capability versions must be specified and agreed through an SLA and DOU. Any disagreement about functional specifications might require the specification to be revised and proposed again. Approval of the version specification determines a contract that allows both potential consumers and providers to proceed independently.

3.6.1 SOA Life cycle: Assemble phase

The *assemble* phase of the SOA life cycle in the governance enablement profile is used to govern a Capability version entity from having its specification approved, through to being realized. Figure 3-15 depicts the transitions and states of the assemble phase of an entity as defined in the SOA life cycle.

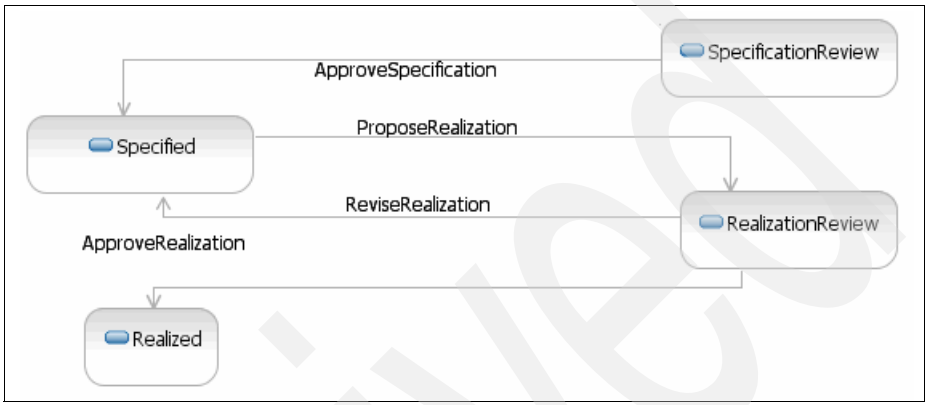


Figure 3-15 SOA life cycle assemble phase

Table 3-33 describes the states of the assemble phase of the SOA life cycle and, for each state, names the transition that moves an entity forward to that state.

Note: Where there is a transition that moves an entity from one state back to a previous state, that transition is not listed in Table 3-33. Refer to Figure 3-15 to see all transitions.

Table 3-33 SOA life cycle assemble phase explanation

Transition	State	Description
Approve specification	Specified	In this state, the majority of the development or assembly of this version of the capability occurs. The details of activities undertaken during this state will be very specific to the development processes being used within the organization. but the key exit criteria is when development (and development test) is complete and a release is ready to be reviewed before being sent to operations for integration testing, configuration and staging. On completion of the development, the version realization is proposed for a realization review.

Archived

Transition	State	Description
Propose realization	Realization review	In this state, stakeholders agree that sufficient testing of the developed realization has taken place, and the quality of the version is such that it can be sent to operations for deployment and configuration into the staging environments.
Approve realization	Realized	The version is installed onto staging (integration, test, pre-production) environments and configured to operate with other deployed applications, processes and services. Testing is undertaken of the SLDs that the capability offers. and if all SLDs can be met then the version is proposed as ready for staging.

Archived

3.6.2 SOA life cycle: Deploy phase

The *deploy* phase of the SOA life cycle in the governance enablement profile is used to govern a capability version entity from being realized, through to being deployed in production. Figure 3-16 depicts the transitions and states of the deploy phase of an entity as defined in the SOA life cycle.

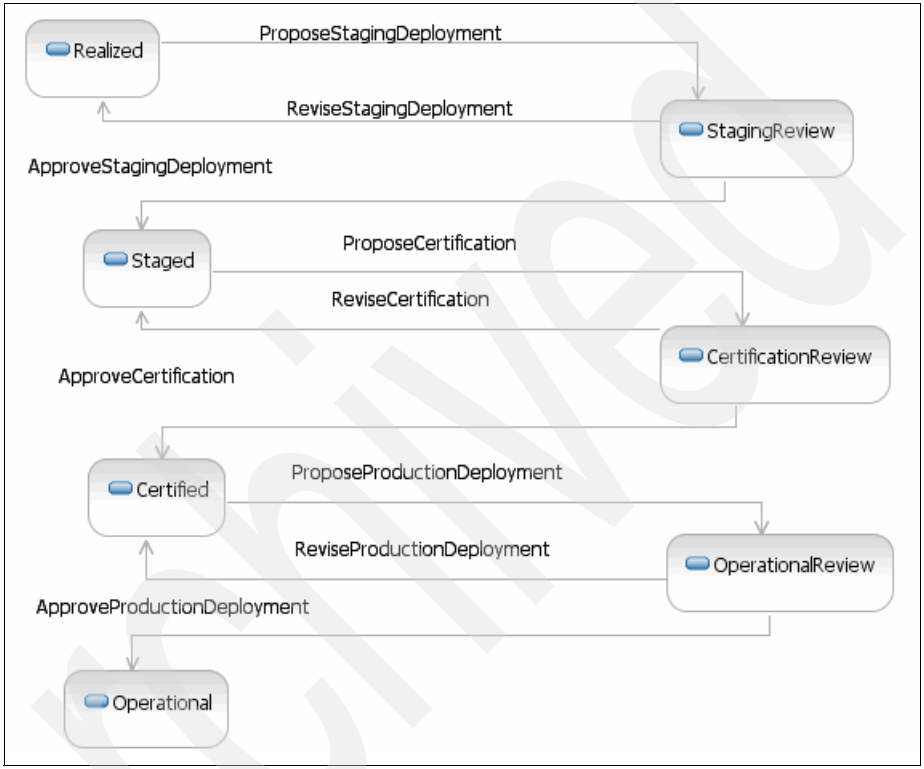


Figure 3-16 SOA life cycle deploy phase

Archived

Table 3-34 describes the states of the deploy phase of the SOA life cycle, and, for each state, names the transition that moves an entity forward to that state.

Note: Where there is a transition that moves an entity from one state back to a previous state, that transition is not listed in Table 3-34. Refer to Figure 3-16 to see all transitions.

Table 3-34 SOA life cycle deploy phase explanation

Transition	State	Description
Propose staging deployment	Staging review	This state identifies when consumers in the staging environment can have access to the SLDs that have been deployed. The review must review the SLAs expected of the capability to ensure that all planned commitments can be met. Any capacity deficit could result in a revision of the staging deployment and configuration, requiring a reproposal to review.
Approve staging deployment	Staged	The SLAs and dependencies between capabilities are tested to prove that when this loosely coupled solution is deployed in an operational environment, most of the changes that deliver business agility have been tested over the likely bounds of use. Once this testing is complete, the version is proposed for certification.
Propose certification	Certification review	The staging and integration testing is confirmed and authority is given to deploy the version to the production environment.
Approve certification	Certified	The certified version, and configuration, is deployed into the production environments and tested.
Propose production deployment	Operational review	The final confirmation and authority is given to release the version of the capability to the business.
Approve production deployment	Operational	The capability version is deployed to production.

Archived

3.6.3 SOA life cycle: Manage phase

The *manage* phase of the SOA life cycle in the governance enablement profile to deprecate, and eventually retire, a deployed Capability version entity when it is no longer required. Figure 3-17 depicts the transitions and states of the manage phase of an entity as defined in the SOA life cycle.

Figure 3-17 SOA life cycle manage phase

Table 3-35 describes the states of the manage phase of the SOA life cycle and, for each state, names the transition that moves an entity forward to that state.

> **Note:** Where there is a transition that moves an entity from one state back to a previous state, that transition is not listed in Table 3-35. Refer to Figure 3-17 to see all transitions.

Table 3-35 SOA life cycle manage phase explanation

Transition	State	Description
Deprecate	Deprecated	The business capability is no longer considered necessary. No more versions of this capability must be produced, but use of existing versions can continue until no longer required.
Retire	Retired	There are no versions of this capability in use in the organization.

Archived

3.7 SLD life cycle

The *SLD* life cycle in the governance enablement profile to govern an SLD from being initially identified, through to being retired when it is no longer in use. The SLD life cycle is separated into multiple phases, which we describe in the following sections.

SLD life cycle: Model and assemble phase

The *model and assemble* phases of the SLD life cycle in the governance enablement profile is used to govern an SLD entity from being initially identified, through to having its specification approved. Figure 3-18 depicts the transitions and states of the model and assemble phases of an SLD entity as defined in the SLD life cycle.

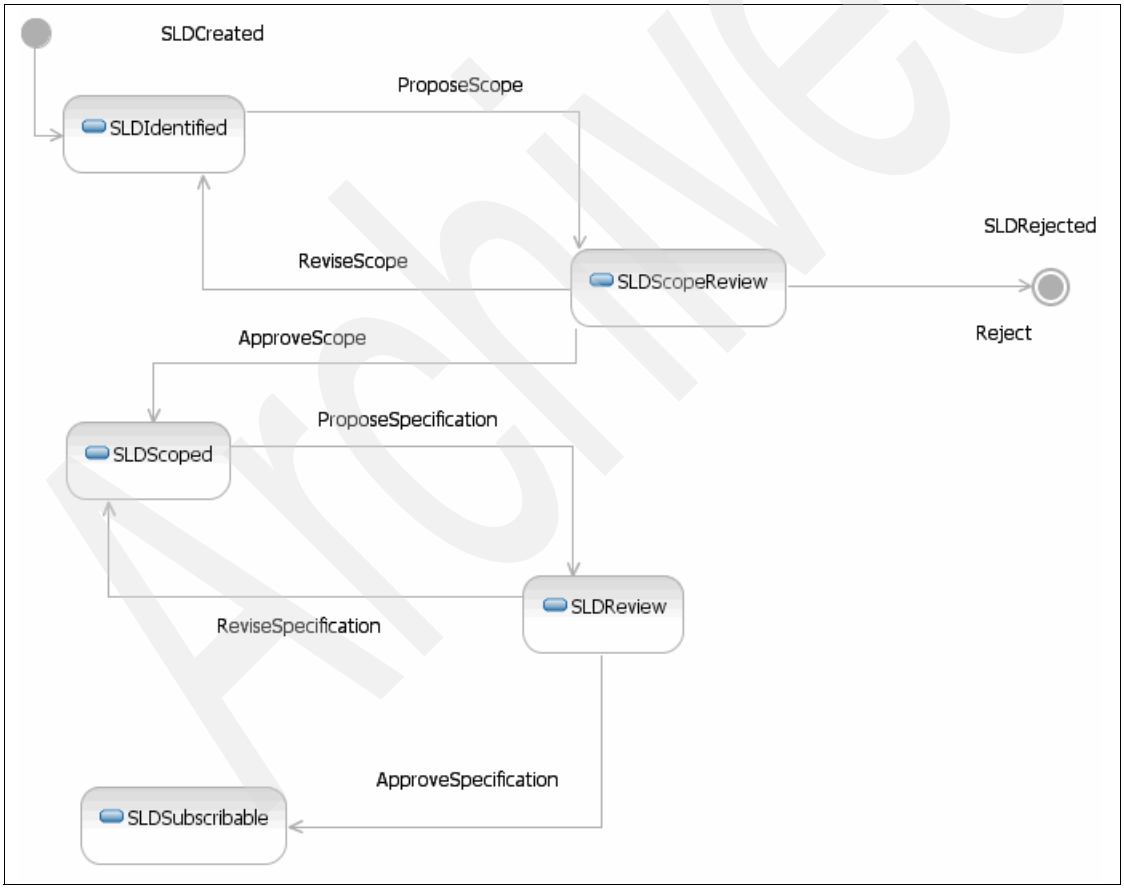


Figure 3-18 SLD life cycle, model and assemble phases

Archived

Table 3-36 describes the states of the model and assemble phases of the SLD life cycle, and, for each state, names the transition that moves an SLD entity forward to that state.

Note: Where there is a transition that moves an SLD entity from one state back to a previous state, that transition is not listed in Table 3-36. Refer to Figure 3-18 to see all transitions.

Table 3-36 SLD life cycle model and assemble phases explanation

Transition	State	Description
Initial state	SLD identified	A new QoS has been identified that can be configured to work with an existing capability version. The scope and qualities of service that will be made available are defined, and then put forward for scope review.
Propose scope	SLD scope review	In this state, the decision is made to proceed with allocating the resources to develop a new SLD.
Approve scope	SLD scoped	The qualities of service are elaborated and a complete specification defined for the SLD. This specification will allow consumers to develop to the SLD if they want to subscribe to a service and setup an appropriate SLA. After this specification is complete, the SLD goes forward for review.
Propose specification	SLD review	The stakeholders review and approve the SLD, allowing consumers to subscribe and develop against it. For SLDs that are developed as part of the version during its SOA life cycle, this review is coincident with the specification review.
Approve specification	SLD subscribable	SLAs can be established, through the agreed endpoints relationship, to reference this SLD. Development can continue against a subscribable SLD but no interactions can be undertaken with its endpoints yet. This means that SLAs can be approved and enter the inactive state if the SLD is subscribable, but cannot be made active until endpoints become online.

Archived

SLD life cycle: Deploy and manage phases

The *deploy and manage* phases of the SLD life cycle in the governance enablement profile is used to deprecate or supercede an SLD entity and to retire a deprecated SLD when it is no longer in use. Figure 3-19 depicts the transitions and states of the deploy and manage phases of an SLD entity as defined in the SLD life cycle.

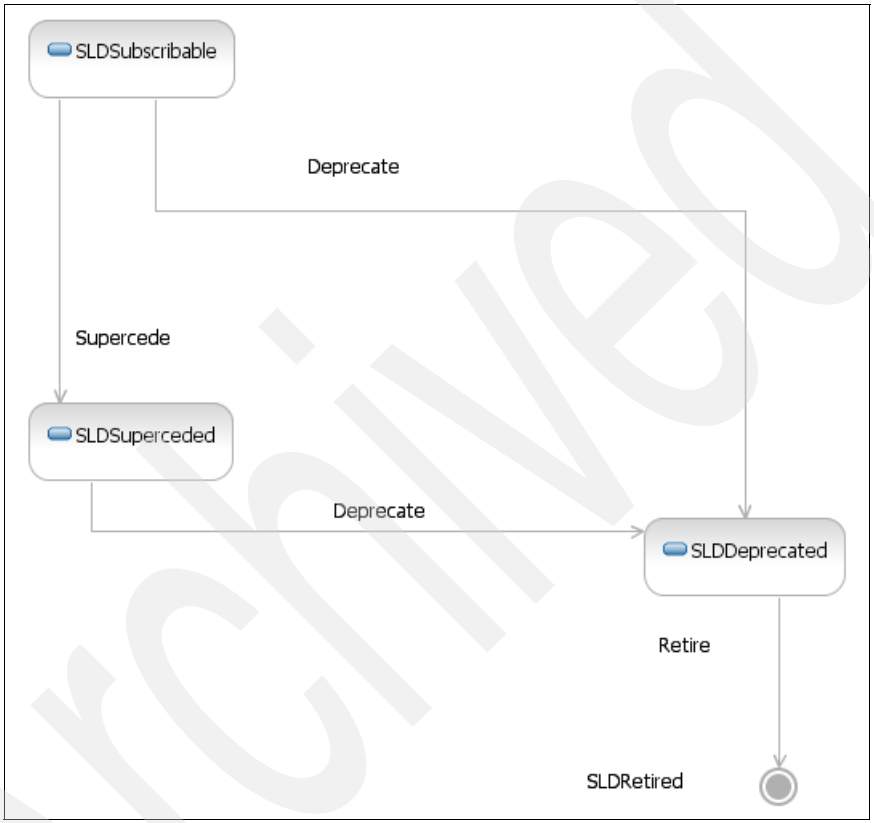


Figure 3-19 SLD life cycle deploy and manage phases diagram

Archived

Table 3-37 describes the states of deploy and manage phases of the SLD life cycle and for each state names the transition that moves an SLD forward to that state.

Table 3-37 SLD life cycle deploy and manage phases explanation

Transition	State	Description
Supercede	SLD superceded	A new compatible SLD becomes subscribable, with active endpoints, and the provider wants to move consumers and their SLAs onto this new provided SLD. No new subscriptions can be made to an SLD in this state.
Deprecate	SLD deprecated	All existing SLAs are moved onto the compatible SLDs, and these endpoints are made inactive. Existing SLAs must be renegotiated to directly reference the compatible SLD.
Retire	SLD retired	The SLD has no consuming SLAs.

3.7.1 SLA life cycle

The *SLA* life cycle in the governance enablement profile is used to govern an SLA entity from being initially identified, through to being activated, and, eventually, terminated when it is no longer required. Figure 3-20 depicts the transitions and states of an SLA entity as defined in the SLA life cycle.

Archived

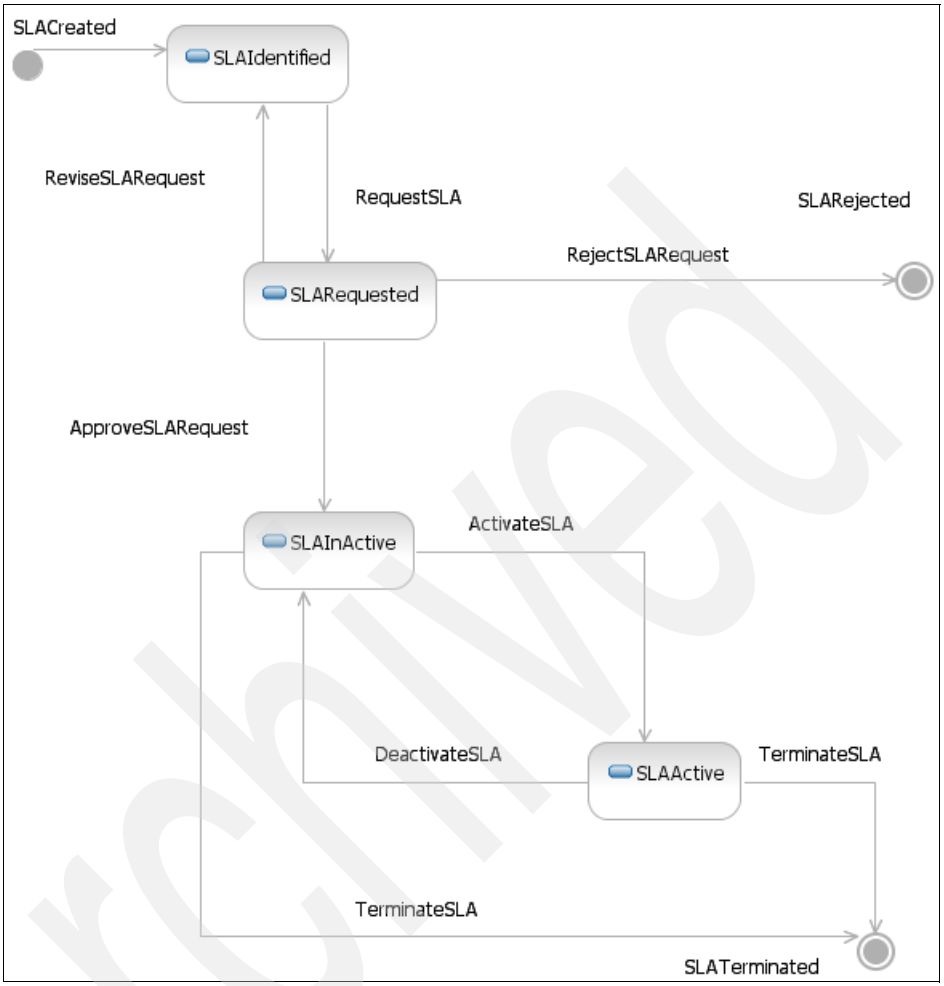


Figure 3-20 SLA life cycle

Table 3-38 describes the states of the SLA life cycle and, for each state, names the transition that moves an SLA entity forward to that state.

> **Note:** Where there is a transition that moves an SLA entity from one state back to a previous state, that transition is not listed in Table 3-38. Refer to Figure 3-20 to see all transitions.

Archived

Table 3-38 SLA life cycle explanation

Transaction	State	Description
Initial State	SLA identified	This state is entered as soon as a consumer, represented by a capability version, requests a dependency on a service version or other capability version that offers the SLD that they require.
Request SLA	SLA requested	The agreed endpoints relationship target has been selected together with details of the required SLA properties and policies. The provider of the selected SLD must approve the request, reject it or ask for it to be revised.
Approve SLA request	SLA inactive	The development team that want to consume the service can continue their development based on the consumption of this specific SLA, but they do not yet have authorization to access any endpoints.
Activate SLA	SLA active	All of the approved endpoints associated with the SLD, that are online, can be invoked using the terms of the SLA. There might be situations where the SLA is deactivated, in which case the SLA enters the SLA inactive state and any further interactions are blocked until it is reactivated.
Terminate SLA	SLA terminated	No interactions from this SLA are permitted.

Archived

3.7.2 Endpoint life cycle

The *endpoint* life cycle in this governance enablement profile is used to govern an endpoint entity from being approved for use, through to being retired. Figure 3-20 depicts the transitions and states of a endpoint entity as defined in the endpoint life cycle.

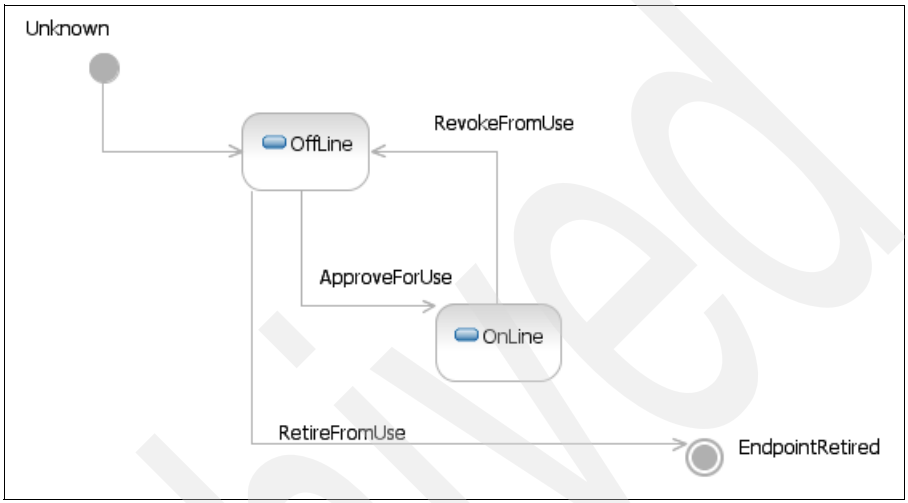


Figure 3-21 Endpoint life cycle

Table 3-39 describes the endpoint life cycle and, for each state, names the transition that moves an endpoint entity forward to that state.

Note: Where there is a transition that moves an Endpoint entity from one state back to a previous state, that transition is not listed in Table 3-39. Refer to Figure 3-21 to see all transitions.

Table 3-39 Endpoint life cycle explanation

Transition	State	Description
Initial state	Offline	An offline endpoint might be deployed and thus reachable by potential consumers, but if protected by mediations that can access the endpoint state, access to this particular endpoint is denied.
Approve for use	Online	Mediations can consider this endpoint as possible routing target.
Retire from use	Endpoint retired	The endpoint has been removed from the environment.

Archived

As the entities pass through these various life cycles, constraints as to which role can perform transitions and if these entities must be related to another entity are enforced through *policies*, which we discuss in the next section.

3.8 Policies in the governance enablement profile

The policies in the governance enablement profile specifies which roles (who) can perform a specification action (what) on entities with particular attributes or in a particular state (when). In this section, we describe each of these policies.

3.8.1 Life cycle transition policies

The *life cycle* transition policies in the governance enablement profile control which roles can perform transitions on entities that are in a specific life cycle. For example, only a user in the business role has the right to transition a business capability entity to propose charter.

Asset life cycle decision rights

Asset life cycle decision rights in the governance enablement profile are policies that are applied to an asset entity as defined in the asset life cycle. Table 3-40 describes the asset life cycle decision rights for an asset entity according to roles.

Table 3-40 Asset life cycle decision rights

Role	Policy URI	Assertion	Transition
Development	urn:AssetLifecycleDecisionRights_Development	AssetLifecycleDecisionRights.AssetDevelopmentTransitionCheck	► Propose scope ► Propose asset specification ► Submit for approval ► Deprecate ► Retire
Business	urn:AssetLifecycleDecisionRights_Business	AssetLifecycleDecisionRights.AssetBusinessTransitionCheck	► Revise scope ► Approve scope ► Revise asset ► Approve
SOA Governance	urn:AssetLifecycleDecisionRights_SOAGovernance	AssetLifecycleDecisionRights.AssetSOAGovernanceTransitionCheck.	► Revise asset specification ► Approve asset specification

Capability life cycle decision rights

Capability life cycle decision rights in the governance enablement profile are policies that are applied to a business capability entity as defined in the capability life cycle. Table 3-41 describes the capability life cycle decision rights for a business capability entity according to roles.

Table 3-41 Capability life cycle decision rights

Role	Policy URI	Assertion	Transition
Business	urn:CapabilityLifecycleDecisionRights_Business	CapabilityLifecycleDecisionRights.CapabilityBusinessTransitionCheck	► Propose charter
SOA Governance	urn:CapabilityLifecycleDecisionRights_SOAGovernance	CapabilityLifecycleDecisionRights.CapabilitySOAGovernanceTransitionCheck	► Reject capability ► Revise charter ► Approve charter ► Deprecate capability ► Retire capability

DOU life cycle decision rights

DOU life cycle decision rights in the governance enablement profile are policies that are applied to DOU entities as defined in the asset life cycle. Table 3-42 describes the capability life cycle decision rights for a business capability entity according to roles.

Table 3-42 Capability life cycle decision rights explanation

Role	Policy URI	Assertion	Transition
Development	urn:DOULifecycleDecisionRights_Development	DOULifecycleDecisionRights.DOUDevelopmentTransitionCheck	► Propose asset specification ► Revise asset specification ► Approve asset specification ► Submit for approval ► Deprecate ► Retire
Business	urn:DOULifecycleDecisionRights_Business	DOULifecycleDecisionRights.DOUBusinessTransitionCheck	► Propose scope ► Revise scope ► Approve scope
SOA Governance	urn:DOULifecycleDecisionRights_SOAGovernance	DOULifecycleDecisionRights.DOUSOAGovernanceTransitionCheck	► Revise asset ► Approve

Archived

Endpoint life cycle decision rights

Endpoint life cycle decision rights in the governance enablement profile are policies that are applied to an endpoint entity as defined in the endpoint life cycle. Table 3-43 describes the endpoint life cycle decisions rights in the governance enablement profile for an endpoint entity according to roles.

Table 3-43 Endpoint life cycle decision rights explanation

Role	Policy URI	Assertion	Transition
Operations	urn:EndpointLifecycleDecisionRights_Operations	EndpointLifecycleDecisionRights.EndpointOperationsTransitionCheck	► Revoke from use ► Approve for use ► Retirue from use

Schema life cycle decision rights

Schema life cycle decision rights in the governance enablement profile are policies that are applied to a schema specification entity as defined in the asset life cycle. Table 3-44 describes the schema life cycle decision rights in the governance enablement profile for a schema specification entity according to roles.

Table 3-44 Schema life cycle decisions rights explanation

Role	Policy URI	Assertion	Transition
Development	urn:SchemaLifecycleDecisionRights_Development	SchemaLifecycleDecisionRights.SchemaDevelopmentTransitionCheck	► Propose scope ► Propose asset specification ► Revise asset specification ► Approve asset specification ► Submit for approval
SOA Governance	urn:SchemaLifecycleDecisionRights_SOAGovernance	SchemaLifecycleDecisionRights.SchemaSOAGovernanceTransitionCheck	► Revise scope ► Approve scope ► Revise asset ► Approve ► Deprecate ► Retire

Archived

Service interface life cycle decision rights

Service interface life cycle decision rights in the governance enablement profile are policies that are applied to a service interface specification entity as defined in the asset life cycle. Table 3-45 describes the service interface life cycle decision rights in the governance enablement profile for a service interface entity according to roles.

Table 3-45 Service interface life cycle decision rights explanation

Role	Policy URI	Assertion	Transition
Development	urn:ServiceInterfaceLifecycleDecisionRights_Development	ServiceInterfaceLifecycleDecisionRights.ServiceInterfaceDevelopmentTransitionCheck	► Propose scope ► Propose asset specification ► Revise asset specificaiton ► Approve asset specification ► Submit for approval
SOA Governance	urn:ServiceInterfaceLifecycleDecisionRights_SOAGovernance	ServiceInterfaceLifecycleDecisionRights.ServiceInterfaceSOAGovernanceTransitionCheck	► Revise scope ► Approve scope ► Revise asset ► Approve ► Deprecate ► Retire

SLA life cycle decision rights

SLA life cycle decision rights in the governance enablement profile are policies that are applied to an SLA entity as defined in the SLA life cycle. Table 3-46 describes the SLA life cycle decision rights in the governance enablement profile for an SLA entity according to roles.

Table 3-46 SLA life cycle decision rights explanation

Role	Policy URI	Assertion	Transition
Development	urn:SLALifecycleDecisionRights_Development	SLALifecycleDecisionRights.SLADevelopmentTransitionCheck	► Request SLA ► Revise SLA request ► Reject SLA request
Operations	urn:SLALifecycleDecisionRights_Operations	SLALifecycleDecisionRights.SLAOperationsTransitionCheck	► Activate SLA ► Deactivate SLA
SOA Governance	urn:SLALifecycleDecisionRights_SOAGovernance	SLALifecycleDecisionRights.SLASOAGovernanceTransitionCheck	► Approve SLA request ► Terminate SLA

Archived

SLD life cycle decision rights

SLD life cycle decision rights in the governance enablement profile are policies that are applied to an SLD entity as defined in the SLD life cycle. Table 3-47 describes the SLD life cycle decision rights in the governance enablement profile for an SLD entity according to roles.

Table 3-47 Service level definition life cycle decision rights explanation

Role	Policy URI	Assertion	Transition
Development	urn:SLDLifecycleDecisionRights_Development	SLDLifecycleDecisionRights.SLDDevelopmentTransitionCheck	► Propose scope ► Propose specification
Operations	urn:SLDLifecycleDecisionRights_Operations	SLDLifecycleDecisionRights.SLDOperationsTransitionCheck	► Revise spefication ► Deprecate ► Supercede
SOA Governance	urn:SLDLifecycleDecisionRights_SOAGovernance	SLDLifecycleDecisionRights.SLDSOAGovernanceTransitionCheck	► Revise scope ► Approve scope ► Approve specification ► Retire

SOA life cycle decision rights

SOA life cycle decision rights are policies that are applied to a capability version entity as defined in the SOA life cycle. Table 3-48 describes the SOA life cycle decision rights in the governance enablement profile for a capability version entity according to roles.

Table 3-48 SOA life cycle decision rights explanation

Role	Policy URI	Assertion	Transition
Development	urn:SOALifecycleDecisionRights_Development	SOALifecycleDecisionRights.SOADevelopmentTransitionCheck	► Propose plan ► Propose specification ► Propose realization ► Revise realization ► Approve realization
Business	urn:SOALifecycleDecisionRights_Business	SOALifecycleDecisionRights.SOABusinessTransitionCheck	► Propose scope ► Revise plan ► Approve plan ► Revise production deployment ► Approve production deployment

Archived

Role	Policy URI	Assertion	Transition
Operations	urn:SOALifecycleDecisionRights_Operations	SOALifecycleDecisionRights.SOAOperationsTransitionCheck	► Propose staging deployment ► Revise staging deployment ► Approve staging deployment ► Propose certification ► Propose production deployment ► Deprecate
SOA Governance	urn:SOALifecycleDecisionRights_SOAGovernance	SOALifecycleDecisionRights.SOASOAGovernanceTransitionCheck	► Revise scope ► Approve scope ► Revise specification ► Approve specification ► Revise certification ► Approve certification ► Reject deprecation ► Retire

3.8.2 Life cycle detail policies

The life cycle detail policies in the governance enablement profile enforce constraints on entities for certain life cycle transitions. We describe these policies in the sections that follow.

Capability life cycle detail rights

Capability life cycle detail rights in the governance enablement profile are policies that are applied to a business capability entity as defined in the capability life cycle. Figure 3-22 shows the capability life cycle detail rights to transition a capability version to charter available.

Archived

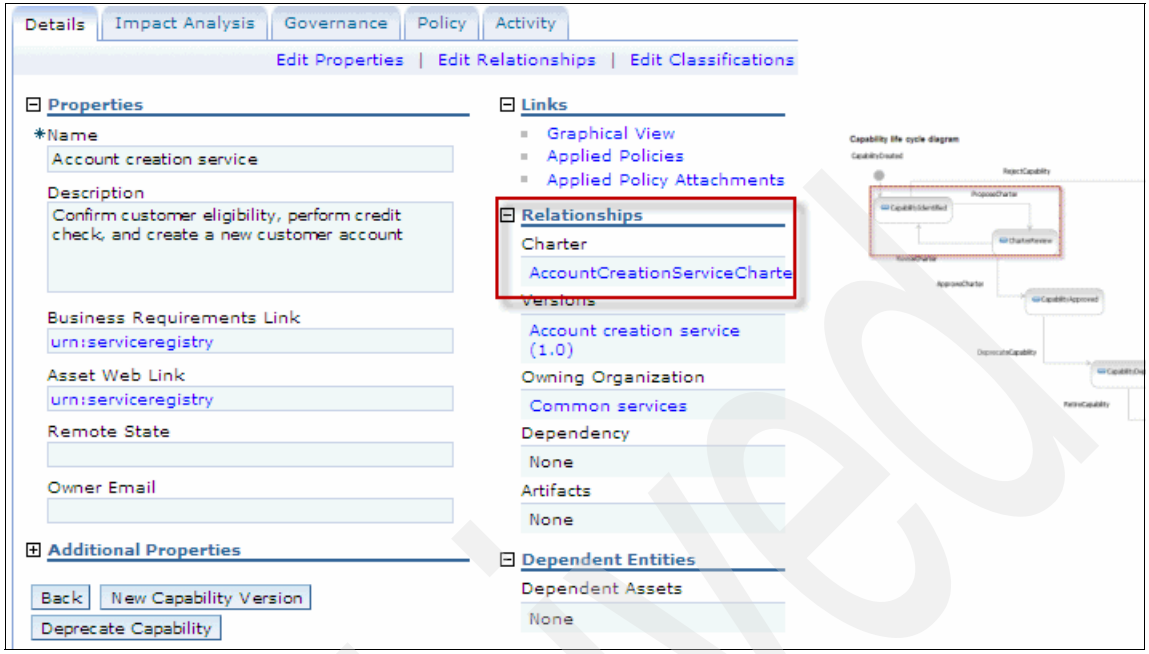


Figure 3-22 Capability life cycle detail rights, propose charter

Table 3-49 describes the capability life cycle detail rights.

Table 3-49 Capability life cycle detail rights explanation

Transition	Policy URI	Assertion	Description
Propose charter	urn:CapabilityLifecycleDetailRights_CharterAvailable	CapabilityLifecycleDetailRights.CapabilityCharterAvailableCheck	For a user to perform the Propose charter transition, a charter must be associated with the Business capability entity, as a target of the Charter relationship,

Archived

Endpoint life cycle detail rights

Endpoint life cycle detail rights in the governance enablement profile are policies that are applied to a endpoint entity as defined in the endpoint life cycle. Table 3-50 describes the endpoint life cycle detail rights.

Table 3-50 Endpoint life cycle detail rights explanation

Transition	Policy URI	Assertion	Description
Revoke From Use	urn:EndpointLifecycleDetailRights_RevokeFromUse	EndpointLifecycleDetailRights.SLDOnlineEndpointCheck	For a user to perform the “Revoke from use” transition on an endpoint entity, any “Active” SLD entity that references that endpoint entity must have at least one other endpoint entity in the “Online” state.
Retire From Use	urn:EndpointLifecycleDetailRights_RetireFromUse	EndpointLifecycleDetailRights.EndpointRetireCheck	For a user to apply the “Retire from use” transition on an endpoint entity, any SLD entity that references that endpoint entity must either be in the “Retired” state, or have at least one endpoint entity in the “Active” state.

Archived

Figure 3-23 shows the endpoint life cycle detail just before an endpoint entity is transitioned to “approve for use” state.

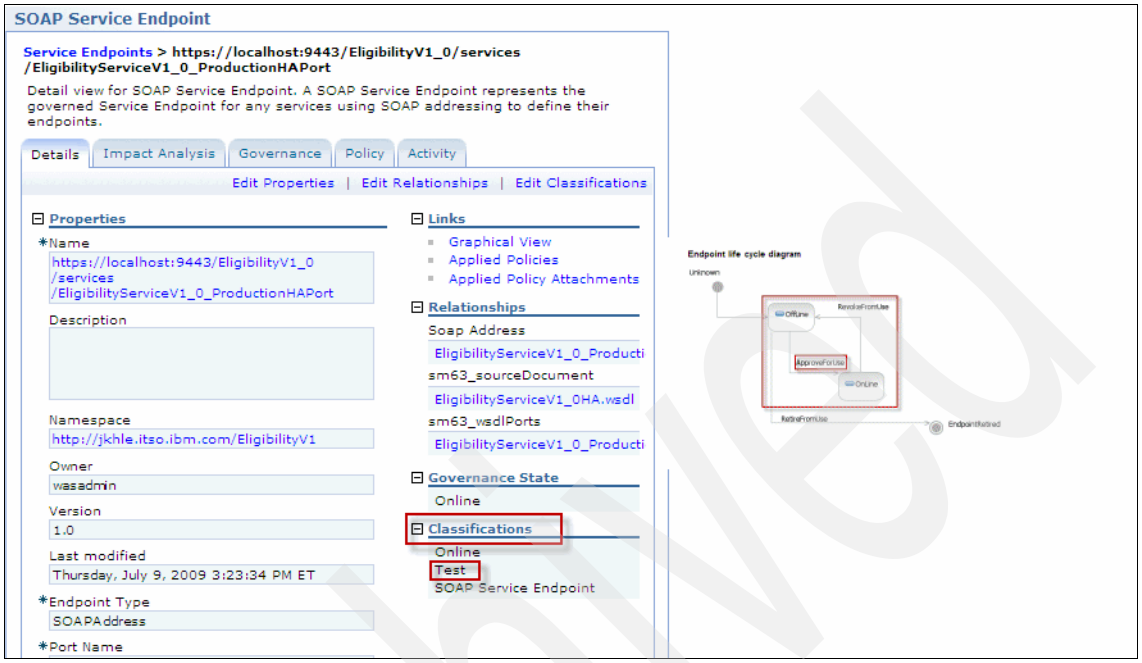


Figure 3-23 Endpoint Life cycle detail rights, “Approve for use”

Table 3-51 describes the endpoint life cycle detail rights.

Table 3-51 Endpoint life cycle detail rights “Approve for use” explanation

Transition	Policy URI	Assertion	Description
Approve For Use	urn:EndpointLifecycleDetailRights_ApproveForUse	EndpointLifecycleDetailRights.EndpointEnvironmentCheck	For a user to perform the “Approve for use” transition, an endpoint entity must have associated environment metadata.

Archived

Rational Asset Manager asset approval rights

Rational Asset Manager asset approval rights in the governance enablement profile are policies that are applied to a Rational Asset Manager asset entity that is undergoing life cycle transitions. Table 3-52 describes the Rational Asset Manager asset approval rights.

Table 3-52 Rational Asset Manager asset approval rights explanation

Transition	Policy URI	Assertion	Description
Approve Asset Specification	urn:RAMAssetApprovalRights_ApproveAssetSpecification	RAMAssetApprovalRights.ApproveAssetSpecificationCheck	For a user to perform the Approve asset specification transition, a remote Rational Asset Manager asset entity must be in an “Approved” state; that is, the remoteState property, if set, must have the value “Approved”
Approve	urn:RAMAssetApprovalRights_Approve.	RAMAssetApprovalRights.ApproveAssetCheck	For a user to perform the Approve transition, a remote Rational Asset Manager asset entity must be in an Approved state; that is, the remoteState property, if set, must have the value “Approved”
Approve Charter	urn:RAMAssetApprovalRights_ApproveCharter	RAMAssetApprovalRights.ApproveCharterCheck	For a user to perform the Approve charter transition, a remote Rational Asset Manager asset entity must be in an Approved state; that is, the remoteState property, if set, must have the value “Approved”
Approve Specification	urn:RAMAssetApprovalRights_ApproveSpecification	RAMAssetApprovalRights.ApproveSpecificationCheck	For a user to perform the Approve specification transition, a remote Rational Asset Manager asset entity must be in an Approved state; that is, the remoteState property, if set, must have the value “Approved”

Archived

Schema life cycle detail rights

Schema life cycle detail rights in the governance enablement profile are policies that are applied to a schema specification entity as defined in the asset life cycle. Figure 3-24 diagrams the schema life cycle detail rights before a schema specification can be transitioned to propose scope.

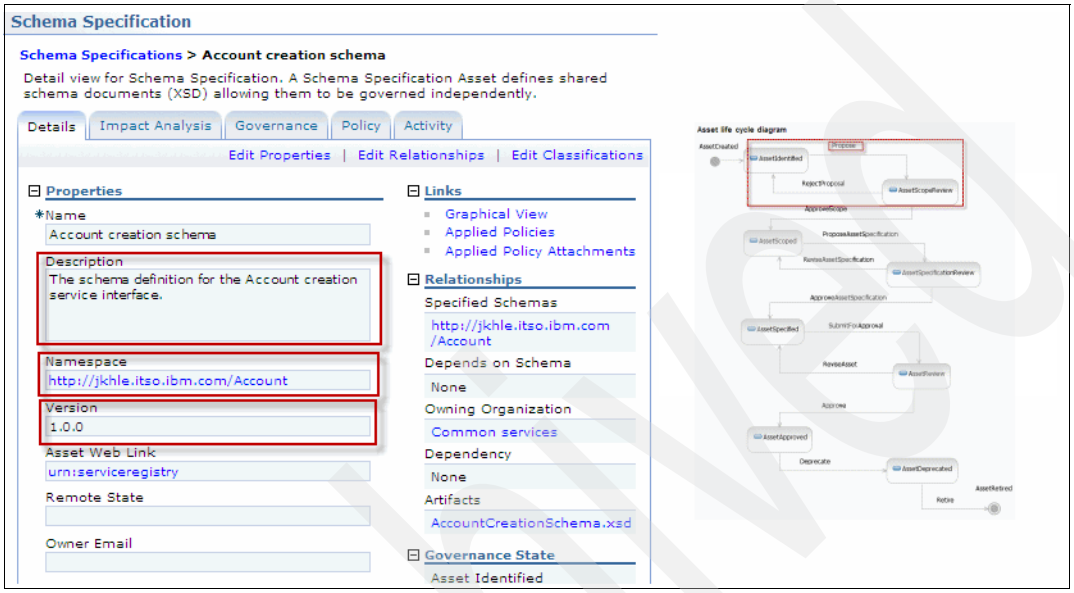


Figure 3-24 Schema life cycle detail rights, propose scope

Table 3-53 describes the Schema life cycle detail rights.

Table 3-53 Schema life cycle detail rights explanation

Transition	Policy URI	Assertion	Description
Propose Scope	urn:SchemaLifecycleDetailRights_ProposeScope	SchemaLifecycleDecisionRights.PropertyValidContentsCheck	For a user to perform the Propose transition, the description, namespace and version properties of a Schema specification entity must all have a value.

Archived

Figure 3-25 shows the schema life cycle detail rights before a schema specification entity can be transitioned to propose scope.

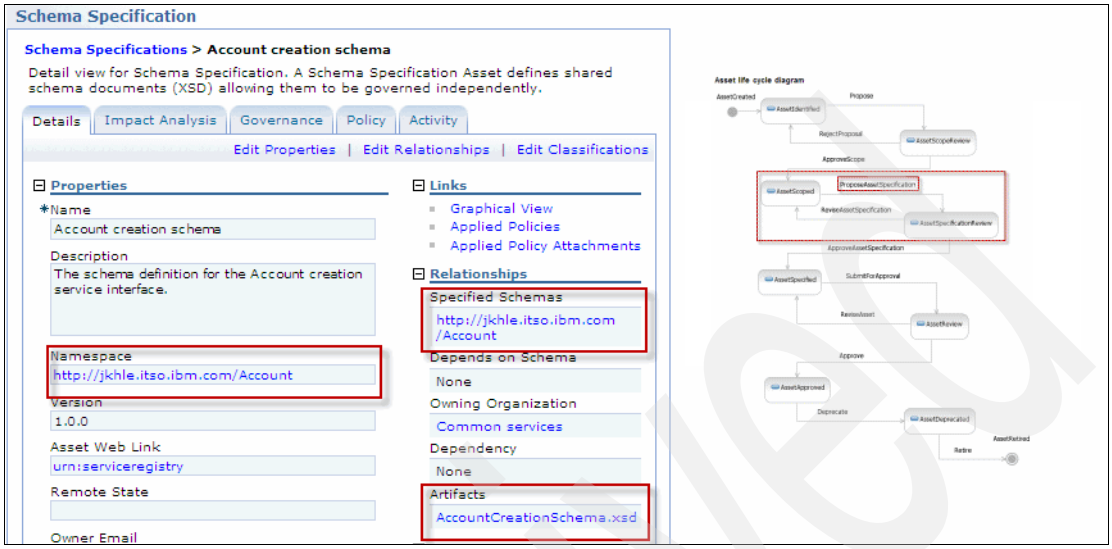


Figure 3-25 Schema life cycle detail rights, propose asset specification

Table 3-54 describes the schema life cycle detail rights to transition a schema specification entity to propose asset specification.

Table 3-54 Schema life cycle detail rights, propose asset specification explanation

Transition	Policy URI	Assertion	Description
Propose Asset Specification	urn:SchemaLifecycleDetailRights_ProposeAssetSpecification	SchemaLifecycleDecisionRights.ArtifactXSDCheck	For a user to perform the propose asset specification transition on a schema specification, the targets of the artifacts relationship must be of type XSD document.
Propose Asset Specification	urn:SchemaLifecycleDetailRights_ProposeAssetSpecification	SchemaLifecycleDecisionRights.NamespaceXSDCheck	For a user to perform the propose asset specification transition on a schema specification, the namespace defined in the schema specification must be the same as that defined in the associated XSD document.

Transition	Policy URI	Assertion	Description
Propose Asset Specification	urn:SchemaLifecycleDetailRights_ProposeAssetSpecification	SchemaLifecycleDecisionRights.XSDSchemaReferencedCheck	For a user to perform the Propose asset transition on a schema specification, the schemas defined in the XSD Document associated with the schema specification must be referenced in the schema specification, as targets of the Specified schemas relationship.

SLA life cycle detail rights

SLA life cycle detail rights in the governance enablement profile are policies that are applied to an SLA entity as defined in the SLA life cycle. Figure 3-26 shows the SLA life cycle detail rights, with request and approve SLA.

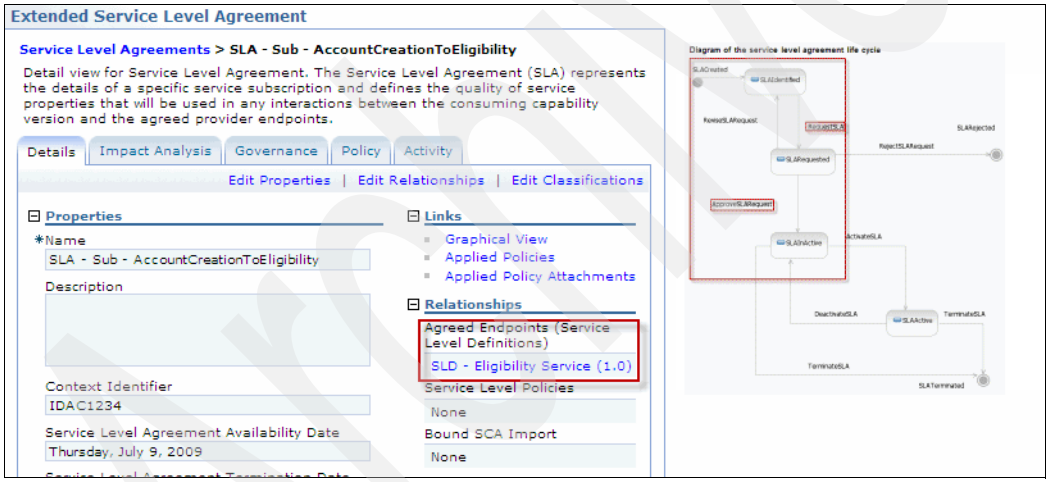


Figure 3-26 SLA life cycle detail rights, request and approve SLA

Table 3-55 describes the SLA life cycle detail rights.

Table 3-55 SLA life cycle detail rights explanation

Transition	Policy URI	Assertion	Description
Request and Approve SLA	urn:SLALifecycleDetailRights_RequestandApproveSLA	SLALifecycleDetailRights.SLAEndpointSubscribableSLDCheck	For a user to perform the request SLA transition or the approve SLA request transition on an SLA entity, the SLA entity must have at least one subscribable SLD.
Activate SLA	urn:SLALifecycleDetailRights_ActivateSLA	SLALifecycleDetailRights.SLASLDOnlineEndpointCheck	For a user to perform the activate SLA transition, an SLD entity associated with an SLA entity must have at least one online endpoint entity, for the environments supported by the SLA entity.

SLD life cycle detail rights

SLD life cycle detail rights in the governance enablement profile are policies that are applied to SLD as defined in the SLD life cycle. Figure 3-27 shows the SLD life cycle detail rights, propose scope.

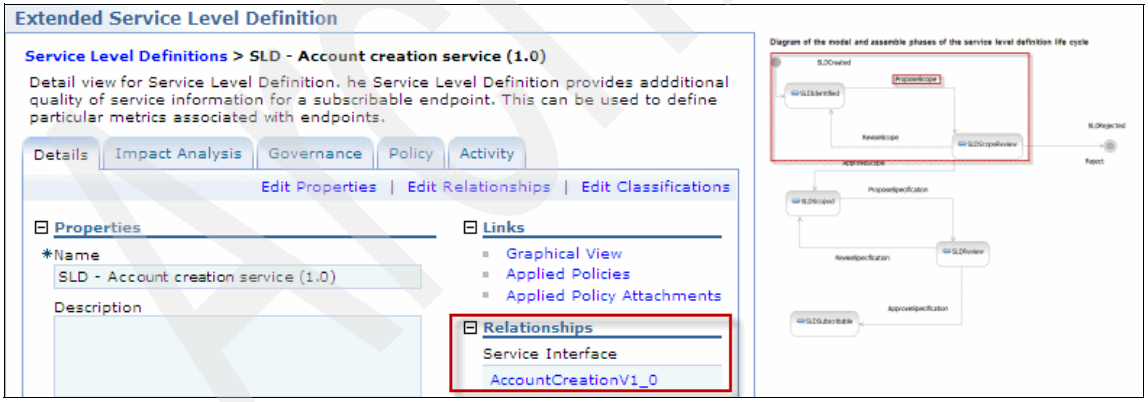


Figure 3-27 SLD life cycle detail rights, propose scope

Table 3-56 lists the SLD life cycle detail rights.

Table 3-56 SLD life cycle detail rights explanation

Transition	Policy URI	Assertion	Description
Propose Scope	urn:SLDLifecycleDetailRights_ProposeScope	SLDLifecycleDetailRights.SLDInterfaceCheck	For a user to perform the Propose scope transition, an SLD must reference a Service interface object

Figure 3-28 shows the SLD life cycle detail rights transition from an SLD to a propose specification.

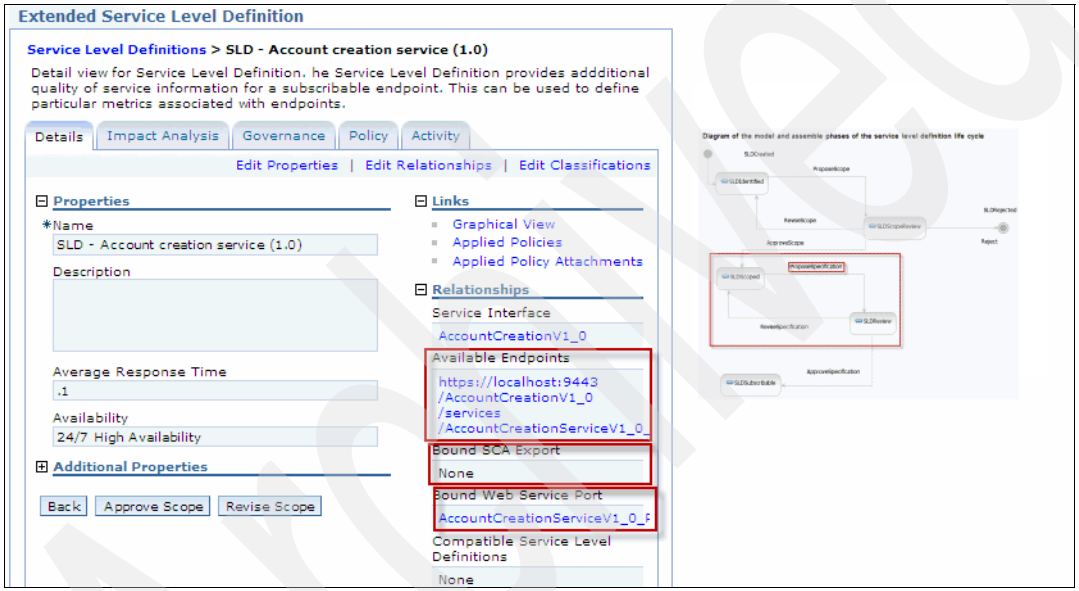


Figure 3-28 SLD life cycle detail rights, propose specification

Archived

Table 3-57 lists the SLD life cycle detail rights.

Table 3-57 SLD life cycle detail rights explanation

Transition	Policy URI	Assertion	Description
Propose Specification	urn:SLDLifecycleDetailRights_ProposeSpecification	SLDLifecycleDetailRights.SLDArtifactCheck	For a user to perform the Propose specification transition, an SLD entity must reference either a Service port entity, Service export entity, or Service endpoint entity

Table 3-58 describes the SLD life cycle detail rights to transition an SLD entity to Retire.

Table 3-58 SLD life cycle rights explanation

Transition	Policy URI	Assertion	Description
Retire	urn:SLDLifecycleDetailRights_Retire	SLDLifecycleDetailRights.SLDRetireCheck	For a user to perform the Retire transition, an SLD entity must have no “Active” or “Inactive” SLAs entity that depend on it

Archived

SOA life cycle detail rights

SOA life cycle detail rights in the governance enablement profile are policies that are applied to a capability version entity. Figure 3-29 on page 140 shows the SOA detail rights for a capability version entity to transition to a propose scope.

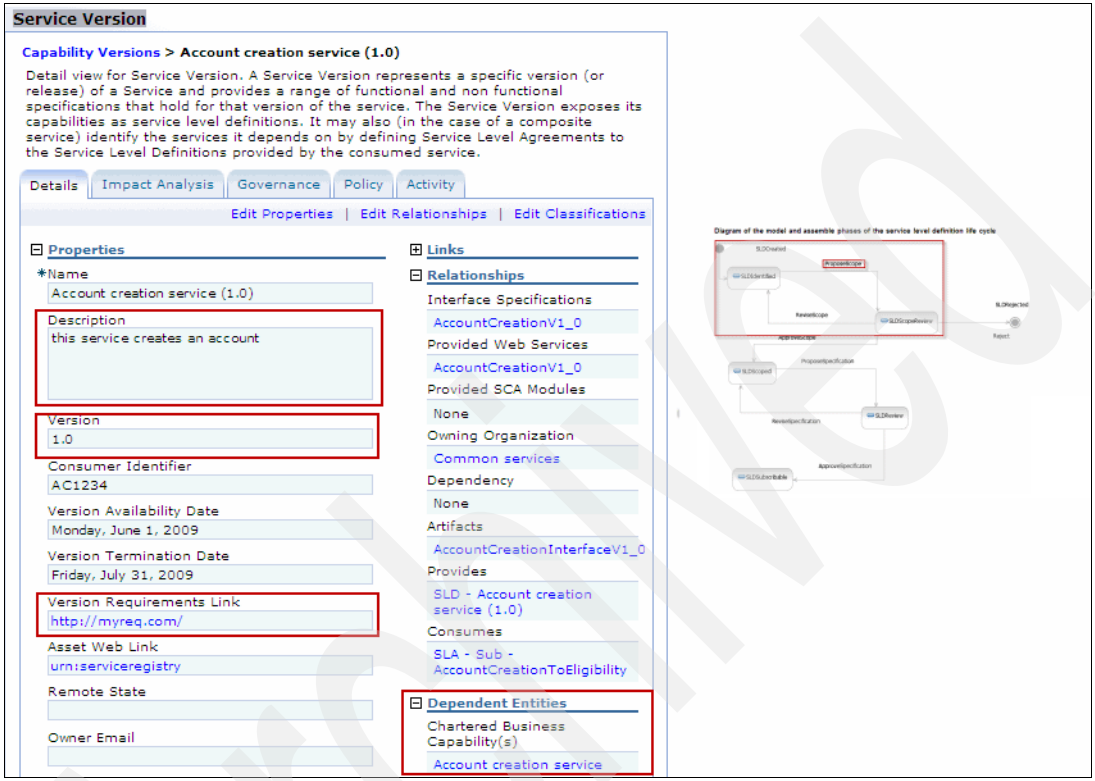


Figure 3-29 SOA life cycle detail rights, propose scope

Archived

Table 3-59 describes the SOA life cycle detail rights to transition a capability version to a propose scope.

Table 3-59 SOA life cycle detail rights explanation

Transition	Policy URI	Assertion	Description
Propose Scope	urn:SOALifecycleDetailRights_ProposeScope	SOALifecycleDetailRights.BusinessCapabilityDefinedCheck	For a user to perform the propose scope transition on a capability version, the capability version must be associated with a business capability.
Propose Scope	urn:SOALifecycleDetailRights_ProposeScope	SOALifecycleDetailRights.CapabilityOwnershipCheck	For a user to perform the propose scope transition on a business capability, an owning organization must be associated with the business capability.
Propose Scope	urn:SOALifecycleDetailRights_ProposeScope	SOALifecycleDetailRights.PropertyValidContentsCheck	For a user to perform the propose scope transition on a capability version, the description, version, and requirements properties must all have a value.

Archived

Figure 3-30 shows the SOA life cycle detail rights for a capability version entity to transition to a propose specification.

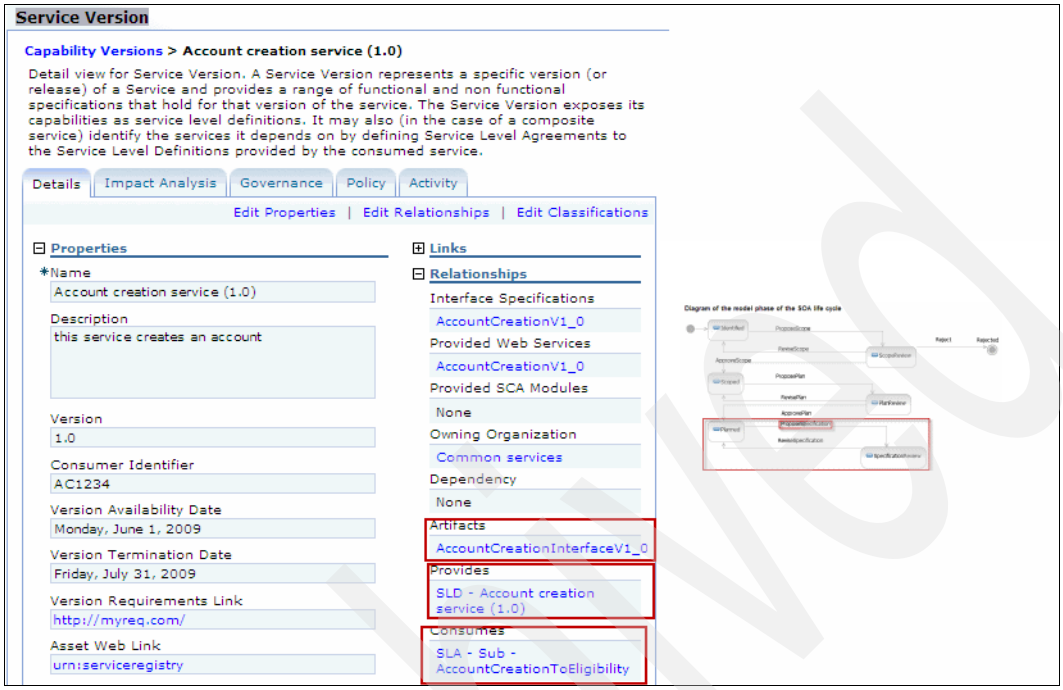


Figure 3-30 SOA life cycle detail rights, propose specification

Table 3-60 describes the SOA life cycle detail rights to transition a capability version to a propose specification.

Table 3-60 SOA life cycle detail rights explanation

Transition	Policy URI	Assertion	Description
Propose Specification	urn:SOALifecycleDetailRights_ProposeSpecification	SOALifecycleDetailRights.SLDMinimumCardinalityCheck	For a user to perform the propose specification transition on a capability version, the capability version must have an associated SLD.

Archived

Transition	Policy URI	Assertion	Description
Propose Specification	urn:SOALifecycleDetailRights_ProposeSpecification	SOALifecycleDetailRights.SLDSubscribableStateCheck	For a user to perform the propose specification transition on a capability version, the SLD associated with the capability version must be in a “Subscribable” state.
Propose Specification	urn:SOALifecycleDetailRights_ProposeSpecification	SOALifecycleDetailRights.SLAMinimumCardinalityCheck	For a user to perform the propose specification transition on a capability version, the capability version must have an associated SLA.
Propose Specification	urn:SOALifecycleDetailRights_ProposeSpecification	SOALifecycleDetailRights.SLAActiveOrInactiveStateCheck	For a user to perform the propose specification transition on a capability version, the SLA associated with the capability version must be in either an “Active” or “Inactive” state.
Propose Specification	urn:SOALifecycleDetailRights_ProposeSpecification	SOALifecycleDetailRights.CapabilityArtifactsCheck	For a user to perform the propose specification, WSDL or SCDL artifacts must be present.

Table 3-61 describes the SOA life cycle detail rights to transition a capability version to a propose staging deployment.

Table 3-61 SOA life cycle detail rights explanation

Transition	Policy URI	Assertion	Description
Propose Staging Deployment	urn:SOALifecycleDetailRights_ProposeStagingDeployment	SOALifecycleDetailRights.SLDStagingSubscribableStateCheck	For a user to perform the propose staging deployment transition on a capability version, any SLD associated with the capability version, that has a staging endpoint, must be in the “Subscribable” state.

Archived

Transition	Policy URI	Assertion	Description
Propose Staging Deployment	urn:SOALifecycleDetailRights_ProposeStagingDeployment	SOALifecycleDetailRights.EndpointStagingOnlineStateCheck	For a user to perform the propose staging deployment transition on a capability version, the staging endpoints associated with that capability version, through the associated SLD, must be in the “Online” state.
Propose Staging Deployment	urn:SOALifecycleDetailRights_ProposeStagingDeployment	SOALifecycleDetailRights.SLAStagingSubscribableStateCheck	For a user to perform the propose staging deployment transition on a capability version, the SLA associated with the capability version, for use in a staging deployment, must be in the “Active” state.

Table 3-62 describes the SOA life cycle detail rights to transition a capability version to a propose production deployment.

Table 3-62 SOA life cycle detail rights explanation

Transition	Policy URI	Assertion	Description
Propose Production Deployment	urn:SOALifecycleDetailRights_ProposeProductionDeployment.	SOALifecycleDetailRights.SLDProductionSubscribableStateCheck	For a user to perform the propose production deployment transition on a capability version, any SLD associated with the capability version, that has a staging endpoint, must be in the “Subscribable” state.
Propose Production Deployment	urn:SOALifecycleDetailRights_ProposeProductionDeployment.	SOALifecycleDetailRights.EndpointProductionOnlineStateCheck	For a user to perform the propose production deployment transition on a capability version, the production endpoints associated with that capability version, through the associated SLD, must be in the “Online” state.

Transition	Policy URI	Assertion	Description
Propose Production Deployment	urn:SOALifecycleDetailRights_ProposeProductionDeployment.	SOALifecycleDetailRights.SLAProductionSubscribableStateCheck	For a user to perform the propose production deployment transition on a capability version, the SLA associated with the capability version, for use in a production deployment, must be in the “Active” state.

3.8.3 Life cycle update policies

The life cycle update policies in the governance enablement profile determine which roles can update or delete objects that are in a specific state in a life cycle. We describe these policies in this section.

Asset life cycle update policies

Asset life cycle update rights in the governance enablement profile are policies that are applied to asset entities as defined in the asset life cycle. Table 3-63 describes the asset life cycle update rights in the governance enablement profile for an asset entity according to roles.

Table 3-63 Asset life cycle update rights explanation

Role	Policy URI	Assertion	Transition
Development	urn:AssetLifecycleUpdateRights_Development	AssetLifecycleUpdateRights.AssetDevelopmentUpdateCheck	► Asset identified ► Asset scoped ► Asset specified ► Asset approved ► Asset deprecated ► Asset retired
Business	urn:AssetLifecycleUpdateRights_Business	AssetLifecycleUpdateRights.AssetBusinessUpdateCheck	► Asset scope review ► Asset review
SOA Governance	urn:AssetLifecycleUpdateRights_SOAGovernance	AssetLifecycleUpdateRights.AssetSOAGovernanceUpdateCheck	► Asset specification review

Capability life cycle update rights

Capability life cycle update rights in the governance enablement profile are policies that are applied to a business capability as defined in the capability life

cycle. Table 3-64 describes the capability life cycle update rights in the governance enablement profile for a business capability entity.

Table 3-64 Capability life cycle update rights explanation

Role	Policy URI	Assertion	Transition
Business	urn:CapabilityLifecycleUpdateRights_Business	CapabilityLifecycleUpdateRights.CapabilityBusinessUpdateCheck	► Capability identified ► Capability deprecated ► Capability retired
SOA Governance	urn:CapabilityLifecycleUpdateRights_SOAGovernance	CapabilityLifecycleUpdateRights.CapabilitySOAGovernanceUpdateCheck	► Charter review ► Capability approved

DOU life cycle update rights

DOU life cycle update rights in the governance enablement profile are policies that are applied to DOU objects as defined in the Asset life cycle. Table 3-65 describes the DOU life cycle update rights for a DOU entity.

Table 3-65 DOU life cycle update rights explanation

Role	Policy URI	Assertion	Transition
Development	urn:DOULifecycleUpdateRights_Development	DOULifecycleUpdateRights.DOUDevelopmentUpdateCheck	► Asset scoped ► Asset specified
Business	urn:DOULifecycleUpdateRights_Business	DOULifecycleUpdateRights.DOUBusinessUpdateCheck	► Asset identified ► Asset scope review ► Asset specification review
SOA	urn:DOULifecycleUpdateRights_SOAGovernance	DOULifecycleUpdateRights.DOUSOAGovernanceUpdateCheck	► Asset review
Administrator	urn:DOULifecycleUpdateRights_Administrator	DOULifecycleUpdateRights.DOUAdministratorUpdateCheck	► Asset approved ► Asset deprecated ► Asset retired

Archived

Endpoint life cycle update rights

Endpoint life cycle update rights in the governance enablement profile are policies that are applied to a endpoint entity as defined in the endpoint life cycle. Table 3-66 describes the endpoint life cycle update rights in the governance enablement profile for a endpoint entity.

Table 3-66 Endpoint life cycle update rights explanation

Role	Policy URI	Assertion	Transition
Operations	urn:EndpointLifecycleUpdateRights_SOAGovernance	EndpointLifecycleUpdateRights.EndpointOperationsUpdateCheck	► Offline ► Online ► Endpoint retired

Schema life cycle update rights

Schema life cycle update rights in the governance enablement profile are policies that are applied to a schema specification entity as defined in the asset life cycle. Table 3-67 describes the schema life cycle update rights in the governance enablement profile for a schema specification entity.

Table 3-67 Schema life cycle update rights explanation

Role	Policy URI	Assertion	Transition
Development	urn:SchemaLifecycleUpdateRights_Development	SchemaLifecycleUpdateRights.SchemaDevelopmentUpdateCheck	► Asset identified ► Asset scoped ► Asset specification review ► Asset specified
SOA Governance	urn:SchemaLifecycleUpdateRights_SOAGovernance	SchemaLifecycleUpdateRights.SchemaSOAGovernanceUpdateCheck	► Asset scope revew ► Asset review
Administrator	urn:SchemaLifecycleUpdateRights_Administrator	SchemaLifecycleUpdateRights.SchemaAdministratorUpdateCheck	► Asset approved ► Asset deprecated ► Asset retired

Service interface life cycle update rights

Service interface life cycle update rights in the governance enablement profile are policies that are applied to a service interface specification entity as defined in the asset life cycle.

Archived

Table 3-68 describes the service interface life cycle update rights in the governance enablement profile for a service interface specification entity.

Table 3-68 Service interface life cycle update rights explanation

Role	Policy URI	Assertion	Transition
Development	urn:ServiceInterfaceLifecycleUpdateRights_Development	ServiceInterfaceLifecycleUpdateRights.ServiceInterfaceDevelopmentUpdateCheck	► Asset identified ► Asset scoped ► Asset specification review ► Asset specified
SOA Governance	urn:ServiceInterfaceLifecycleUpdateRights_SOAGovernance	ServiceInterfaceLifecycleUpdateRights.ServiceInterfaceSOAGovernanceUpdateCheck	► Asset scope review ► Asset review
Administrator	urn:ServiceInterfaceLifecycleUpdateRights_Administrator	ServiceInterfaceLifecycleUpdateRights.ServiceInterfaceAdministratorUpdateCheck	► Asset approved ► Asset deprecated ► Asset retired

SLA life cycle update rights

SLA life cycle update in the governance enablement profile rights are policies that are applied to SLA entity as defined in the SLA life cycle. Table 3-69 describes the SLA life cycle update rights for an SLA entity.

Table 3-69 SLA update rights explanation

Role	Policy URI	Assertion	Transition
Development	urn:SLALifecycleUpdateRights_Development	SLALifecycleUpdateRights.SLADevelopmentUpdateCheck	► SLA identified ► SLA requested
Operations	urn:SLALifecycleUpdateRights_Operations	SLALifecycleUpdateRights.SLAOperationsUpdateCheck	► SLA active ► SLA inactive

Archived

SLD life cycle update rights

SLD life cycle update in the governance enablement profile rights are policies that are applied to an SLD entity as defined in the SLD life cycle. Table 3-70 describes the SLD life cycle update rights for an SLD entity.

Table 3-70 SLD life cycle update rights explanation

Role	Policy URI	Assertion	Transition
Development	urn:SLDLifecycleUpdateRights_Development	SLDLifecycleUpdateRights.SLDDevelopmentUpdateCheck	► SLD identified ► SLD scoped
SOA Governance	urn:SLDLifecycleUpdateRights_SOAGovernance	SLDLifecycleUpdateRights.SLDSOAGovernanceUpdateCheck	► SLD scope review ► SLD review ► SLD deprecated
Operations	urn:SLDLifecycleUpdateRights_Operations	SLDLifecycleUpdateRights.SLDOperationsUpdateCheck	► SLD review ► SLD subscribable ► SLD superceded

SOA life cycle update rights

SOA life cycle update rights in the governance enablement profile are policies that are applied to a capability version as defined in the SOA. Table 3-71 describes the SOA life cycle update rights for a capability version entity.

Table 3-71 SOA life cycle update rights explanation

Role	Policy URI	Assertion	Transition
Development	urn:SOALifecycleUpdateRights_Development	SOALifecycleUpdateRights.SOADevelopmentUpdateCheck	► Scoped ► Planned ► Specified ► Realization review
Business	urn:SOALifecycleUpdateRights_Business	SOALifecycleUpdateRights.SOABusinessUpdateCheck	► Identified ► Plan review

Archived

Role	Policy URI	Assertion	Transition
SOA Governance	urn:SOALifecycleUpdateRights_SOAGovernance	SOALifecycleUpdateRights.SOASOAGovernanceUpdateCheck	► Scope review ► Specification review ► Certification review ► Operational review ► Deprecated ► Retired
Operations	urn:SOALifecycleUpdateRights_Operations	SOALifecycleUpdateRights.SOAOperationsUpdateCheck	► Realized ► Staging review ► Staged ► Certified ► Operational

Archived

Part 2

Building WSRR solutions

Archived

© Copyright IBM Corp. 2009. All rights reserved.

Archived

4

JKHL Enterprises case study

This book and subsequent IBM Redpapers™ publications use a common example organization to illustrate service-oriented architecture (SOA) governance using IBM WebSphere Service Registry and Repository (WSRR) as the authoritative registry and repository. This chapter introduces that case study and includes the following topics:

- Introduction to the case study
- JKHLE SOA governance vision and requirements
- JKHLE runtime environment
- The JKHLE SOA governance solution
- JKHLE service governance with the WSRR scenario
- JKHLE organizational structure and personas

© Copyright IBM Corp. 2009. All rights reserved.

4.1 Introduction to the case study

JKHL Enterprises is a fictitious supply company facing the typical challenges that arise when striving to reach the potential benefits of SOA solutions. JKHLE is an example that appears in other IBM materials. We adapted the example in some places to illustrate particular points that are relevant to building an SOA governance solution with WSRR.

JKHL Enterprises (JKHLE) is a mature company that adopted SOA to deliver the ability to respond rapidly to changing business needs. SOA allows JKHLE to reconfigure existing services and to maintain a clear mapping between business needs and IT implementations.

JKHLE has taken an incremental approach to adopting SOA by adding function as the company matures. As part of this SOA adoption, JKHLE believes that SOA governance is essential for success. They established an *SOA Center of Excellence* with associated roles and communication plans. Currently, they have a manual, paper-based approach to service life cycle governance and are facing challenges that are impeding their reaching the full potential of SOA.

JKHLE has identified the following pain points that are obstructing them from reaching their goals with their current solution:

- Inability to identify new service candidates and prioritize them
- Inability to find or reuse services leading to duplication of services
- Inability to apply and enforce a set of standards leading to interpretability issues
- No well-defined process for enhancing and versioning services
- No method to visualize the impact of changes
- No management insight to the health and performance of the SOA
- Inconsistent enforcement of operational policies throughout the runtime environment
- Inflexibility of the enterprise service bus (ESB) infrastructure because the ESB is still tightly coupled to the provider

4.2 JKHLE SOA governance vision and requirements

JKHLE has captured a set of requirements for SOA governance or, more specifically, a service governance solution that builds from their current SOA

infrastructure. The SOA governance solution provides them with the SOA insight that they need to meet business agility goals.

Figure 4-1 shows the vision for a holistic closed-loop governance solution that provides the following benefits:

- Design and runtime registry and repository, which we illustrate in this book
- A runtime environment that supports policy enforcement and dynamic endpoint resolution
- A runtime management solution that can monitor the health of the services

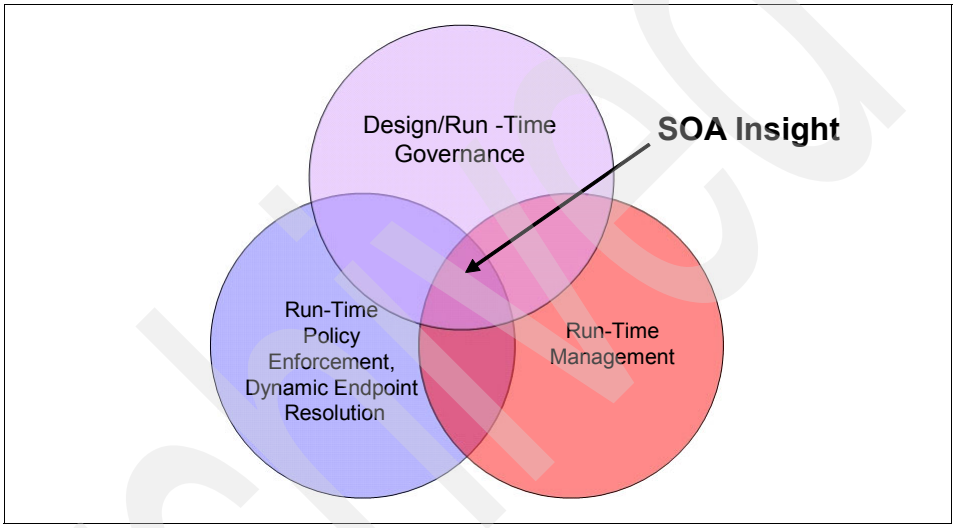


Figure 4-1 The SOA governance solution vision for JKHLE

JKHLE also requires the following capabilities from a registry and repository:

- Govern the service life cycle from new service creation through to service retirement
- Provide a prescriptive approach to service versioning
- Govern, provide visual impact, and notify interested parties of service changes
- Provide multiple methods to find and publish services to and from the registry
- Manage the consumer and provider relationship
- Enforce design time standards by applying policies
- Manage and govern WS-Polices with the services to which they are applied
- Produce custom reports to provide management team insight such as business value and operational metrics

Archived

- Easy access to the registry from the ESBs in order to perform dynamic endpoint lookups, consumer contract validation, and policy resolution
- Integrate the registry with IBM Tivoli Composite Application Manager (ITCAM) for SOA to capture the metrics metadata

JKHLE found that IBM WebSphere Service Registry and Repository (WSRR) meets their requirements. By implementing an SOA governance solution with WSRR, JKHLE can secure value from the following capabilities:

- Governing new service creation and determining the priorities as well as the funding for the establishment of these services
- Identifying who owns services so that they can better drive requirements against services and foster trust as well as harbor reuse
- Ensuring services adhere to standards and leading practices, thus reducing deployment and management risk
- Ensuring a consistent service change management capability which reduces potential fatal impact to the business
- Providing a more explicit controlled service versioning process for services that reduces operational and maintenance costs as well as the impact of unwarranted change on service consumers
- Managing service consumers, thus enabling more efficient capacity planning
- Increasing ESB flexibility
- Providing reports that give management team insight to the business value of SOA and operational metrics
- Enforcing a starter set of domains based on WS-Policy throughout the runtime environments

4.3 JKHLE runtime environment

The JKHLE runtime environment currently consists of a federation of ESBs:

- IBM WebSphere DataPower X15
- IBM WebSphere Enterprise Service Bus
- IBM WebSphere Message Broker
- Other third-party ESBs

JKHLE takes advantage of IBM Tivoli Composite Application Manager (ITCAM) for SOA for runtime management of their SOA. ITCAM for SOA provides monitoring and management of services and mediations in an SOA environment. It monitors a wide variety of metrics on a large number of application server

runtime environments and ESBs. It also manages mediations in WebSphere Enterprise Service Bus.

4.4 The JKHLE SOA governance solution

JKHLE selected WSRR as their authoritative registry and repository to aid in the implementation of SOA governance and, more specifically, in the implementation of service governance. Figure 4-2 shows the complete SOA governance vision for JKHLE and the products that enable that solution.

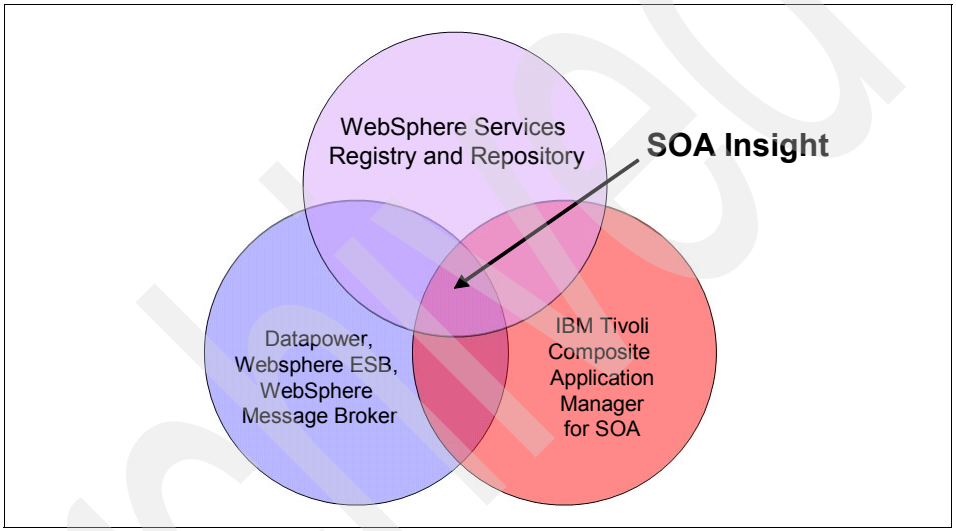


Figure 4-2 JKHLE SOA governance vision

This book focuses on how JKHLE customizes the WSRR governance enablement profile (GEP) to enable their service governance process. The complete JKHLE solution is detailed in a series of IBM Redpapers publications:

- *Integrating WebSphere Service Registry and Repository with WebSphere Process Server and WebSphere ESB*, REDP-4557
- *Integrating WebSphere Service Registry and Repository with WebSphere MQ and WebSphere Message Broker*, REDP-4558
- *Integrating WebSphere Service Registry and Repository with WebSphere DataPower*, REDP-4559
- *Integrating WebSphere Service Registry and Repository with IBM Tivoli Security Policy Manager*, REDP-4561

Archived

4.5 JKHLE service governance with the WSRR scenario

JKHLE decided that the WSRR governance enablement profile fulfills most of their needs for the governance of services. However, they want to make some customizations to the governance enablement profile to align it more closely with their defined governance processes.

JKHLE customizes the governance enablement profile by making the following modifications to the business model and the life cycles of the profile. We provide step-by-step instructions on how to make these changes in Chapter 3, “Introduction to the governance enablement profile” on page 77.

- The governance enablement profile governs service interfaces separately in the service interface specification entity, which has its own life cycle. The governance process that JKHLE defined does not require governance at this fine-grained level. Thus, JKHLE chose to remove the service interface specification entity and govern the service interface in the capability version.
- When reviewing the model, JKHLE liked the notion of what the document of understanding (DOU) entity represents but decided to change the name of the document to *subscription request*. At JKHLE, a subscription request represents an agreement of intent between the consumer and the provider for a specific capability version. JKHLE also simplify the asset life cycle, which is applied to the subscription request by removing the life cycle states that are not required.
- For schemas that are shared throughout multiple services, JKHLE used the schema specification entity. They applied the asset life cycle that they customized for the subscription request to the schema specification as well. JKHLE govern schemas that are specific to a service and not shared across services using the capability version entity along side the service interface.
- JKHLE changed the life cycles on the service level definition and the capability version by removing the life cycles states that their processes do not require.

JKHLE also completed the following tasks to implement this solution:

- JKHLE configured WSRR security according to the users and roles as described in the next section, JKHLE organizational structure and personas. For more information about configuring security, see Chapter 7, “Security” on page 263.
- To help enforce proper service definition in the registry, JKHLE defined a design time policy, as described in Chapter 9, “Policies” on page 301.
- JKHLE defined how services are promoted to their target runtime environments, as described in Chapter 8, “Promotion” on page 289.

- JKHLE defined and created a set of reports that provide management insight, as described in Chapter 10, “Reports” on page 349.
- Finally, JKHLE governed their services in the scenarios, as described in Part 3, “Scenarios” on page 399.

4.6 JKHLE organizational structure and personas

JKHLE has the following organizational structure (as shown in Figure 4-3):

- *JKHL Enterprises* is the a top-level organization that represents the complete enterprise
- *Common services* is a child organization of JKHLE and is the department that develops and delivers services that are shared throughout the enterprise
- *Commercial* is a child organization of JKHLE that represents the commercial line of business

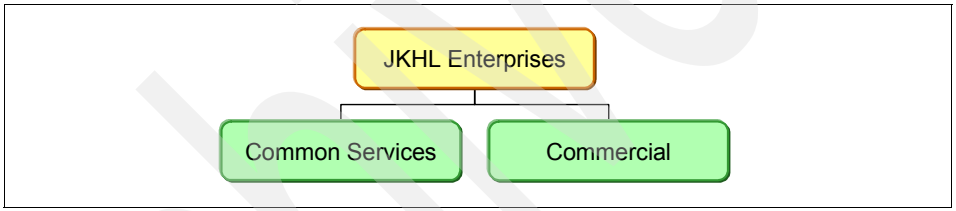


Figure 4-3 JKHLE organizational structure

Archived

JKHLE identified the users and user roles as shown in Figure 4-4.

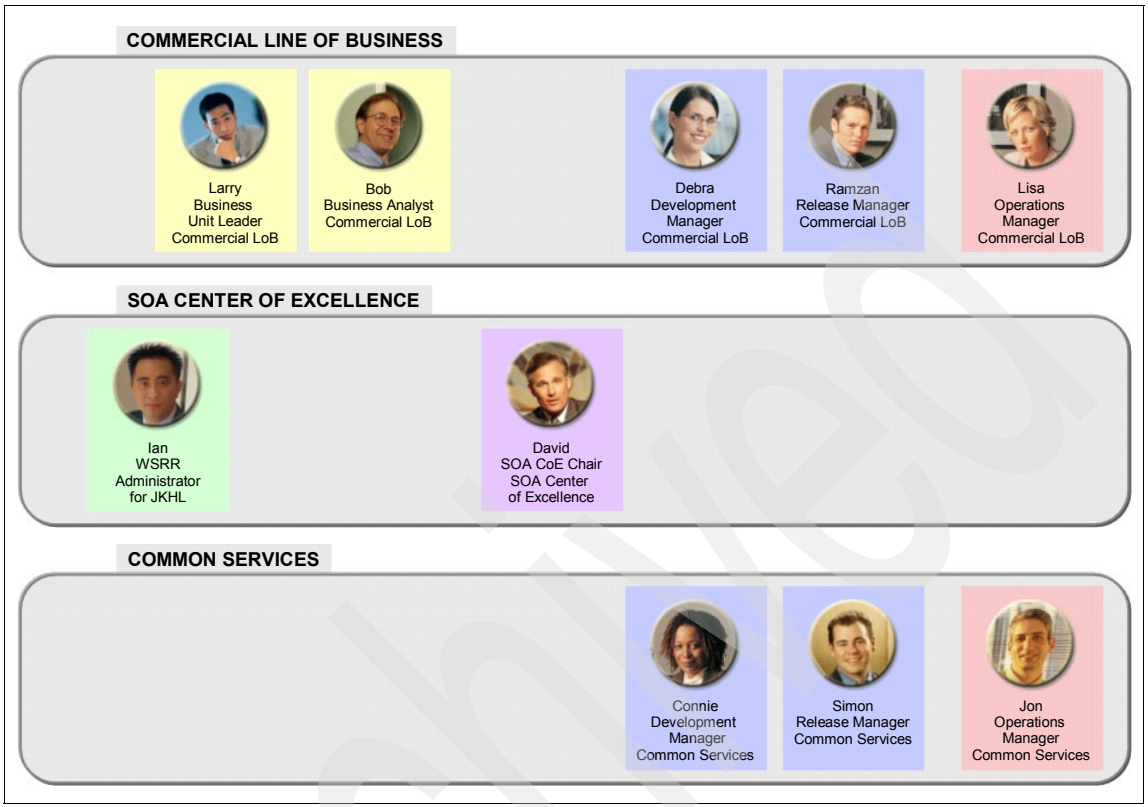


Figure 4-4 JKHLE users and user roles

Within the JKHLE case study, the following JKHLE employees participate in various phases of the service governance process:

- *Larry*, who is the Business Unit Leader for the JKHLE Commercial Line of Business
- *Bob*, who is the Business Analyst for the JKHLE Customer Care Application
- *Debra*, who is the Development Manager for the JKHLE Customer Care Web Application
- *Ramzan*, who is the Release and Project Manager for Debra in the JKHLE Commercial Line of Business
- *Connie*, who is the Development Manager for the JKHLE Common Services team
- *Simon*, who is the Release and Project Manager for Connie in the JKHLE Common Services team

- *Jon*, who is the Operations Release Manager for the JKHLE Common Services data center
- *Lisa*, who is the Operations Release Manager for the JKHLE Commercial applications and data center
- *David*, who is the JKHLE SOA Center of Excellence chairman
- *Ian*, who is the WSRR Administrator responsible for configuring WSRR to support the JKHLE governance processes and policies

Archived

Archived

5

Modeling in WebSphere Service Registry and Repository Studio

This chapter introduces modeling in WebSphere Service Registry and Repository Studio. It includes the following sections:

- Overview of WebSphere Service Registry and Repository Studio
- Using models with WebSphere Service Registry and Repository Studio
- Extending templates versus editing a template
- Customizing the governance enablement profile for JKHL Enterprises
- Applying JKHLE's customizations to the governance profile taxonomy
- Generating a profile
- Transferring .emx models between clients

© Copyright IBM Corp. 2009. All rights reserved.

5.1 Overview of WebSphere Service Registry and Repository Studio

WebSphere Service Registry and Repository Studio (WSRR Studio) is a stand-alone Eclipse application that you can use to complete the following tasks:

- Create and edit WSRR configuration profiles
- Generate business models, classification systems, and life cycles from Unified Modeling Language (UML) diagrams
- Create and edit stored artifacts
- Generate custom reports

> **Note:** Prior to WebSphere Service Registry and Repository V6.3, this functionality was referred to as the *WebSphere Service Registry and Repository Eclipse plug-in*. It is now renamed to the *WebSphere Service Registry and Repository content plug-in*.

WSRR Studio is available for Microsoft Windows® and Linux® platforms.

Because WSRR Studio is based on Eclipse, the suite also includes the standard plug-ins that are shipped with the Eclipse framework, such as the ability to synchronize with a Concurrent Versions System (CVS) code control server. You can also install additional Eclipse plug-ins into the development environment.

The remainder of this chapter focuses on the modelling component of WSRR Studio. (We discuss reporting in detail in Chapter 10, “Reports” on page 349.) We do not discuss the WSRR content plug-in in this book. For more information, see the WSRR Information Center at:

http://publib.boulder.ibm.com/infocenter/sr/v6r3/topic/com.ibm.sr.doc/twsr_pluginsetup.html

5.2 Using models with WebSphere Service Registry and Repository Studio

This section discusses the following:

- Configuration profiles
- Profile templates
- System models
- Configuration profile files

- Business model systems
- Classification systems
- Life cycles

5.2.1 Configuration profiles

You can use WSRR Studio to create and customize configuration profiles. A *profile* is a package of all the WSRR configuration artifacts, including the business model definitions, life cycle definitions, classification systems, governance policies, security roles and permission mappings, as well as custom plug-ins and user interface customizations. WSRR Studio uses UML tooling to visually build business models, classification systems, and life cycles.

5.2.2 Profile templates

WSRR Studio includes two *profile templates* from which you can customize a profile:

- Basic profile: WebSphere Service Registry and Repository V6.3
- Governance enablement profile: WebSphere Service Registry and Repository V6.3

These templates are a starting point, but you can extend and customize them to create a profile that is tailored to your company's need.

5.2.3 System models

System models are loaded into WSRR during application startup and cannot be changed or customized in any way. Their UML representations are included in WSRR Studio. You can create new business model systems that can reference these system models. For example, you can create a new business model class that has a relationship to a WSDL document, which is included in the WSDL_1.1 model. The system models are part of WSRR and are not included in a WSRR configuration profile.

Archived

Table 5-1 lists the system models.

Table 5-1 WSRR system models

Business Model Name	Included classes
Base Model	BaseObject Document ExtensionLogicalObject Logical Object
IBMSCA	ExportDocument ImportDocument ModuleDocument
Policy	Policy PolicyAttatchment PolicyDocument PolicyExpression PolicyReference
WSDL_1.1	SOAPAddress SOAPBinding WSDLBinding WSDLDocument WSDLFault WSDLInput WSDLMessage WSDLOperation WSDLOutput WSDLPart WSDLPort WSDLPortType WSDLService
WSRRBaseModel	GenericDocument GenericObject GraphQuery PropertyQuery QueryObject XMLDocument
XSDModel	AttributeDeclaration ComplexTypeDefinition ElementDeclaration LocalAttribute SimpleTypeDefinition XSDDocument XSDType

Archived

5.2.4 Configuration profile files

When you create a new profile in WSRR Studio, a corresponding *configuration profile files* directory is created. If no changes are made to the profile, the configuration files in this directory are the same as those files that ship with the product configuration. For example, if you create a profile based on the governance enablement profile—the WebSphere Service Registry and Repository V6.3 template—and then export the profile to a WSRR system, the profile is identical to the governance enablement profile that is included with the WSRR server installation.

5.2.5 Business model systems

The template that you use to create a new profile affects which business models are marked as read-only and, thus, determines whether you can generate business model definition files from them. Technically, all non-system models can be replaced in WSRR by an administrator; however, this practice is recommended only for advanced users. So, WSRR Studio marks business models that are not intended to change as read-only to discourage users from editing them.

Table 5-2 compares the read-only business models.

Table 5-2 Comparison of read-only business models

Business model full name	Business model short name	WSRR configuration name (label)	Basic Profile	Governance enablement profile	Included Classes
Asset Model	6_3_ALEModel	WSRR Asset Business Model	Read-only	Read-only	Asset Organization
GEP63	6_3_ProfileModel	Governance enablement profile business model	Editable	Read-only	ApplicationVersion BusinessApplication BusinessCapability Business Process BusinessService CapabilityVersion DOU ProcessVersion ServiceInterfaceSpecification ServiceLevelAgreement ServiceLevelDefinition ServiceVersion
GPX63	6_3_GovernanceProfile-Extensions	Governance Profile Business Model Extensions	Editable	Editable	ExtendedServiceLevel-Agreement ExtendedServiceLevel-Deffinition
Service Discovery Model	6_3_ServiceDiscoveryModel	Technical Model	Read-only	Read-only	EnterpriseApplication EnterpriseModule

Archived

Business model full name	Business model short name	WSRR configuration name (label)	Basic Profile	Governance enablement profile	Included Classes
Service Model	6_3_ServiceModel	WSRR Service Model	Read-only	Read-only	Connection ExtensionLogicalObject ExtensionServiceEndpoint GlobalElement GlobalType ManualClientMQEndpoint ManualClientChannelTableM Q- Endpoint ManualEndpoint ManualMQEndpoint ManualSOAPEndpoint MQEndpoint Queue QueueManager SCAExport SCAImport Schema Service ServiceBinding ServiceEndpoint ServiceExport ServiceImport ServiceInterface ServiceLibrary ServiceMessage ServiceModule ServiceOperation ServicePort SOAPAddress SOAPServiceEndpoint WSDLExport WSDLImport

The difference between the basic profile and governance enablement profile templates is what is included in the configuration profile files directory by default. As we described previously, the contents of this directory structure mimic exactly the corresponding profiles that ship with WSRR.

Note: The basic profile contains the same UML models as the governance enablement profile.

By default, the UML models that are included in a template are set to not generate business model configuration files (`generateOWL=false`). If you chose the basic profile template as a starting point and then add new models, only the configuration files for the new models are generated and, thus, none of the governance enablement profile models, such as the GEP63 business model, are added into the configuration profile.

If you want to include some or all of the governance enablement profile models, you can change the `generateOWL` property to `true` for each of the required

models. If you use the governance enablement profile template as a starting point, the GEP63 business model, for example, is already present in the configuration profile files. So, you do not need to generate it.

> **Note:** The GEP63 business model is read only in the governance enablement profile but is editable in the basic profile.

5.2.6 Classification systems

As with the business model definitions, the UML diagrams for the taxonomies that are included in the basic profile and the governance enablement profile templates are same so that you can use the ALEBusinessDomians taxonomy in a customized profile based on the basic profile template if required.

Table 5-3 describes each taxonomy.

Table 5-3 Taxonomies in the basic profile and governance enablement profile

Taxonomy Name	Label	Description
ALEBusinessDomains	Business Domains	Not included in the Basic Profile by default. This classification system is a duplicate of the default classifications stored in Rational Asset Manager. Rational Asset Manager provides a classification system to classify assets into various business domains. This classification system allows these classifications to be synchronized from Rational Asset Manager into WSRR. If the taxonomy is updated in Rational Asset Manager this should also be updated in WSRR.
core	core	Included in both profiles by default. Required by WSRR to support XSD templates, Document Groups, SCA Libraries, and Validation Reports.
PolicySetTaxonomy	Policy Set Classifications	Included in both profiles by default. Used to classify discovered policy sets and associated policy files.
PolicyTaxonomy	Policy Classifications	Included in both profiles by default. Used to classify policy logical expressions/policies based on their domain.
GovernanceProfileTaxonomy	Governance Profile Taxonomy	Included in both profiles by default. Used to classify environments and business domains.

Archived

5.2.7 Life cycles

The UML for the life cycle is the same in the governance enablement profile and the basic profile templates, although the basic profile's configuration profile files do not contain all of the composite life cycles of the governance enablement profile. You can add the other composite life cycles into a customized profile if needed.

Note, however, that you cannot toggle generation of the life cycle files in WSRR Studio at the composite life cycle level. Therefore, if you regenerate the life cycle from a basic profile template, the composite life cycles from the governance enablement profile will also be generated. To produce a configuration profile with only the composite life cycle from the basic profile, you need to delete the other composite life cycles so that only the default life cycle remains.

5.3 Extending templates versus editing a template

WSRR Studio encourages the extension of template profiles. By extending a template profile, when a template is updated in a maintenance release of WSRR Studio, you can apply the update by replacing the relevant UML and configuration files. Then, you can simply regenerate the customized profile, and the updated profile will include the maintenance fix or fixes.

Although extending a template profile is encouraged, you can also override the read-only settings on the business models. In some circumstances, it might make more sense to edit the governance enablement profile rather implementing it again from the Basic Profile. The end result would be a subset of what is contained in the governance enablement profile. The advantage is that it takes less work to start from the governance enablement profile and remove items from it, but the disadvantage is that any changes have to be merged manually in when a maintenance release of WSRR Studio is shipped. This can be made easier with the use of a code control system, such as CVS.

Archived

5.4 Customizing the governance enablement profile for JKHL Enterprises

Note: In the examples in this book, we use a case study about a fictional company named *JKHL Enterprises* (JKHLE). For information about this case study, see Chapter 4, “JKHL Enterprises case study” on page 153.

JKHLE has identified a number of customizations that they want to apply to the governance enablement profile. They want to use WSRR Studio to make the relevant changes to the GEP63 and GPX63 business models; the SLA, asset, and SOA governance life cycles; as well as the governance profile taxonomy.

JKHLE begins by installing WSRR Studio as described in the WSRR Information Center:

http://publib.boulder.ibm.com/infocenter/sr/v6r3/topic/com.ibm.sr.studio.doc/twsr_install_studio.html

Note: The scenarios that we use in this chapter require a minimum of WebSphere Service Registry and Repository Studio V6.3 Fix Pack 1. Note that some of the screen captures that we use in this chapter were taken from the V6.3 GA release. Therefore, the figures that we show might differ from WebSphere Service Registry and Repository V6.3 Fix Pack 1.

5.4.1 Creating the configuration project for JKHLE

JKHLE wants to reduce the complexity of the governance enablement profile rather than extend it. They can either start with the basic profile template and extend it, or they can reduce the governance enablement profile template by overriding the read-only setting on the GEP63 business model. They want to keep the governance enablement profile as close to its original state as possible so that if, at a future stage, they want to migrate from their reduced profile to the governance enablement profile, they can do so with minimal disruption.

After considering the options, as described in 5.3, “Extending templates versus editing a template” on page 170, JKHLE decides to base the customized profile on the governance enablement profile template.

To customize the profile on the governance enablement profile, JKHLE begins by creating a new configuration project as follows:

1. Click **File** → **New** → **WSRR Configuration Project**, as shown in Figure 5-1.

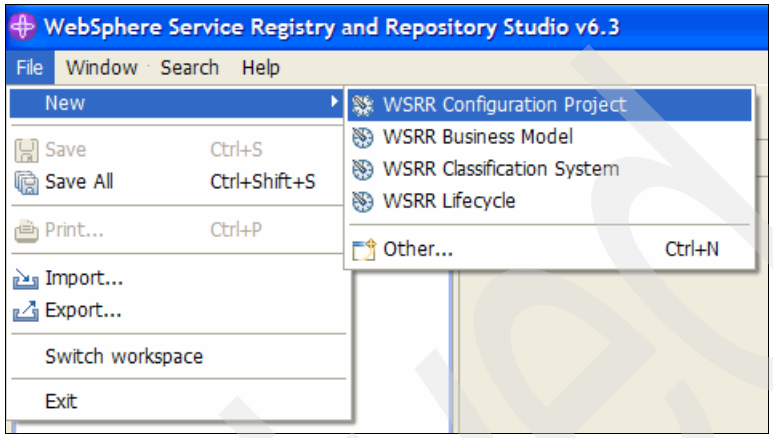


Figure 5-1 Creating a new configuration project

2. The Create a WSRR Configuration Project window opens, as shown in Figure 5-2 on page 173. In this window, complete these steps:
   a. Enter a configuration project name, such as `JKHL Enterprises Configuration Profile`.
   b. Enter a location or leave the default location selected.
   c. Select the “Governance Enablement Profile - WSRR v6.3” option.
   d. Click **Finish**.

Archived

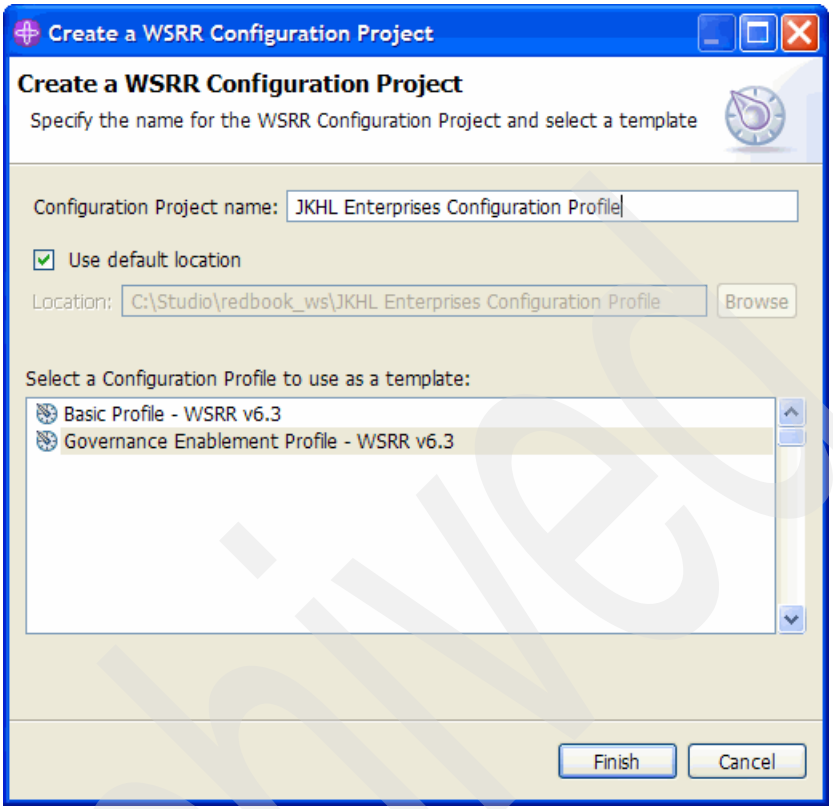


Figure 5-2 Specifying the new configuration project details

3. Review the configuration project as shown in Figure 5-3.

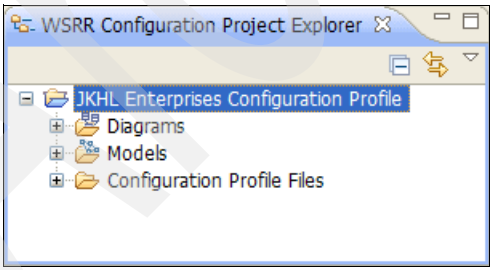


Figure 5-3 Configuration profile

Archived

After you create the configuration project, it consists of the following folders:

- Diagrams

  Includes the UML Model of the WSRR business models, life cycles, and classification systems.

- Models

  Includes the same UML models as the diagrams folder as well as all the individual artifacts that make up the diagram, for example each class and trigger.

- Configuration Profile Files

  Includes all the artifacts that make up a standard WSRR configuration profile, which is where the generated files are stored.

5.4.2 Applying JKHLE's customizations to the GEP63 business model

This section describes the customizations that JKHLE makes to the GEP63 business model.

Removing the read-only setting on the GEP63 business model

Because JKHLE wants to simplify the GEP63 business model from the governance enablement profile template, they need to remove the read-only setting on the GEP63 business model system.

Archived

To remove this setting, in the WSRR Configuration Project Explorer, follow these steps:

1. Select **JKHL Enterprises Configuration Profile** → **Diagrams** → **6_3_ProfileModel**, as shown in Figure 5-4.

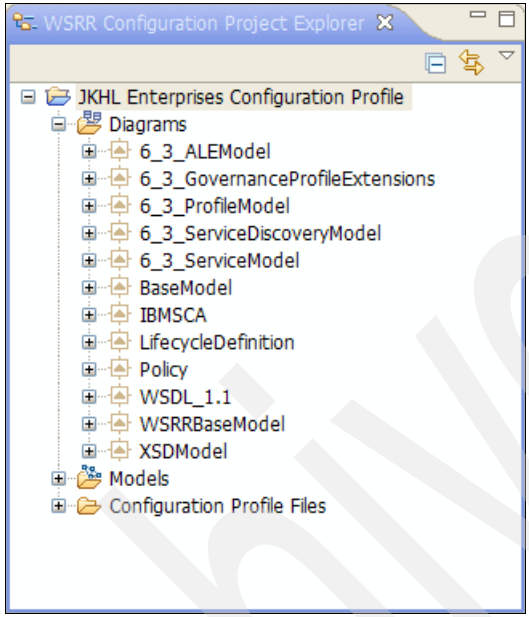


Figure 5-4 Selecting the profile

Archived

Note: When you select the 6_3_ProfileModel node, its name is populated with the name from the model to *<<BusinessModelPackage, ReadOnly>> GEP63* (see Figure 5-3).

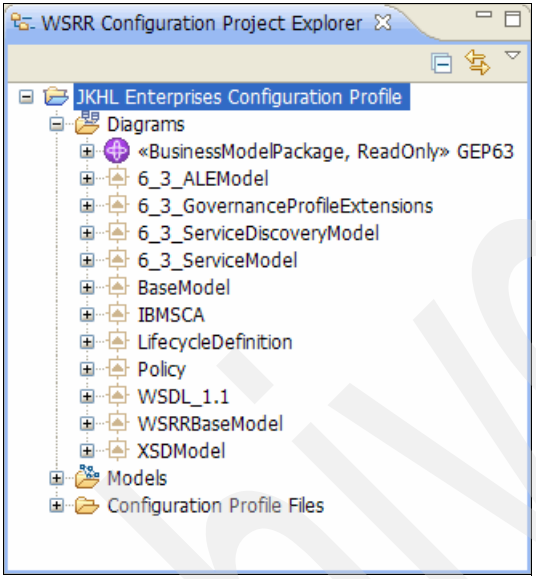


Figure 5-5 The 6_3_ProfileModel with the name populated

2. Click **<<BusinessModelPackage>> GEP63**.

Archived

3. On the Properties tab, shown in Figure 5-6, click **Profiles**, and then select **ReadOnlyProfile**. Click **Remove Profile**.

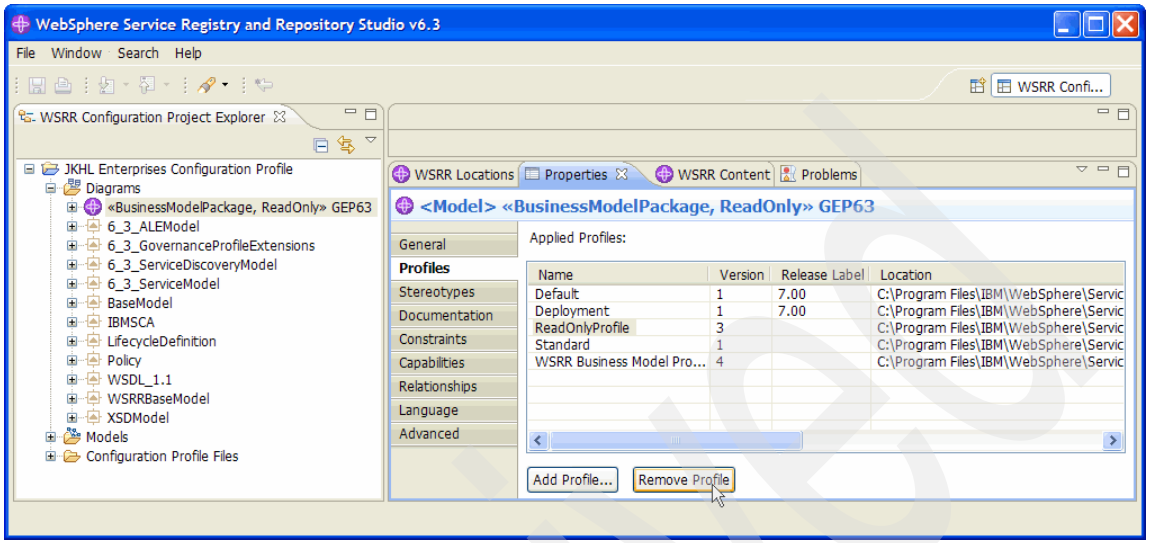


Figure 5-6 Removing the ReadOnlyProfile

4. In the warning message that displays, click **OK** to override the default read-only setting of the GEP63 business model in the governance enablement profile template.

> **Note:** The name of the node in the WSRR Configuration Project Explorer is now called *<<BusinessModelPackage>> GEP63* rather than *<<BusinessModelPackage, ReadOnly>> GEP63*.

Archived

Manipulating the ServiceInterfaceSpecification class

JHKLE wants to remove the ServiceInterfaceSpecification class. So, they need to update the interfaceSpecifications relationship so that it goes directly from the CapabilityVersion to the ServiceInterface class by following these steps:

1. Expand **<<BusinessModelPackage, ReadOnly>> GEP63**, and double-click **GEP63ArchitectureModel** to open it. Figure 5-7 shows this model.

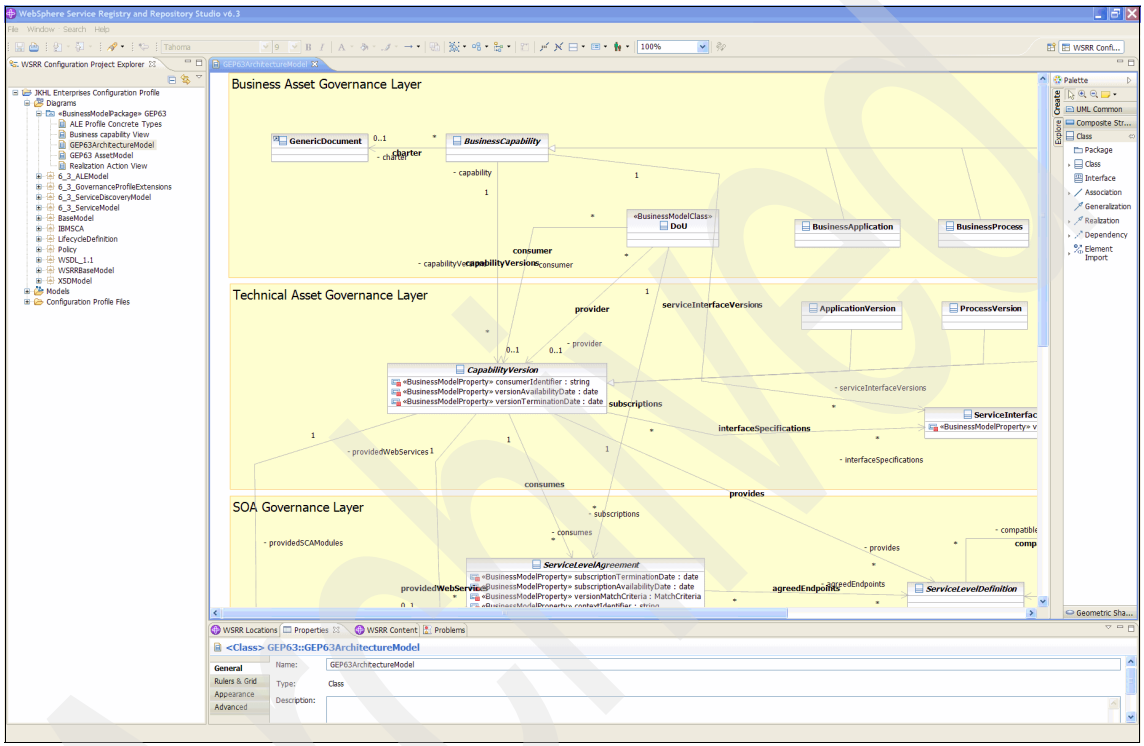


Figure 5-7 GEP63 Architectural Model

2. Click and drag the arrowhead for the interfaceSpecifications relationship from the ServiceInterfaceSpecification class to the ServiceInterface class (see Figure 5-8 and Figure 5-9).

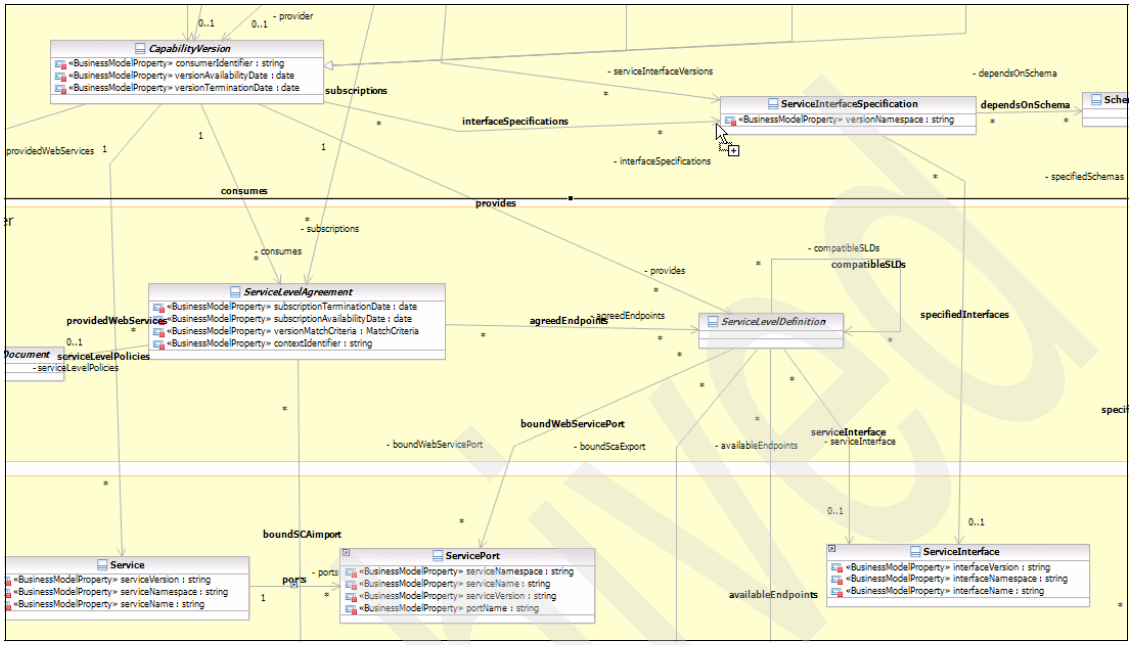


Figure 5-8 Selecting the interfaceSpecifications relationship

Archived

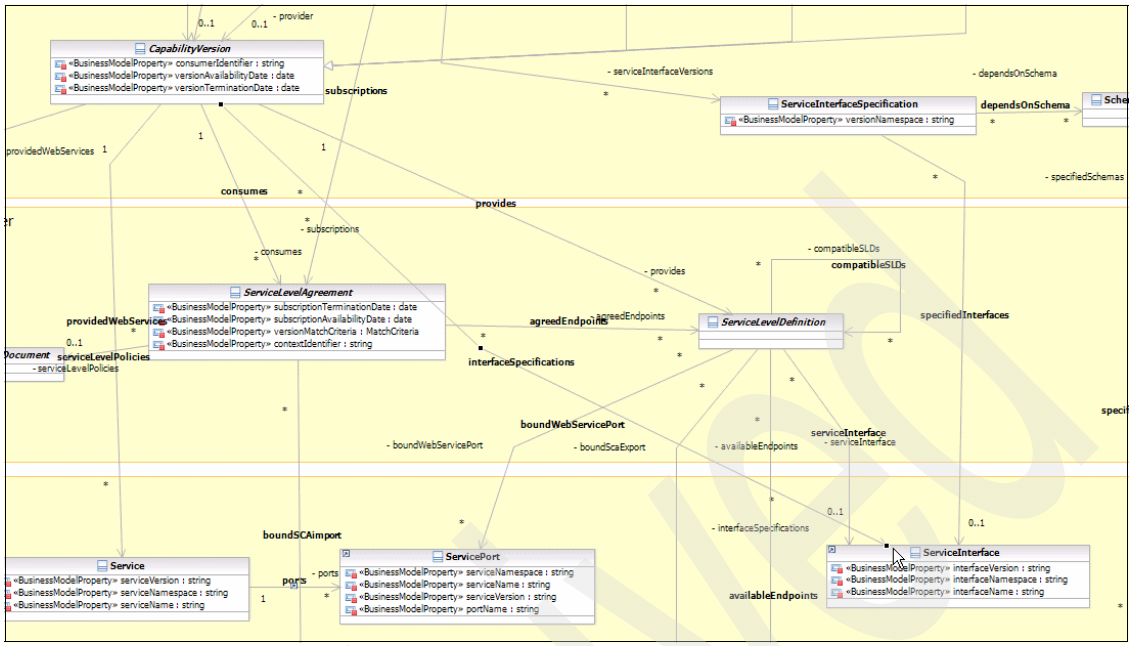


Figure 5-9 Moving the interfaceSpecifications relationship to the ServiceInterface class

Archived

3. Now that the relationship is updated, you can safely delete the ServiceInterfaceSpecification. Right-click the ServiceInterfaceSpecification class, and click **Delete from Model**, as shown in Figure 5-10.

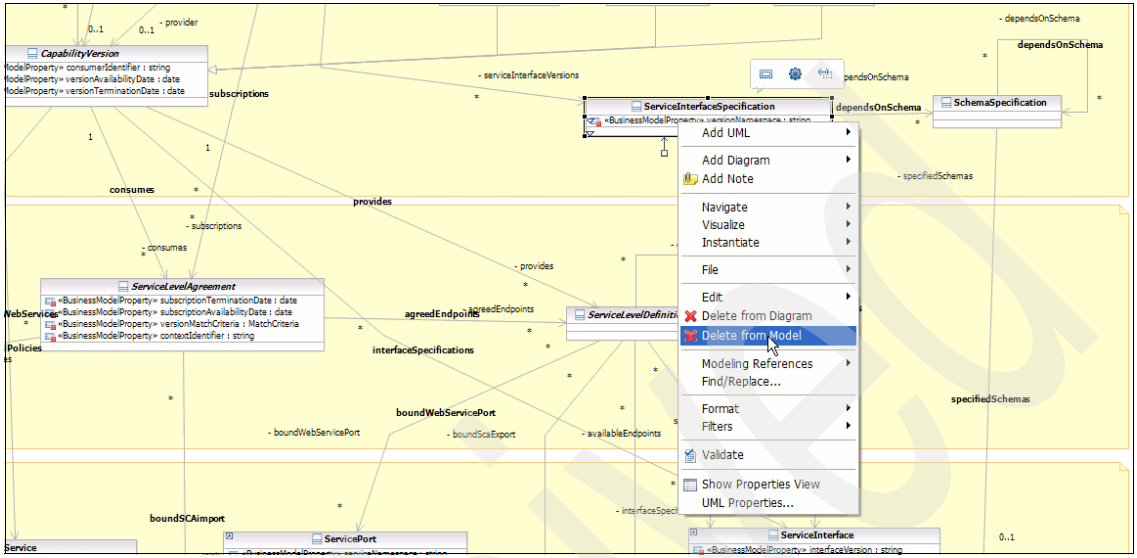


Figure 5-10 Deleting the ServiceInterfaceSpecification

Changing the label of the document of understanding to Subscription Request

JKHLE decided not to change the business model Uniform Resource Identifier (URI) of the document of understanding (DOU) and, instead, change just the label. Changing just the label gives them the flexibility to move to the governance enablement profile later and keeps the configuration changes to a minimum. (Changing the URI requires that you update the governance policies, user interface, and other configuration files that reference the business model classes URI.)

Archived

To change the label of the DOU, follow these steps:

1. Click the DOU class in the model. Then, on the Properties tab, click **General** as shown in Figure 5-11.

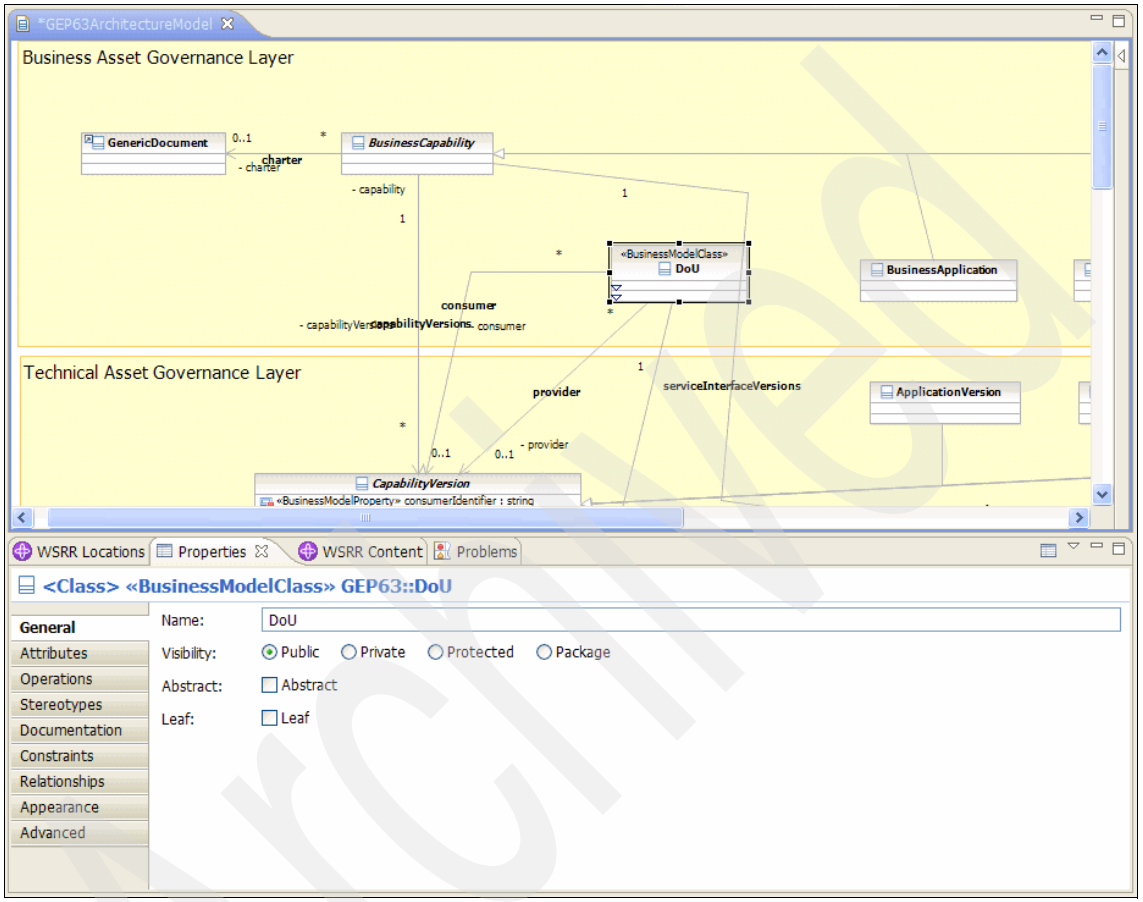


Figure 5-11 Selecting the DOU

Note that the name of the class is *DoU*. JKHLE does not want to change the actual class name of the DOU, because doing so requires them to update all references to this class in the configuration profile. To satisfy their requirements, JKHLE needs to change just the label, so that to the user, the class displays as a Subscription Request.

2. Go to the Documentation tab and enter the following description:

```
A Subscription Request represents an agreement of intent between the
consumer and the provider for a specific capability version.
```

3. On the Advanced tab (as shown in Figure 5-12), edit the Business Model Class comment field. Enter the following text:

   `A Subscription Request represents an agreement of intent between the consumer and the provider for a specific capability version.`

   In the label field, enter the following text:

   `Subscription Request`

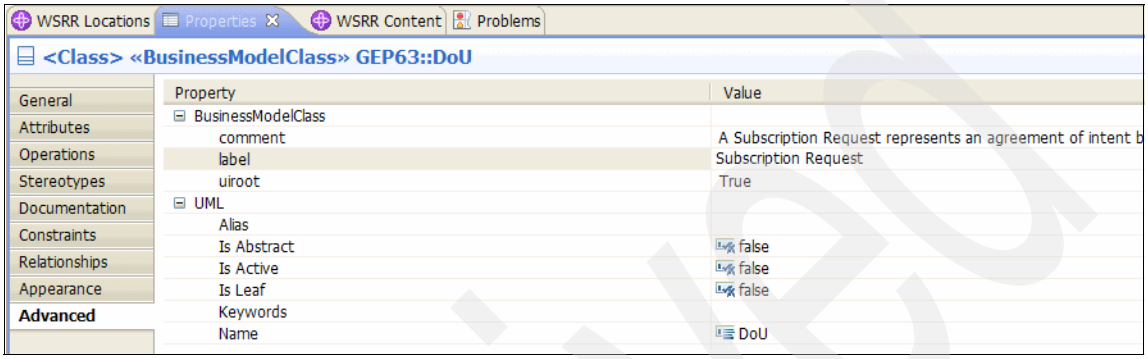


Figure 5-12 Modifying the properties on the DOU class

Archived

Updating the relationships

This section describes how JKHLE update the consumer, provider, and subscriptions relationships.

JKHLE updated the consumer relationship as follows:

1. Click the **consumer** relationship in the model, as shown in Figure 5-13.

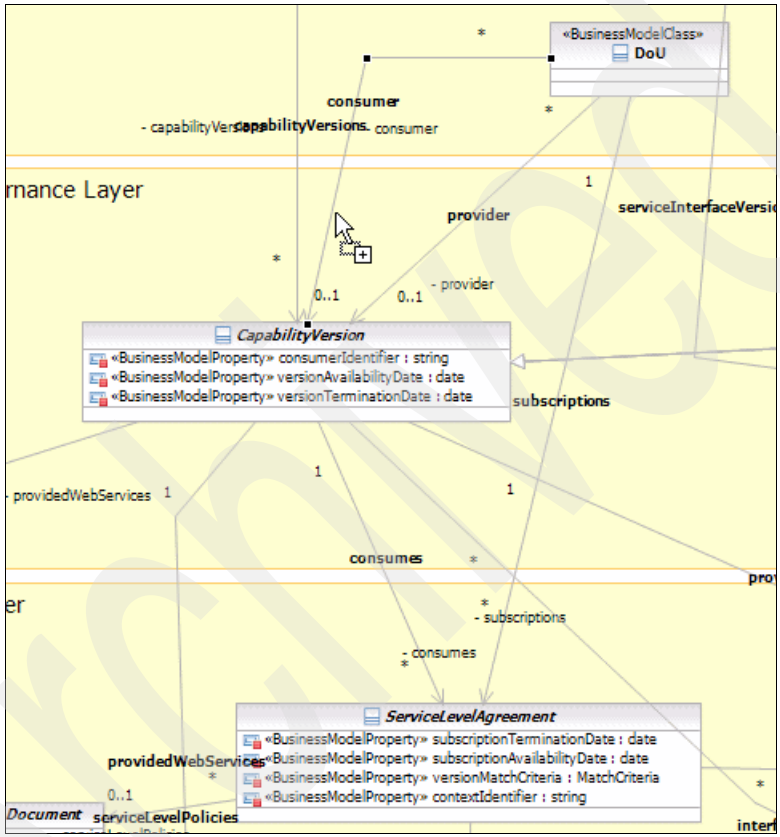


Figure 5-13 Selecting the consumer relationship

2. On the Properties tab, click **Advanced**. Edit the Business Model Class comment field. Enter the following text:

```
The consumer relationship identifies the capability version that
consumes the service.
```

Archived

JKHLE updated the provider relationship as follows:

1. Click the **provider** relationship in the model.
2. Then, in the Properties tab, click **Advanced**. Edit the Business Model Class comment field. Enter the following text:

   ```
   The provider relationship identifies the capability version that the
   consumer wishes to subscribe to.
   ```

JKHLE updated the subscriptions relationship as follows:

1. Click the **subscriptions** relationship in the model.
2. In the Properties tab, click **Advanced**. Edit the Business Model Class comment field. Enter the following text:

   ```
   The subscriptions relationship lists those Service Level Agreements
   between the consumer and the provider for the specified subscription
   request.
   ```

Configuring WebSphere Service Registry and Repository Studio to generate business the GEP63 business model

Now that JKHLE has updated the GEP63 business model, they need to configure WSRR Studio to generate the relevant business model:

1. Expand **JKHL Enterprises Configuration Profile** → **Diagrams** → **<<BusinessModelPackage**>> → **GEP63**.
2. On the Properties tab, click **Stereotypes**, and then toggle **generateOWL** to **true**.

Updating the Extended service level definition

JHKLE wants to extend the GPX63 business model system. With the extension classes, JHKLE can add specific metadata to the business model classes with minimal effort. The user interface configuration files in the governance enablement profile are already configured to use the extension classes. Thus, minimal user interface configuration is required to update this class.

> **Note:** The class name of the service level agreement and service level definition are the same in the GEP63 and GPX63 business model definitions. Just their namespace is different. The labels for the service level agreement (SLA) and service level definition (SLD) classes in the GPX63 are *Extended service level agreement* and *Extended service level definition*.

Archived

JKHLE updates the Extended SLD as follows:

1. In the WSRR Configuration Project Explorer, click **JKHL Enterprises Configuration Profile** → **Diagrams** → **6_3_GovernanceProfileExtensions**.

   Note: When you select the 6_3_GovernanceProfileExtensions node, its name is populated with the name from the model to *<<BusinessModelPackage>> GPX63*.

2. Expand **<<BusinessModelPackage>> GPX63**. Then, double-click **GPX63 Extension classes**, as shown in Figure 5-14.

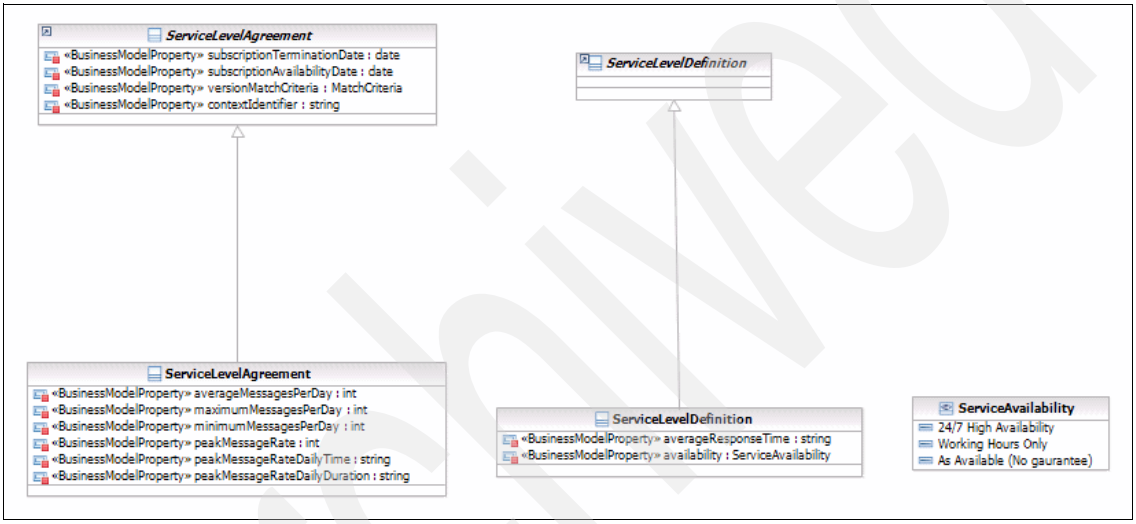


Figure 5-14 GPX63 Extension classes

Updating the service level definition

JKHLE wants to have the same properties on their service level definition (SLD) artifact as on their SLA. Table 5-4 lists these properties.

Table 5-4 Table of properties for the Extended SLA

ID	Type	Visibility	IsStatic	Multiplicity
`averageMessagesPerDay`	int	Private	False	0..1
`maximumMessagesPerDay`	int	Private	False	0..1
`minimumMessagesPerDay`	int	Private	False	0..1
`peakMessageRate`	int	Private	False	0..1

Archived

ID	Type	Visibility	IsStatic	Multiplicity
`peakMessageRateDailyTime`	string	Private	False	0..1
`peakMessageRateDailyDuration`	string	Private	False	0..1

To reuse an existing property in the same business model but in a different class, use the same property ID needs. The property type also has to be consistent with the original property.

JKHLE updates the SLD as follows:

1. Click **Service Level Definition** in the UML diagram.
2. On the Properties tab, click **Attributes**.
3. Right-click the grid, and click **Insert New Attribute**, as shown in Figure 5-15.

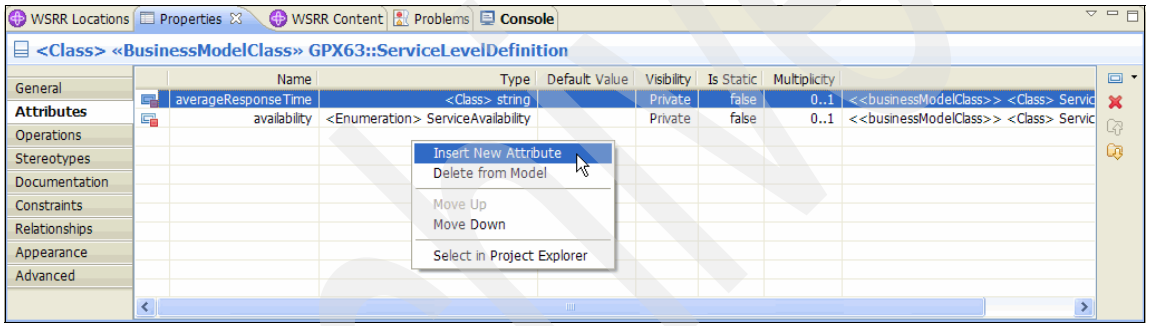


Figure 5-15 Inserting a new attribute

4. Click the name of the newly created attribute to edit it, for example **attribute1** as shown in Figure 5-16.

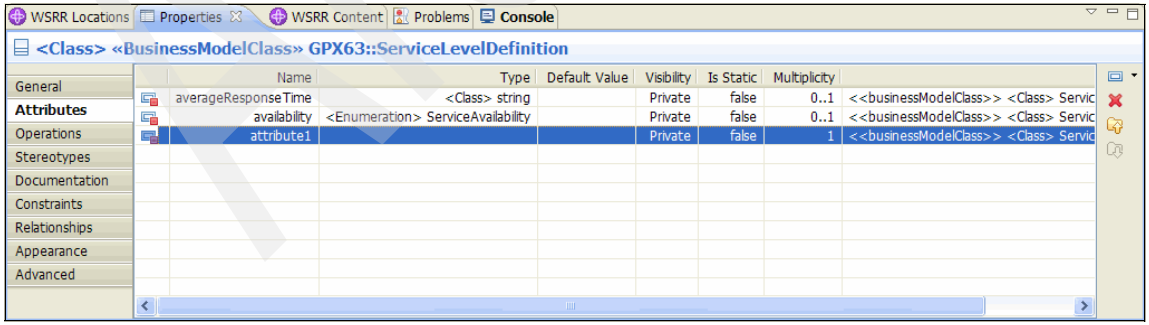


Figure 5-16 Selecting an attribute

5. Enter `averageMessagesPerDay` in the name column.
6. Click the Type field, and then click the ellipsis icon (...).
7. In the **Select Element for Type** window, enter `int` as the search text.
8. Click **`int - XSDDataTypes::XSDDataTypes::int`**, as shown in Figure 5-17. Then, click **OK**.

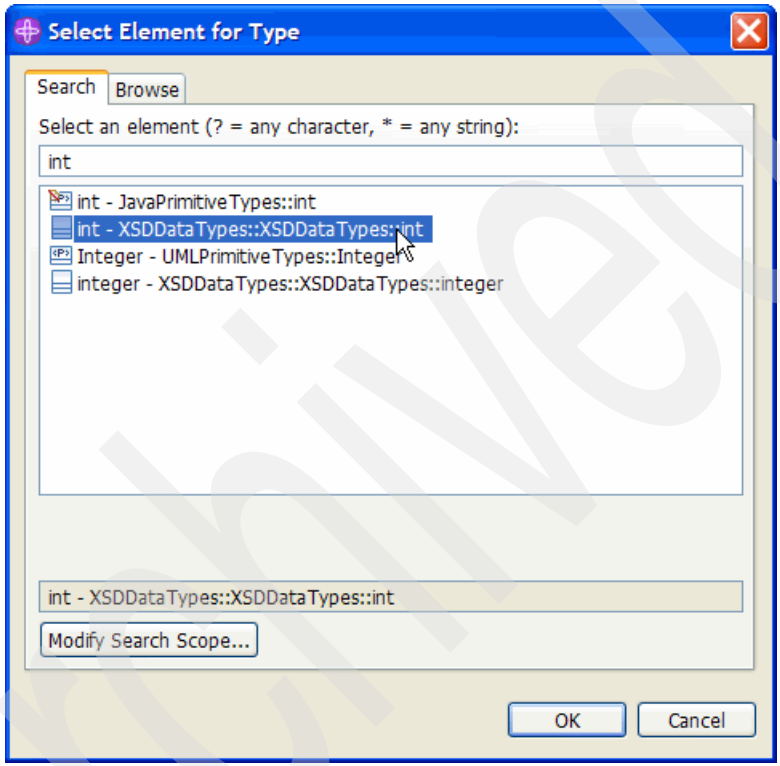


Figure 5-17 Selecting an element

Archived

9. Click the **Multiplicity** column for this attribute. Then, click the down arrow and select **0..1**, as shown in Figure 5-18.

	Name	Type	Default Value	Visibility	Is Static	Multiplicity	
	averageResponseTime	<Class> string		Private	false	0..1	<<businessModelClass>> <Class> Servi
	availability	<Enumeration> ServiceAvailability		Private	false	0..1	<<businessModelClass>> <Class> Servi
	averageMesagesPerDay	<Class> int		Private	false	0..1	<<businessModelClass>> <Class> Servi

Figure 5-18 averageMessagesPerDay attributed added to the Extended SLD class

10. Click **averageMessagesPerDay** in the SLD class on the UML diagram, as shown in Figure 5-19.

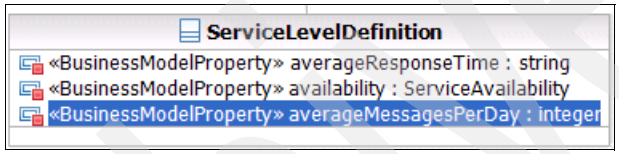


Figure 5-19 averageMessagesPerDay attribute on the SLD

11. On the Properties tab, click **Advanced**. Then, edit the ID field. Enter the following information (as shown in Figure 5-20):

```
averageMessagesPerDay
```

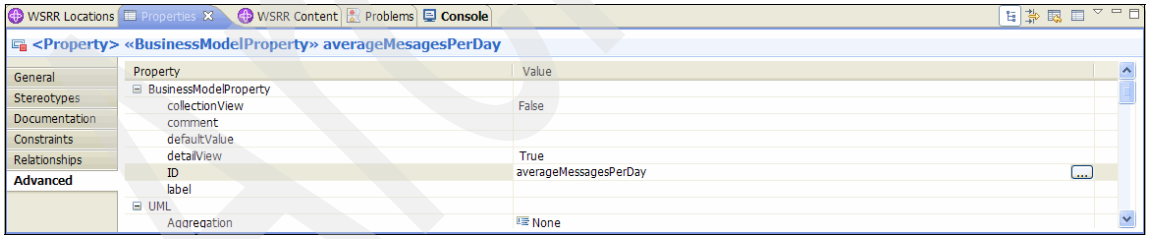


Figure 5-20 Adding an ID to the averageMessagesPerDay attribute

12. Repeat this procedure for the other remaining attributes as shown in Table 5-4. When you select the type, always use the XSD data type, for example as follows:

```
int - XSDDataTypes::XSDDataTypes::int and string -
XSDDataTypes::XSDDataTypes::string
```

Archived

Figure 5-21 shows the completed attributes table.

Name	Type	Default Value	Visibility	Is Static	Multiplicity	Owner
averageResponseTime	<Class> string		Private	false	0..1	<<businessModelClass>> <Class> ServiceLevelDefinition
availability	<Enumeration> ServiceAvailability		Private	false	0..1	<<businessModelClass>> <Class> ServiceLevelDefinition
averageMesagesPerDay	<Class> int		Private	false	0..1	<<businessModelClass>> <Class> ServiceLevelDefinition
maximumMessagesPerDay	<Class> int		Private	false	0..1	<<businessModelClass>> <Class> ServiceLevelDefinition
minimumMessagesPerDay	<Class> int		Private	false	0..1	<<businessModelClass>> <Class> ServiceLevelDefinition
peakMessageRate	<Class> int		Private	false	0..1	<<businessModelClass>> <Class> ServiceLevelDefinition
peakMessageRateDailyTime	<Class> string		Private	false	0..1	<<businessModelClass>> <Class> ServiceLevelDefinition
peakMessageRateDailyDuration	<Class> string		Private	false	0..1	<<businessModelClass>> <Class> ServiceLevelDefinition

Figure 5-21 Table of attributes on the customized SLD

Figure 5-22 shows the final SLD class.

Figure 5-22 Updated SLD class

5.4.3 Applying JKHLE's customizations to the life cycles

JKHLE wants to streamline the governance process and reduce the number of approvals that are required in the governance enablement profile. They want to reduce the complexity of the following composite life cycles:

- Asset life cycle
- SLD life cycle
- SOA life cycle

This section explains how to apply the customizations to the life cycles.

Archived

Configuring WebSphere Service Registry and Repository Studio to generate OWL and SACL files

JKHLE needs to configure WSRR Studio so that it generates both the ontology file and the corresponding SACL file from the UML life cycle models as follows:

1. Click **JKHL Enterprises Configuration Project** → **Diagrams** → **LifecycleDefinition**.

> **Note:** If you have already expanded the LifecycleDefinition node, it will be called *<<LifecycleModel>> Lifecycle Definition*.

2. Select the **<<LifecycleModel>> Lifecycle Definition** node.
3. On the Properties tab, click **Stereotypes**. Then, toggle **generateOWL** to **true**, and Toggle **generateSACL** to **true**, as shown in Figure 5-23.

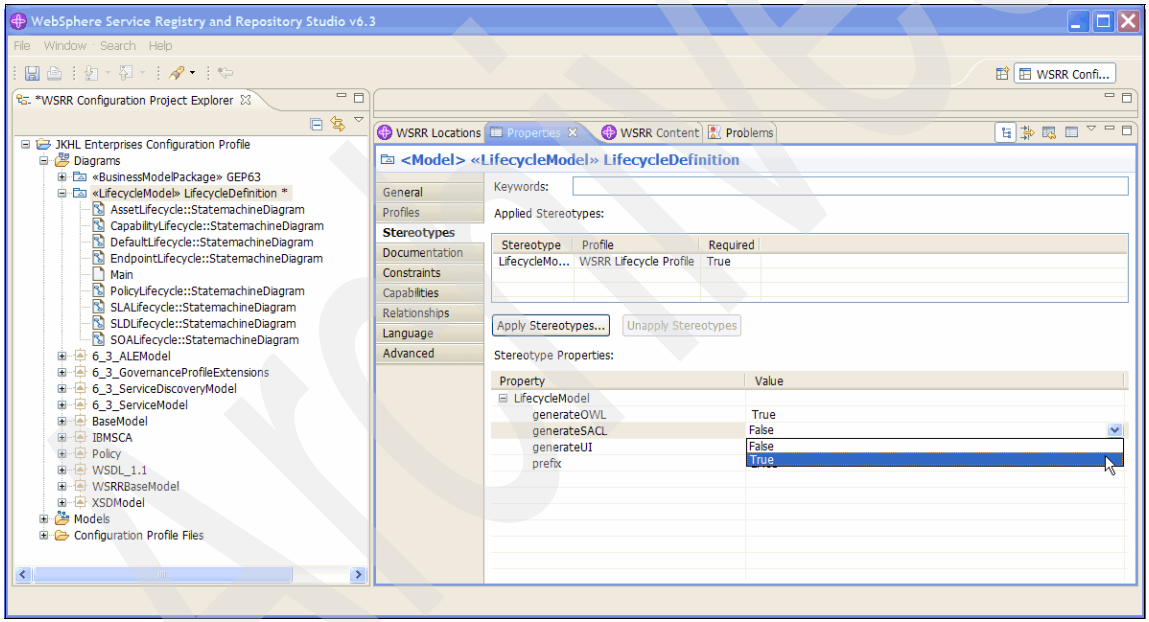


Figure 5-23 Setting the generateOWL and generateSACL values to true

Updating the asset life cycle

JKHLE wants to make the asset life cycle simpler by removing the following states:

- AssetScoped
- AssetSpecificationReview
- AssetSpecified
- AssetReview

JKHLE updates the asset life cycle as follows:

1. Expand **JKHL Enterprises Configuration Project** → **Diagrams** → **<<LifecycleModel>> Lifecycle Definition**.
2. Double-click **AssetLifecycle::StatemachineDiagram**.

Archived

3. Click and drag the arrowhead for the ApproveAsset transition from the AssetReview state to the AssetScopeReview state (see Figure 5-24 and Figure 5-25).

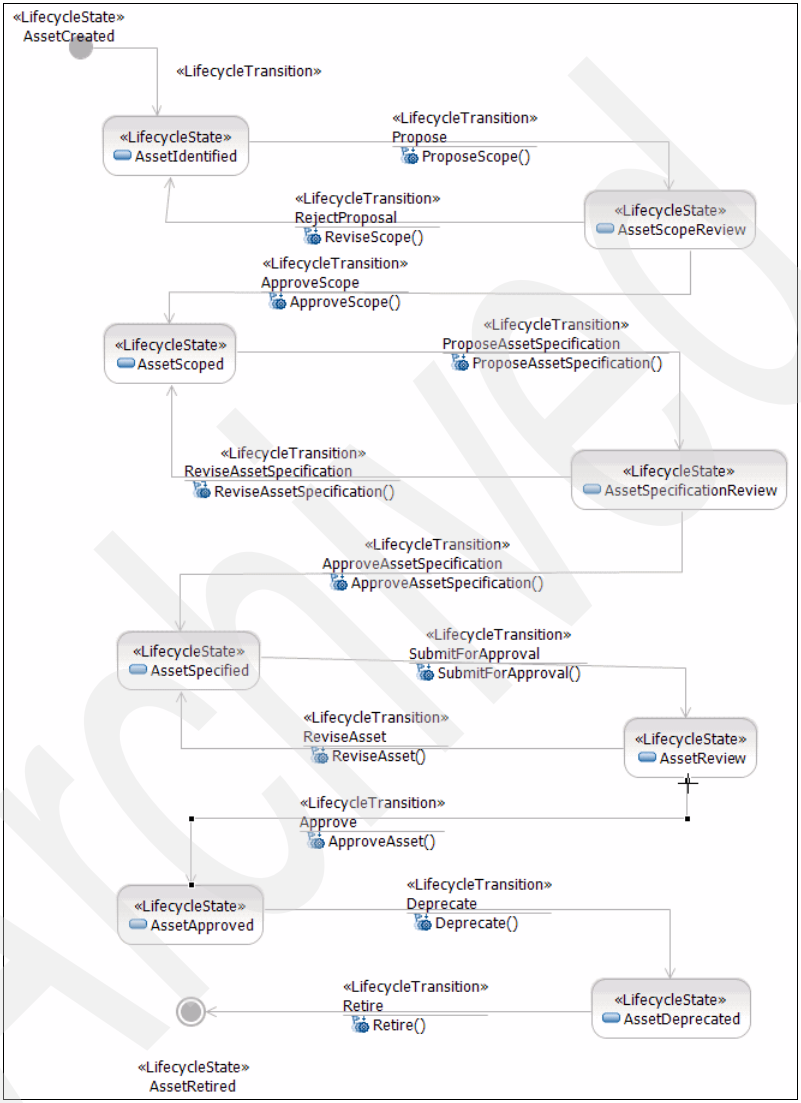


Figure 5-24 Selecting the ApproveAsset transition

Archived

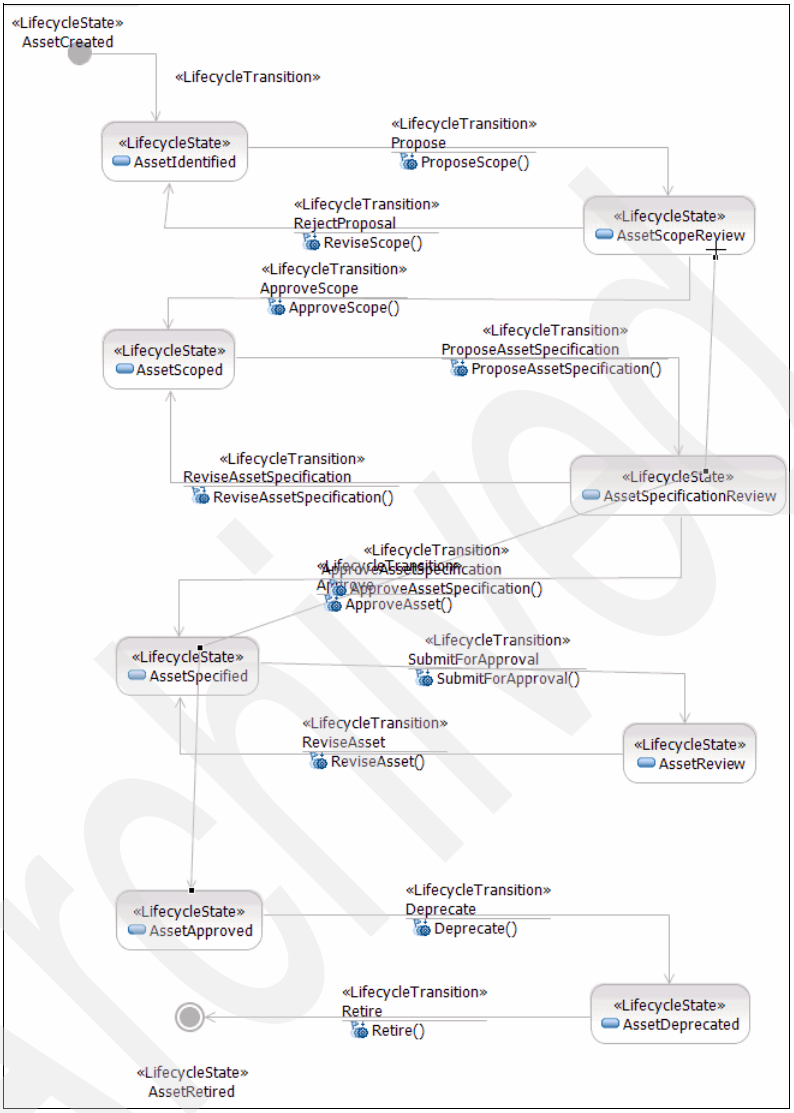


Figure 5-25 Moving the ApproveAsset transition to the AssetScopeReview state

Archived

4. Right-click the **AssetScoped** state, and click **Delete from Model**, as shown in Figure 5-26.

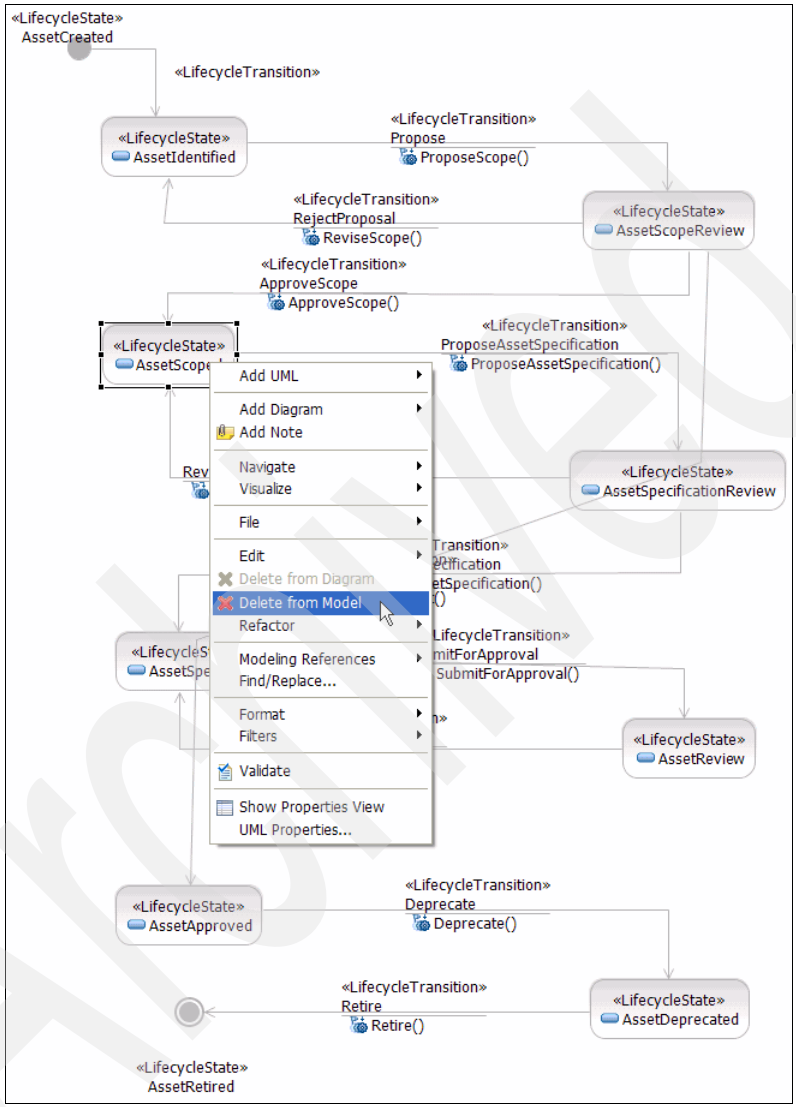


Figure 5-26 Deleting the AssetScoped state from the asset life cycle

5. Right-click the **AssetSpecificationReview** state, and click **Delete from Model**.
6. Right-click the **AssetSpecified** state, and click **Delete from Model**.
7. Right click the **AssetReview** state, and click **Delete from Model**.

8. Finally, click **File** → **Save**.

The model should now look as shown in Figure 5-27.

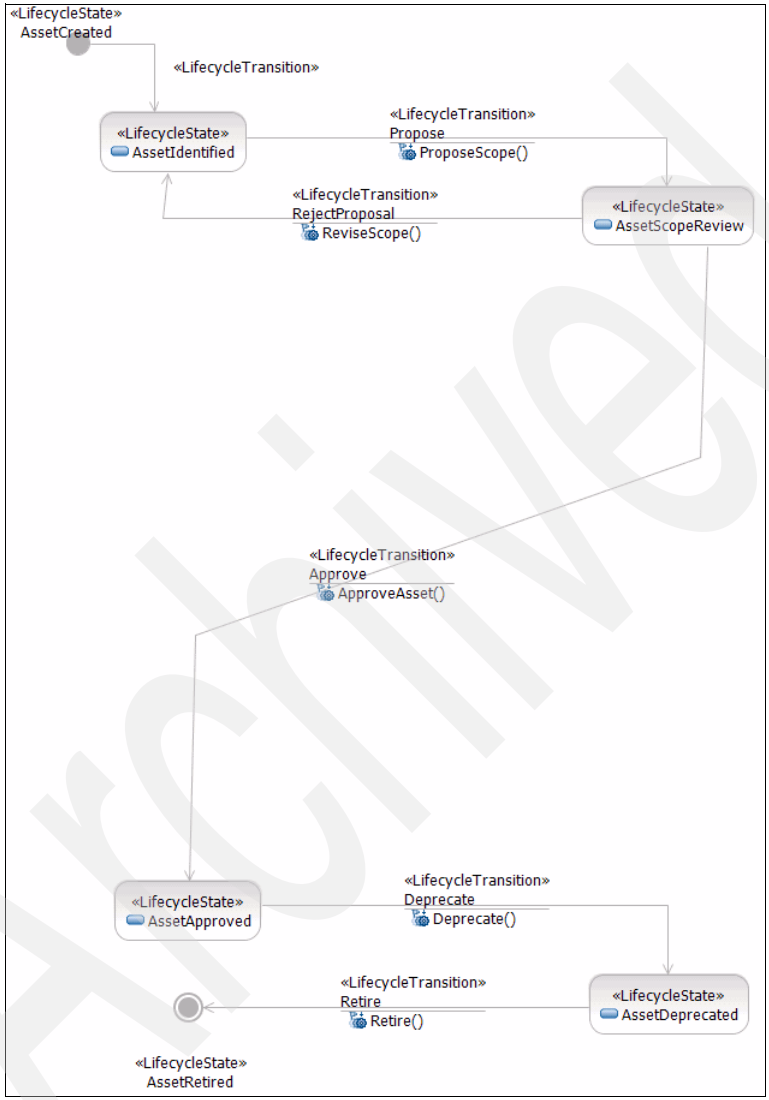


Figure 5-27 JKHLE's completed asset life cycle

Archived

Updating the SLD life cycle

JKHLE wants to make the SLD life cycle simpler by removing the following states:

- SLDScoped
- SLDReview

JKHLE updates the SLD life cycle as follows:

1. Expand **JKHL Enterprises Configuration Project** → **Diagrams** → **<<LifecycleModel>> Lifecycle Definition**.
2. Double-click **SLDLifecycle::StatemachineDiagram**.
3. Click and drag the arrowhead of the ApproveScope relationship from the SLDScopeReview state to the SLDSubscribable state.
4. Right-click the **SLDScoped** state, and click **Delete from Model**.
5. Right-click the **SLDReview** state, and click **Delete from Model**.
6. Finally, click **File** → **Save**.

Archived

Figure 5-28 shows the finished SLD life cycle.

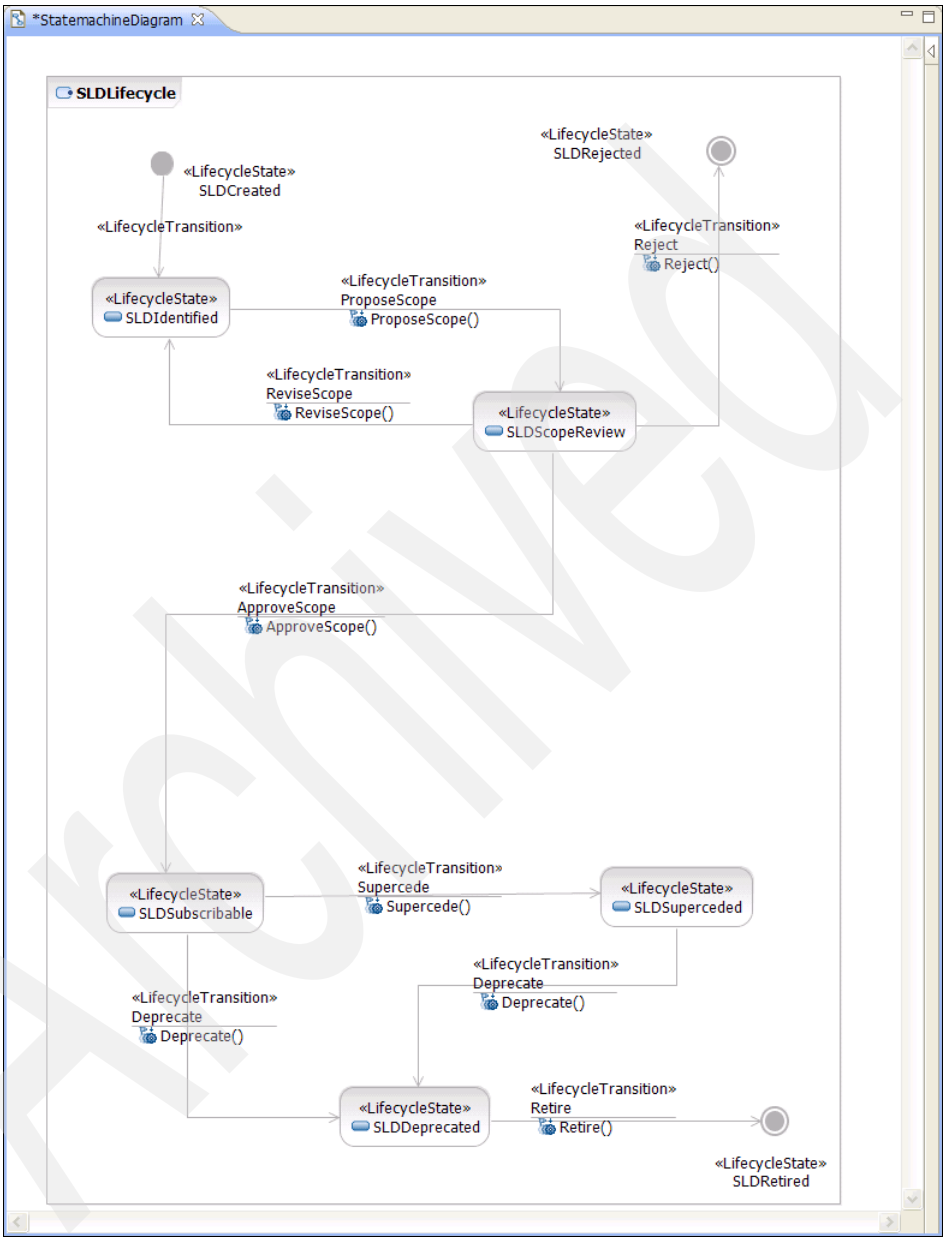


Figure 5-28 JKHLE's completed SLD life cycle

Archived

Updating the SOA governance life cycle

JKHLE updates the SOA life cycle by removing the following states:

- PlanReview
- Planned
- SpecificationReview
- RealizationReview
- Realized
- StagingReview
- CertificationReview
- OperationalReview

Then, JKHLE adds a state of **Superceded** between the Operational and Deprecated state.

To remove the states:

1. Expand **JKHL Enterprises Configuration Project** → **Diagrams** → **<<LifecycleModel>> Lifecycle Definition**.
2. Double-click **SOALifecycle::StatemachineDiagram**.
3. Click and drag the ApproveSpecification transition from the SpecificationReview state to the Scoped state.
4. Click and drag the ReviseSpecification transition from the SpecificationReview state to the Specified state.
5. Click and drag the ReviseSpecification transition from the Planned state to the Scoped state.
6. Right-click the **SpecificationReview** state, and click **Delete from Model**.
7. Right-click the **Planned** state, and click **Delete from Model**.
8. Right-click the **PlanReview** state, and click **Delete from Model**.

Archived

Figure 5-29 shows the life cycle at this stage.

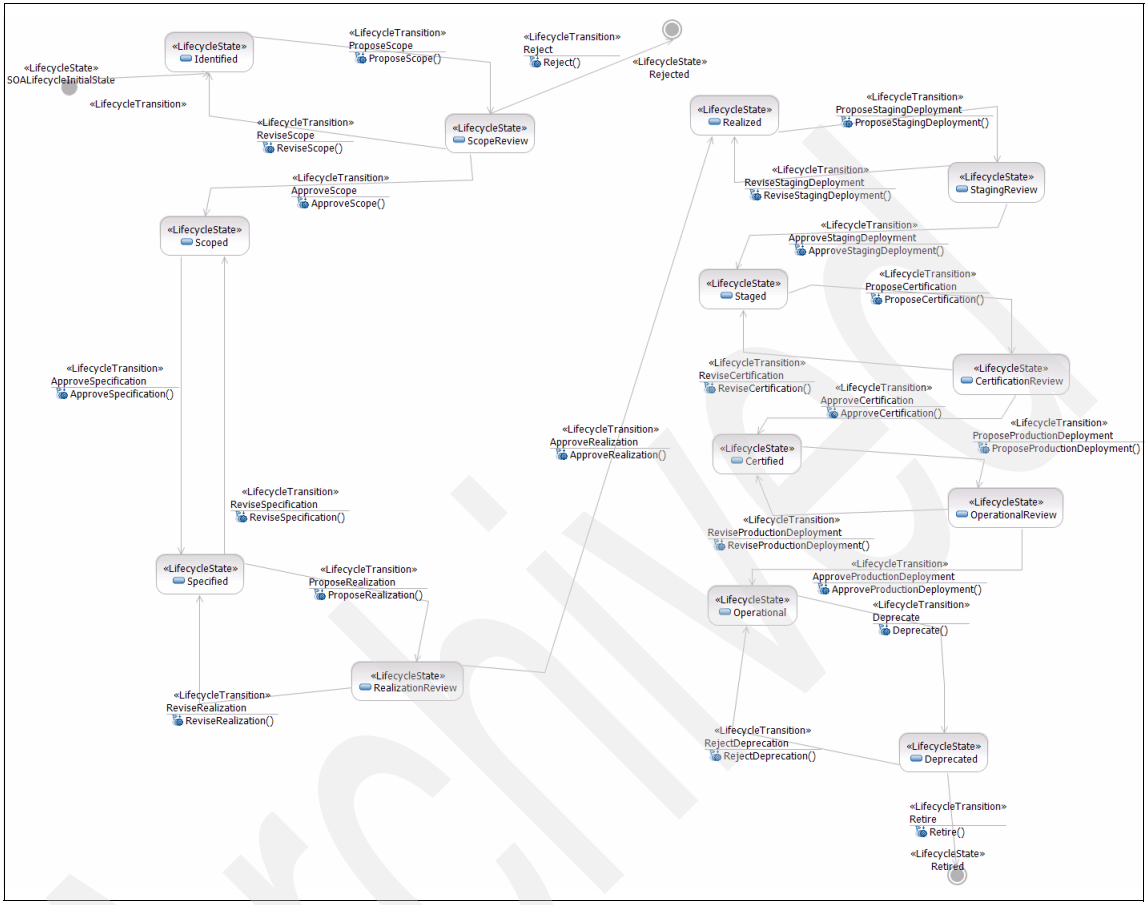


Figure 5-29 Partially updated SOA life cycle

9. Click and drag the ApproveStagingDeployment transition from the StagingReview state to the Specified state.
10. Right-click the **RealizationReview** state, and click **Delete from Model**.
11. Right-click the **Realized** state, and click **Delete from Model**.
12. Right-click the **StagingReview** state, and click **Delete from Model**.
13. Click and drag the ApproveCertification transition from the CertificationReview state to the Staged state.
14. Right-click the **CertificationReview** state, and click **Delete from Model**.
15. Click and drag the ApproveProductionDeployment transition from the OperationalReview state to the Certified state.

16. Right-click the **OperationalReview** state, and click **Delete from Model**.

Figure 5-30 shows the life cycle at this stage.

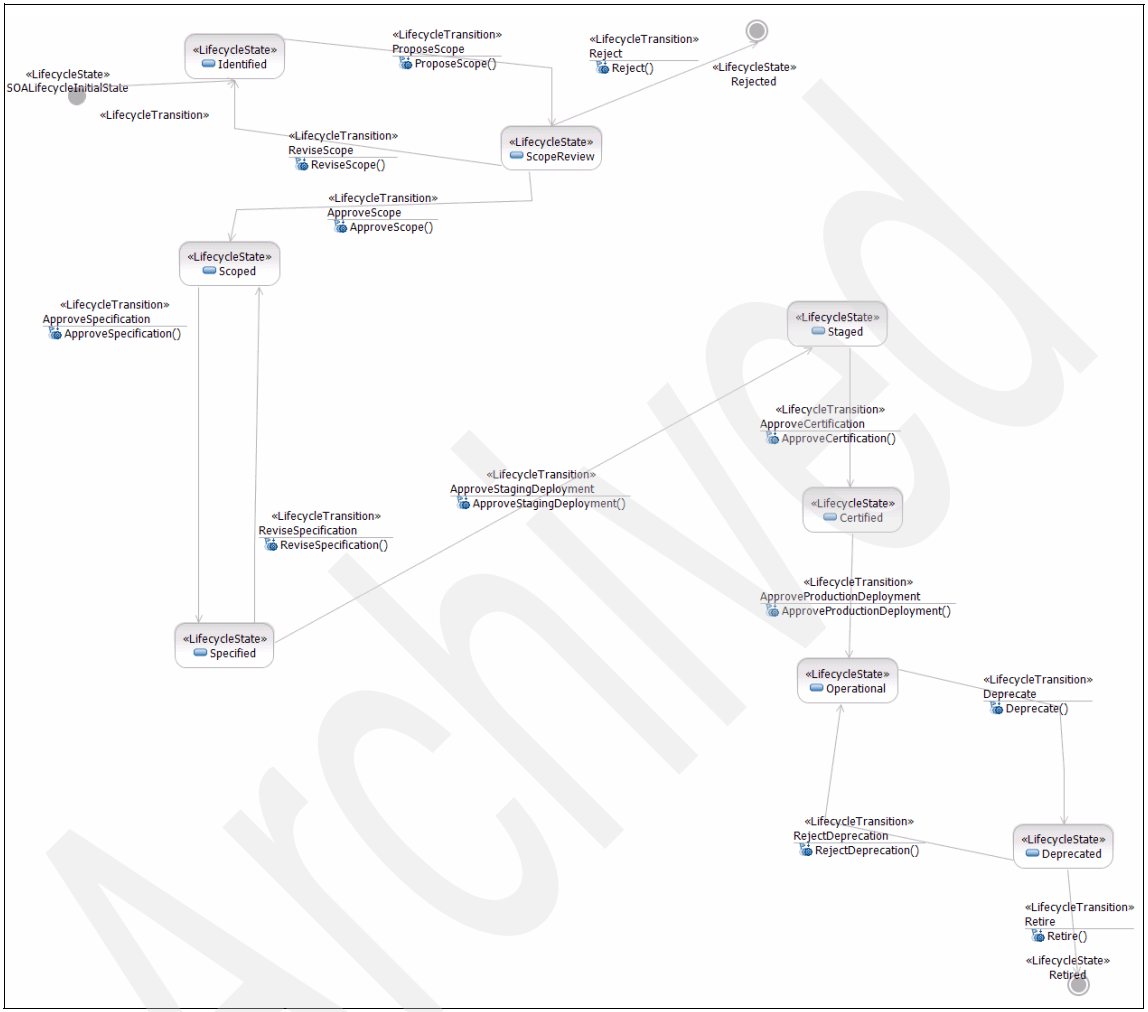


Figure 5-30 SOA life cycle with states removed

Archived

17. Now, to add a superceded state, click Show Palette (if it is hidden), as shown in Figure 5-31.

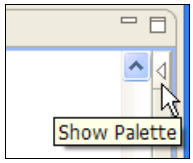


Figure 5-31 Show Palette icon

18. Click **State** under **State Machine** in the Palette frame (as shown in Figure 5-32).

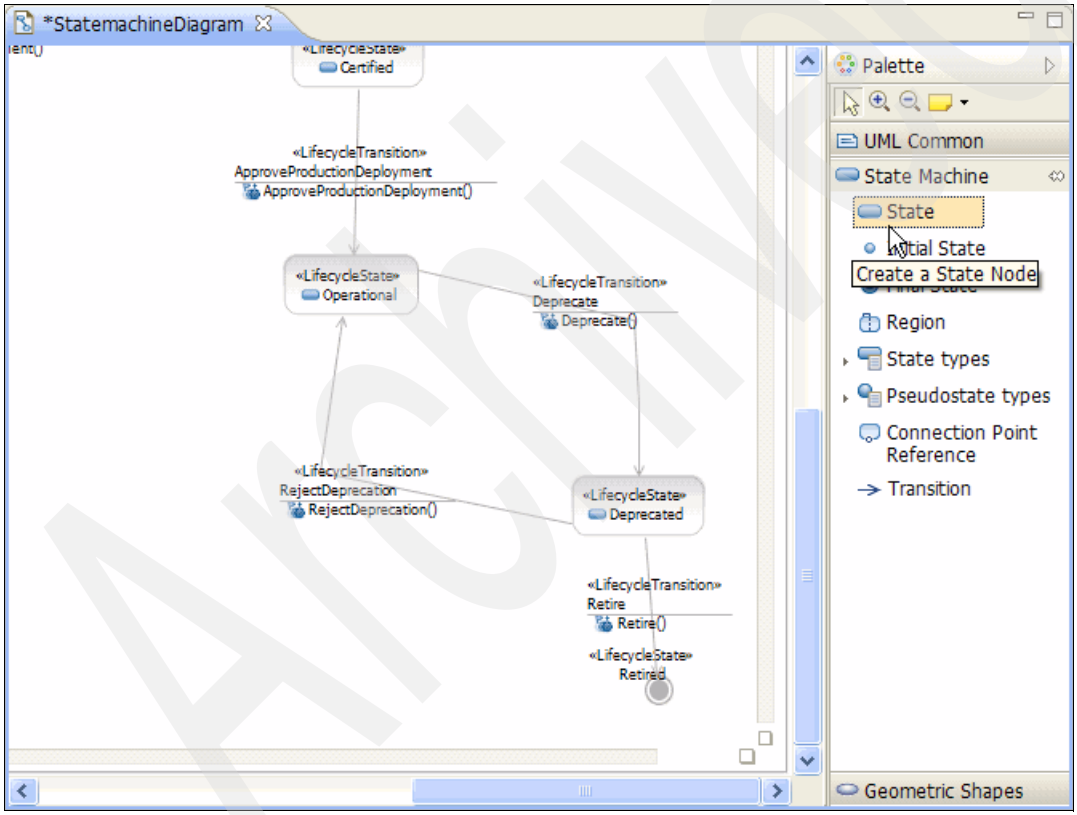


Figure 5-32 Selecting a state object from the palette

Archived

19. Click the UML model, as shown in Figure 5-33.

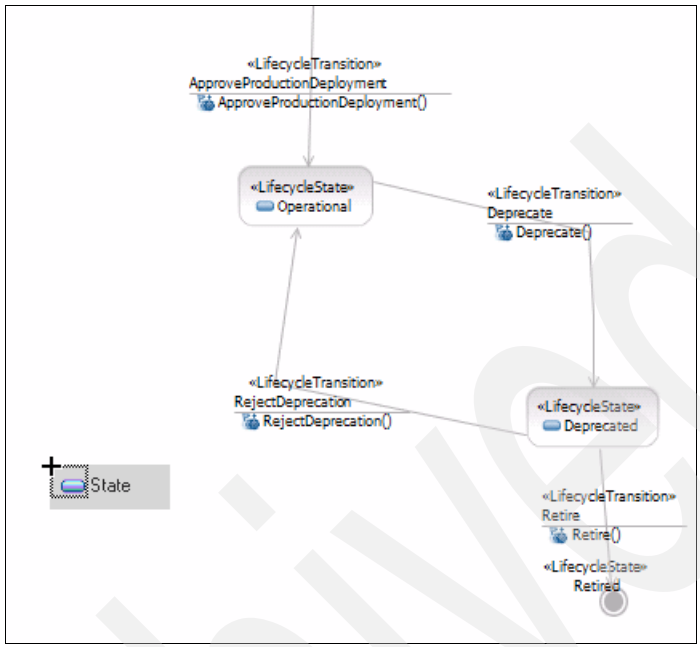


Figure 5-33 Dropping a state onto the UML diagram

20. On the Properties tab, click **General**. Then, edit the Name field. Enter the following text (as shown in Figure 5-34):

```
Superceded
```

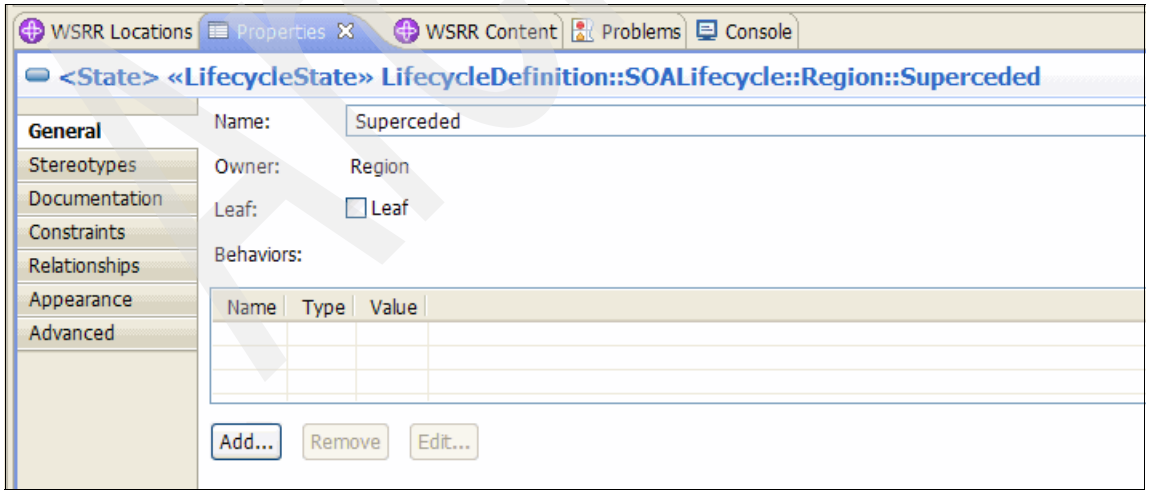


Figure 5-34 Editing the Superceded state

21. On the Advanced tab, edit the LifecycleState comment field. Enter the following text:

    `The superceded state indicates that the capability is deployed in production and in use, however is no longer subscribable since a newer version of the capability is available.`

    Edit the LifecycleState ID field and enter the following text:

    `Superceded`

    Edit the LifecycleState label field and enter the following text:

    `Superceded`

22. Next, to add a supercede transition, click **Transition** under **State Machine** in the Palette frame, as shown in Figure 5-35.

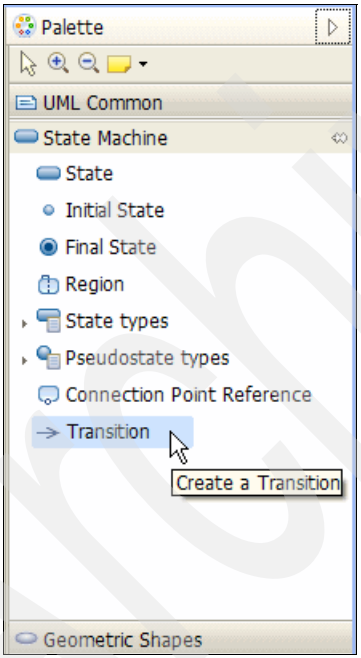


Figure 5-35 Selecting a transition object from the palette

Archived

23. Click the Operational state in the UML Model and drag the transition to the Superceded state, as shown in Figure 5-36.

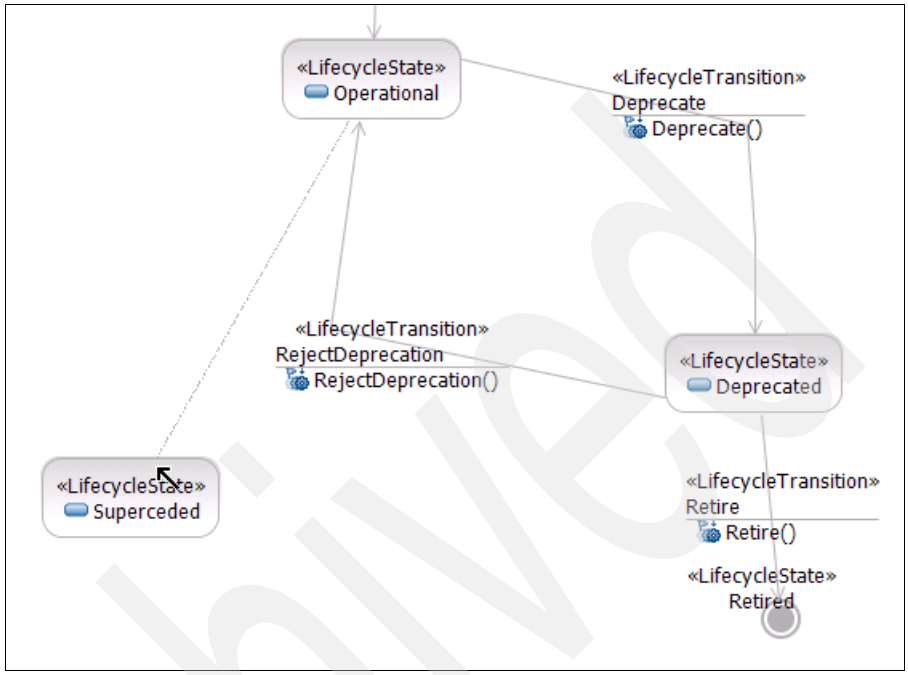


Figure 5-36 Adding a transition to the UML diagram

24. Label the transition as Supercede, by clicking the new transition to select it and then clicking the transition again to open the edit field. Then, enter `Supercede`.

Archived

25. Right-click the superceded transition, and then click **Add UML** → **Trigger** → **Signal Event**, as shown in Figure 5-37.

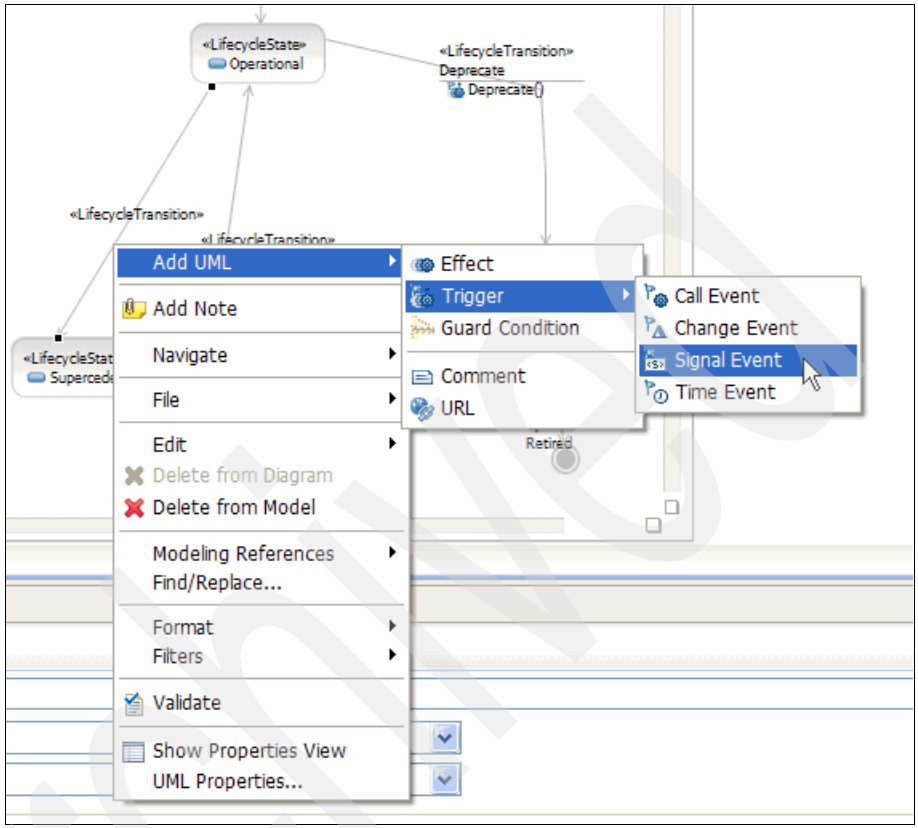


Figure 5-37 Adding a signal event to the transition

26. Click **Select Existing Element**, as shown in Figure 5-38.

Figure 5-38 Re-using an exiting element

27. Click **Browse**. Then, click **JKHL Enterprises Configuration Profile** → **Models** → **LifecycleModel** → **Events**. Click **<<LifecycleSignal>> Supercede**.

28. Next, to add a Deprecate2 transition, click **Transition** under **State Machine** in the palette.

Archived

29. Click the Superceded state in the UML Model and drag the transition to the Deprecated state.
30. Label the transition as `Deprecate2`.

> **Note:** Each transition name in a life cycle must be unique. In our testing, we already had a transition labelled *Deprecate*. Thus, we labeled this transition *Deprecate2*.

31. Right-click the **Deprecate2** transition, and then click **Add UML** → **Trigger** → **Signal Event**. Click **Select Existing Element**.
32. Click **Browse**. Then, click **JKHL Enterprises Configuration Profile** → **Models** → **LifecycleModel** → **Events**. Click **<<LifecycleSignal>> Deprecate**.
33. Finally, click **File** → **Save**.

Figure 5-39 shows the altered life cycle.

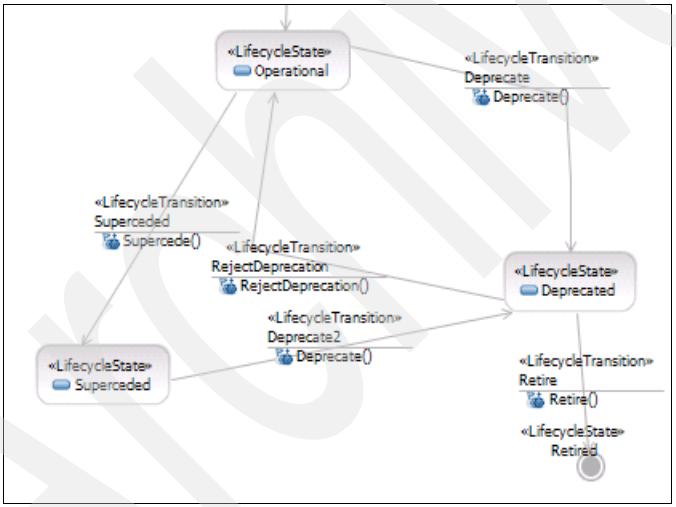


Figure 5-39 Part of JKHLE's SOA life cycle with a superceded state added

Archived

Figure 5-40 shows the finished SOA governance life cycle for JKHLE.

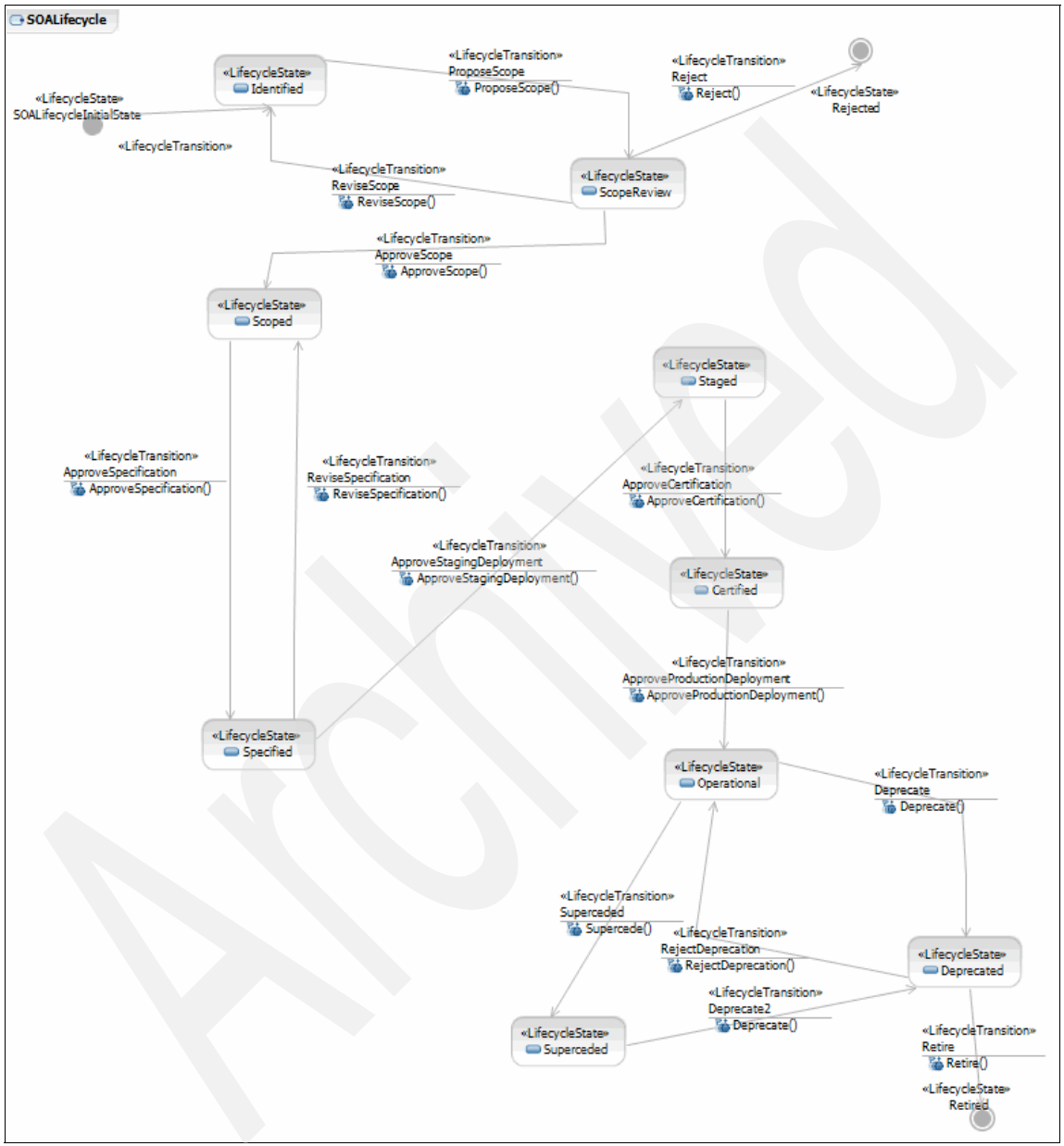


Figure 5-40 JKHLE's completed SOA life cycle

5.5 Applying JKHLE's customizations to the governance profile taxonomy

This section describes how to apply the customizations from JKHLE to the governance profile taxonomy to add a pre-production environment as follows:

1. Click **JKHL Enterprises Configuration Project** → **Diagrams** then double-click **GovernanceProfileTaxonomy**, as shown in Figure 5-41.

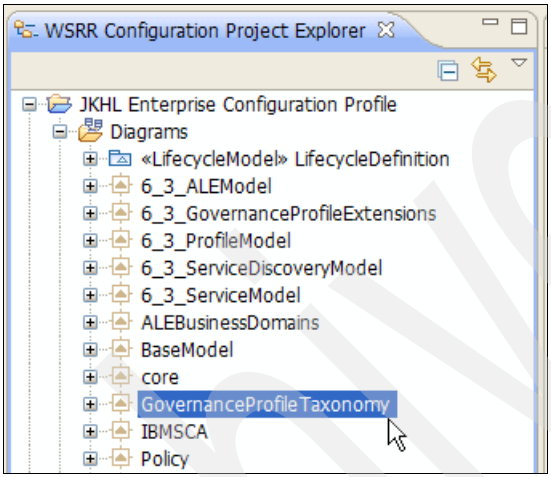


Figure 5-41 Selecting the governance profile taxonomy

2. Expand **<<ClassificationSystemPackage>> GovernanceProfileTaxonomy**.
3. Double-click **Class Diagram**, shown in Figure 5-42, to display a UML representation of the taxonomy.

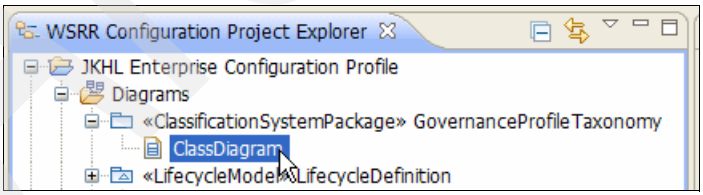


Figure 5-42 Selecting the governance profile taxonomy class diagram

Archived

4. Click **Class** under **Class** in the Palette frame, as shown in Figure 5-43.

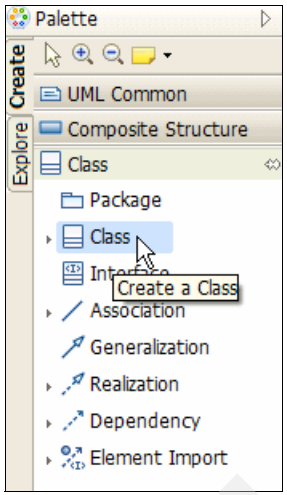


Figure 5-43 Selecting a class object from the palette

5. Click the UML diagram as shown in Figure 5-44.

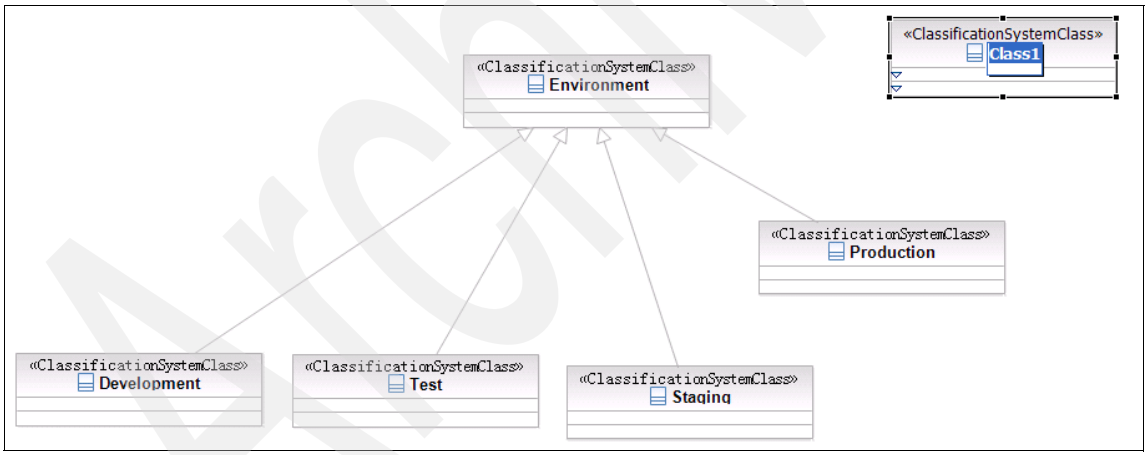


Figure 5-44 Adding a new class to the governance profile taxonomy

Archived

6. On the Properties tab, click **General**. Then, enter `Pre-Production` in the Name field.
7. On the Advanced tab, enter the following text in the Classification Class comment field:

   `The Pre-production environment`
8. In the Classification Class label, enter the following text:

   `Pre-Production`
9. Click **Generalization** under **Class** in the palette.
10. Click the Pre-Production class in the UML Model, and drag the generalization to the Environment class, as shown in Figure 5-45.

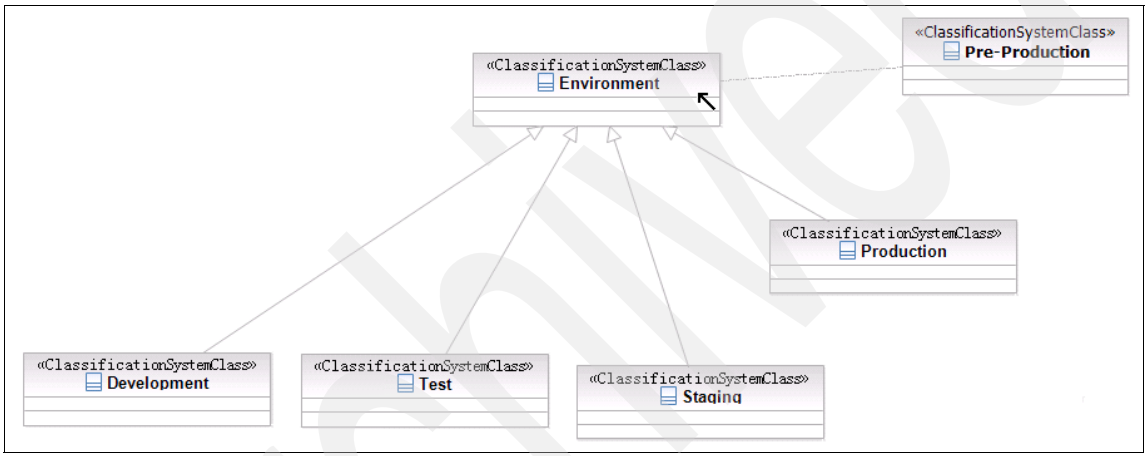


Figure 5-45 Adding a generalization from the Pre-Production to the Environment class

Archived

Configuring WSRR Studio to generate an owl file for the governance profile taxonomy

To configure WSRR Studio to generate an owl file, JKHLE completes the following steps:

1. Click **JKHL Enterprises Configuration Project** → **Diagrams** → **<<ClassificationSystemPackage>> GovernanceProfileTaxonomy**.
2. On the Properties tab, click **Stereotypes**. Then, toggle **generateOWL** to **true**, and Toggle **generateSACL** to **true**, as shown in Figure 5-46.

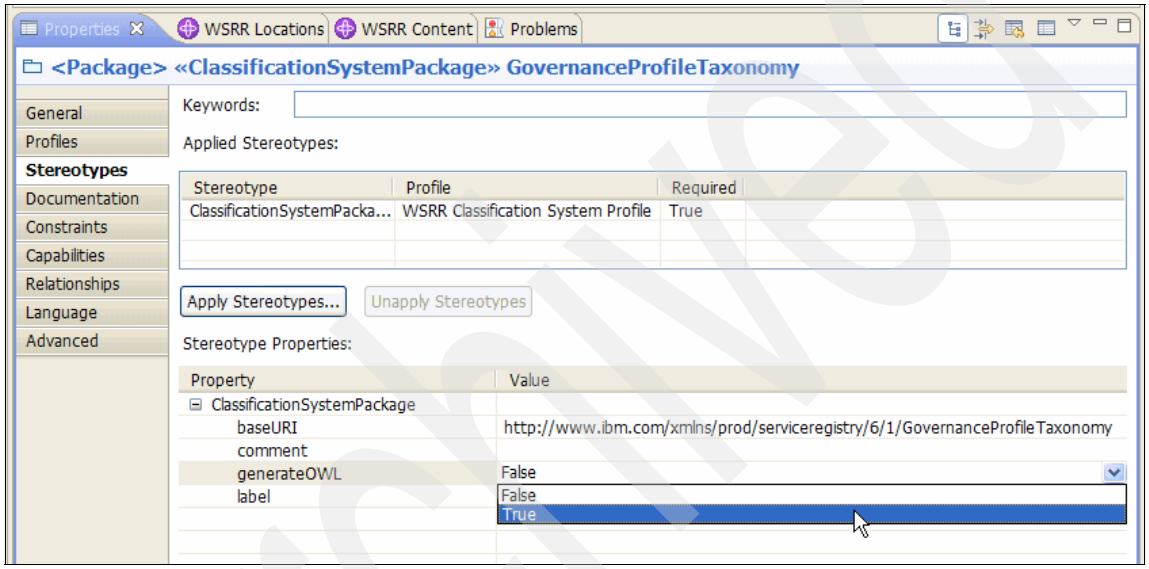


Figure 5-46 Setting the generateOWL value to true

Archived

5.6 Generating a profile

Now that JKHLE has completed all the necessary updates, they can generate a WSRR configuration profile from their customized UML models. To generate a profile, right-click the JKHL Enterprises Configuration Profile, and then click **WSRR** → **Generate All Artifacts** as shown in Figure 5-47.

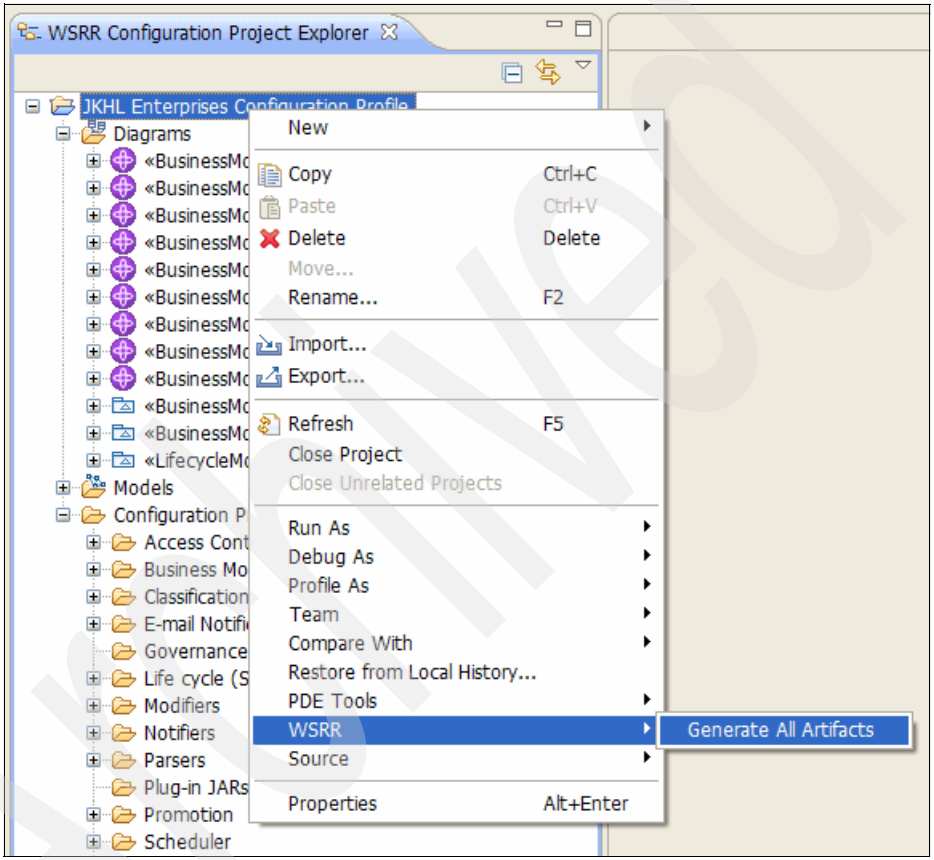


Figure 5-47 Generating the profile

WSRR Studio includes a very basic change control mechanism. If you generate the configuration files and you have made no manual changes to the files, the generation process overrides the appropriate configuration file or files. However, if you have made a manual change to a configuration file (for example, if you edit the XML file), WSRR Studio does not overwrite the configuration file automatically but creates a new configuration file of the same name with the term *(Generated)* applied to the name, for example *Governance Enablement Profile*

Business Model (Generated). This naming convention allows you to merge any manual changes to the generated configuration files.

5.7 Transferring .emx models between clients

You can export and share the UML models that you create or edit in WSRR Studio with other WSRR Studio users. Each model is stored as an `.emx` file on the file system of the machine that is running WSRR Studio.

5.7.1 Exporting .emx files

To export an .emx file:

1. Right-click the JKHL Enterprises Configuration Profile, and click **Export**.
2. In the **Export** dialog box:
   a. Click **General** → **File System**.
   b. Click **Next**.
   c. Expand the **JKHL Enterprises Configuration Project** node.
   d. Clear the Configuration Profile Files and the UML Models nodes.
   e. Click **UML Models** to display its contents. Then, select only the following `.emx` files:
      - `6_3_GovernanceProfileExtensions.emx`
      - `6_3_ProfileModel.emx`
      - `LifecycleDefinition.emx`

      Note: These `.emx` files are available for download as additional material with this book. See Appendix B, “Additional material” on page 621 for more information.

   f. Click **Browse**, and navigate to **Desktop** → **My Computer** → **Local Disk (C:)** → **temp** or an alternate target directory.
   g. Confirm that the Configuration Project name is **JKHL Enterprises Configuration Project**.
   h. Click **Finish**.

Archived

5.7.2 Importing .emx files

To import `.emx` files, you first need to create a new WSRR Configuration Profile using the same profile template that you used to create the models. The JKHLE customizations were based on the governance enablement profile template. Follow these steps:

1. Click **File** → **New** → **WSRR Configuration Project**.
2. In the Create a WSRR Configuration Project window:
   a. Enter a configuration project name, such as `JKHLE Profile`.
   b. Enter a location or leave the default location selected.
   c. Select the “Governance Enablement Profile - WSRR v6.3” option.
   d. Click **Finish**.
3. Close the project so that the `.emx` files can be replaced on the file system.
4. Right-click **JKHLE Profile**. Then, click **Import**.
5. Click **General** → **File System**.
6. Click **Next**.
7. Click **Browse** (next to the “From Directory” field) and navigate to **Desktop** → **My Computer** → **Local Disk (C:)** → **temp** or an alternate target directory.
8. Select the following `.emx` files (as shown in Figure 5-48 on page 216):
   – `6_3_GovernanceProfileExtensions.emx`
   – `6_3_ProfileModel.emx`
   – `LifecycleDefinition.emx`
9. Click **Browse** (next to the “Into folder” field). In the Import into Folder dialog box, expand **JKHLE Profile**. Then, click **UML Models**. Click **OK**.

   Note: If you import the `.emx` files into the root directory rather than the UML Models directory, duplicate UML models are created.

10. Select **Overwrite existing resources without warning**.
11. Click **Finish**.

Archived

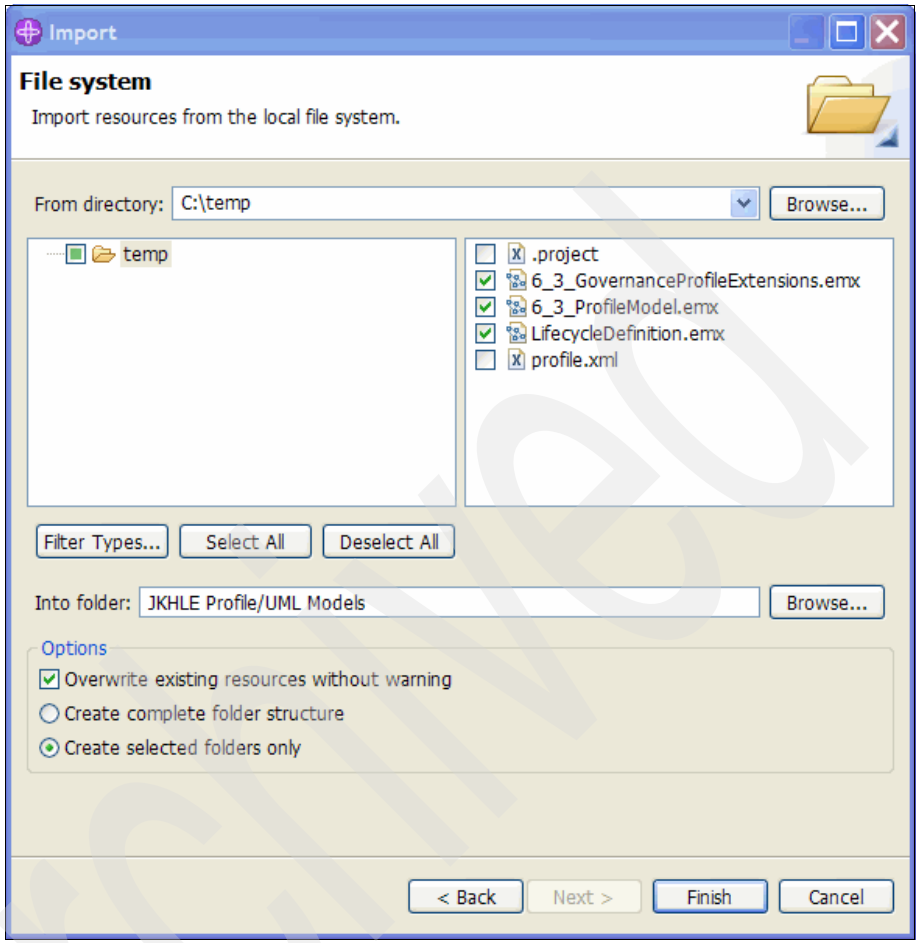


Figure 5-48 Import .emx files

The UML models in the profile are updated with the information from the `.emx` files. You can now generate the profile, which will include the changes in the UML files.

6

WebSphere Service Registry and Repository Web user interface

You can use the WebSphere Service Registry and Repository (WSRR) Web user interface (Web UI) to use and manage all aspects of WSRR from a compatible Web browser. The Web user interface is configured as a standard part of the WSRR installation.

This chapter describes how to customize aspects for the WSRR Web UI and includes the following topics:

- Overview of the Web UI
- WSRR Web UI definition files
- Themes
- Customizing the WSRR Web UI for the JKHLE business scenario
- Troubleshooting

© Copyright IBM Corp. 2009. All rights reserved.

6.1 Overview of the Web UI

The Web user interface for WSRR displays in a window that is divided into the following areas, as illustrated in Figure 6-1.

- Banner and menu bar
- Content in the main frame
- Footer

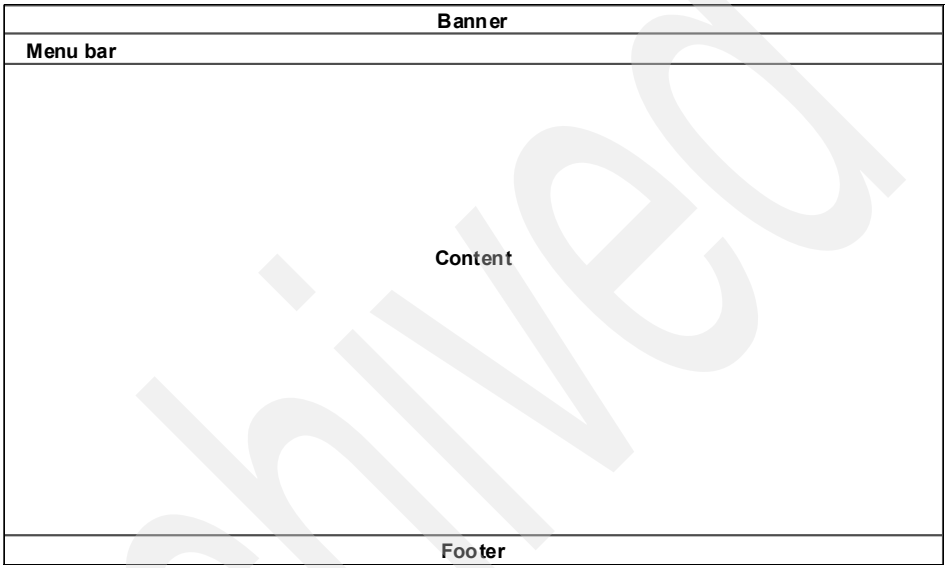


Figure 6-1 Web UI layout

Archived

Using the My Preferences page, you can also display a navigation tree in a side pane, if required, as shown in Figure 6-2.

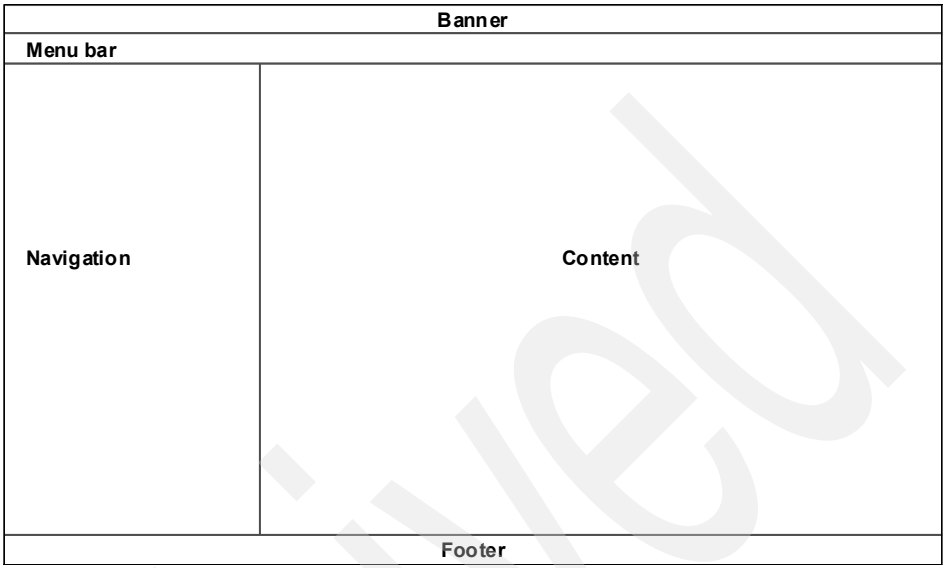


Figure 6-2 Navigation tree

The side pane can be on the left or the right, depending on the language that is displayed. The views displayed in the Web UI are dependent on the WSRR access control role that is associated with the user who is logging on.

6.2 WSRR Web UI definition files

Definition files define the content that displays in the navigation and work area frames of the WSRR Web UI. The set of system definition files that are installed with the product define the default user interface. You can customize the definition files to provide your own business views to WSRR data. In addition, definitions in one file might reference definitions that are in another file. Thus, a hierarchy exists between the various types of definition file, as illustrated in Figure 6-3.

Archived

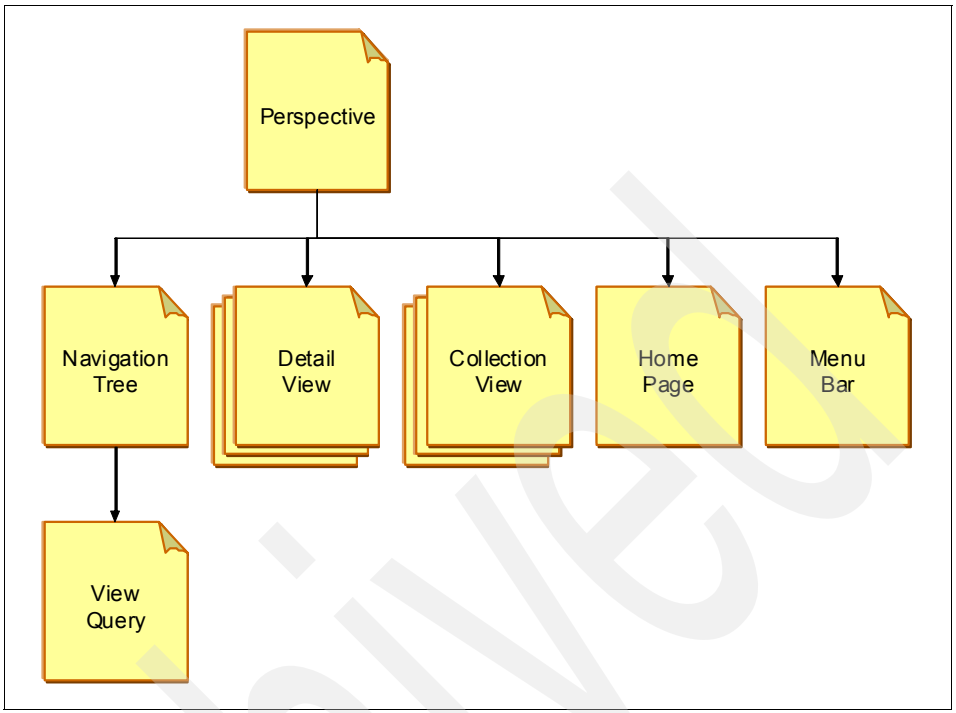


Figure 6-3 WSRR Web UI definition files

The following sections describe the different types of XML definition files that the WSRR Web UI uses.

6.2.1 Perspective

Perspective definition files include a collection of views that display the data in WSRR in a consistent manner. Different perspectives provide different views to the data that is stored in WSRR. You can use a perspective to define the following views:

- Menu bar
- Navigation tree
- Detail view
- Collection view

The `<roles>` element in a perspective defines the WSRR user roles that are permitted to view the perspective. When a user logs in to the WSRR Web UI, the list of roles to which that user is mapped is determined. If the list of roles to which the user is mapped and the list of roles specified in a perspective contain a matching role, then the user is permitted to view the perspective.

The banner frame in the WSRR Web UI displays a list box that contains an entry for each perspective that the user is allowed to view. The user can switch between perspectives by selecting the relevant entry in the list box, as shown in Figure 6-4.

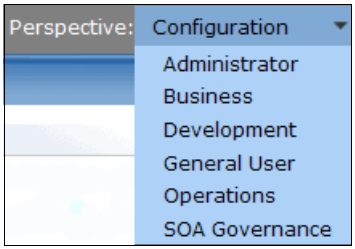


Figure 6-4 Perspective roles

Perspective definition files specify the menu bar, navigation tree, home page, detail views, and collection views that are displayed to users when viewing that perspective. Therefore, the perspective definition is a starting point when customizing the Web UI.

A perspective definition also specifies the following information:

- Whether the user can save a search created by the query wizard or by refining a collection view using one or more filters
- Which filters are available on collection views and query wizard results and whether filters are enabled for the perspective

A perspective definition contains mappings between display type names and view definitions, for both collection and detail view definitions. When a display type name is used to refer to a view definition, the current perspective is used to get the name of the actual view definition. Display type names are used by the following definition types:

- View query definitions
- Detail view definitions
- Collection view definitions

6.2.2 Menu bar

Menu bar definition files specify the structure of the menu bar. The menu bar includes the following available options in WSRR:

- The *Home* menu returns to the home page for the current perspective.
- The *Active Profile* menu provides views of the configuration items that are currently in use in the registry.

Archived

- The *Manage Profiles* menu provides views of the complete set of configuration items for WSRR.
- The *Actions* menu provides views to work with documents, preferences, subscriptions, and searches.
- The *View* menu provides views of business-oriented or abstract entities.
- The *Tasks* menu provides filtered views for a collection of entities. You can use a view to focus on the tasks that are specific to the role that is associated with a perspective.
- The *My Service Registry* menu provides views to work with subscriptions, searches, favorites, and preferences.
- The *Help* menu displays WSRR online help information.

There are a number of menu bar definition files supplied with WSRR that can be exported and used as examples to design your own menu bar definitions. For example, a menu bar definition file can contain a menu bar definition for a specific user role.

Figure 6-5 shows the default menu options that are available for the SOA governance perspective.

Figure 6-5 Default SOA governance menu bar

Table 6-1 shows the default menu option values for each role type.

Table 6-1 Default menu options

	Home	**Active Profile**	**Manage Profile**	**Actions**	**View**	**Tasks**	**My Service Registry**	**Help**
Administrator	✓			✓	✓		✓	✓
Business	✓			✓	✓	✓	✓	✓
Configuration	✓	✓	✓				✓	✓
Development	✓			✓	✓	✓	✓	✓
General User	✓				✓		✓	✓

Archived

	Home	Active Profile	Manage Profile	Actions	View	Tasks	My Service Registry	Help
Operations	✓			✓	✓	✓	✓	✓
SOA Governance	✓			✓	✓	✓	✓	✓

6.2.3 Navigation tree

Navigation tree definition files define the name and the structure of a navigation tree. A number of navigation tree definition files ship with WSRR. You can export and use these files as examples to design your own navigation tree definitions.

A navigation tree definition file contains a definition for a specific user role. It consists of several nodes that can be nested. Each node can be an HTML link that you click to perform an action. Figure 6-6 shows an example of a navigation tree for the General User role.

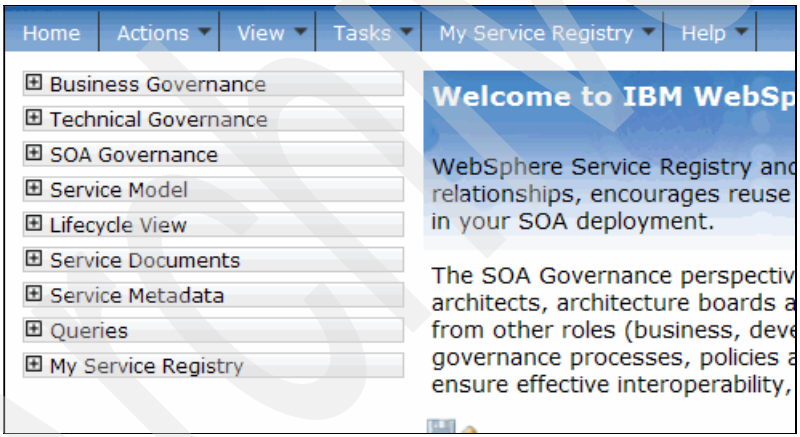


Figure 6-6 Example navigation tree

A link to the home page and a search box opens at the top of every navigation tree. Figure 6-7 shows the search box that displays in the menu bar. You can configure the search box not to display.

Figure 6-7 Search box

Archived

Users can choose to display the navigation tree from the My Preferences window, as shown in Figure 6-8.

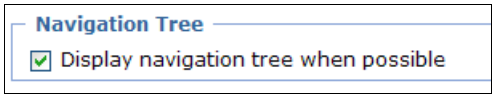


Figure 6-8 Selecting display of the navigation tree

6.2.4 View query

A *view query* definition file specifies a query to run against WSRR when a link in the navigation tree or an option in the menu bar is clicked. The results are shown in a *collection view*. A view query definition file can contain multiple view query definitions. Figure 6-9 shows in the browser location bar, the query executed when service level definitions is selected from the navigation tree.

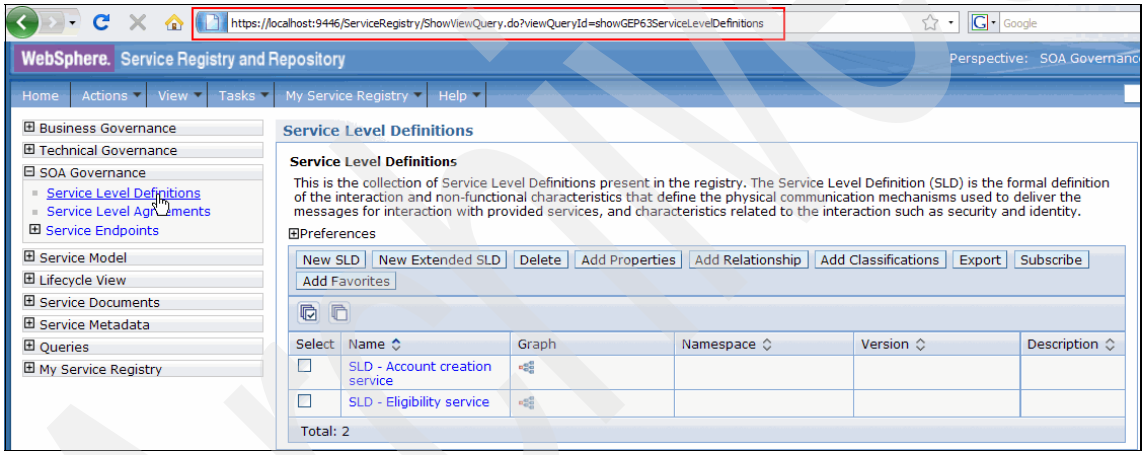


Figure 6-9 Running a query

6.2.5 Detail view

A *detail view* in the WSRR Web UI displays the details of a specific instance of an underlying WSRR entity. The properties that display are dependent on the type of the entity. A detail view can also display the target objects of a relationship and the classifications that are set on the object without the need to navigate to a separate collection view or editor. A detail view definition file contains only a single detail view definition. Figure 6-10 shows an example detail view for the input message of a WSDL operation.

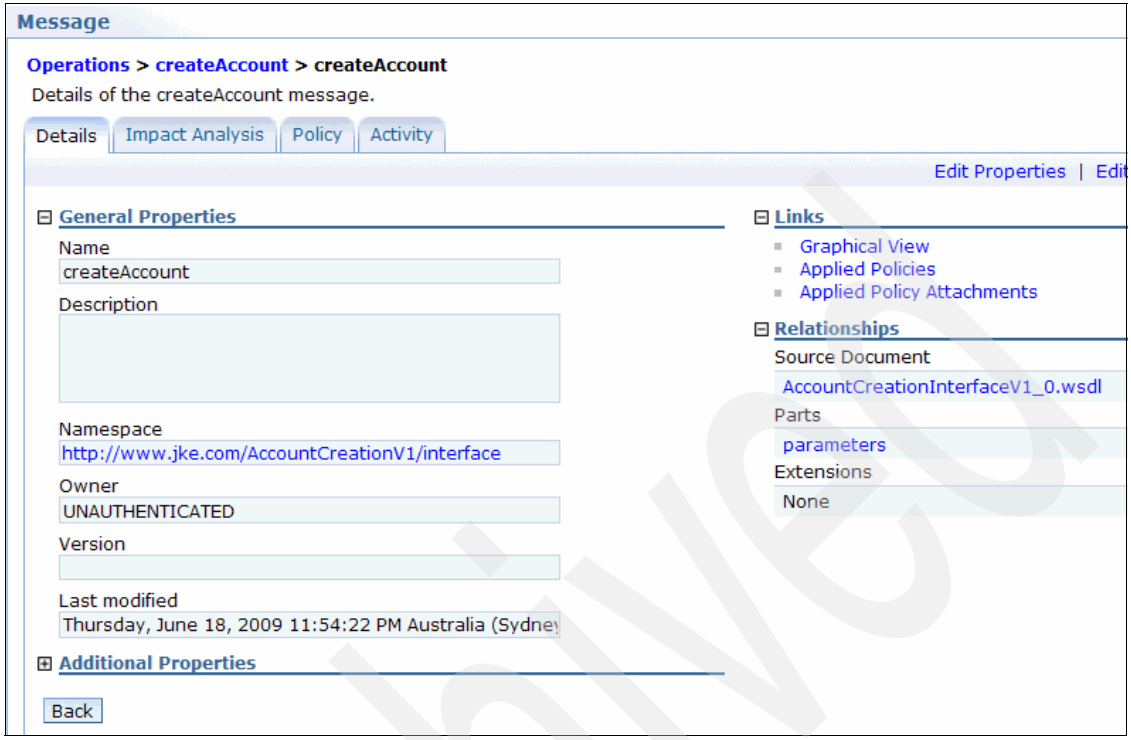


Figure 6-10 Example detail view

6.2.6 Collection view

A *collection view* in the WSRR Web UI displays a collection of the underlying entities that are stored in the repository. A collection view definition file describes the properties of the entity that are displayed in columns and the action buttons that are displayed. Figure 6-11 shows the buttons displayed on the service endpoint view.

Archived

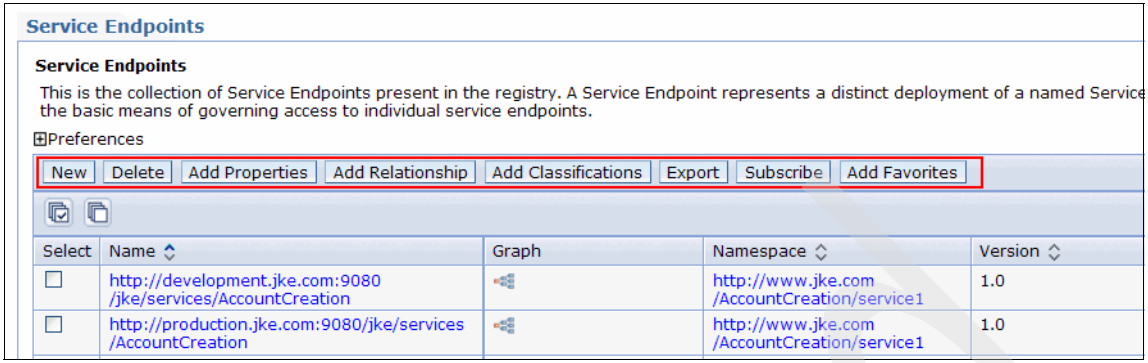


Figure 6-11 Service endpoints buttons

Example 6-1 shows an example button configuration.

Example 6-1 Example button configuration

```
<button-definitions>
   <button-definition>
      <button-message message-key="collectionButton.addProperties" />
      <button-action>addProperty</button-action>
</button-definition>
```

6.2.7 Home page

Each perspective has its own *home page*, which provides a range of functions to make it easier to find and work with WSRR objects. The home page displays when you switch to a perspective. The home page is built from the panels described in Table 6-2.

Table 6-2 Home page panels

Panel	Description
Welcome	Displays a home page title and welcome text.
Business objects	Lists the names of all the business model templates that are stored in the registry. To display a business object collection, click one of the business model template names listed in the business objects panel, which displays the entities that were created from that business model template.
Saved searches	Lists the names of the searches that are saved in the registry. Clicking the name of a search executes that search and displays the search results.
Service Documents	Lists all the document types that WSRR supports. To display the collection of documents of a specific type, click the document type in the list.

Archived

Panel	Description
Service Metadata	Lists all the types of objects that can be derived from WSDL documents or SCA integration modules that are loaded into the registry. To display the collection of objects of a specific type, click the object type in the list.
Load documents	Loads a document with all its dependent documents or saves the documents as a document group.
Browse by classification	Displays objects that have a specific classification.
External help resources	Provides access to WSRR help pages and external support pages.
About	Displays WSRR product information.

Each perspective has a separate home page configuration, and you can customize each configuration to suit your requirements.

Each home page configuration is specified by an XML definition file. You specify XML elements that control the design of each panel. A home page definition file contains a definition for a specific user role.

You can modify an existing home page configuration using any of the following methods:

- Modify the configuration directly using the Web UI
- Update the definition file offline and then load it into WSRR. You can retrieve the existing definition file before you make modifications.

Archived

Figure 6-12 illustrates the default home page layout for the General User role.

Figure 6-12 General user home page

Archived

6.3 Themes

Themes define the look of the WSRR Web UI. There are two different kinds of theme definition:

- Logon themes

  Only one logon theme can be active at a time, and this theme is used by all users.

- User themes

  Each user can choose a preferred theme by navigating to **My Service Registry** → **My Preferences** and selecting the required option, as shown in Figure 6-13.

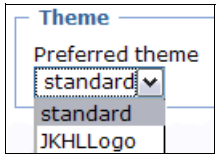


Figure 6-13 Selecting a user theme

By customizing a theme, you can tailor the look and feel of the WSRR Web UI for a particular organization or group of users. You can modify the following elements:

- Banner (Company or Product logo)
- Colors including fonts and backgrounds
- Font size
- Left to right or right to left layout
- Images
- Footer

A theme definition is a compressed file (.zip) that contains a set of files that define the look and feel of the user interface. A standard user theme definition file and logon theme definition file ship with WSRR.

Archived

Figure 6-28 illustrates how the standard themes can be exported and used as to design a customized theme definition.

Figure 6-14 Example customized theme

Archived

6.4 Customizing the WSRR Web UI for the JKHLE business scenario

> **Note:** In the examples in this book, we use a case study about a fictional company named *JKHL Enterprises* (JKHLE). For information about this case study, see Chapter 4, “JKHL Enterprises case study” on page 153.

JKHLE has identified the following customizations that they want to apply to the WSRR Web UI:

- Remove references to the service interface specification because this is removed from the business model at JKHLE.
- Remove the abstract asset links in the WSRR user interface because JKHLE prefers that users see only links to the concrete types.
- Remove the life cycle states and tasks that refer to the same life cycle states to reflect the changes that were made to the following life cycles:
  - Service level definition life cycle
  - Asset life cycle
  - SOA life cycle
- Use the term *Subscription Request* rather than *document of understanding* (DOU) to represent an agreement between two areas of the organization.
- Limit the number of relationships that display in a chain of objects when a graph view is opened.
- Brand the WSRR Web UI with the JKHLE company logo.

You can edit the WSRR Web UI configuration files either in the WSRR Web UI or in WSRR Studio. WSRR studio is recommended when making a large number of changes. To edit the configuration files in WSRR Studio, first open the WSRR Studio client. Then, open the JKHL Enterprises Configuration Profile or create a new project based on the governance enablement profile. For more information about modeling changes, refer to Chapter 5, “Modeling in WebSphere Service Registry and Repository Studio” on page 163.

6.4.1 Removing the service interface specification from the WSRR Web UI

Because JKHLE does not require the service interface specification, they want to remove all references to this object in the WSRR Web UI. This section describes how to remove these references.

Note: We suggest that you comment out entries that are not required rather than deleting them so that you can back out changes more easily.

Removing the service interface specification from the home pages

The following home pages reference the service interface specification and need to be updated:

- GEPDevelopmentHomePage
- GEPSOAGovernanceHomePage
- GEPUserHomePage

Archived

To update these pages, follow these steps:

1. Expand **JKHL Enterprises Configuration Profile** → **Configuration Profile Files** → **Web UI Configuration** → **Home Pages**. Then, double-click **GEPDevelopmentHomePage** to edit the file (Figure 6-15).

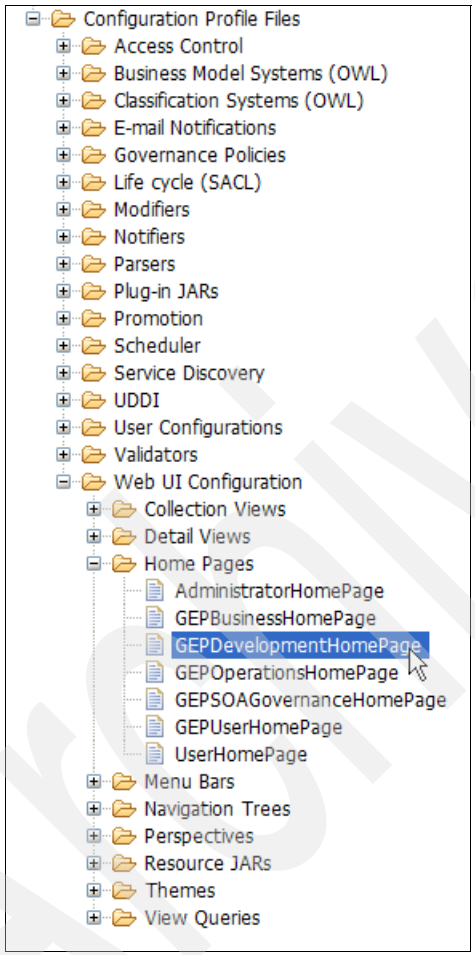


Figure 6-15 Selecting the menu bar UI configuration file

Archived

2. Comment out any reference to the `ServiceInterfaceSpecification` in this file, as shown in Example 6-2 and Figure 6-16.

Example 6-2 ServiceInterfaceSpecification

```
<directory-type-entry resource-bundle="GEP63Resources"
resource-key="navigationtree.unified.GEP63.concepts.ServiceInterface
Specification"
view-query-id="showGEP63ServiceInterfaceSpecifications" />
```

```
        <directory-type icon="WSRR_Banner_Generic_Blue.gif">
            <title-message resource-bundle-name="GEP63Resources"
                message-key="navigationtree.unified.GEP63.TechnicalGovernance" />
            <directory-type-entries>

                <directory-type-entry resource-bundle="GEP63Resources"
                    resource-key="navigationtree.unified.GEP63.concepts.CapabilityVersion"
                    view-query-id="showGEP63CapabilityVersions" />
                <directory-type-entry resource-bundle="GEP63Resources"
                    resource-key="navigationtree.unified.GEP63.concepts.ApplicationVersion"
                    view-query-id="showGEP63ApplicationVersions" />
                <directory-type-entry resource-bundle="GEP63Resources"
                    resource-key="navigationtree.unified.GEP63.concepts.ProcessVersion"
                    view-query-id="showGEP63ProcessVersions" />
                <directory-type-entry resource-bundle="GEP63Resources"
                    resource-key="navigationtree.unified.GEP63.concepts.ServiceVersion"
                    view-query-id="showGEP63ServiceVersions" />

<!--
                <directory-type-entry resource-bundle="GEP63Resources"
                    resource-key="navigationtree.unified.GEP63.concepts.ServiceInterfaceSpecification"
                    view-query-id="showGEP63ServiceInterfaceSpecifications" />
 -->

                <directory-type-entry resource-bundle="GEP63Resources"
                    resource-key="navigationtree.unified.GEP63.concepts.SchemaSpecification"
                    view-query-id="showGEP63SchemaSpecifications" />

            </directory-type-entries>
        </directory-type>
```

Figure 6-16 Commenting out the ServiceInterfaceSpecifiction

3. Close the text editor as shown in Figure 6-17.

Figure 6-17 Closing the editor pane

4. Click **Yes** to save the changes, as shown in Figure 6-18.

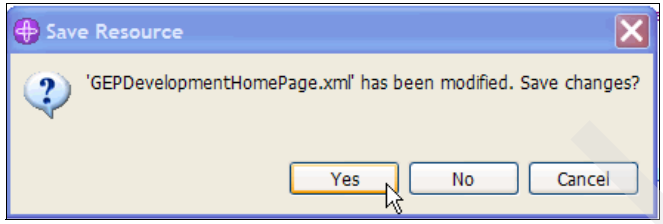


Figure 6-18 Saving changes

Note: You can also use Ctrl+S to save the changes and then close the file.

5. Repeat steps 1 through 4 for the GEPSOAGovernanceHomePage.
6. Repeat steps 1 through 4 for the GEPUserHomePage.

Removing the service interface specification from the menu bars

The following menu bars reference the service interface specification and need to be updated:

- GEPDevelopmentMenuBar
- GEPSOAGovernanceMenuBar
- GEPUserMenuBar

To update these menu bars, follow these steps:

1. Expand **JKHL Enterprises Configuration Profile** → **Configuration Profile Files** → **Web UI Configuration** → **Menu Bars**. Then, double-click **GEPDevelopmentMenuBar** to edit the file.
2. Comment out each of the `menu-item` elements that has an ID of `unified.ServiceInterfaceSpecification` or that begins with this ID, as shown in Example 6-3 and Example 6-4.

Example 6-3 createServiceInterfaceSpecification

```
<menu-item id="createServiceInterfaceSpecification"
resource-bundle-name="LM63Resources"
message-key="navigationtree.task.LM63.ServiceInterfaceSpecificationL
ifecycle.AssetScoped"
```

```
url="CreateGenericObject.do?businessModelTemplateName=http%3A//www.i
bm.com/xmlns/prod/serviceregistry/profile/v6r3/GovernanceEnablementM
odel%23ServiceInterfaceSpecification">
</menu-item>
```

Example 6-4 AsserIdentified

```
<menu-item
id="unified.ServiceInterfaceSpecification.AssetIdentified"
resource-bundle-name="LM63Resources"
message-key="navigationtree.unified.LM63.ServiceInterfaceSpecificati
onLifecycle.AssetIdentified"
view-query-id="showLM63ServiceInterfaceSpecificationAssetIdentified"
/>
```

3. Close the text editor, and save your changes.
4. Repeat steps 1 through 3 for the GEPSOAGovernanceMenuBar.
5. Repeat steps 1 through 3 for the GEPUserMenuBar.

Removing the service interface specification from the navigation tree

The following navigation trees reference the service interface specification and need to be updated:

- GEPDevelopmentNavigationTree
- GEPSOAGovernanceNavigationTree
- GEPUserMenuBar

To update these navigation trees, follow these steps:

1. Expand **JKHL Enterprises Configuration Profile** → **Configuration Profile Files** → **Web UI Configuration** → **Navigation Trees**. Then, double-click **GEPDevelopmentNavigationTree** to edit the file.
2. Comment out each of the `node id` elements that has an ID of `unified.ServiceInterfaceSpecification` or that begins with this ID.
3. Close the text editor, and save your changes.
4. Repeat steps 1 through 3 for the GEPSOAGovernanceMenuBar.
5. Repeat steps 1 through 3 for the GEPUserMenuBar.

Archived

6.4.2 Removing the abstract asset links from the WSRR UI

This section describes how to remove the asset life cycle and asset tasks from the WSRR UI.

Removing the asset life cycle and asset tasks from the menu bars

The following menu bars reference the asset life cycle and asset tasks and need to be updated:

- GEPBusinessMenuBar
- GEPDevelopmentMenuBar
- GEPSOAGovernanceMenuBar

To update these menu bars, follow these steps:

1. Expand **JKHL Enterprises Configuration Profile** → **Configuration Profile Files** → **Web UI Configuration** → **Menu Bars**. Then, double-click **GEPBusinessMenuBar** to edit the file.
2. Comment out each of the `menu-item` elements that has an ID of `unified.AssetLifecycle` or that begins with this ID.
3. Close the text editor, and save your changes.
4. Repeat steps 1 through 3 for the GEPDevelopmentMenuBar.
5. Repeat steps 1 through 3 for the GEPSOAGovernanceMenuBar.

Removing the asset life cycle from the navigation tree

The following navigation trees reference the asset life cycle and need to be updated:

- GEPBusinessNavigationTree
- GEPDevelopmentNavigationTree
- GEPSOAGovernanceNavigationTree

To update these navigation trees, follow these steps:

1. Expand **JKHL Enterprises Configuration Profile** → **Configuration Profile Files** → **Web UI Configuration** → **Navigation Trees**. Then, double-click **GEPBusinessNavigationTree** to edit the file.
2. Comment out each of the `node id` elements that has an ID of `unified.AssetLifecycle` or that begins with this ID.
3. Close the text editor, and save your changes.

Archived

4. Repeat steps 1 through 3 for the GEPDevelopmentMenuBar.
5. Repeat steps 1 through 3 for the GEPSOAGovernanceMenuBar.

6.4.3 Removing the life cycle states from the WSRR UI that were removed from the model

The menu bar and navigation tree definition files need to be updated to reflect the changes that were made in the modelling chapter to remove some of the states from the SOA, Asset and SLD life cycles.

Table 6-3 through Table 6-6 show the life cycle states and their equivalent key in the menu bar and navigation tree files.

Table 6-3 states to be removed from the SOA life cycle

Life cycle state	Key
PlanReview	unified.SOALifecycle.PlanReview
Planned	unified.SOALifecycle.Planned
SpecificationReview	unified.SOALifecycle.SpecificationReview
RealizationReview	unified.SOALifecycle.RealizationReview
Realized	unified.SOALifecycle.Realized
StagingReview	unified.SOALifecycle.StagingReview
CertificationReview	unified.SOALifecycle.CertificationReview
OperationalReview	unified.SOALifecycle.OperationalReview

Table 6-4 States to be removed from the Schema Specification life cycle

Life cycle state	Key
AssetScoped	unified.SchemaSpecification.AssetScoped
AssetSpecificationReview	unified.SchemaSpecification.AssetSpecificationReview
AssetSpecified	unified.SchemaSpecification.AssetSpecified
AssetReview	unified.SchemaSpecification.AssetReview

Archived

Table 6-5 States to be removed from the DOU (Subscription Request) life cycle

Life cycle state	Key
AssetScoped	unified.DoU.AssetScoped
AssetSpecificationReview	unified.DoU.AssetSpecificationReview
AssetSpecified	unified.DoU.AssetSpecified
AssetReview	unified.DoU.AssetReview

Table 6-6 States to be removed from the SLD life cycle

Life cycle state	Key
SLDScoped	unified.SLDLifecycle.SLDScoped
SLDReview	unified.SLDLifecycle.SLDReview

Removing the life cycle states from the menu bars

The following menu bars reference the life cycle states and need to be updated:

- GEPBusinessMenuBar
- GEPDevelopmentMenuBar
- GEPOperationsMenuBar
- GEPSOAGovernanceMenuBar

To update these menu bars:

1. Expand **JKHL Enterprises Configuration Profile** → **Configuration Profile Files** → **Web UI Configuration** → **Menu Bars**. Then, double-click **GEPBusinessMenuBar** to edit the file.
2. Locate the `Lifecycles` menu-item element as shown in Example 6-5.

   Example 6-5 Lifecycles menu-item element

   ```
   <menu-item id="Lifecycles" resource-bundle-name="LM63Resources"
   message-key="menubar.unified.LM63.LifecycleView">

               <menu-item id="unified.CapabilityLifecycle" ...
   ```

3. Within this element, comment out all the `menu-id` elements that have an ID which matches the key in tables Table 6-3 on page 238 through Table 6-6 on page 239. Example 6-6 shows this element.

   Example 6-6 PlanReview

   ```
   <menu-item id="unified.SOALifecycle.PlanReview"
   resource-bundle-name="LM63Resources"
   ```

```
message-key="navigationtree.unified.LM63.SOALifecycle.PlanReview"
view-query-id="showLM63SOALifecyclePlanReview" />
```

4. Close the text editor, and save your changes.
5. Repeat steps 1 through 4 for the GEPDevelopmentMenuBar.
6. Repeat steps 1 through 4 for the GEPOperationsMenuBar.
7. Repeat steps 1 through 4 for the GEPSOAGovernanceMenuBar.

Removing the life cycle states from the navigation trees

The following navigation trees reference the life cycle states and need to be updated:

- GEPBusinessNavigationTree
- GEPDevelopmentNavigationTree
- GEPOperationsNavigationTree
- GEPSOAGovernanceNavigationTree

To update these navigation trees:

1. Expand **JKHL Enterprises Configuration Profile** → **Configuration Profile Files** → **Web UI Configuration** → **Navigation Tree**. Then, double-click **GEPBusinessNavigationTree** to edit the file.
2. Locate the `Lifecycles` node ID element as shown in Example 6-7.

Example 6-7 Lifecycles node ID element

```
<node id="Lifecycles" node-resource-bundle="LM63Resources"
node-resource-key="menubar.unified.LM63.LifecycleView">

        <node id="unified.CapabilityLifecycle"..
```

3. Within this element, comment out all the `node id` elements that have an ID that matches the key in Table 6-3 on page 238 through Table 6-6 on page 239. Example 6-6 shows this element.

Example 6-8 PlanReview

```
<node id="unified.SOALifecycle.PlanReview"
node-resource-bundle="LM63Resources"
node-resource-key="navigationtree.unified.LM63.SOALifecycle.PlanRevi
ew" view-query-id="showLM63SOALifecyclePlanReview" />
```

4. Close the text editor, and save your changes.
5. Repeat steps 1 through 4 for the GEPDevelopmentNavigationTree.

6. Repeat steps 1 through 4 for the GEPOperationsNavigationTree.
7. Repeat steps 1 through 4 for the GEPSOAGovernanceNavigationTree.

Figure 6-19 illustrates the changes to the navigation tree for the development perspective that is applied when the updated configuration profile is activated in WSRR.

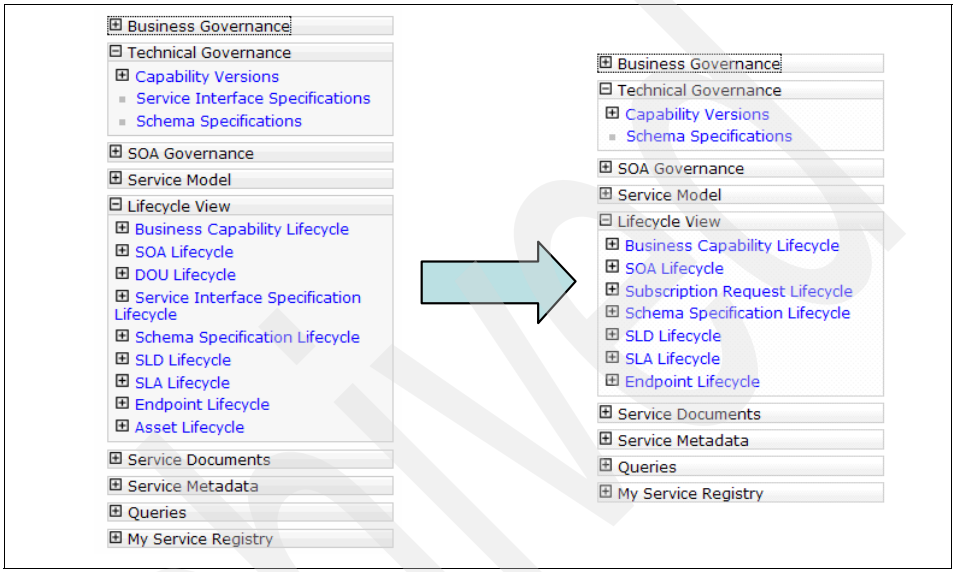


Figure 6-19 Changes to the navigation tree

6.4.4 Removing the items from the tasks menu bar that reference the removed life cycle states

The menu bar definition files need to be updated to reflect the changes that were made in the modelling chapter to remove some of the life cycle states from the SOA, Asset and SLD life cycles.

Note: An equivalent of the tasks menu does not exist in the Navigation Tree.

Archived

Tables Table 6-7 to Table 6-10 show the life cycle states and their equivalent view queries to find objects in the said state.

Table 6-7 states to be removed from the SOA life cycle

Life cycle state	view query
PlanReview	showLM63SOALifecyclePlanReview
Planned	showLM63SOALifecyclePlanned
SpecificationReview	showLM63SOALifecycleSpecificationReview
RealizationReview	showLM63SOALifecycleRealizationReview
Realized	showLM63SOALifecycleRealized
StagingReview	showLM63SOALifecycleStagingReview
CertificationReview	showLM63SOALifecycleCertificationReview
OperationalReview	showLM63SOALifecycleOperationalReview

Table 6-8 States to be removed from the Schema Specification life cycle

Life cycle state	Key
AssetScoped	showLM63SchemaSpecificationAssetScoped
AssetSpecificationReview	showLM63SchemaSpecificationAssetSpecificationReview
AssetSpecified	showLM63SchemaSpecificationAssetSpecified
AssetReview	showLM63SchemaSpecificationAssetReview

Table 6-9 States to be removed from the DOU (Subscription Request) life cycle

Life cycle state	Key
AssetScoped	showLM63DoUAssetScoped
AssetSpecificationReview	showLM63DoUAssetSpecificationReview
AssetSpecified	showLM63DoUAssetSpecified
AssetReview	showLM63DoUAssetReview

Archived

Table 6-10 States to be removed from the SLD life cycle

Life cycle state	Key
SLDScoped	showLM63SLDScoped
SLDReview	showLM63SLDReview

Removing items from the task menu bars

The following menu bars reference the life cycle states and need to be updated:

- GEPBusinessMenuBar
- GEPDevelopmentMenuBar
- GEPOperationsMenuBar
- GEPSOAGovernanceMenuBar

To update these menu bars:

1. Expand **JKHL Enterprises Configuration Profile** → **Configuration Profile Files** → **Web UI Configuration** → **Menu Bars**. Then, double-click **GEPBusinessMenuBar** to edit the file.
2. Locate the `Tasks` menu-item element, as shown in Example 6-9.

Example 6-9 Tasks

```
<menu-item id="Tasks"
         resource-bundle-name="LM63Resources"
         message-key="menubar.unified.LM63.Tasks">

         <menu-item id="unified.CapabilityLifecycle"...
```

3. Within this element, comment out all the `menu-item` elements that have a `view-query-id` that matches the view queries in Table 6-7 on page 242 through Table 6-10 on page 243. Example 6-6 shows this element.

Example 6-10 PlanReview

```
<menu-item id="unified.SOALifecycle.PlanReview"
resource-bundle-name="LM63Resources"
message-key="navigationtree.task.LM63.SOALifecycle.PlanReview"
view-query-id="showLM63SOALifecyclePlanReview" />
```

4. Close the text editor, and save your changes.
5. Repeat steps 1 through 4 for the GEPDevelopmentMenuBar.
6. Repeat steps 1 through 4 for the GEPOperationsMenuBar.
7. Repeat steps 1 through 4 for the GEPSOAGovernanceMenuBar.

Archived

6.4.5 Updating the task names in the menu bar to reflect the changes in the runtime environments

JKHLE wants to update the tasks in the menu bar so that the tasks that promote the capability version to each runtime environment fit with the new life cycle. They need to update the resource key definitions so that the menus in the task bar are defined as shown in Table 6-11.

Table 6-11 Resource keys and new task names

Task (Resource key)	Task Name
navigationtree.task.LM63.SOALifecycle.Specified	Define SLD & prepare for Staging
navigationtree.task.LM63.SOALifecycle.Staged	Prepare for Pre-Production
navigationtree.task.LM63.SOALifecycle.Certified	Prepare for Production

To update the task names, follow these steps:

1. Expand **JKHL Enterprises Configuration Profile** → **Configuration Profile Files** → **Resource JARs**. Then, double-click **ProfileResources**.

   The property files in this resource bundle contain values for the labels, menus, navigation trees, screens, and buttons that display in the WSRR Web UI. JKHLE has a requirement to update the only the English language property files.

   Note: These instructions assume that a suitable compression tool is installed that allows you to open and update a file directly from within a compressed file.

2. Open the file `LM63Resources_en.properties` file.
3. Find the resource keys listed in Table 6-11 and replace the values for these keys with the values shown in the table. Example 6-11 shows the updated items.

Example 6-11 Updated resource values

```
navigationtree.task.LM63.SOALifecycle.Specified = Define SLD & prepare for Staging
navigationtree.task.LM63.SOALifecycle.RealizationReview = Versions for Realization Review
navigationtree.task.LM63.SOALifecycle.Realized = Version Staging
```

4. Save the changes, and update the archive file.

6.4.6 Adding life cycle tasks in the UI

JKHLE wants the SOA governance role to also work with planned capability versions. To update the menu bar tasks for the SOA governance role, follow these steps:

1. Expand **JKHL Enterprises Configuration Profile** → **Configuration Profile Files** → **Web UI Configuration** → **Menu Bars**. Then, double-click **GEPSOAGovernanceMenuBar** to edit the file.
2. Locate the `Tasks` menu-item element as shown in Example 6-12.

Example 6-12 Tasks

```
<menu-item id="Tasks"
         resource-bundle-name="LM63Resources"
         message-key="menubar.unified.LM63.Tasks">

         <menu-item id="unified.CapabilityLifecycle"...
```

Within the `Tasks` menu-item element, locate the `unified.SOALifecycle` menu-id element. Add the entry shown in Example 6-13 *after* the `unified.SOALifecycle.ScopeReview` menu-id element.

Example 6-13 Scoped

```
<menu-item id="unified.SOALifecycle.Scoped"
resource-bundle-name="LM63Resources"
message-key="navigationtree.task.LM63.SOALifecycle.Scoped"
view-query-id="showLM63SOALifecycleScoped" />
```

JKHLE also requires that the Operations role has access to capability versions that are in the specified state and that are in the process of being deployed to the stagging environment. To add this entry:

1. Expand **JKHL Enterprises Configuration Profile** → **Configuration Profile Files** → **Web UI Configuration** → **Menu Bars**. Then, double-click **GEPOperationsMenuBar** to edit the file.
2. Locate the `Tasks` menu-item element. Within the `Tasks` menu-item element, locate the `unified.SOALifecycle` menu-id element. Add the entry shown in Example 6-14 *after* the `unified.SOALifecycle.SpecificationReview` menu-id element.

Example 6-14 Specified

```
<menu-item id="unified.SOALifecycle.Specified"
resource-bundle-name="LM63Resources"
message-key="navigationtree.task.LM63.SOALifecycle.Specified"
view-query-id="showLM63SOALifecycleSpecified"/>
```

6.4.7 Replacing the term “DOU” with the term “Subscription Request” in the WSRR Web UI

JKHLE does not want the term *DOU* displayed in the WSRR Web UI. Instead, they want to replace this term with the term *Subscription Request*, as follows:

1. Expand **JKHL Enterprises Configuration Profile** → **Configuration Profile Files** → **Resource JARs**. Then, double-click **ProfileResources**.

2. JKHLE needs to update the following English language property files:

   – `GEP63Resources_en.properties`
   – `LM63Resources_en.properties`

   Open each property file in a text editor, and search for references to the term *DOU*.

   Note: Do *not* use a global search and replace.

   Take care in checking the context of the reference to *DOU* before you replace it with *Subscription Request*. Some occurrences are singular and some are plural, as shown in Example 6-15.

   Example 6-15 Sample text from a property file

   ```
   collection.view.DoU.title = DOUs
   detail.view.DoU.title = DOU Detail View
   ```

   Example 6-16 shows the same entry after changing the term.

   Example 6-16 Updated property file entry

   ```
   collection.view.DoU.title = Subscription Requests
   detail.view.DoU.title = Subscription Request Detail View
   ```

   Note: Do not modify the key values.

3. Save the changes, and update the archive file.

Archived

6.4.8 Updating the SOALifecycleDecisionRights governance policy

The WSRR UI that comes with the GEP is configured to display a button for each of the available governance transitions that the current user is permitted to perform, such as the one shown in Figure 6-20.

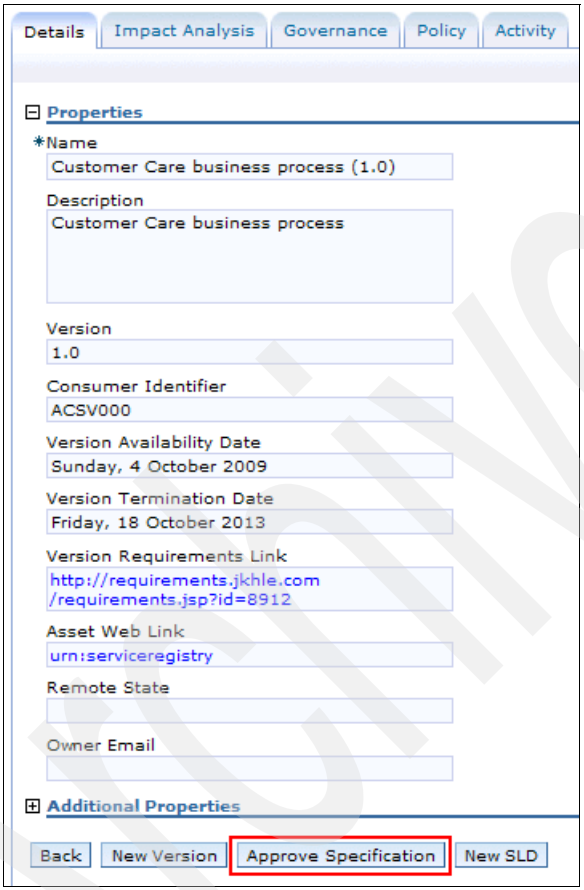


Figure 6-20 Example of a custom button for a governance transition

The governance transition buttons are defined in the artifacts detail view, but the buttons only display in the WSRR UI if the appropriate governance policy validation rules also return true. Because JHKLE customized the SOALifecycle governance enablement profile, they also need to update the SOALifecycleDecisionRights policy to allow the roles that come with the profile to transition the capability version artifacts when the policy is in a particular state.

Archived

JKHLE needs to make the following changes:

- Remove permission for a user in the Business Role to transition a capability version through ApproveProductionDeployment.
- Remove permission for a user in the SOA Governance Role to transition a capability version through ApproveCertification.
- Add permission for a user in the Operation Role to transition a capability version through:
  - ApproveCertification
  - ApproveProductionDeployment
  - Supercede

The policy itself contains these rules twice—once for the transition operation filter and again for the validate operation filter. The transition filter grants or denies permission for whether the user can perform the transition. The validate filter, however, is used to determine whether the button displays.

You can make this update in WSRR Studio either by updating the XML file or by loading the policy file into WSRR as content and using the WSRR policy editor to make the changes. If you use the WSRR policy editor, you need to import the updated policy file back into WSRR studio. For detailed information about policies, see Chapter 9, “Policies” on page 301.

To make these changes in WSRR Studio:

1. Expand **JKHL Enterprises Configuration Profile** → **Configuration Profile Files** → **Governance Policies**. Then, double-click **SOALifecycleDecisionRights** to edit the file.
2. Make the appropriate changes as we described previously.

> **Note:** An updated SOALifecycleDecisionRights file is included with the additional materials supplied with this book. (See Appendix B, “Additional material” on page 621 for more information.) It is recommended that you compare these files to see the exact changes that are required.

Archived

6.4.9 Modifying default user preferences

The WSRR Web UI default preferences for all users are defined in the default UI preferences file. JHKLE wants to set the default display depth of the graph to limit the number of relationships that display in a chain of objects as follows:

1. Expand **JKHL Enterprises Configuration Profile** → **Configuration Profile Files** → **User Configurations**. Then, double-click **DEFAULT_UI_PREFERENCES** to edit the file.
2. Edit the depth property to be *2*, as shown in Figure 6-21.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE properties SYSTEM "http://java.sun.com/dtd/properties
<properties>
   <comment/>

   <!-- set the default Graph View/Auto-suggest properties
   <!-- orientation can be topBottom or leftRight
   <entry key="orientation">leftRight</entry>

   <!-- maxAutoObjects can be >=1 or automatic
   <!-- (number of boxes to display)
   <entry key="maxAutoObjects">50</entry>

   <!-- displayMode can be external, contents or all
   <entry key="displayMode">all</entry>

   <!-- depth limits the hierarchy of entities displayed
   <!-- (can be auto, 1, 2, 3, 4 or unlimited)
   <entry key="depth">2</entry>
```

Figure 6-21 Updating the graph view to default to a depth of 2

Note: After these changes are persisted to the WSRR database, these changes are not applied until the next time a user logs in.

Archived

Figure 6-22 shows the updated WSRR Web UI graphical view that is applied after the configuration profile is uploaded. The default display depth of the graph is now limited to two levels.

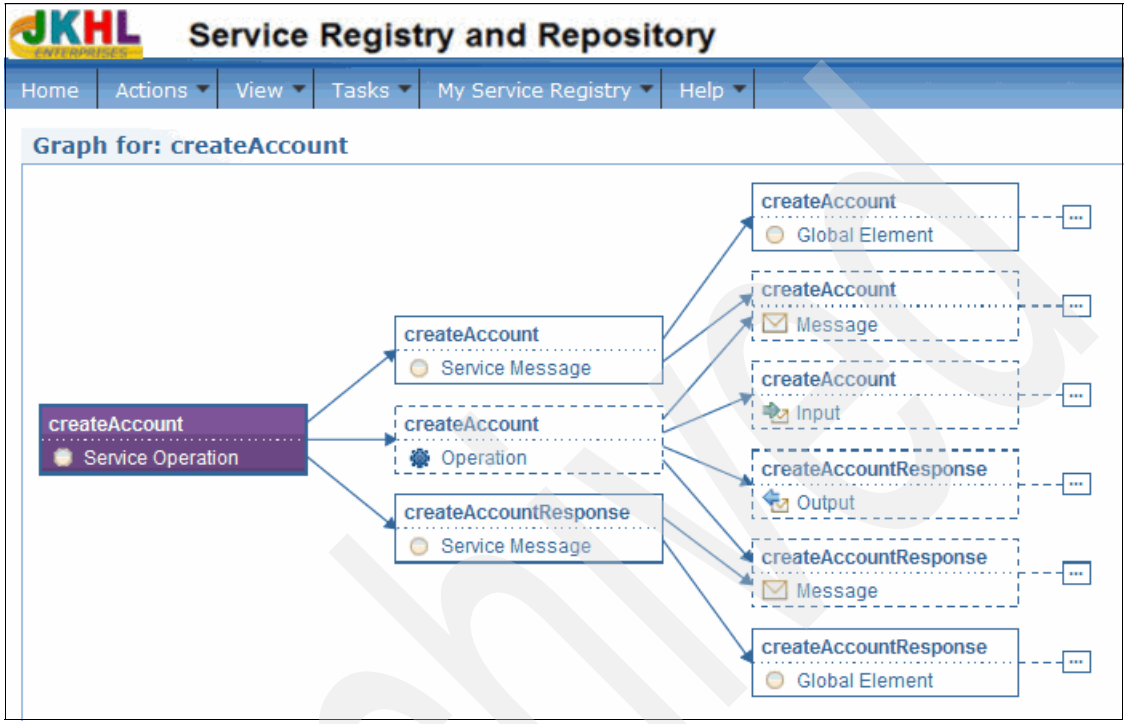


Figure 6-22 Displaying a graph with a depth of two levels

A user can, at any time, change the number of levels that display for a particular graph by changing the Graph Display Depth option on the graph view, as shown in Figure 6-23.

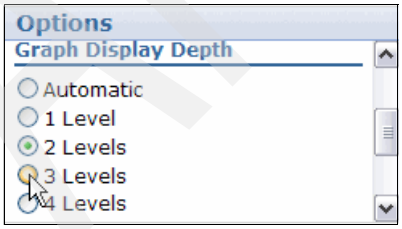


Figure 6-23 Changing the graph display depth

6.4.10 Updating the configuration profile on the WSRR server

This section describes how to export a profile, and load and activate that profile in WSRR.

Exporting a profile

To export a profile:

1. Save and close any open files in WSRR Studio.
2. Right-click **JKHL Enterprises Configuration Profile**, and click **Export**, as shown in Figure 6-24.

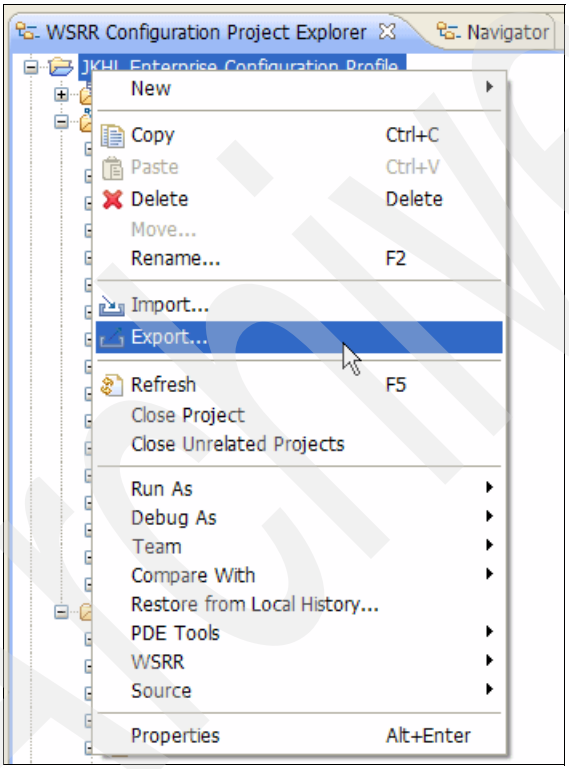


Figure 6-24 Exporting a profile

Archived

3. Expand **WebSphere Service Registry and Repository (WSRR),** and select **WSRR Configuration Profile**, as shown in Figure 6-25. Then, click **Next**.

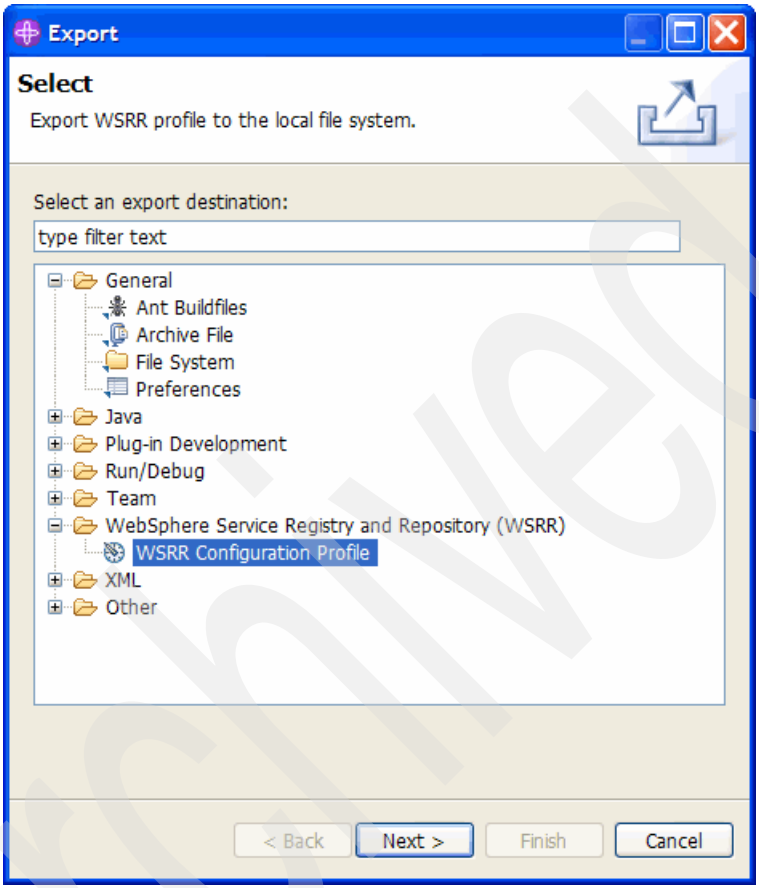


Figure 6-25 Exporting a WSRR configuration profile

Archived

4. Specify a target directory as shown in Figure 6-26, and click **Finish**.

Figure 6-26 Specifying the name of the export file

Loading and activating the profile in WSRR

To load and activate the profile in WSRR, log on to WSRR, and select the **Configuration** perspective. Then, follow these steps:

1. Click **Manage Profile** → **Configuration Profile**. Then, click **Load Configuration Profile**.
2. Browse to the location the configuration profile was saved to and select the configuration profile (for example `JKHL Enterprises Configuration Profile.zip`). Click **Open**.
3. Enter a configuration profile name of `JKHL Enterprises Configuration Profile`, and click **OK**.
4. Select the **JKHL Enterprises Configuration Profile**, and click **Make Active**.

6.4.11 Creating the JKHLE theme

Note: WSRR Studio does not currently support updating themes.

To create a new theme:

1. Export the standard theme.
2. Edit the theme files.
3. Load the new theme.
4. Make the theme available to users.

Export the standard theme

To export a theme, log on to WSRR, and select the Configuration perspective. Then, follow these steps:

1. Click **Active Profile** → **Web UI Configuration** → **Themes** → **User Themes**, as shown in Figure 6-27.

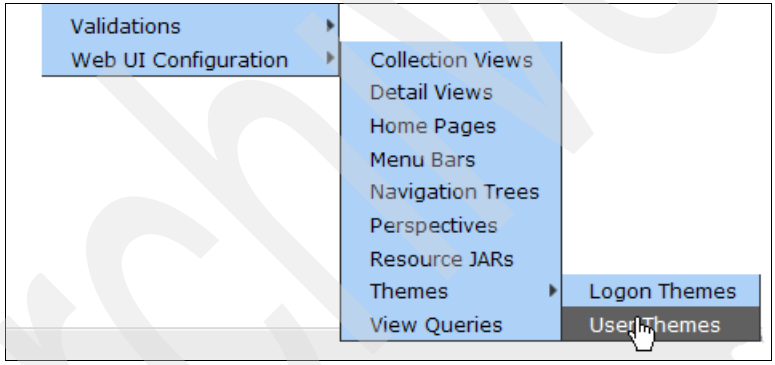


Figure 6-27 Selecting User Themes menu option

Archived

2. Select the standard theme, and click **Export**, as shown in Figure 6-28.

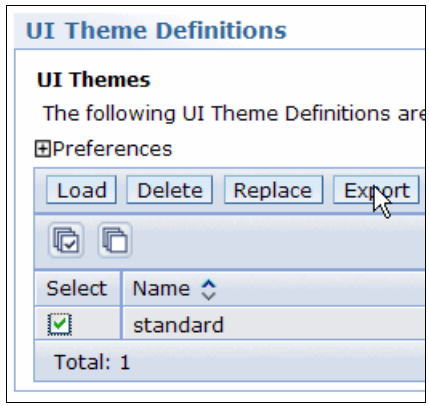


Figure 6-28 Exporting the standard theme

3. Save the `standard.zip` file to your local file system.

Edit the theme files

The compressed file that you saved to your local directory includes the CSS, JSP, and image files that are used to define the standard theme. You use these files as the basis for your new theme. To edit the theme files, follow these steps:

1. Extract the contents of the compressed file to your local file system.
2. Update the banner logo:
   a. Copy the JHKLE logo to the `images` directory.
   b. Copy the updated WSRR graphic to the `images` directory. (This new graphic has black text on a white background.)
   c. Edit the `banner.jsp` file that is located in the `standard` directory, as shown in Example 6-17.
   d. Save the file.

Example 6-17 Updating the JHKLE logo

```
<%@ taglib uri="/WEB-INF/sr-utils.tld" prefix="sr" %>

<div class="banner">
  <div class="bannerlogo" id="bannerlogo">
    <div class="bannerleft">
      <div class="bannerright"></div>
        <img src="theme/${currentTheme}/images/jkhlLogo.jpg" alt=""
id="jkhllogo"/>
```

```
        <img src="theme/${currentTheme}/images/blackregistry.jpg"
alt="<sr:message key='banner.wsrr.logo'/>" title="<sr:message
key='banner.wsrr.logo'/>" id="productname"/>
        <img src="theme/${currentTheme}/images/companyLogo.gif"
alt="<sr:message key='banner.IBM.logo'/>" title="<sr:message
key='banner.IBM.logo'/>" id="ibmlogo"/>
    </div>
  </div>
  <div class="bannerbottom">
    <div class="bannerbottomleft"></div>
    <div class="bannerbottomright"></div>
  </div>
</div>
```

e. Open the `pageStyles.css` file that is located in the `css` directory. Search for the text *Banner*.

f. Update the `.bannerlogo img#webspherelogo` setting by replacing the text `img#webspherelogo` with `img#jkhllogo` as shown in Example 6-18.

Example 6-18 Updating the banner logo

```
.bannerlogo img#jkhllogo {
  position: absolute;
  top: 3px;
  left: 13px;
}
.bannerlogo img#productname {
  position: absolute;
  top: 7px;
  left: 107px;
}}
```

3. Change the banner background color:

   a. Open the `pageStyles.css` file that is located in the `css` directory.

   b. Search for the text *Banner*.

   c. Update the `.bannerlogo` setting by changing the `background-color` setting to white as shown in Example 6-19.

   d. Comment out the entries for `background: url` for the `.bannerlogo`, `.bannerleft`, and `.bannerright`.

Example 6-19 Updating the background color

```
.bannerlogo {
  width: 100%;
```


Archived

```
  height: 30px;
  background-color: #ffffff;
/*  background: url(../images/bannerRepeat.png) top left repeat-x;
*/
}
.bannerleft {
  width: 100%;
  height: 30px;
/*  background: url(../images/bannerLeft.png) top left no-repeat; */
}
.bannerright {
  width: 100%;
  height: 30px;
/*  background: url(../images/bannerRight.png) top right no-repeat;
*/
}}
```

4. Finally, modify the banner text color to be black:
   a. In the `pageStyles.css` file search for the text *misc*.
   b. Update the banner text color settings to black as shown in Example 6-20.
   c. Save the file.

Example 6-20 Changing the text color on the banner

```
/********/
/* misc */
/********/
.signoutcontainer {
  position: absolute;
  right: 80px;
  top: 8px;
  z-index: 2000;
}
.signout {
  color: black;
}
.signout a {
  color: black;
}
.signout a:visited {
  color: black;
}
.signout a:hover {
  color: black;
```

Archived

```
}
.signoutcontainer .menubar ul, .signoutcontainer .menubar
a.menutitle {
  position: relative;
  top: -5px;
  color: black;
}
```

Note: If you use Internet Explorer as your browser, we recommend that you use Internet Explorer V7 or above. It you are using V6, you need to make the following additional modification to the style sheet to change the menu text to black:

```
.ie6 .menubar a {
  text-decoration: none !important;
  color: #000000 !important;
}
```

This change results in all menu text displaying in black, not just the banner menus.

5. Create a new compressed file that contains the full directory structure, as shown in Figure 6-29.

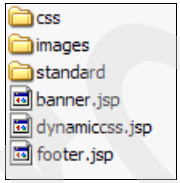


Figure 6-29 Theme directory structure

Refer to the WSRR information center for more detail about customizations that you can make to themes:

http://publib.boulder.ibm.com/infocenter/sr/v6r3/topic/com.ibm.sr.doc/rwsr_webui_ref_themes_create.html

Load the new theme

You can now load the new theme that is contained in the compressed file into WSRR as follows:

1. Click **Active Profile** → **Web UI Configuration** → **Themes** → **User Themes**.
2. Click **Load**.

3. Click **Browse**. Navigate to the new compressed theme file, and click **Open**.
4. Provide the UI theme configuration item name, as shown on Figure 6-30, and click **OK**.

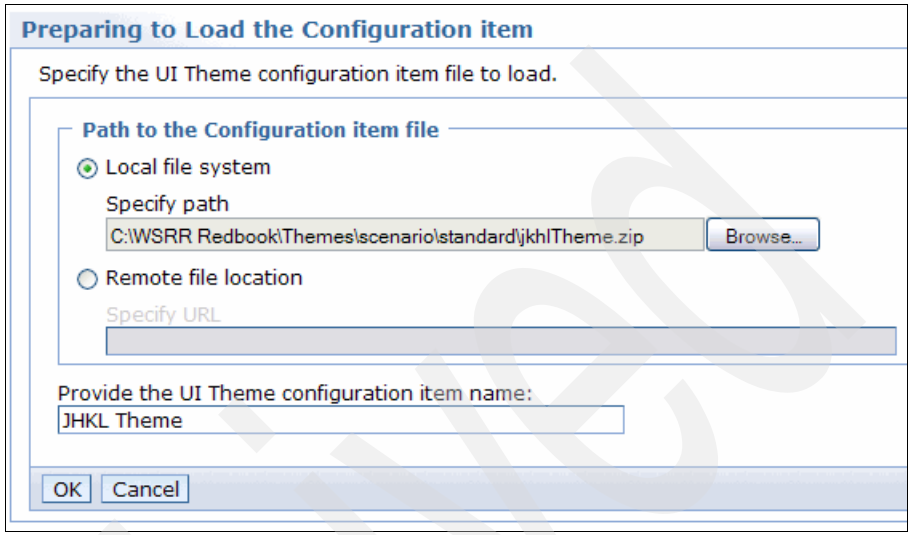


Figure 6-30 Loading the theme configuration file

Archived

Make the theme available to users

To make the theme available to for users, follow these steps:

1. Click **Active Profile** → **Web UI Configuration** → **Themes** → **User Themes**.
2. Select the JKHLE Theme, and click **Make Available**, as shown in Figure 6-31.

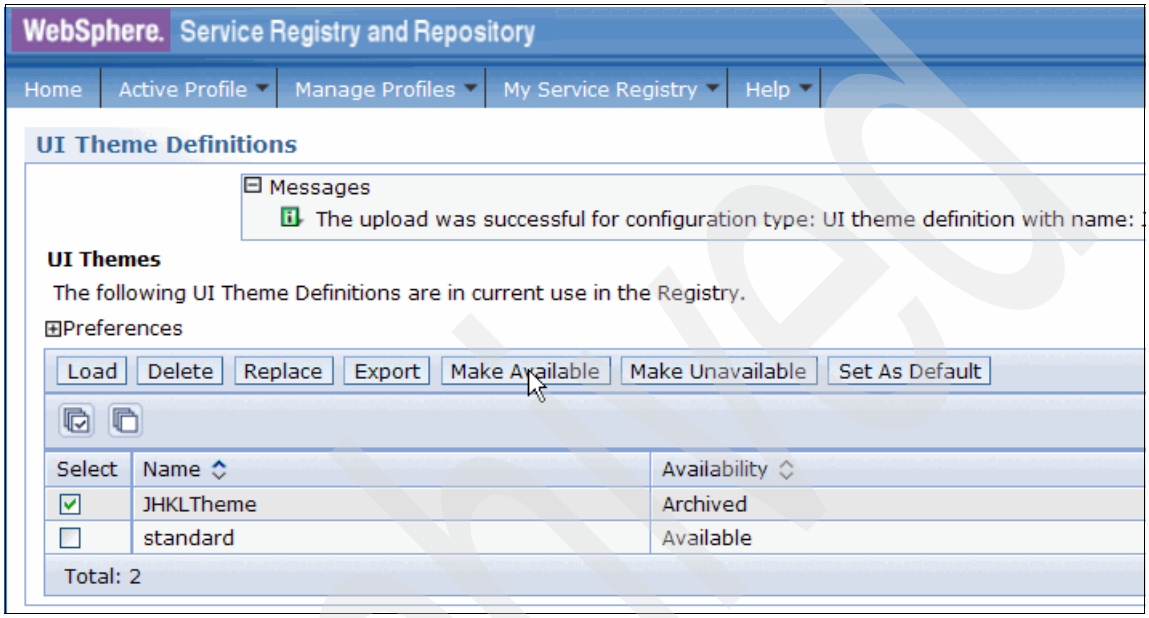


Figure 6-31 Making the theme available

You can set the theme as the default theme by selecting the theme and then clicking **Set As Default**.

Alternatively, users can a preferred theme by clicking **My Service Registry** → **My Preferences** and selecting the preferred theme from the drop-down list, as shown in Figure 6-32.

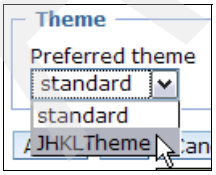


Figure 6-32 Selecting the user theme

The new JKHLE theme is now applied, as shown in Figure 6-33.

Figure 6-33 The new JKHLE theme

6.5 Troubleshooting

The WSRR Web UI validates the definition files when starting and during use. The WSRR Information Center includes a section that describes how to troubleshoot some of the validation problems that can occur:

http://publib.boulder.ibm.com/infocenter/sr/v6r3/topic/com.ibm.sr.doc/twsr_configrn_webui27.html

Archived

Archived

7

Security

This chapter describes how to secure a WebSphere Service Registry and Repository system. It includes the following topics:

- Overview of security
- WebSphere Application Server security
- WSRR security
- Implementing WebSphere Application Server security at JKHL Enterprises
- Implementing WSRR fine-grained security at JKHLE

Note: This chapter is based on WebSphere Application Server V7 security.

Although this chapter discusses some aspects of WebSphere Application Server security, it is not intended to be a complete reference for WebSphere Application Server security. For more information about WebSphere Application Server security, see:

http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/index.jsp?topic=/com.ibm.websphere.base.doc/info/aes/ae/welc6topsecuring.html

© Copyright IBM Corp. 2009. All rights reserved.

7.1 Overview of security

WebSphere Service Registry and Repository (WSRR) is a J2EE application that runs within WebSphere Application Server and that capitalizes on the security capabilities that are provided by the WebSphere Application Server run time. In particular, WSRR uses the J2EE security to guarantee that only authenticated users can access various WSRR resources. WSRR complements and extends the security provided by WebSphere Application Server with fine-grained access control of the artifacts that are held within WSRR.

7.2 WebSphere Application Server security

This section addresses security concerns that apply generally to WebSphere Application Server.

7.2.1 J2EE security

The J2EE programming model provides role-based authorization. *Role-based authorization* is often realized by URLs or method calls on an EJB component and is viewed as a coarse-grained approach to security implementation because this approach enables secure access to functional areas of the system. In a system that needs protection and that includes individual components identified as resources, role-based authorization can map resources to the J2EE roles that are permitted to access these resources.

J2EE *roles* are collections of permissions in the J2EE application and are logical, application-specific names that are mapped to users, groups, or a combination of both users and groups at deployment time. Users or a group that consists of a class of users belong to roles in the role-based authorization scheme. Based on whether users or groups of users belong to a J2EE role, access to protected resources is permitted or denied. User registries typically contain information about various groups that exist in an organization and the individual users who make up these various groups. In J2EE programming model, users or group of users are mapped to J2EE roles, either declaratively or by programming technique using an application programming interface (API).

J2EE roles are not the same as groups or a collection of class of users. Instead, J2EE roles are logical and include application-specific names on a per application basis that can be mapped to users or groups or a combination of both users and groups.

Archived

7.2.2 Administrative security

Administrative security determines whether security is used at all, the type of registry against which authentication takes place, and other values, many of which act as defaults. Administrative security can be thought of as a “switch” that activates a wide variety of security settings for WebSphere Application Server. Values for these settings can be specified, but the values do not take effect until administrative security is activated and WebSphere Application Server is restarted. The settings include the authentication of users, the use of Secure Sockets Layer (SSL), and the choice of the user account repository. In particular, application security, including authentication and role-based authorization, is not enforced unless administrative security is active.

For more information, refer to:

http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/topic/com.ibm.websphere.base.doc/info/aes/ae/csec_global.html

7.2.3 Application security

Application security enables security for the applications in your environment. This type of security provides application isolation and requirements for authenticating application users.

For more information, refer to:

http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/topic/com.ibm.websphere.nd.doc/info/ae/ae/csec_appsecurity.html

7.2.4 Administrative roles

WebSphere Application Server V7 provides eight administrative roles that provide differing degrees of authority to perform certain application server administrative functions. These roles apply to the Web-based administrative console as well as the system management scripting interface. You can map these administrative J2EE roles to users or groups.

> **Note:** To start WSRR, at a minimum, you must have WebSphere Application Server administrative permission set to the Monitor role.

Archived

Table 7-1 describes the administrative roles. For a more detailed explanation, refer to:

http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/topic/com.ibm.websphere.base.doc/info/aes/ae/rsec_adminroles.html

Table 7-1 WebSphere Application Server administrative roles

Administrative User Role	Description
Monitor	Has the least permissions. Primarily confines the user to viewing the application server configuration and current state.
Administrator	Includes Operator and Configurator permissions. Also includes permission to access sensitive data including server password, Lightweight Third Party Authentication (LTPA) password and keys, and so on.
Operator	Includes Monitor permissions. Can also change the runtime state (for example, can start or stop services).
Configurator	Includes Monitor permissions. Can also change the WebSphere Application Server configuration.
Deployer	Can complete both configuration actions and runtime operations on applications.
Admin Security Manager	Has privileges for managing users and groups from within the administrative console and determines who has access to modify users and groups using administrative role mapping. Only the administrative security manager role can map users and groups to administrative roles. By default, Admin ID is granted to the administrative security manager.
iscadmins	Has administrator privileges for managing users and groups from within the administrative console only.
Auditor (WebSphere Application Server V7 only)	Includes the Monitor persmissions. Can also view and modify the configuration settings for the security auditing subsystem.

Archived

7.2.5 Special subjects

To help administer security, WebSphere Application Server also provides the special subjects listed in Table 7-2. A *special subject* is a generalization of a particular class of users that can be mapped to users or groups. For more information, refer to:

http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/topic/com.ibm.websphere.base.doc/info/aes/ae/usec_tselugrad.html

Table 7-2 WebSphere Application Server special subjects

Special Subject	Description
None	No user is mapped to a role.
AllAuthenticated	Allows all users who can authenticate with the user registry to be mapped to a role.
Everyone	Allows all users, regardless of whether they can authenticate with the user registry, to be mapped to a role.

7.2.6 WSRR role mappings

WSRR is a J2EE application that is deployed into WebSphere Application Server and that is listed as *ServiceRegistry* (as shown in Figure 7-1).

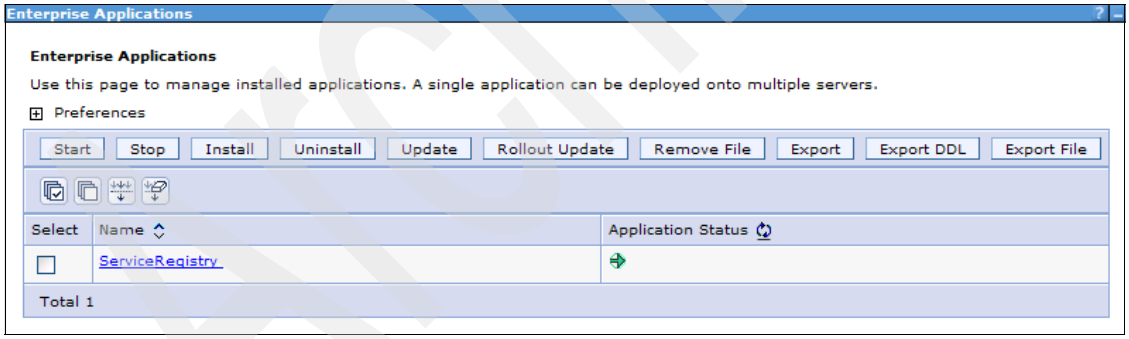


Figure 7-1 WSRR application deployed into WebSphere Application Server

Archived

As a J2EE application, WSRR defines two security roles within its deployment descriptor, as listed in Table 7-3. When application security is enabled in WebSphere Application Server, the application server makes use of these security roles to enforce decisions on which users can access WSRR and which users are denied. For users to access WSRR using the Web user interface, scripting interface, or programmatically, you need to map these users to the J2EE role of User or Administrator.

Table 7-3 WSRR J2EE roles

WSRR J2EE Role	Description
Administrator	Access to all administrative functions including any operations on the MBean.
User	Access to all non-administrative function. A common setting is to map the `AllAuthenticatedUsers` principal to the User role. You then apply further security to each WSRR user using the internal fine-grained access control mechanism.

Note: Unless specified otherwise during the WSRR installation, the default configuration maps the special subject `AllAuthenticated` to both the User and Administrator roles, enabling all users who can authenticate (with the specified user registry) full access to WSRR.

7.2.7 WSRR administrator RunAs role

Several components within WSRR need to access privileged resources at run time. For example, a number of components need to access configuration information at application startup in order to perform various initialization tasks. Other components need to perform tasks within WebSphere Application Server that can be performed only by a WebSphere administrator.

Generally speaking, access to resources within a J2EE application takes place using the credentials of the currently authenticated user. However, any EJBs and servlets in a J2EE application can specify a *RunAs* identity that is used when when the EJB or servlet makes calls to other components. WSRR makes use of this mechanism to allow the credentials of a specific user to be used whenever a component needs to access privileged resources or perform privileged tasks.

The RunAs identity that is specified in the WSRR deployment descriptors is a logical role-name. That is, it is the logical name of a J2EE role that is defined by WSRR. The J2EE role in WSRR that is authorized to access the privileged resources or perform the privileged actions is the *Administrator* role. To make credentials for a user in the Administrator role available to WebSphere

Application Server at run time, a user ID and password must be associated with the RunAs role. You can perform this association when installing WSRR, but you can also specify a user ID and password after installing WSRR using the WebSphere Administrative Console. The user who you specified must be a member of the WSRR J2EE Administrator role and must also be at least a member of the WebSphere Operator role.

For more information, refer to:

http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/topic/com.ibm.websphere.nd.doc/info/ae/ae/tsec_taurunas.html

7.3 WSRR security

This section addresses security concerns that apply specifically to WSRR.

7.3.1 Levels of authentication

Because WSRR is a J2EE application that is deployed inside WebSphere Application Server, requesters who attempt to access WSRR managed entities must be able to complete the following tasks:

- Successfully complete WebSphere Application Server authentication at the server level, supplying the credentials or token appropriate for the configured authentication mechanism and user registry.
- Successfully pass the J2EE, JACC, RBAC checks at the Web or EJB container level for the WSRR application. The requester's principal (or the principal of a group to which the requester belongs) must be mapped to a role and defined by the WSRR deployment descriptor (that is, the Administrator or User) that permits access to the relevant WSRR API method.
- Successfully pass fine-grained access control checks within the WSRR application itself.

Archived

7.3.2 Configuration properties

WSRR permits customization of its security access control framework by providing two properties:

- `com.ibm.sr.authz.property.query.checking`

  Controls whether authorization checking is performed on the results of a property query. This property can be either enabled or disabled (default).

- `com.ibm.sr.authz.default.access`

  Controls the default level of access permitted to the artifacts contained in WSRR. The possible values for this property are `all` (default) or `none`. When `all` is set for this property, access to all artifacts is permitted until explicit permission is granted to a role or roles. When an explicit permission is granted to a role or roles, only roles with explicit permission to the newly protected artefact have access. If this property is set to `none`, WSRR denies all access to all roles until explicit permission is granted to a role or roles.

For information about how to set these properties, see:

http://publib.boulder.ibm.com/infocenter/sr/v6r3/topic/com.ibm.sr.doc/cwsr_configrn_accesscontro_cust.htmle

7.3.3 Fine-grained access control

WSRR fine-grained access control is *role-based access control* (RBAC). This control is achieved by exploiting an authorization engine that is based on a common Extensible Access Control Markup Language (XACML). These fine-grained access control roles are different from the J2EE Java Authorization Contract for Containers (Java ACC) RBAC roles that are defined declaratively in the WSRR deployment descriptor. For each role that is defined in WSRR, you can add fine-grained access rules or permissions to define the entitlement of these roles.

Incoming WSRR API requests that relate to Create, Retrieve, Update, Delete, Transition, and Governance operations on WSRR artifacts trigger fine-grained access control checks from policy enforcement points. At the WSRR API policy enforcement point, before the common authorization engine is invoked to determine the fine-grained access control decision, the application server requests the `WSSubject` context for each WSRR request. This context is set up after a successful Java Authentication and Authorization Service (JAAS) login.

This information is used to set the common authorization engine Subject handler data, which identifies the requester's user principal as well as the group principal of any group to which the user belongs. When determining the fine-grained

access decision, the common authorization engine uses this Subject handler data to determine the requester's User and Group principals. The WSRR fine-grained access control determines entitlement by finding the requester's role (in the WSRR authorization principal-to-role mapping) and then applying the authorization rules for that role.

Fine-grained access rules define actions that can be performed on a set of target WSRR artifacts. Fine-grained access control supports rules for create, retrieve, update, delete (CRUD), transition, and governance operations. Each of these actions is treated as a separate authorization control domain. For example, a permission to delete an artifact does not imply permission to update the same artifact. This type of permission has to be granted explicitly. Figure 7-2 shows the different authorization domains.

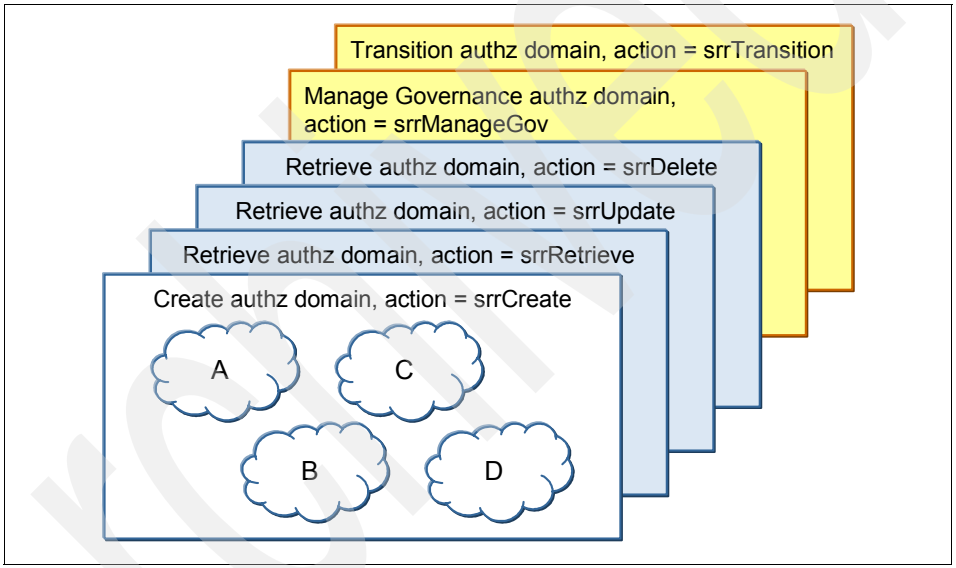


Figure 7-2 WSRR fine-grained security access control authorization model

Permissions are granted to roles. The number and names of the roles used to control access within WSRR is configurable by the user. Typically, each role that is defined maps to a group of users who are defined within the user registry (for example LDAP). Permissions are then granted to each of the defined roles as appropriate, and all users who belong to a given role have access permissions granted to that role.

The artifacts to which a particular permission applies are defined using a subset of XPath. Using this subset, you can specify sets of artifacts by type (for example WSDLDocument), by modelled and user-defined properties, and by classification. In this way, you can define sets of artifacts generically.

The governance enablement profile

The governance enablement profile (GEP) has six roles defined, as shown in Figure 7-3, which are completely independent of the Administrator and User J2EE roles that we discussed earlier. Note that the governance enablement profile is configured with the default access configuration property (`com.ibm.sr.authz.default.access`) set to `all`. Thus, access to all artifacts within the registry is unrestricted until a permission is granted to protect a particular set of artifacts. After this permission is granted, explicit permission is then required for the defined access type to the defined set of artifacts.

New Delete

Select	Role	Users	Groups	Permissions
☐	WSRRUser	0	1	3
☐	SOAGovernance	0	1	1
☐	Development	0	1	1
☐	Business	0	1	1
☐	Operations	0	1	1
☐	WSRRAdmin	0	1	1

Total: 6

Figure 7-3 Governance enablement profile roles

Permissions in the governance enablement profile restrict users in the WSRRUser role from retrieving artifacts until those artifacts are in a stable state within WSRR (that is, after the artifacts have completed the development cycle and are available for reuse). For example, users in the WSRRUser role should be able to view only online endpoints.

The governance enablement profile includes the following default permissions, as listed in Table 7-4:

- The *Retrieve Owned Entities* permission grants access to users in the WSRRUser role to retrieve artifacts that they created originally.
- The *Retrieve All Entities* permission grants access to all GenericObjects in WSRR to users in all roles except WSRRUser. Without such a permission, every user has access to all GenericObjects. After the permission is added, only roles that have that permission can retrieve GenericObjects, thus restricting access to GenericObjects to users in the WSRRUser role.
- The *Retrieve Organizations* permission grants access to users in the WSRRUser role to retrieve organization business model instances.
- The *User Visibility* permission grants access to users in the WSRRUser role to retrieve artifacts when those artifacts are in a stable state. Note that users in other roles can retrieve these artifacts also because those users have been explicitly granted retrieve permission on all GenericObjects.

Note that the governance enablement profile includes rules only for the srrRetrieve domain. Because no rules are defined for other domains and because the default access property is set to all, anyone who is authenticated at the J2EE level can create, update, delete, transition, and manage governance.

Table 7-4 Default governance enablement profile authorization rules

Name	Action (Domain)	Roles	XPath target
Retrieve Owned Entities	srrRetrieve	WSRRUser	/WSRR/GenericObject[@owner='$currentUser']
Retrieve All Entities	srrRetrieve	SOAGovernance Development Business Operations WSRRAdmin	/WSRR/GenericObject
Retrieve Organizations	srrRetrieve	WSRRUser	/WSRR/GenericObject[@primaryType='http://www.ibm.com/xmlns/prod/serviceregistry/v6r3/ALEModel#Organization']
User Visibility	srrRetrieve	WSRRUser	/WSRR/GenericObject[exactlyClassifiedByAnyOf('http://www.ibm.com/xmlns/prod/serviceregistry/lifecycle/v6r3/LifecycleDefinition#AssetApproved','http://www.ibm.com/xmlns/prod/serviceregistry/lifecycle/v6r3/LifecycleDefinition#CapabilityApproved','http://www.ibm.com/xmlns/prod/serviceregistry/lifecycle/v6r3/LifecycleDefinition#Operational','http://www.ibm.com/xmlns/prod/serviceregistry/lifecycle/v6r3/LifecycleDefinition#SLDSubscribable','http://www.ibm.com/xmlns/prod/serviceregistry/lifecycle/v6r3/LifecycleDefinition#Online','http://www.ibm.com/xmlns/prod/serviceregistry/lifecycle/v6r3/LifecycleDefinition#SLAActive')]

Viewing artifacts in the WSRR user interface

As discussed previously, the property query checking configuration setting (com.ibm.sr.authz.property.query.checking) affects the results of property queries against the specified retrieve permissions. The default governance enablement profile setting for this property is disabled. The WSRR user interface uses property queries to display the results that are included in a collection view. Thus, with the default setting of disabled, if a user in the WSRRUser role looks at a collection view of endpoints, offline as well as online endpoints display, as shown in Figure 7-4.

Archived

Figure 7-4 Collection view, as displayed to a WSRRUser showing an offline endpoint

Although the offline endpoints display, if a user selects an offline endpoint, that user is then restricted from viewing the content (as indicated in Figure 7-5). If you do not want to return artifacts for which a user does not have permission to retrieve, then enable the property query checking configuration setting.

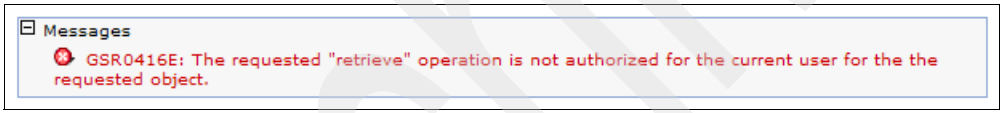


Figure 7-5 Error a when a fine-grained access control rule is violated

> **Note:** Enabling property query checking can have a performance overhead. So, use it only if absolutely necessary.

For more information about fine-grained access control, see:

http://www.ibm.com/developerworks/websphere/library/techarticles/0705_orchard/0705_orchard.html
http://publib.boulder.ibm.com/infocenter/sr/v6r3/topic/com.ibm.sr.doc/twsr_configrn_userroles_fine_grained_access.html

7.4 Implementing WebSphere Application Server security at JKHL Enterprises

Note: In the examples in this book, we use a case study about a fictional company named *JKHL Enterprises* (JKHLE). For information about this case study, see Chapter 4, “JKHL Enterprises case study” on page 153.

JKHLE configured their WebSphere Application Server with the corporate LDAP user registry. JKHLE has the following WSRR requirements for WebSphere Application Server security:

- The WebSphere Application Server administrator does not have administration access to the WSRR system.
- WSRR is available to all users who are registered in the LDAP User Registry.
- The administration functionality of WSRR is available only to those in the LDAP group *grpadmin*.

Note: You need to repeat the WebSphere Application Server configuration, as described in this section on all WebSphere Application Server instances (for example, governance, staging, pre-production, and production).

7.4.1 Granting a user the WebSphere Application Server Operator role

JKHLE does not want to grant the WebSphere Application Server administrator WSRR administration rights. As discussed previously, a WebSphere Application Server administrator needs to start WSRR in at least the Operator role. Thus, JKHLE wants to grant a selected LDAP user to start WSRR in the WebSphere Application Server Operator role and has defined the user *WSRRSuper* in LDAP for this purpose.

To map the Operator role to the WSRRSuper user principal, perform the following steps using the WebSphere Application Server administrative console:

1. In the WebSphere Application Server navigation tree, click **Users and Groups**. Click **Administrative user roles**. Then, click **Add**.
2. In the Administrative user roles panel shown in Figure 7-6, select **Operator** from the Role(s) list. Then, specify `WSRRSuper` in the “Search string” input field, and click **Search**. When WSRRSuper is returned in the Available list box,

select **WSRRSuper**, and click the top arrow to move it to the “Mapped to role” list box.

3. Click **OK**. Then, click **save**.

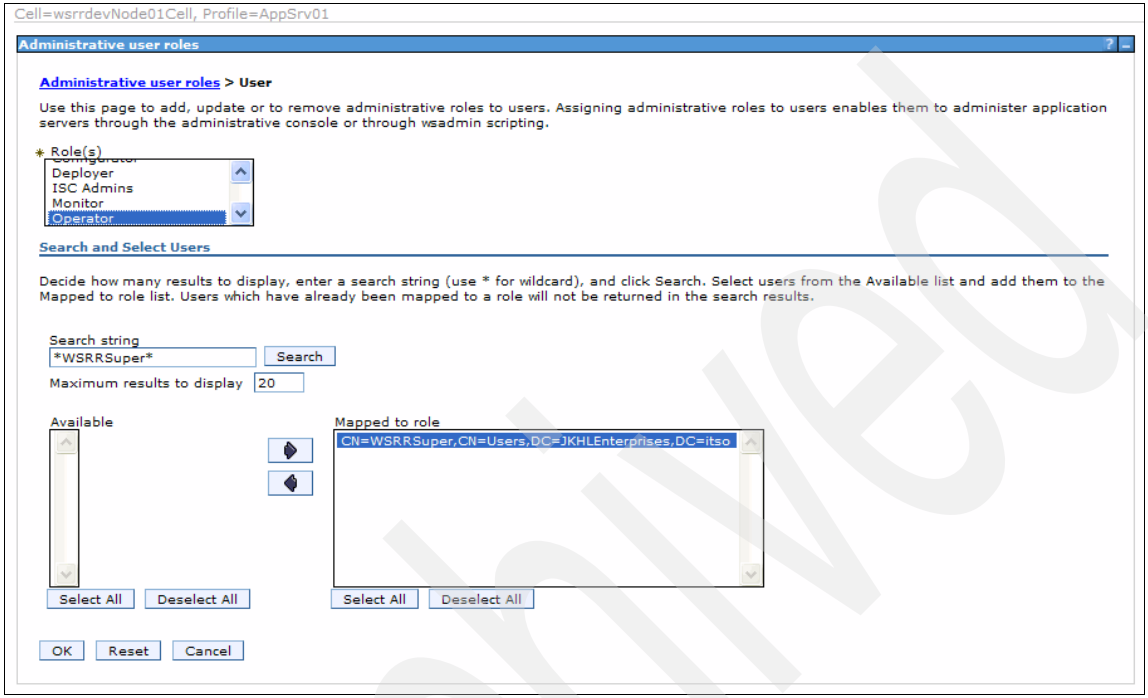


Figure 7-6 Adding WSRRSuper to the Operator role

4. Confirm that WSRRSuper is added to the Operator J2EE role as shown in Figure 7-7.

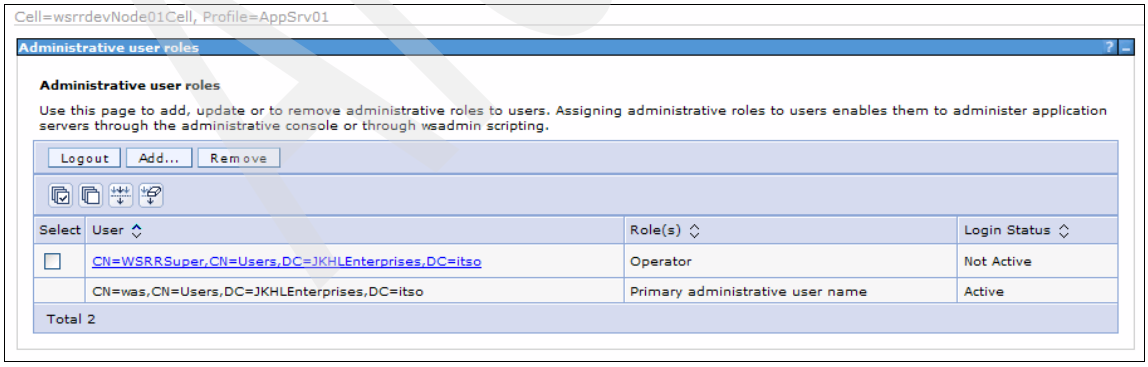


Figure 7-7 WSRRSuper added to the Operator J2EE role

Archived

7.4.2 Mapping the LDAP group to the WSRR Administrator role

JKHLE mapped WSRRSuper to the WSRR Administrator J2EE role. They also want to map the LDAP group *grpadmin* to the same role so that users in this LDAP group can administer the WSRR system. WSRR users who are not in *grpadmin* will not have administrator privileges. JKHLE wants to allow anyone registered in the LDAP User Registry access to WSRR.

To map the LDAP group, follow these steps from the WebSphere Application Server administration console:

1. Click **Applications** in the navigation tree.
2. Click **Application Types**.
3. Click **WebSphere enterprise applications**.
4. Click **ServiceRegistry** in the returned list of Enterprise Applications.
5. Click **Security role to user/group mapping**.
6. Configure the User role to the special subject *All Authenticated* if it is not already configured. Select **User**. Then, click **Map Special Subjects** and click **All Authenticated in Application's Realm**.
7. Configure WSRRSuper and grpAdmin to the Administrator role as follows:
   a. Remove any existing special subjects by selecting **Administrator**. Then, click **Map Special Subjects**, and click **None**.
   b. Add WSRRSuper to the Administrator role by selecting **Administrator**. Click **Map Users**. Specify `WSRRSuper` in the "Search string" input field. Click **Search**. When the WSRRSuper is returned in the Available list box, select **WSRRSuper** and click the top arrow to move it to the Selected list box. Click **OK**.
   c. Add the *grpadmin* group to the Administrator role by selecting **Administrator**. Then, click **Map Groups**, and specify `grpadmin` in the Search string input field. Click **Search**. When the grpadmin user is returned in the Available list box, select **grpadmin**, and click the top arrow to move it to the Selected list box. Click **OK**.

Archived

8. Confirm the mapping as shown in Figure 7-8.

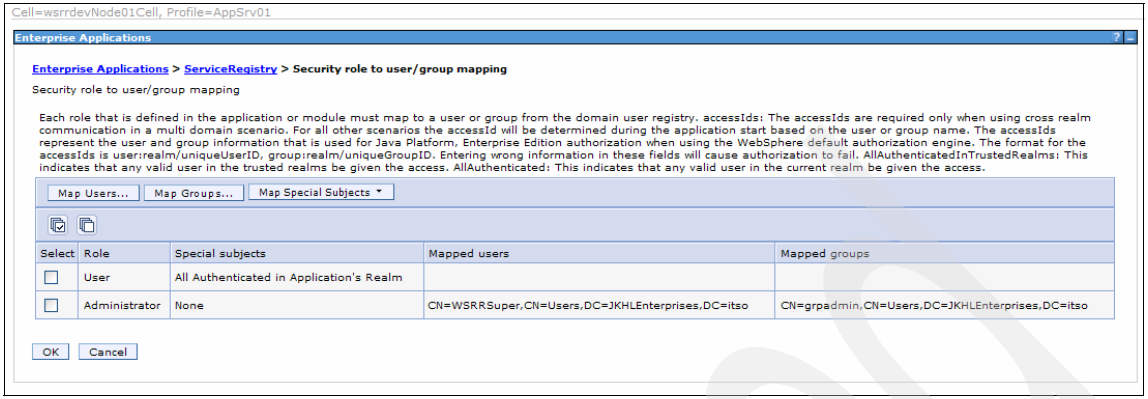


Figure 7-8 WSRR security role to user and group mapping

9. Click **OK**, and then click **Save**.

7.4.3 Configuring the WSRR Administrator RunAs role

To configure the WSRR Administrator RunAs role, follow these steps in the WebSphere Application Server administration console:

1. Click **Applications** in the navigation tree.
2. Click **Application Types**.
3. Click **WebSphere enterprise applications**.
4. Click **ServiceRegistry** in the returned list of Enterprise Applications.
5. Click **User RunAs roles**.

Archived

6. In the Enterprise Application panel, shown in Figure 7-9, enter `WSRRSuper` in the username field, and enter the password for WSRRSuper in the password field. Select the **Administrator** role and then click **Apply**.

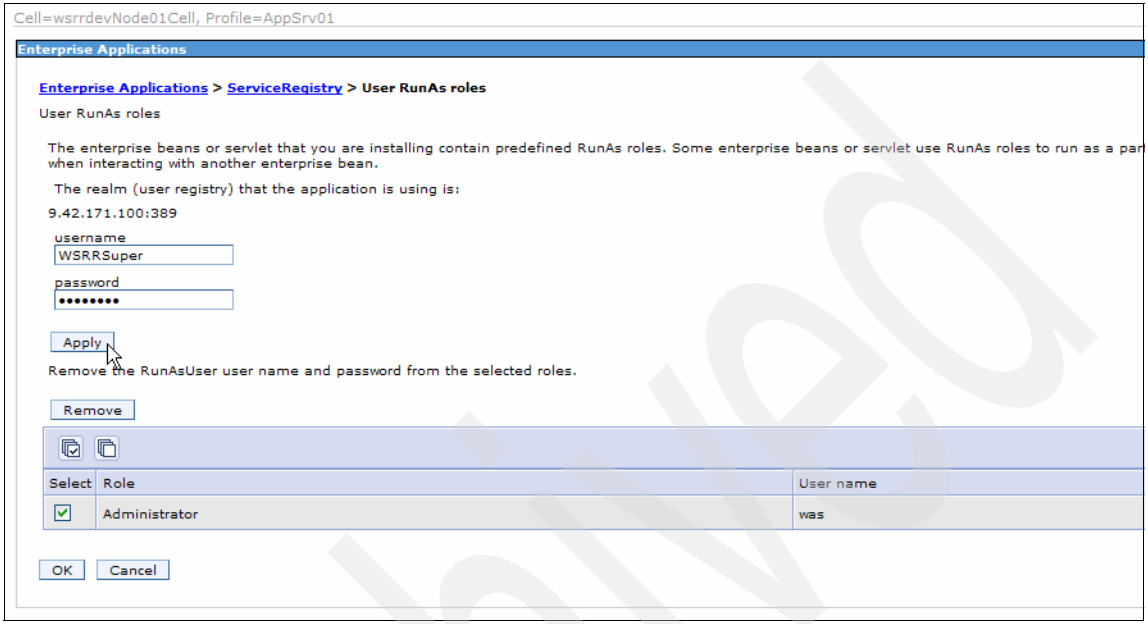


Figure 7-9 User RunAs role mapping

7. Confirm that the user name that is associated with the Administrator RunAs role is now WSRRSuper.
8. Click **OK**, and then click **Save**.

Archived

7.5 Implementing WSRR fine-grained security at JKHLE

JKHLE has the following requirements for their WSRR security:

- Each of the governance enablement profile roles are mapped to at least one LDAP group to ease maintenance so that users can be added and removed from specific LDAP groups and the changes are reflected automatically in WSRR. JKHLE wants to map WSRR roles to the LDAP users and group as shown in Table 7-5.

Table 7-5 WSRR role to group and user principal mappings

WSRR Role	Group Name	User Name
WSRRAdmin	grpadmin	WSRRSuper
Business	grpbusiness	
Development	grpdevelopment	
SOAGovernance	grpsoa	
Operations	grpoperations	
WSRRUser	AllAuthenticated	

- Users authenticated in the WSRRUser role cannot create, update, or delete artifacts in WSRR; however, they can create, update, and their own saved searches.

7.5.1 Mapping users and groups to roles in WSRR

This section discusses:

- The WSRRAdmin role
- The Business, Development, Operations, and SOA Governance roles
- The WSRRUser role
- Manage Roles table

The WSRRAdmin role

To map the WSRRAdmin role, follow these steps:

1. Log in to the WSRR console as someone in the WSRR J2EE Administrator role (that is, either WSRRSuper or a user in the grpadmin group).
2. Hover over the current WSRR Perspective, and click **Configuration** in the drop-down menu.

Archived

3. Select **Active Profile** → **Access Control** → **Roles**, as shown in Figure 7-10.

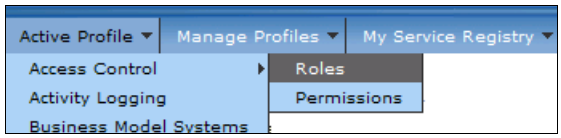


Figure 7-10 Selecting the Roles option

4. Click the number in the cell that relates to WSRRAdmin and Users, as shown in Figure 7-11.

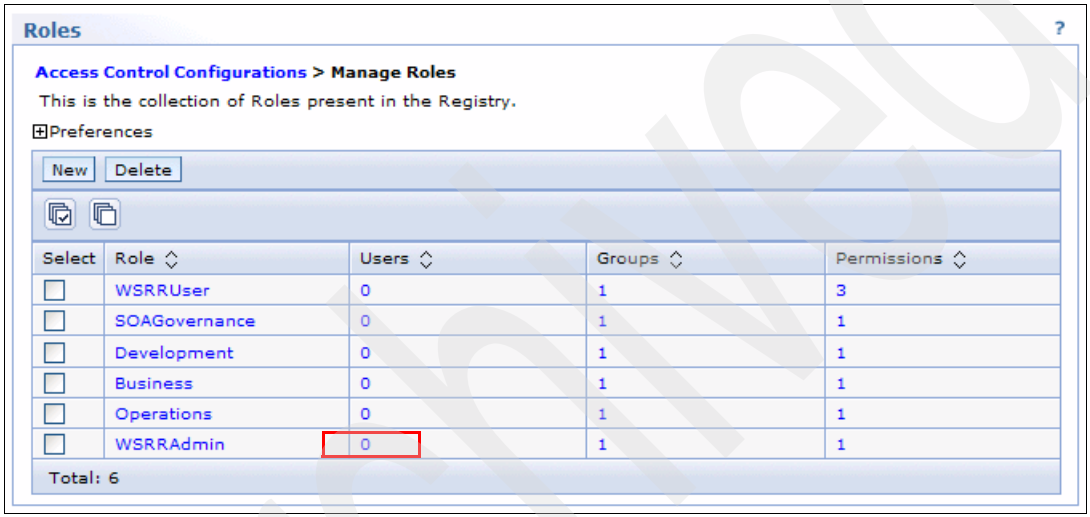


Figure 7-11 Manage roles

5. Enter `WSRRSuper` in the Search input field. Note that *Include* should be set to *Users*.
6. Click **Search**. When WSRRSuper is returned in the Users/Groups list box, select **WSRRSuper**, and click **Add**. WSRRSuper is then listed in the Selected Users list box.
7. Select **Groups** from the Include drop-down menu.
8. Enter `grpadmin` in the Search input field. Then, click **Search**. When the grpadmin group is returned in the Users/Groups list box, select the grpadmin group, and then click **Add**. The grpadmin group is now listed in the Selected Groups list box.
9. If an AllAuthenticatedUsers entry exists in the Selected Groups list box, select **AllAuthenticatedUsers**, and click **Remove.**

Archived

10. Verify that the WSRRAdmin Users/Groups settings are as shown in Figure 7-12. Then, click **OK**.

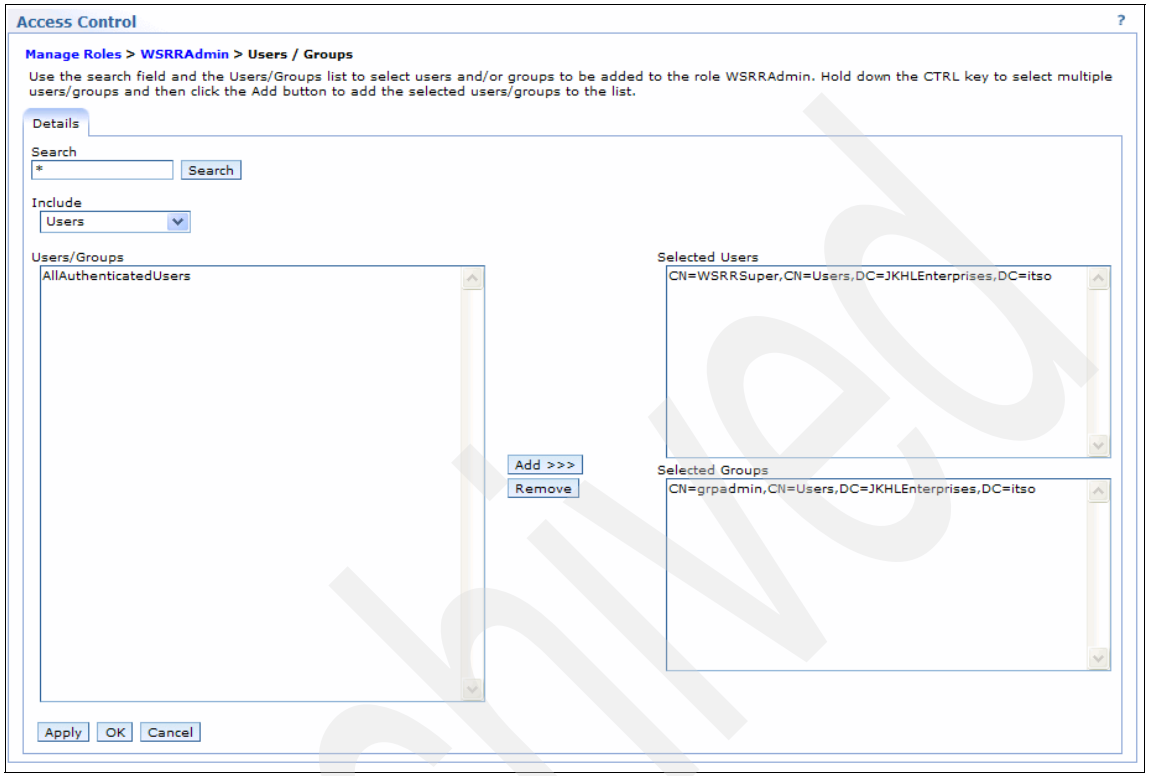


Figure 7-12 Mapping fine-grained security access control role WSRRAdmin

The Business, Development, Operations, and SOA Governance roles

Repeat the steps described in “The WSRRAdmin role” on page 280 for the Business, Development, Operations, and SOA Governance roles. Table 7-5 on page 280 lists the User/Group names for these roles.

Archived

The WSRRUser role

JKHLE wants to grant the WSRRUser role to all users who are registered in their corporate LDAP. To do so, follow these steps:

1. Click **Active Profile** → **Access Control** → **Roles**.
2. Click the number in the cell that relates to WSRRUser and Groups.
3. Confirm that the Selected Groups list includes the special subject *AllAuthenticatedUsers* (as shown in Figure 7-13). If this subject is not included, select the AllAuthenticatedUsers special subject, and click **Add**. The AllAuthenticatedUsers subject is not listed the Selected Groups list box. Click **OK**.

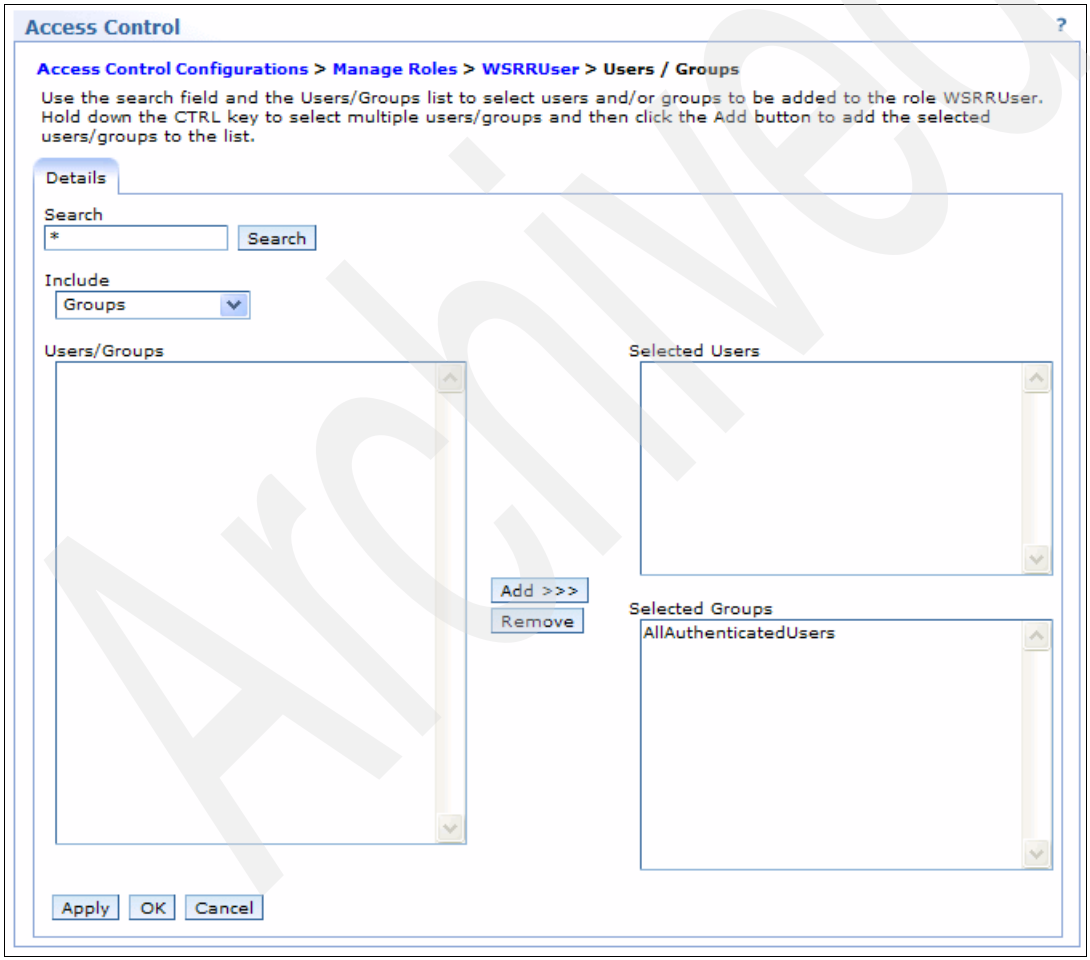


Figure 7-13 Access control users and groups

Archived

Manage Roles table

After mapping the WSRR governance enablement profile fine-grained security access control roles to LDAP user registry, JKHLE's Manage Role table is configured as shown in Figure 7-14.

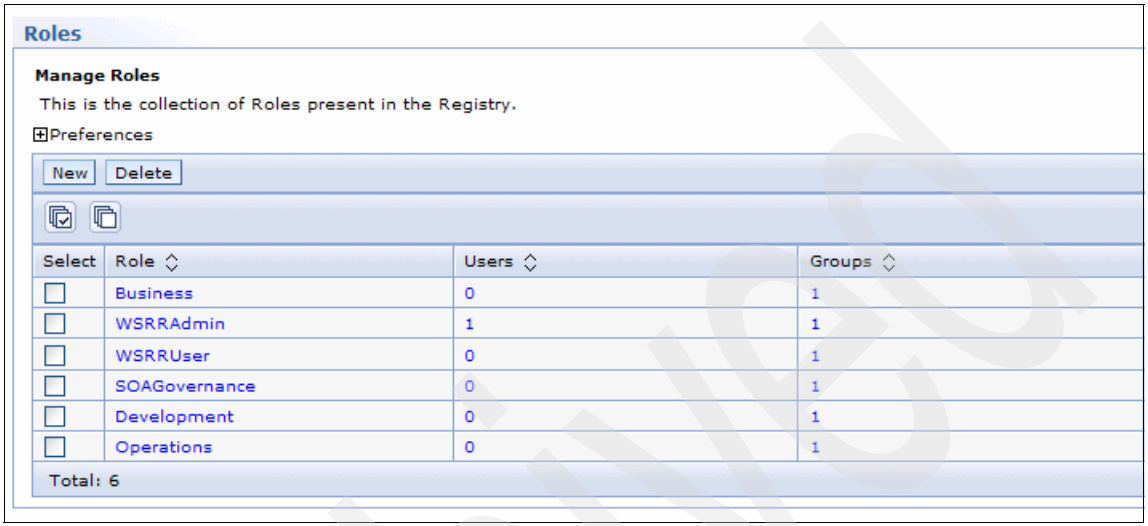


Figure 7-14 Summary of WSRR governance enablement profile fine-grained role mappings

7.5.2 Mapping permissions to roles in WSRR

JKHLE wants to make the WSRRUser role a read-only perspective into WSRR. They do no not want users in that role to be able to create, update, delete, or change the governance of artifacts in WSRR. The easiest way to achieve this level of permissions is to restrict objects at the BaseObject level because all artifacts in WSRR inherit permissions from the BaseObject level. However, JKHLE does want users in this role to be able to create, update, and delete saved searches and subscriptions. Thus, they need to configure security rules as described in Table 7-6.

Archived

Table 7-6 Additional authorization rules for JKHLE

Name	Action (Domain)	Roles	XPath target
Create BaseObject	srrCreate	SOAGovernance Development Business Operations WSRRAdmin	`/WSRR/BaseObject`
Create PropertyQuery	srrCreate	WSRRUser	`/WSRR/PropertyQuery`
Create Subscription	srrCreate	WSRRUser	`/WSRR/Subscription`
Delete BaseObject	srrDelete	SOAGovernance Development Business Operations WSRRAdmin	`/WSRR/BaseObject`
Delete PropertyQuery	srrDelete	WSRRUser	`/WSRR/PropertyQuery[@owner='$currentUser']`
Delete Subscription	srrDelete	WSRRUser	`/WSRR/Subscription[@owner='$currentUser']`
Update BaseObject	srrUpdate	SOAGovernance Development Business Operations WSRRAdmin	`/WSRR/BaseObject`
Update PropertyQuery	srrUpdate	WSRRUser	`/WSRR/PropertyQuery[@owner='$currentUser']`
Update Subscription	srrUpdate	WSRRUser	`/WSRR/Subscription[@owner='$currentUser']`
Manage Governance BaseObject	srrManageGov	SOAGovernance Development Business Operations WSRRAdmin	`/WSRR/BaseObject`

Archived

Create BaseObject rule

To create the BaseObject rule, follow these steps:

1. Log in to the WSRR console as someone in the WSRR J2EE Administrator role (that is, either WSRRSuper or a user in the grpadmin group).
2. Hover over the current WSRR Perspective and click **Configuration** in the drop-down menu.
3. Click **Active Profile** → **Access Control** → **Roles**.
4. Click the number in the cell that relates to WSRRAdmin and Permissions as shown in Figure 7-15.

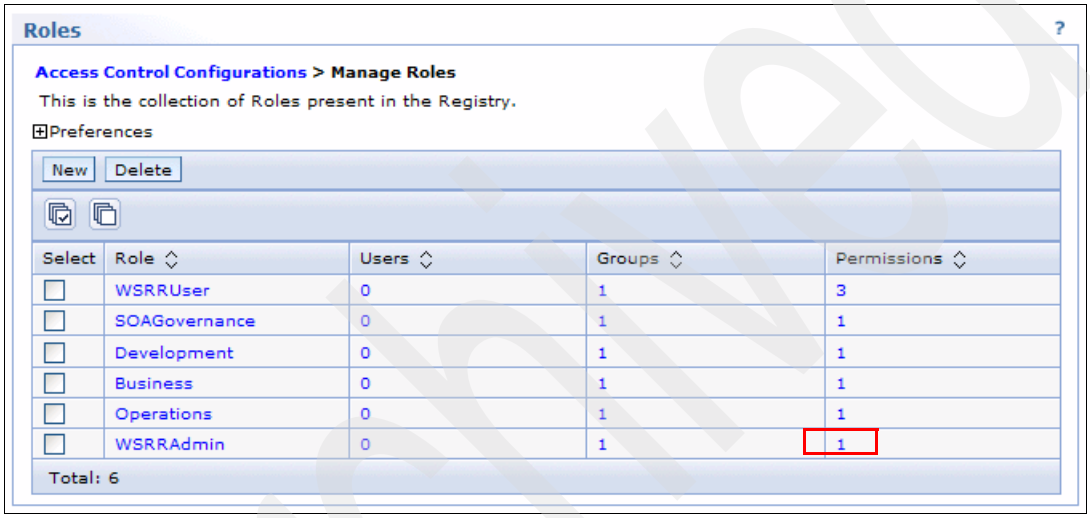


Figure 7-15 Manage roles

Archived

5. Click **Create a New Permission**.
6. To create a new permission, enter `Create BaseObject` in the Name input field. Leave the Permission Action as `Create`, and leave the Permission Target (XPATH) as `/WSRR/BaseObject`. Click **OK**.

   The permissions are now listed as shown in Figure 7-16.

Create a New Permission | Add an Existing Permission | Copy to another Role | Delete

Select	Permission Name	Action	XPATH target
☐	Create BaseObject	srrCreate	/WSRR/BaseObject
☐	Retrieve all entities	srrRetrieve	/WSRR/GenericObject

Total: 2

Figure 7-16 Permissions

7. Select the **Create BaseObject** permission. Then, click **Copy to another Role**.
8. Select the **SOAGovernance**, **Development**, **Business** and **Operations** roles, and click **Select Roles**.

Delete PropertyQuery

To delete the PropertyQuery, follow these steps:

1. Log in to the WSRR console as someone in the WSRR J2EE Administrator role (that is, either WSRRSuper or a user in the grpadmin group).
2. Hover over the current WSRR Perspective, and click **Configuration** in the drop-down menu.
3. Click **Active Profile** → **Access Control** → **Roles**.
4. Click the number in the cell that relates to WSRRUser and Permissions (Figure 7-15 on page 286).
5. Click **Create a New Permission**.
6. Then, enter `Delete PropertyQuery` in the Name input field. Select **Delete** from the Permission Action drop-down menu.
7. Enter `/WSRR/PropertyQuery[@owner='$currentUser']` in the Permission Target (XPATH) input field.
8. Click **OK**.

Repeat this process to configure the remaining permissions as listed in Table 7-6 on page 285.

Archived

8

Promotion

This chapter introduces the promotion features of WebSphere Service Registry and Repository (WSRR), the types of promotion modes that WSRR reports, and how to filter what is promoted. It also includes the promotion topologies that WSRR supports.

This chapter includes the following topics:

- Overview of the WSRR promotion feature
- Promotion methods
- Implementing promotion at JKHLE
- Editing the sample configuration file
- Troubleshooting

Note: WSRR APAR IZ59696 and WSRR V6.3 Fix Pack 1 is required for the promotion features to work correctly.

© Copyright IBM Corp. 2009. All rights reserved.

8.1 Overview of the WSRR promotion feature

The WSRR promotion feature provides a mechanism for pushing WSRR artifacts from a single WSRR instance into multiple WSRR instances. With the promotion feature, you can manage artifacts that relate to multiple environment topologies (for example test and production) in a central WSRR instance, commonly known as the *governance registry*. You can configure WSRR to promote (copy) an artifact that is transitioned from one state to another in its life cycle and to promote all of that artifact's dependents as well, including the metadata and governance states, to a specific WSRR instance.

Figure 8-1 shows an example topology.

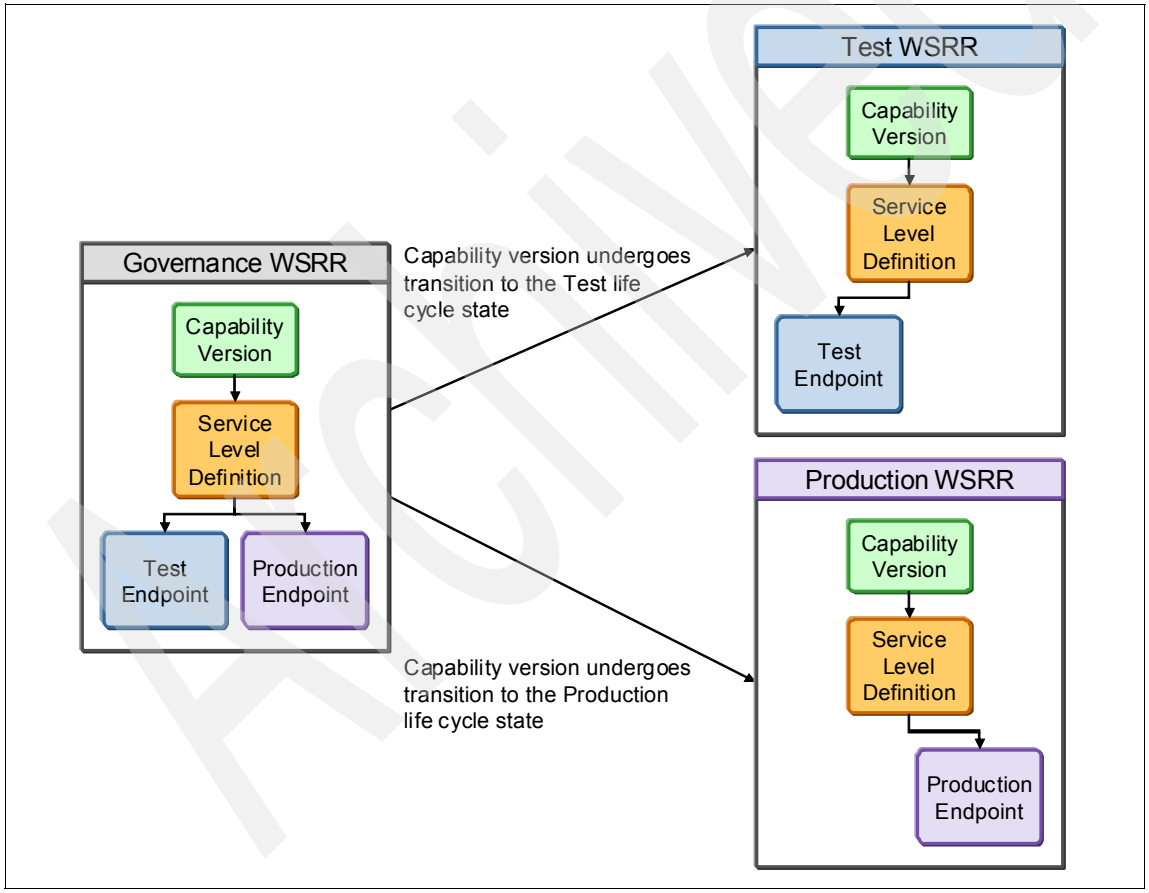


Figure 8-1 Example WSRR topology

This example topology includes the following components:

- A central governance WSRR instance where all WSRR artifacts are loaded, managed, and governed
- A test WSRR instance where test-related artifacts are promoted when a certain point in the life cycle is met
- A production WSRR instance where production-related artifacts are promoted when a certain point in the life cycle is met

8.2 Promotion methods

You can configure a promotion method synchronously, asynchronously, or manually. Each of these methods has its own advantages, which we will discuss later in this section. For the purpose of this discussion, we consider a *ServiceVersion* that is going through the SOA life cycle and that is configured to be promoted to a *Test WSRR* instance on the *ApproveStagingDeployment* transition (as illustrated in Figure 8-2).

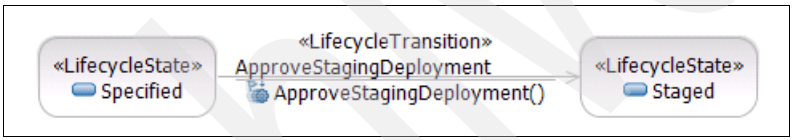


Figure 8-2 Approve Staging Deployment transition

8.2.1 Synchronous promotion

Synchronous promotion uses the same transaction context to update both the governance and target WSRR instance. In the scenario described previously, when an ApproveStagingDeployment transition is initiated on the ServiceVersion, an attempt is made to promote the ServiceVersion and its dependents into the Test WSRR instance. If this attempt fails, the transaction is rolled back in the governance registry where the process was initiated, and the ServiceVersion is returned to the Specified state.

Advantage

The main advantage of synchronous promotion is that the governance registry and target registry are guaranteed to be synchronized after the promotion operation completes, whether the promotion succeeds with a commit or fails with a rollback.

Considerations

If you configure promotion to promote to two or more target runtimes in the same transition, note that the promotion process must complete on all of the target instances. Otherwise, the promotion is rolled back in every instance.

Also, note that synchronous promotion uses RMI over IIOP to communicate between the governance registry and the target registries. Therefore, you can use this method where there is a network connection and where the appropriate ports are opened.

8.2.2 Asynchronous promotion

Asynchronous promotion is performed at a later time, as part of a scheduled task. When the artifact is transitioned, the transition succeeds immediately, and a JMS message is sent to `jms/TransitionSuccessTopic`. When the scheduled task is executed sometime later, the information about the artifact is read from the JMS topic, and the object is promoted in the same manner as in synchronous promotion.

> **Note:** The state of the object is taken from the registry at the time that the scheduled task runs. The state of the artifact is not stored at the time it is transitioned.

Advantage

The transition in the governance registry is not rolled back if the promotion fails. However, in this case, you can configure a failure transition to effectively roll back the transition in the event of promotion failure and, therefore, provide feedback that the promotion failed.

Considerations

Asynchronous promotion is not recommended. However, if asynchronous promotion is needed, consider the following issues:

- The governance and target WSRR instances will not be synchronized between the time when the transition occurs and when the scheduled task is executed.
- The artifact that is transitioned, its metadata, and any other artifacts and their metadata in the graph might be updated between when the transition takes place and the scheduled task being run. Thus, the updated artifacts are promoted into the target registries, rather than a snapshot of when they were transitioned.

Archived

- If promotion is configured to promote to two or more target runtimes in the same transition, the promotion process must complete on all the target instances. Otherwise, the promotion is rolled back to every target instance. If configured, the governance registry transitions the artifact using the failure transition.
- As with synchronous promotion, asynchronous promotion also uses RMI over IIOP to communicate between the governance registry and the target registries. Therefore, use this method only where there is a network connection and where the appropriate ports are opened.

8.2.3 Manual promotion to a file

With *manual* promotion, content is written to a specified file at a configured file path location on the governance registry. In the scenario presented previously, when the ServiceVersion goes through the ApproveStagingDeployment transition, a promotion file is written to the file system on the governance registry, the transition succeeds immediately, and the ServiceVersion is updated to the Staged state. You must import the promotion file manually to the target WSRR instance or instances.

Advantage

You can use manual promotion when the governance registry cannot communicate directly with the target WSRR instances. Thus, manual promotion is required if there is no network connection or if there are firewall restrictions between the governance and runtime registries.

Considerations

The main issue with using manual promotion is that when an artifact is transitioned in the governance registry, a promotion file is created on the local file system. The artifact in the governance registry is transitioned successfully to the new state; however, the target registries are still in the old state. Thus, the target registries are not synchronized with the governance registry for an indefinite period of time until the promotion file is loaded into the target runtime or runtimes. Manual promotion, as the name implies, requires manual intervention and is reliant on a human task.

8.2.4 Promotion filtering by classification

Promotion filtering helps promote a subset of objects in the collection of the total governed objects to the target environment. Figure 8-3 shows a capability version in the governance registry that contains multiple endpoints that are classified according to their environment (such as Test and Production).

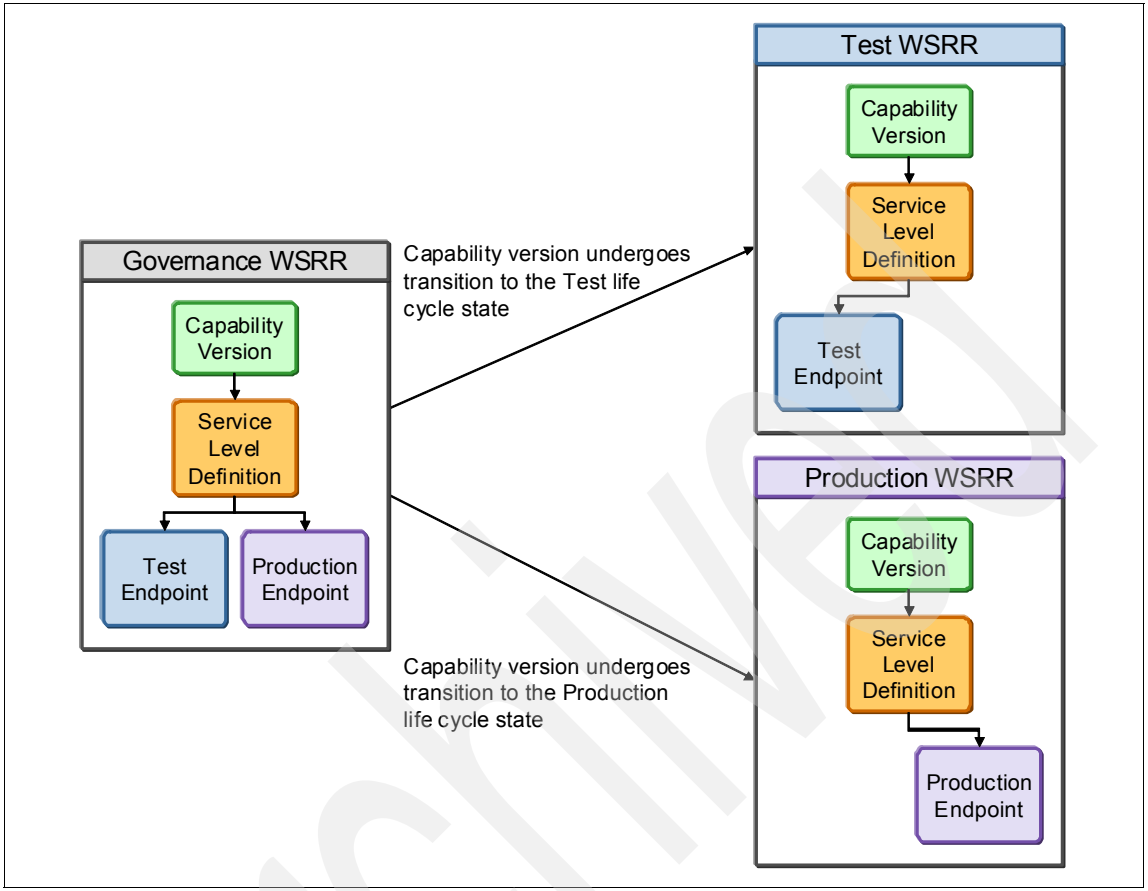


Figure 8-3 Promotion filtering

When the capability version is promoted to the Test life cycle state, it is desirable that only the Test endpoint is promoted to the Test WSRR instance. The same is true for the Production endpoint when the capability version is transitioned to the Production life cycle state. This behavior can be achieved by classifying the endpoint according to the target environment.

> **Note:** If an object is not classified according to an environment, it is assumed that the object is required in all environments.

For more information about classification filtering, refer to the WSRR Information Center:

http://publib.boulder.ibm.com/infocenter/sr/v6r3/topic/com.ibm.sr.doc/rwsr_promotion_filtering_addingclassif.html

8.3 Implementing promotion at JKHLE

Note: In the examples in this book, we use a case study about a fictional company named *JKHL Enterprises* (JKHLE). For information about this case study, see Chapter 4, “JKHL Enterprises case study” on page 153.

JKHLE wants to configure synchronous promotion between the governance and runtime registries. The JKHLE deployment topology features one central governance WSRR instance and three runtime WSRR instances in the form of a staging, preproduction, and production environment (as illustrated in Figure 8-4).

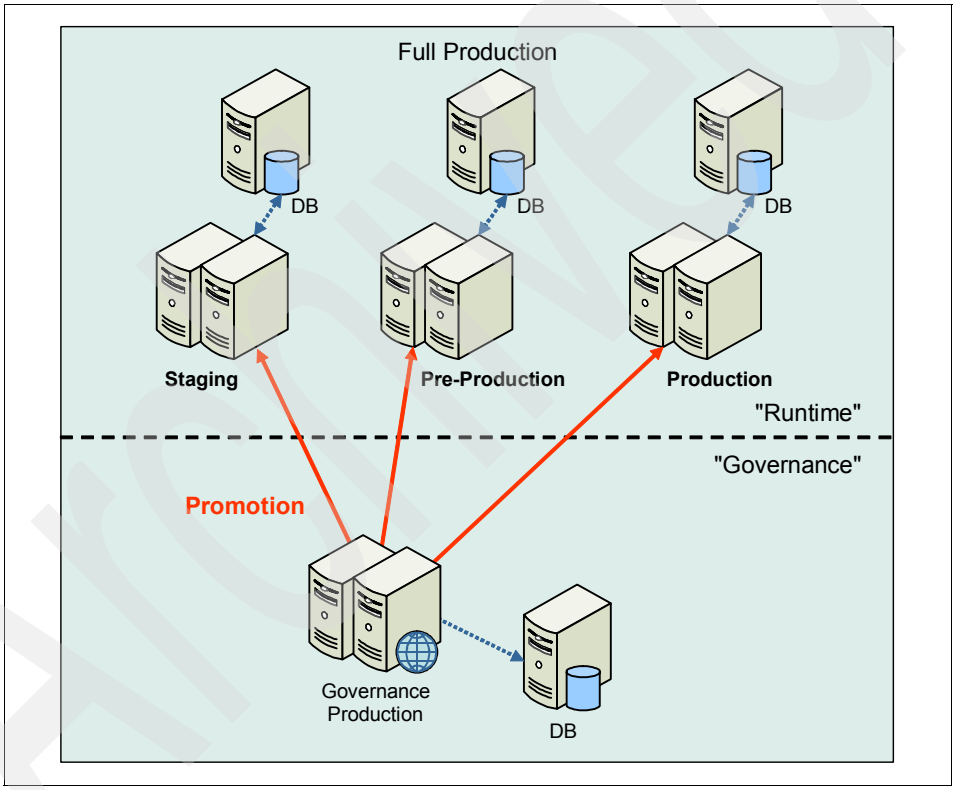


Figure 8-4 Deployment topology

In addition to promoting the ServiceVersion and its dependents when it goes through an appropriate life cycle transition, JKHLE also wants the runtime registries to update the Endpoints when they are transitioned from online to offline and vice versa.

Table 8-1 describes JKHLE's required configuration.

Table 8-1 Promotion configuration

Artifact	Transition	Target Environment
Service Version	http://www.ibm.com/xmlns/prod/servicereg istry/lifecycle/v6r3/LifecycleDefiniti on#**ApproveStagingDeployment**	http://www.ibm.com/xmlns/prod/servicereg istry/6/1/GovernanceProfileTaxonomy# **Staging**
Service Version	http://www.ibm.com/xmlns/prod/servicereg istry/lifecycle/v6r3/LifecycleDefiniti on#**ApproveCertification**	http://www.ibm.com/xmlns/prod/servicereg istry/6/1/GovernanceProfileTaxonomy# **Pre-Production**
Service Version	http://www.ibm.com/xmlns/prod/servicereg istry/lifecycle/v6r3/LifecycleDefiniti on#**ApproveProductionDeployment**	http://www.ibm.com/xmlns/prod/servicereg istry/6/1/GovernanceProfileTaxonomy# **Production**
EndPoint	http://www.ibm.com/xmlns/prod/servicereg istry/lifecycle/v6r3/LifecycleDefiniti on#**ApproveForUse**	http://www.ibm.com/xmlns/prod/servicereg istry/6/1/GovernanceProfileTaxonomy# **Staging** http://www.ibm.com/xmlns/prod/servicereg istry/6/1/GovernanceProfileTaxonomy# **Pre-Production** http://www.ibm.com/xmlns/prod/servicereg istry/6/1/GovernanceProfileTaxonomy# **Production**
EndPoint	http://www.ibm.com/xmlns/prod/servicereg istry/lifecycle/v6r3/LifecycleDefiniti on#**RevokeFromUse**	http://www.ibm.com/xmlns/prod/servicereg istry/6/1/GovernanceProfileTaxonomy# **Staging** http://www.ibm.com/xmlns/prod/servicereg istry/6/1/GovernanceProfileTaxonomy# **Pre-Production** http://www.ibm.com/xmlns/prod/servicereg istry/6/1/GovernanceProfileTaxonomy# **Production**

Archived

8.4 Editing the sample configuration file

We supply a sample promotion configuration file with this book. Before you can use this file, you need to change the environment information in the file as follows:

1. Open the `PromotionProperties.xml` file in a text editor.
2. Update the following environment properties within the configuration file for each of the staging, preproduction, and production environments:
   - security enabled (Set to false if security is off)
   - wsrrUser
   - wsrrPassword
   - server name
   - port

   For a more detailed explanation of these properties, refer to:

   http://publib.boulder.ibm.com/infocenter/sr/v6r3/topic/com.ibm.sr.doc/rwsr_promotion_config_environments.html

Archived

To edit the sample configuration file, follow these steps:

1. Log in to the WSRR console as someone in the WSRR J2EE Administrator role (that is, either *WSRRSuper* or a user in the *grpadmin* group).
2. Hover over the current WSRR Perspective, and click **Configuration** in the drop-down menu.
3. Select **Active Profile** → **Promotion**, as shown in Figure 8-5.

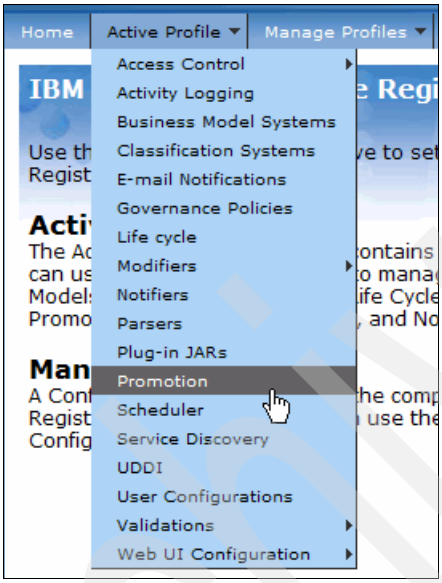


Figure 8-5 Selecting the Promotion Configuration page

You can either replace the `PromotionProperties` configuration file or edit it from within the WSRR user interface. To edit the file, click **PromotionProperties**. Alternatively, you can export the file, edit it in an external tool, and then replace it in WSRR.

Archived

4. Select **PromotionProperties** and click **Replace Promotion Properties** as shown in Figure 8-6.

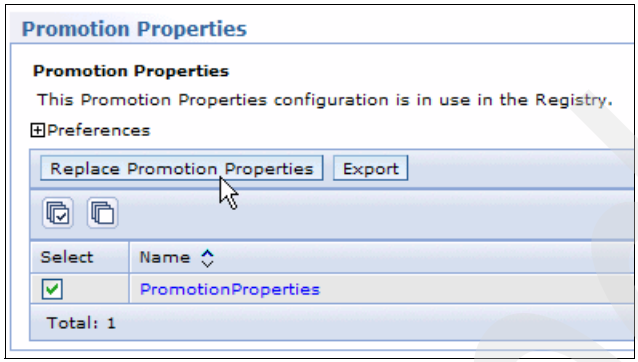


Figure 8-6 Replacing the promotion configuration file

5. Click **Browse**, navigate to the sample promotion configuration file, and click **Open**. Click **OK** to replace the file.

8.5 Troubleshooting

Note that the promotion process changes the creation time stamp of an entity from its original value to the current date and time in the destination WSRR instance.

We recommend that you disable the correlator modifier for artifacts that are promoted from a governance registry to a target registry. You can disable the correlator modifier by specifying the class `SMCorrelatorModifierDisableOnImport` rather than `SMCorrelatorModifier` in the modification properties configuration file on the target registry. For more information see:

http://publib.boulder.ibm.com/infocenter/sr/v6r3/topic/com.ibm.sr.doc/twsr_correlator_modifier_activate.html

When performing a WSRR promotion capability configuration on a single host, set a custom property `com.ibm.websphere.orb.uniqueServerName` to be `true` for the JVMs of both the source and target application servers that are hosting WSRR. Refer to WSRR Information Center for more details:

http://publib.boulder.ibm.com/infocenter/sr/v6r3/topic/com.ibm.sr.doc/rwsr_promotion_limitations_testsingle.html

Archived

To avoid transaction timeouts during promotion process, configure a longer transaction timeout value in for WSRR hosting the WebSphere Application Server instance. Refer to the WSRR Information Center for more details:

http://publib.boulder.ibm.com/infocenter/sr/v6r3/topic/com.ibm.sr.doc/cwsr_planning_install16_soap_local.html

Archived

9

Policies

One of the key goals of service-oriented architecture (SOA) is to provide policy-driven interactions between services. These policies can lead to business agility and faster time to value because you can change behavior by modifying the applicable policies. However, to maintain a compliant solution, the policies themselves must be managed as closely as the service implementations.

This chapter describes the capabilities provided by IBM WebSphere Service Registry and Repository (WSRR) in support of policy management, policy authoring, policy life cycle governance, and policy enforcement tasks. It includes details about the following topics:

- Overview of policy management in WSRR
- Applying policy to the JKHLE business scenario
- Implementing policy
- References

© Copyright IBM Corp. 2009. All rights reserved.

9.1 Overview of policy management in WSRR

Policies specify the requirements that must be met so that a Web service can be consumed by a client. For example, a Web service might require that all messages are digitally signed and encrypted; in this example, the requirement for signature and encryption is the policy and the Web service itself is referred to as the subject of the policy; the entity to which the policy applies.

A policy-driven approach to an SOA offers a way to represent a variety of requirements, including business, technical, and operational requirements, into expressions that can be interpreted and acted upon throughout the life cycle of a service. Expressing requirements or capabilities in a self-documenting, human readable form or automated machine enforceable directives, enables SOA to be more consumable, manageable, and traceable. Policies also foster business agility and quicker time-to-value because modifying policies is easier than modifying services or consuming applications.

In SOA solutions featuring policy-driven iterations, the policies themselves must be managed for solutions to be consistent and compliant. WSRR provides a set of key capabilities to support effective policy management for both design and runtime related policies. WSRR also provides enforcement of design-time governance policies.

9.1.1 Supported policy frameworks and concepts in WSRR

The implementation of service policy management in WSRR is based on the following standards:

- Web Services Policy 1.5 Framework (WS-Policy) defines an abstract model and an XML-based expression grammar for declaring policies.
- Web Services Policy 1.2 Attachment (WS-PolicyAttachment) defines mechanisms for associating policies with the subjects to which they apply.

The following types of policy-related entity can be stored in the registry:

- *Policy documents* are XML documents that conform to the WS-Policy standard. They declare policies together with the subjects to which the policies apply. You can load policy documents into the registry by using the Web UI or the API.
- *Policies* specify policy requirements. Policies are derived from the policy declarations contained in a policy document and are created automatically when a policy document is loaded.

- *Policy attachments* associate policies with the subjects to which the policies apply. Policy attachments are derived from the policy declarations contained in a policy document and are created automatically when a policy document is loaded.

In addition to defining policy in a policy document, the WS-PolicyAttachment standard also supports embedding policy declarations in a WSDL file. If you load a WSDL file that contains policy declarations, WSRR creates corresponding policies and policy attachments automatically. The following mechanisms for embedding policy declarations in a WSDL file are supported:

- WS-Policy Attachment practice (recommended):
  - Policy expressions (`wsp:Policy`) elements are included as child elements of a WSDL document `wsdl11:definition` element.
  - Policy expressions are referenced using policy reference (`wsp:PolicyReference`) elements included as a child of the WSDL document element to which the policy applies.
- Policy declarations (`wsp:Policy`) elements are embedded directly in the specific elements, in the WSDL document, to which the policy applies.

In both cases, the subject of a policy is the WSDL element in which the policy declaration, either `wsp:PolicyReference` (recommended) or `wsp:Policy`, is contained.

Archived

Figure 9-1 illustrates the relationship between the various policy-related entities that are stored in the registry.

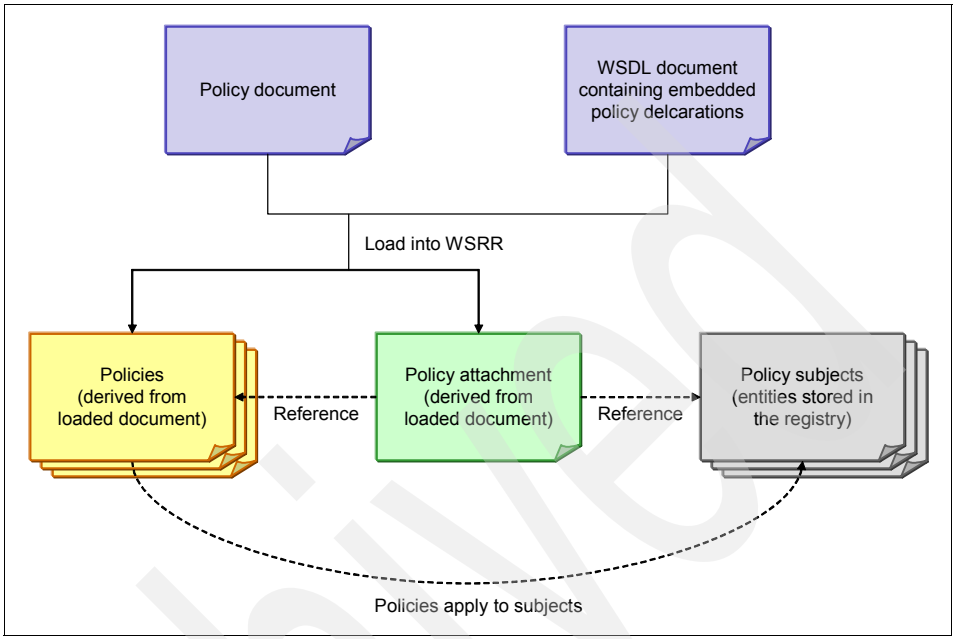


Figure 9-1 Policies stored in WSRR

9.1.2 Policy domains

Policy requirements can be expressed in relation to various fields of interest, such as security, privacy, reliability, and so forth. These various categorized fields of interest are termed as *policy library domains* or *policy domains*. Additionally, policy requirements can be expressed in the context of business, architecture, or operational. At the highest level of abstraction, business policies capture requirements closer to the business domain in business vocabulary. At the lowest level of abstraction, operational policies capture capabilities in terms of operational environment. Architectural polices address the architectural requirements that are needed to map business policy requirements to operational capabilities.

In this section, we describe the WSRR policy domains.

WSRR supported domains

WSRR provides library domains to help facilitate increased time-to-value and accelerated adoption of policy in SOA solutions. WSRR currently supports the following policy domains based on the WS-Policy standards for loading, viewing, editing, and attaching policies:

- Web Services Message Transmission Optimization Mechanism (MTOM) Policy 1.0

  Defines a behavior in which an endpoint requires and generates messages serialized as specified in SOAP MTOM.

- WebSphere WS-Security Policy 1.2 Extensions

  Identifies additional assertions supported by WebSphere Application Server that are relevant for WS-Security Policy.

- WSRR metadata governance policy

  Provides a way to govern services by ensuring that the metadata that describes those services conforms to leading practices.

- Web Services Atomic Transaction Policy 1.0

  Supports the Atomic Transaction 1.0 coordination type that is used in WS-Coordination, which allows application operations that cross heterogeneous service boundaries to be considered as part of the same atomic transaction.

- Web Services Atomic Transaction Policy 1.1

  Supports the Atomic Transaction 1.1 coordination type that is used in WS-Coordination, which allows application operations that cross heterogeneous service boundaries to be considered as part of the same atomic transaction.

- Web Services Business Activity Policy 1.0

  Supports the Business Activity coordination type that used in WS-Coordination, which supports coordination of application activities that are long running and that involve interactions between multiple heterogeneous services. This domain allows service invocations to be actioned or compensated consistently according to the overall context.

- Web Services Business Activity Policy 1.1

  Supports the Business Activity coordination type that is used in WS-Coordination, which supports coordination of application activities that are long running and that involve interactions between multiple heterogeneous services. This domain allows service invocations to be actioned or compensated consistently according to the overall context.

Archived

- WS-Reliable Messaging Policy 1.0

  Describes an assertion to indicate that WS-Reliable Messaging must be used to ensure reliable message delivery between a service consumer (Source) and provider (Destination).

- WS-RM Policy 1.1

  Describes an assertion to indicate that WS-Reliable Messaging must be used to ensure reliable message delivery between a service consumer (Source) and provider (Destination) or vice-versa.

- WS Security Policy 1.1

  Defines assertions to ensure secure exchange of messages between services.

- WS Security Policy 1.2 December 2005

  Defines assertions to ensure secure exchange of messages between services.

- WebSphere WS Security Policy 1.2 Extensions

  Identifies additional Assertions supported by WebSphere Application Server that are relevant for the WS-Security Policy.

WSRR WS-Policy extensions for domain growth

WSRR provides a componentized approach to supporting policy standards that allows the supported domains to grow over time. WSRR supports the following extensions to WS-Policy standards to perform policy library management:

- Policy taxonomy

  You can use policy taxonomy to list the supported domains, grouped by category. In WSRR, you can browse policies by domains for quick identification of the supported policies. If the policy taxonomy identifies policies by policy classes in addition to policy domains, then you can use policy classes to further refine the type of policies.

- Policy domain

  Each policy domain is identified by its namespace and prefix. The author of the policy selects a policy domain from the available domains when authoring or creating new policies. When a policy expression defines assertions from a particular domain, an attribute is added to the policy to indicate that domain and a classification is added to the policy. The Uniform Resource Identifier (URI) of the classification matches the domain namespace.

Archived

- Policy class or policy type

  In each domain there are certain key types of policy that describe required behaviors using specific combinations of assertion. These policy classes are identified in each policy domain allowing a policy author to define the high-level goals of the policy and then to elaborate with the specific assertions that support that policy class. When authoring and selecting these policy classes, descriptions are provided to guide the policy author on the purpose of each class and its intended use. The class is added as an attribute to the policy expression and is also provided as a classification, to aid location of relevant policies in the event that the policy taxonomy includes a policy class.

9.1.3 Publishing and discovering policies

The section discusses how to publish policies to WSRR. It covers how to use the Service Discovery feature in WSRR to discover and publish policy sets. Finally, it touches on the policies exposed when loading an Service Component Architecture (SCA) model into WSRR for governance.

Publish policies to WSRR

You can use multiple methods to publish existing policy documents to WSRR to be governed. WSRR provides a rich Web-based user interface, shown in Figure 9-2, to load the policy documents and to visualize their dependencies. For developers, WSRR provides both an Eclipse and Visual Studio plug-in. You can also publish policy documents to WSRR using one of the provided APIs (Java, SOAP, REST, or UDDI).

Archived

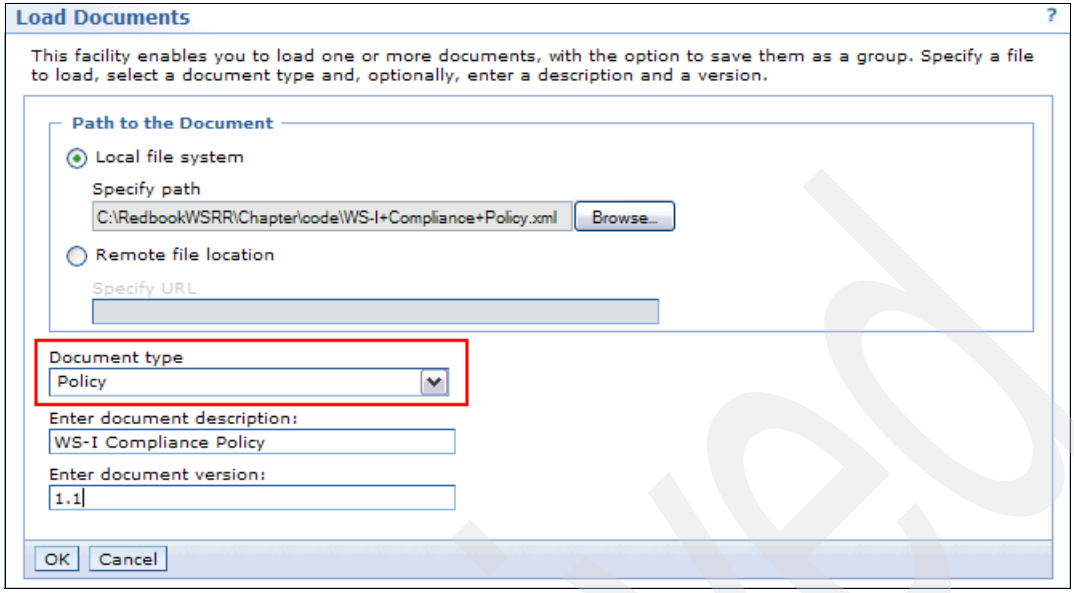


Figure 9-2 Web-based user interface to load WS-Policy documents

When you load the policy documents into WSRR, they are parsed into their logical constructs, policies, and policy attachments, which you can then annotate with metadata for easy discovery.

Policy sets

A *policy set* is defined as a collection of assertion about how services are defined. You can use a policy set to simplify security configurations. Policy sets can be discovered and published to the following applications:

- WebSphere Application Server V6.1 with Fix Pack 23 or later, with Web Services Feature Pack 17 or later installed
- WebSphere Application Server V7.0
- WebSphere Message Broker V6.1

Discovering policy sets

After you set up the Service Discovery and Scheduler configurations, policy sets can be discovered automatically using the service discovery scheduler or manually by running a scheduled task immediately. When discovering policy sets, if a policy set exists in WSRR with the same name as one that is being discovered, the existing policy set and associated policy files are not overwritten, because it is assumed that policy sets with the same name contain the same

content, however, the policy set and associated policy files are classified with all the types of systems that they were discovered from.

Publishing policy sets to policy enforcement points

A policy set and its referenced policy files can be published to either WebSphere Application Server or WebSphere Message Broker by governing the policy set file and taking the file through a life cycle. The transition that triggers the publication of the policy set is defined in the Service Discovery configuration files in the `<publish-transitionURI>` element. When a policy set is put into a state in its life cycle by the transition URI defined in the `<publish-transitionURI>` element, the policy set is published to the instances defined in the discovery configuration that have the `<publish-transitionURI>` element.

For more information about how to configure WSRR for the Policy Set Feature, see the WebSphere Service Registry and Repository V6.3 Information Center:

http://publib.boulder.ibm.com/infocenter/sr/v6r3/index.jsp

SCA module promoted properties

When you load an SCA module that includes promoted properties into WSRR, the promoted properties are exposed as mediation policies in the form of WS-Policy. You can then use WSRR to store and govern the mediation policy information. Mediation policies can help you control service requests by dynamically overriding module properties. For example, you can apply different mediation policies in different contexts by creating mediation policy conditions.

> **Note:** Mediation policies are concerned with the control of mediation flows in terms of mediation functions, such as routing, transformation, conversion, or logging, rather than non-functional quality characteristics, such as transactions, security, and so on.

When developing an SCA module that needs to make use of a mediation policy, include a *policy resolution* mediation primitive in the mediation flow. At run time, the policy resolution mediation primitive obtains mediation policy information from the registry.

For more information about mediation policies, see *Integrating WebSphere Service Registry and Repository with WebSphere Process Server and WebSphere ESB*, REDP-4557.

9.1.4 Authoring, editing, and attaching policies

You can use the WSRR policy editor to author new policies and to edit existing policies. The policy editor includes library templates that you can use to create and edit policies (Figure 9-3). For information about how to use the policy editor, see 9.3, “Implementing policy” on page 316.

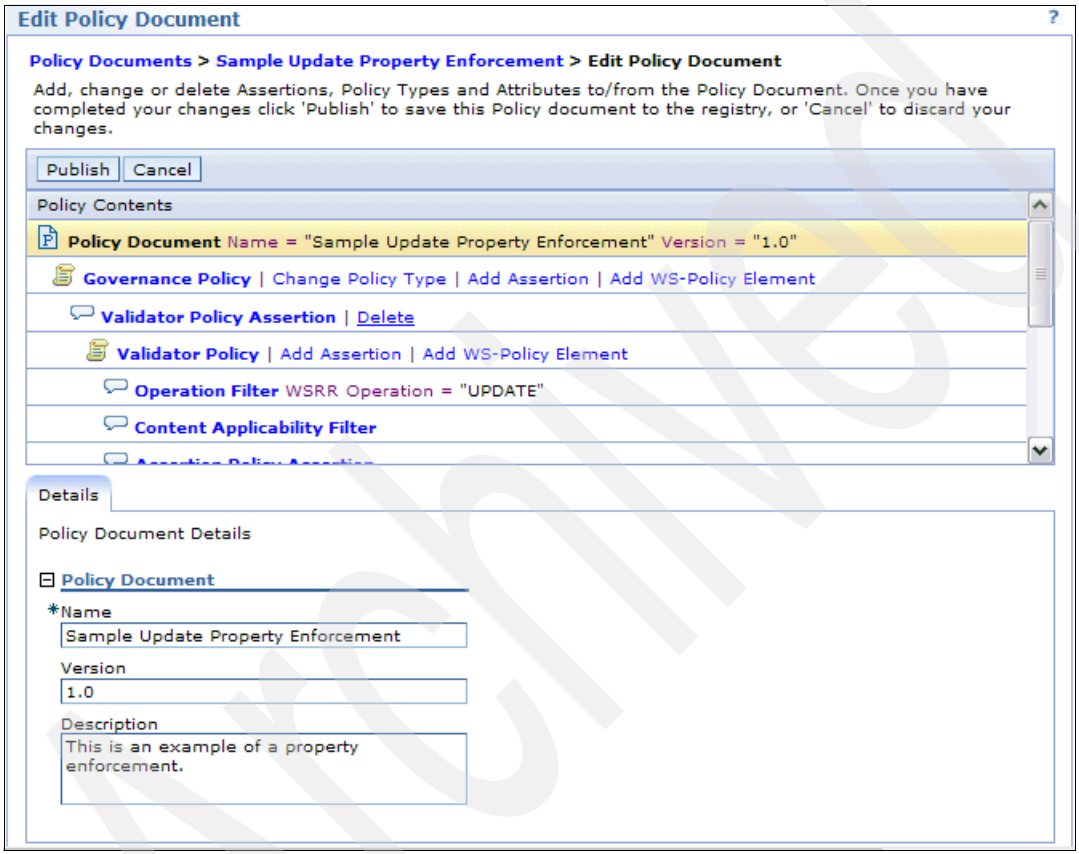


Figure 9-3 Edit Policy Document

Archived

You can also attach policies to services. To attach a policy to an entity, from the details view of the entity, select the Policy tab. Figure 9-4 shows the policy attachment editor and where to attach a policy on a Web service.

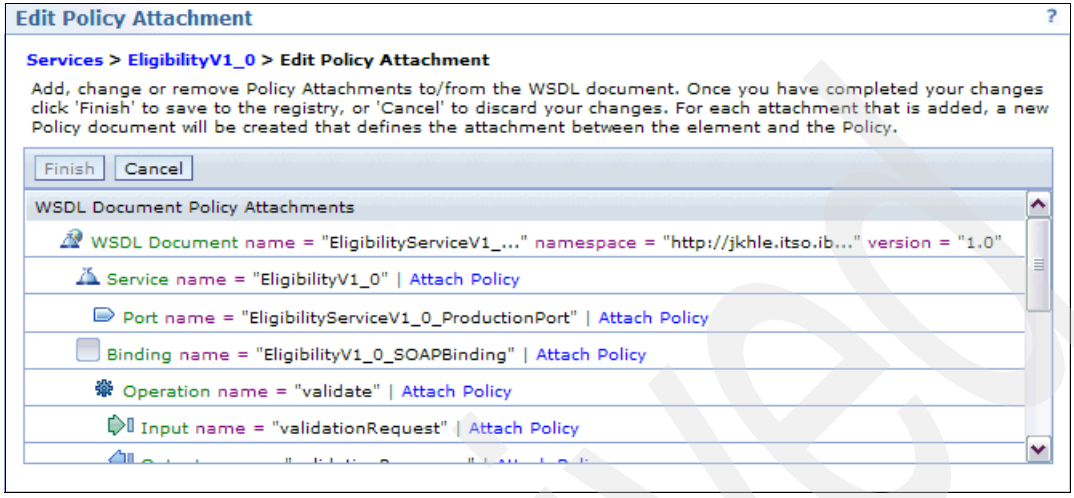


Figure 9-4 Edit Policy Attachment

After you create policies and attach them to services, policy enforcement points can query WSRR for the services and their attached policies to enforce them.

9.1.5 Policy enforcement

In simple terms, a *policy enforcement point* is the entity that enforces policies, such as access control or another policy decision, in response to a request from a user who is waiting to access a resource. In this section, we discuss the WSRR support for design and runtime policy enforcement points.

Design time

WSRR provides a Web Service Interoperability (WS-I) and a governance policy validator to enforce design-time policies in the registry. WSRR exposes a framework that allows you to write custom validator plug-ins to enforce governance policies.

WS-I validator

The WS-I validator validates that the Web Service Definition Language (WSDL) documents that are stored and managed in WSRR adhere to WS-I Basic Profile 1.1. The WS-I validator can check for compliance when any create, update, state transition, or make governable operation is performed on a WSDL document. You can configure the WS-I validator for the WSDL documents that require

validation, the actions to take when validations fail, and whether to generate a validation report.

To enable the WS-I validator policy, you create and configure a governance policy and then deploy that policy to WSRR. For detailed instructions about how to create a governance policy to enable the WS-I validator, see 9.3.1, “Creating a WS-I compliance policy” on page 316.

Governance policy validator

The governance policy validator controls operations that can be performed on specific entities in WSRR, based on the metadata (properties, relationships, and classifications) to which it is attached. The governance policy validator can control the following operations on selected entities:

- `create`
- `update`
- `delete`
- `transition`
- `validate`
- `make_governable`
- `remove_governance`

The governance policy validator is configured by WS-Policy. The WSRR policy editor has a template library for the various assertions that the governance policy validator can enforce.

The governance policy validator enforces the assertions found in Table 9-1. You can use combinations of these assertions to create very powerful governance policies. For example, you can use the protection and a property assertion to ensure that only users in the architect role can modify a particular property with a property value between a specific range.

Table 9-1 Governance policy assertions

Assertion Name	Description
EntityAssertion	Allows the operation if the entity is of a specified entity type.
PropertyAssertion	Allows the operation if the entity has a property of a given name, type, and value (or the value lies in a specified range or belongs to a specified set of values).
ClassificationAssertion	Allows the operation if the entity has a specified set of classifications.
RelationshipAssertion	Allows the operation if the entity has a relationship of a given name to an entity of a specified type.

Assertion Name	Description
RelationshipCountAssertion	Allows the operation if the entity has a relationship of a given name to one or more entities of a specified type and the number of such relationship targets that satisfy the assertion lies within specified minimum and maximum values.
RelatedToByAssertion	Allows the operation if another entity of a specified type has a relationship of a given name to this entity.
RelatedToByCountAssertion	Allows the operation if one or more other entities of a specified type have a relationship of a given name to this entity, and the number of such relating entities that satisfy the assertion lies within specified minimum and maximum values.
NagateAssertion	Allows the operation if at least one of a specified set of assertion fails.
PluginAssertion	Calls a method on a Java class to determine if the operation is allowed.
ProtectionAssertion	Allows the operation if the user who is performing the operation is in a specified WSRR role.
UniqueAssertion	Allows the operation if the name of the entity is unique.

You can find detailed instructions about how to create a governance policy using the WSRR policy editor in 9.3, “Implementing policy” on page 316.

Validation plug-ins

Validation plug-ins are called before certain registry operations take place and provide hooks to validate and potentially modify the content and properties of objects that are stored in the registry. An error can be returned from a validation plug-in to prevent registry operations from completing.

Runtime

Policies authored, governed, and attached to service in WSRR, can be retrieved by runtime policy enforcement points and enforced. WSRR itself does not enforce runtime policies. Example policy enforcement points are:

- IBM WebSphere Application Server
- IBM WebSphere DataPower
- IBM WebSphere Enterprise Service Bus
- IBM WebSphere Message Broker
- Other vendor ESBs and applications

9.1.6 Policy life cycle governance

It is important to manage the life cycle of a policy along side the services to which they are attached so that you can analyze the impact of the policies if they are changed. For example, changing a policy might result in a new version of a service, in which case, all consumers of that service must be notified. Lack of visibility about policies and their impact if the policies changed can lead to system outages and poor or no policy enforcement.

WSRR provides a prescriptive approach to policy governance in the governance enablement profile. The governance enablement profile provides an effective way to manage a policy life cycle in an iterative manner spanning throughout all policy library domains. This governance enablement profile model provides the following phases, as illustrated in Figure 9-5:

- Author

  In this phase of policy life cycle, policies are identified, designed, authored, and collected with other related policies. Additionally, stakeholders who are affected by the policies are identified in this phase.

- Transform

  In this phase, policies are transformed into actionable form.

- Enforce

  In this phase, actionable policies are implemented and enforced either using automatic or manual approaches.

- Monitor

  In this phase, metrics related to policy enforcement are gathered and analyzed in order to understand the impact of policy enforcement, as well as govern the original policy requirements and expectations.

- Retired

  In this phase, policies are retired, having served their useful purpose and reached their end-of-life.

Archived

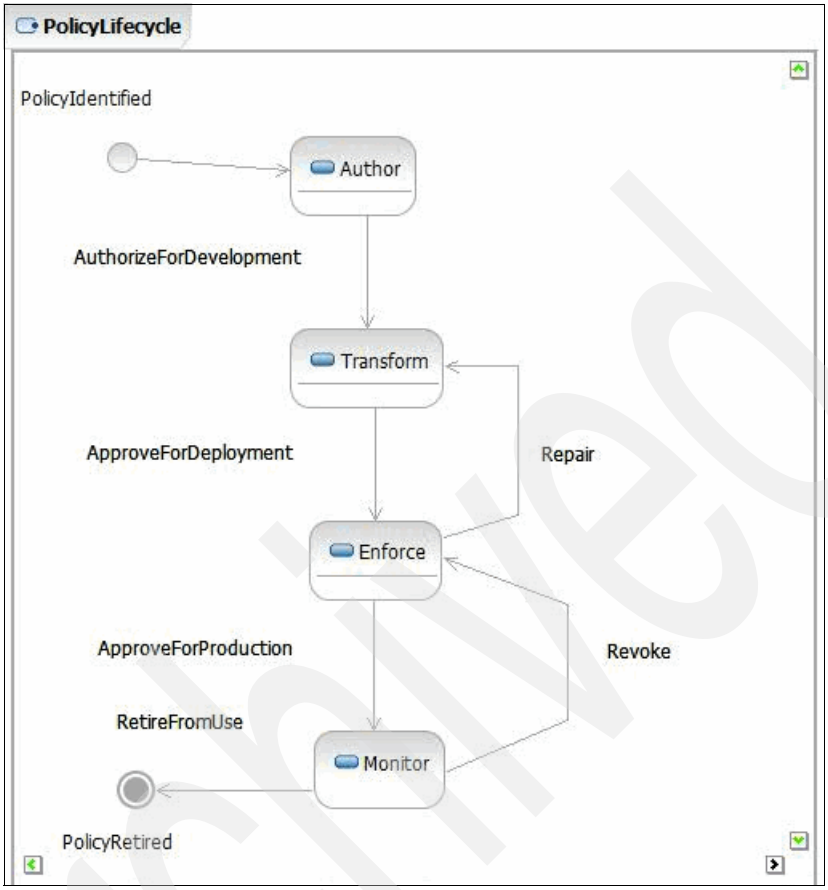


Figure 9-5 Policy life cycle state transition diagram

9.2 Applying policy to the JKHLE business scenario

In the JKHLE case study, we want to apply the following policies:

- Use the policy authoring tool to create a WSRR service metadata governance policy that enforces WS-I compliance checking, reporting, and enforcement on a WSDL file. This policy uses the WS-I validator plug-in. See 9.3.1, “Creating a WS-I compliance policy” on page 316.
- Use the WSRR policy authoring tool to create and implement a WSRR service metadata governance policy that enforces a mandatory service metadata attribute specification. See 9.3.2, “Creating an updated property enforcement policy” on page 329.

Archived

- Define and implement a WebSphere ESB mediation module policy resolution primitive to dynamically control a mediation flow on a message-by-message basis. See *Integrating WebSphere Service Registry and Repository with WebSphere Process Server and WebSphere ESB*, REDP-4557.

9.3 Implementing policy

This section provides step-by-step details about how to create two governance policy validator policies using the WSRR policy editor. It also explains how to deploy these policies into WSRR to be enforced by the governance policy validator.

The first policy exercises the WS-I policy validator, which uses a plug-in assertion. The second policy uses a property assertions to enforce specification of property values for `updatedBy` and `updatingPersonLocation` on the business services entity type object in WSRR at the time of entity modification.

9.3.1 Creating a WS-I compliance policy

The WS-I policy and its WS-I validator plug-in are an excellent example of how to create a custom validator plug-in for which you can create and configure WS-Policies.

In this section, we demonstrate how to create the WS-I policy that is enforced by a custom plug-in validator. The WS-I validator plug-in is already preloaded into WSRR.

Archived

To create the WS-I policy:

1. Start the WSRR policy editor. Change to the SOA governance perspective. On the home page, in the Service Document box, click **Policy Documents** to open the Policy Documents view. From the Policy Documents view, click **New** as shown in Figure 9-6.

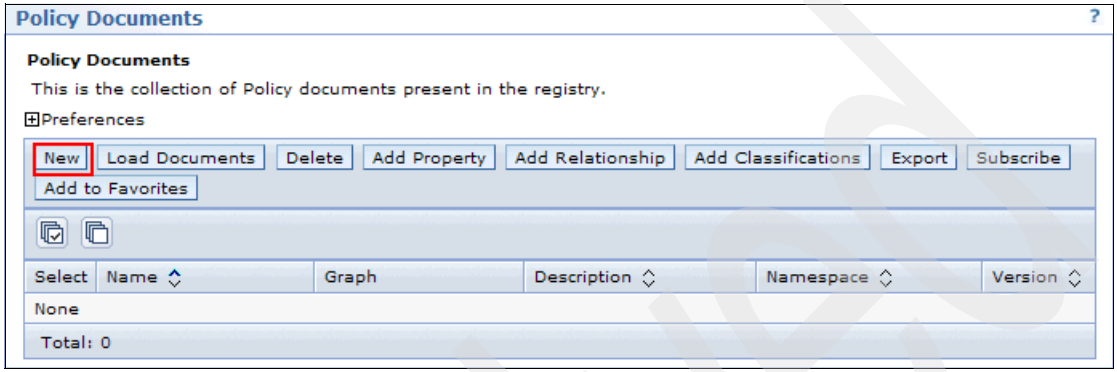


Figure 9-6 Opening the policy editor

When you create a WS-Policy using the WSRR policy editor, you first need to select a WS policy framework. WSRR support the following policy frameworks:

– WS Policy Framework 1.2
– WS Policy Framework 1.5
– WS Policy Framework 1.5 2006

WS Policy Framework 1.5 is the most current policy framework. You should use it when creating new policies unless the policy enforcement points require the previous policy frameworks.

2. Select **WS Policy Framework 1.5**, and click **Next**.

Archived

3. Enter the following information, as shown in Figure 9-7:
   - Name: `WS-I Compliance Policy`
   - Version: `1.0`
   - Description: `This policy validates WS-I Compliance`

Figure 9-7 Entering policy details

4. Select the policy domain for creating Governance Policies. As mentioned in 9.1.2, “Policy domains” on page 304, WSRR supports several policy domains.

Archived

5. Click **Select Policy Domain**. Then, from the **Policy Domains** drop-down menu, select **WSRR Service Metadata Governance Policy 6.2 Domain** and click **Apply**, as shown in Figure 9-8.

Figure 9-8 Select the policy domain

Archived

6. Next, specify the identifier of the Policy Expression to determine how this policy is referenced in policy attachments or policy references. In the Details section, click **Add Property**. Then, from the **Optional Properties** drop-down menu, select **Policy Identifier** and click **Add**. Enter `urn:WSICompliancePolicy` in the Policy Identifier field, as shown in Figure 9-9.

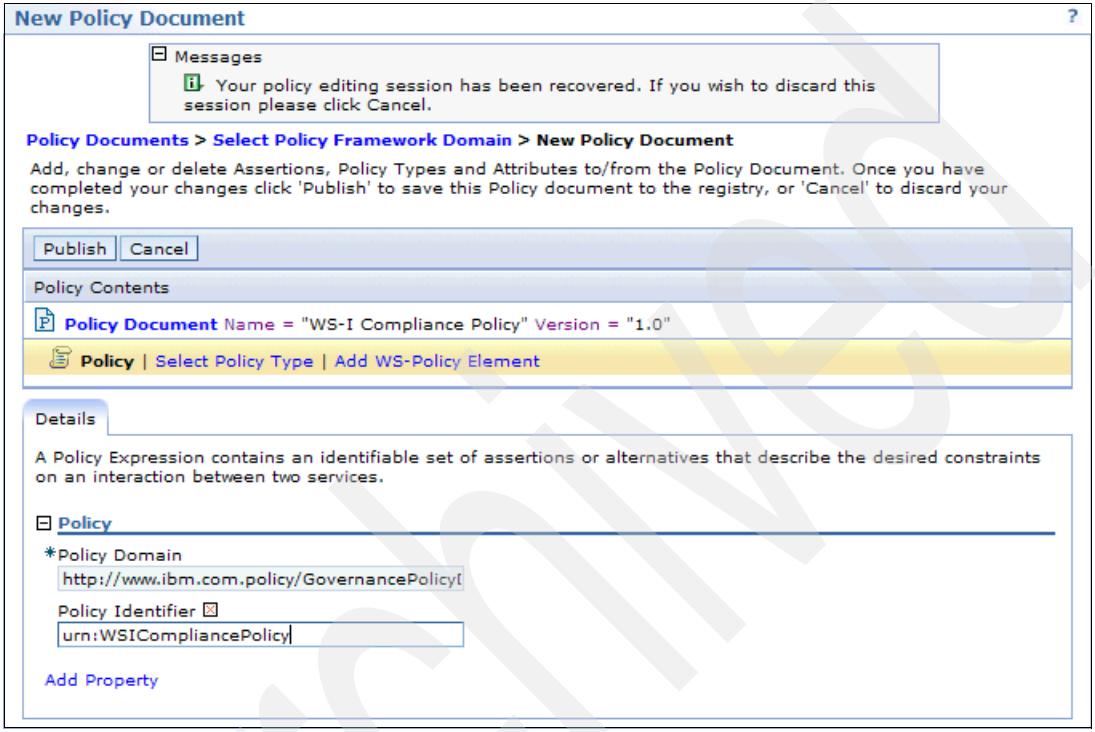


Figure 9-9 Entering the Policy Identifier

> **Note:** The policy editor saves the session automatically. If the session expires or if the browser has an issue, refresh the browser or restart the policy editor. The policy editor prompts you if it recovers the previous session. In Figure 9-9, the session was recovered.

Archived

7. Next, select the type of policy from the domain's list of policy. In the WSRR Service Metadata Governance Policy 6.2 domain, only one policy type is available. Click **Select Policy Type**. Then, from the **Policy Type** drop-down menu, select **Governance Policy**, and click **Select**. The Policy Class property with the value of GPPolicy is added to the Governance Policy, as shown in Figure 9-10.

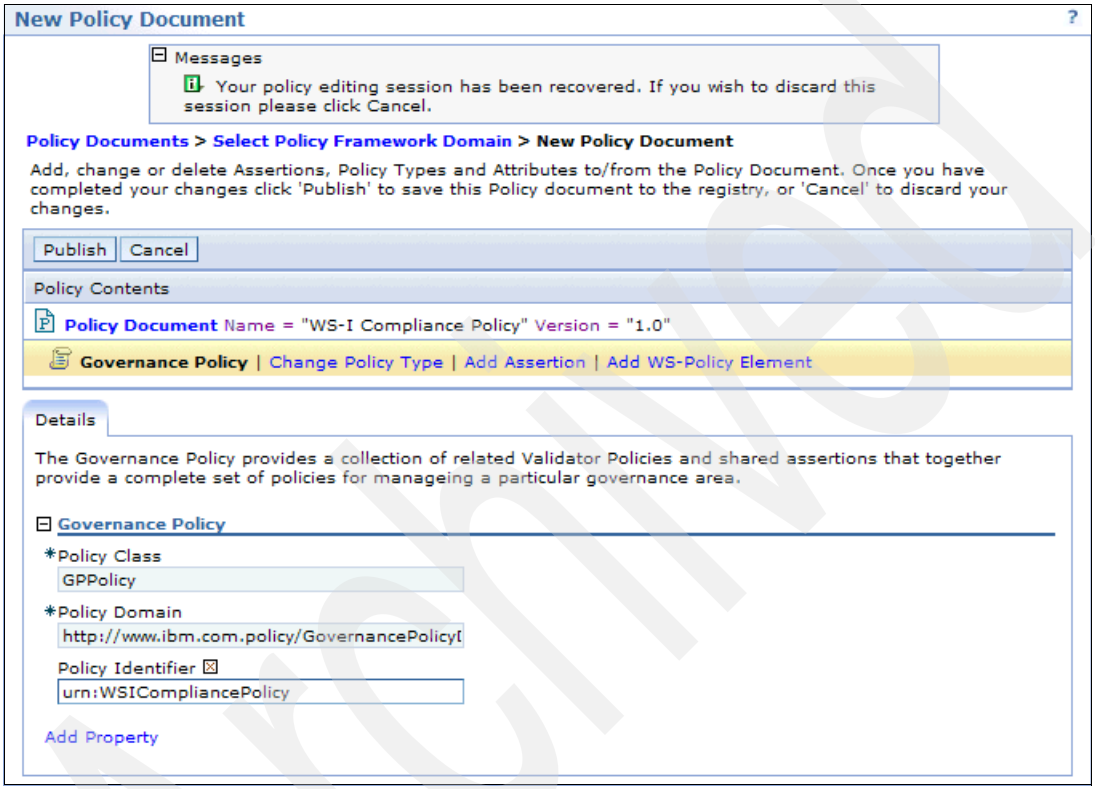


Figure 9-10 Results of selecting policy type of Governance Policy

8. Now, add the Validator Policy assertion to the Governance Policy. The Validator Policy assertion is required for the Governance Policy to work properly. You can add the other assertions outside of the Validator Policy element and reference them from within the Validator Policy using the

Archived

`AssertionRef` assertion. Click **Add Assertion**. Then, from the **Assertion Options** drop-down menu, select **Validator Policy Assertion** and click **Add**, as shown in Figure 9-11.

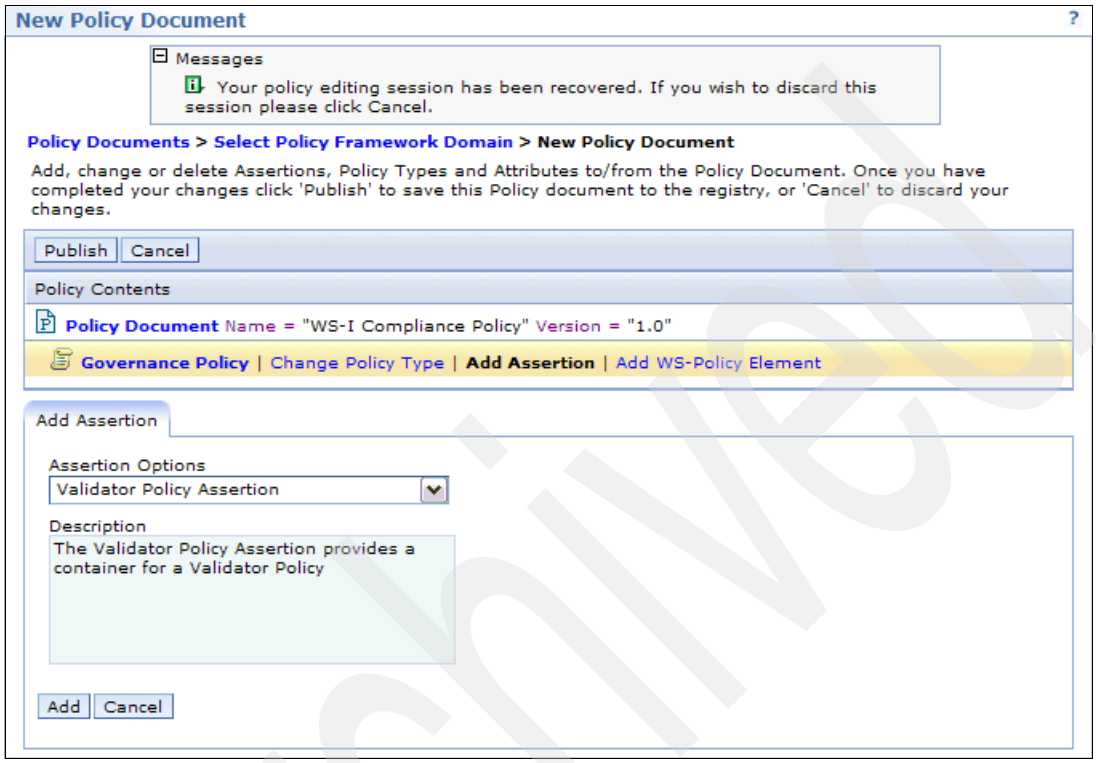


Figure 9-11 Adding Validator Policy Assertion

Archived

The policy editor has the Validator Policy Assertion highlighted in the editor, which allows you to add properties to it, as shown in Figure 9-12.

Figure 9-12 Add Property to Validator Policy Assertion

9. Add a description name to the Validator Policy. Click **Add Property**. Then, from the **Optional Properties** drop-down menu, select **Name**, and click **Add**. Enter `plugin_wsi_validation_policy` in the Name field. The policy editor saves the policy automatically.

   In the next steps, you specify the operations and the content on which this Validator Policy is enforced. The valid entries for the operation field are:

   - `CREATE`
   - `UPDATE`
   - `DELETE`
   - `VALIDATE`
   - `TRANSITION`
   - `MAKE_GOVERNABLE`
   - `REMOVE_GOVERNANCE`

Archived

10. In the editor pane, click **Operation Filter**. Then, in the **Details** section, enter `CREATE` in the WSRR Operation field, as shown in Figure 9-13.

Figure 9-13 Specify CREATE in the WSRR Operation field

11. In the editor pane, click **Content Applicability Filter**. Then, in the Details section, click **Add Property**. From the **Optional Properties** drop-down menu, select **Target XPath**, and click **Add**.

Archived

12. The Target XPath property specifies the entities to which the Assertion Policy applies. It is an XPath expression. The target entity for the WS-I policy is the WSDLDocuments. Enter `/WSRR/WSDLDocument` in the Target XPath field, as shown in Figure 9-14, to specify that the policy is applied to all WSDL Documents.

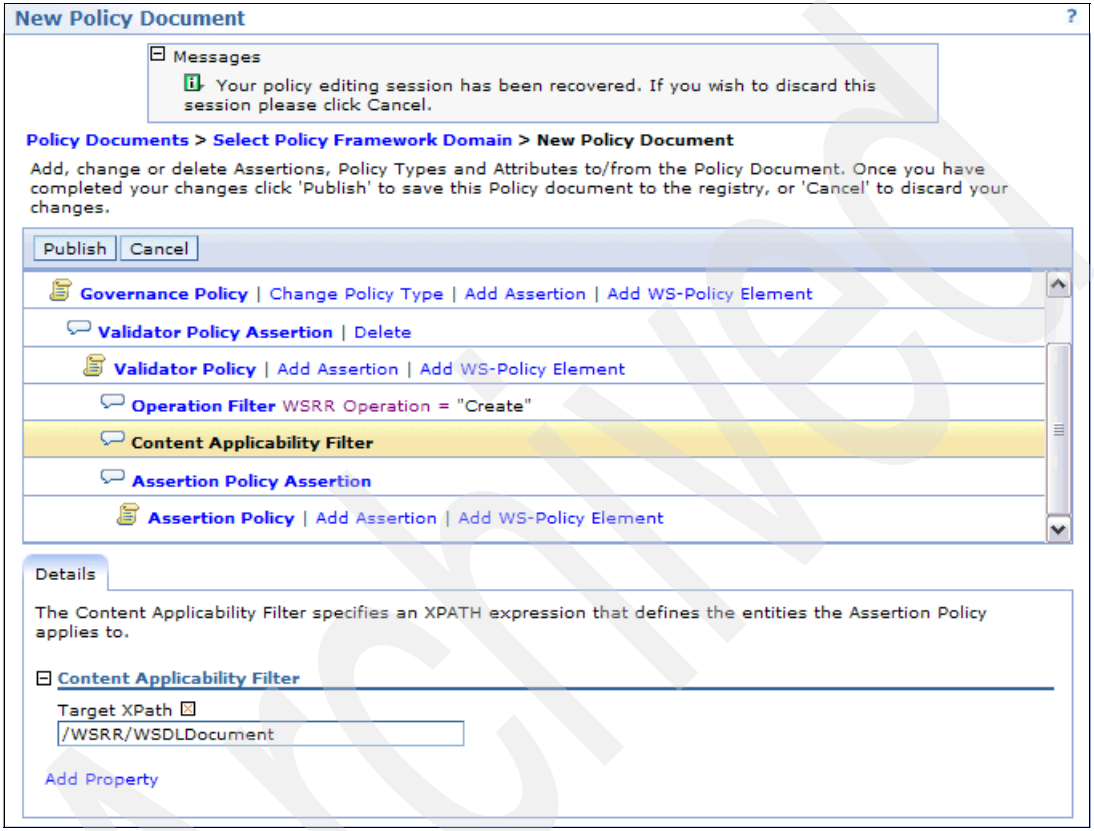


Figure 9-14 Specify Content Applicability Filter

Next, you add an assertion policy to the validator policy. An assertion allows you to restrict the set of entities on which an operation can be performed. If an entity matches against the Validator Policy that contains an assertion, then the assertion determines whether the operation that is associated with the ValidatorPolicy is allowed. The governance policy validator has 11 configurable assertions available.

To use the WS-I validator plug-in, a plug-in assertion is added. A plug-in assertion calls a method on a Java class to determine if an operation is allowed. Configuration options are passed to the Java class.

Archived

13. From the editor pane, in the Assertion Policy row, click **Add Assertion**. Then, from the **Assertion Options** drop-down menu, select **Plugin Assertion** and then click **Add**. Enter `com.ibm.sr.policy.wsi.WSIComplianceValidator` in the Class field, as shown in Figure 9-15.

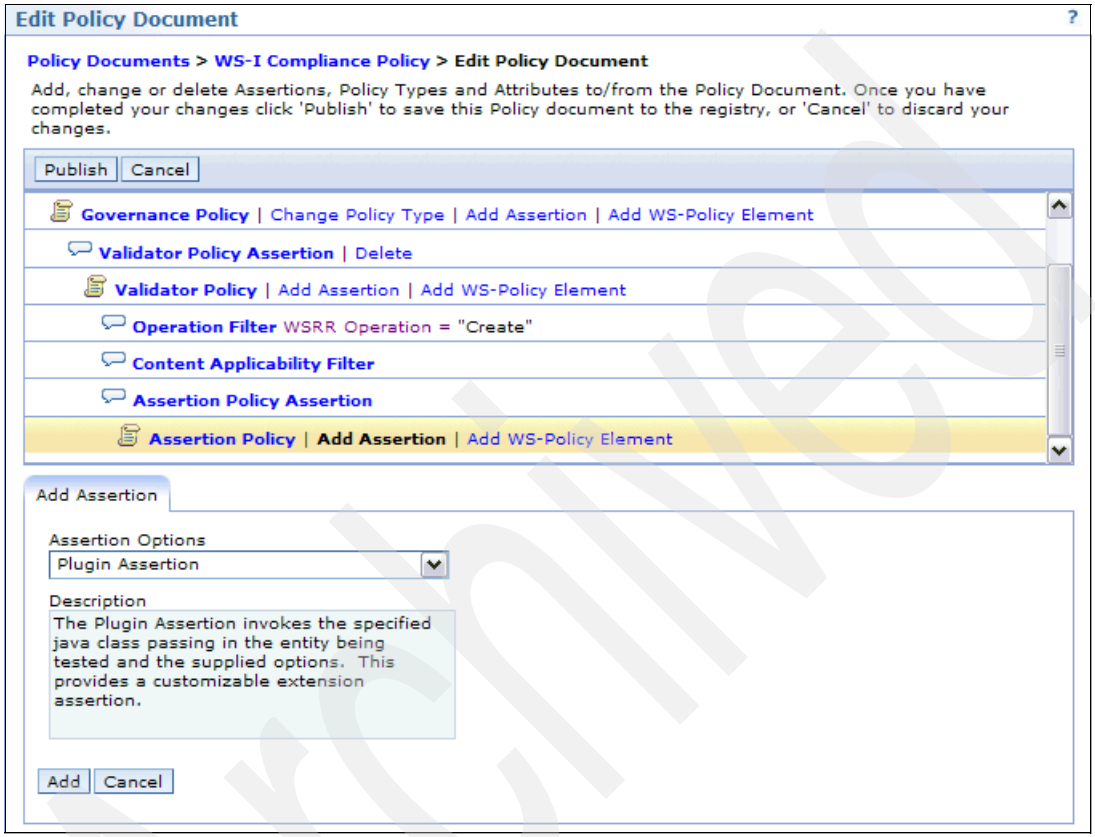


Figure 9-15 Add Plugin Assertion

When using a Plugin Assertion, configuration parameters can be passed to the Java class by adding an Options property. The parameters are separated by semicolons (;). The WS-I validator plug-in allows specification of which rule to enforce and whether a report is generated.

Archived

14. Click **Add Property**. Then, from the **Optional Properties** drop-down menu, select **Name**, and click **Add**. Select **Options**, and click **Add**. Then, enter `Invoking WS-I Compliance Validation Confirmation` in the Name field, and enter `rules=WSIBasicProfile11;report=true` in the **Options** field, as shown in Figure 9-16.

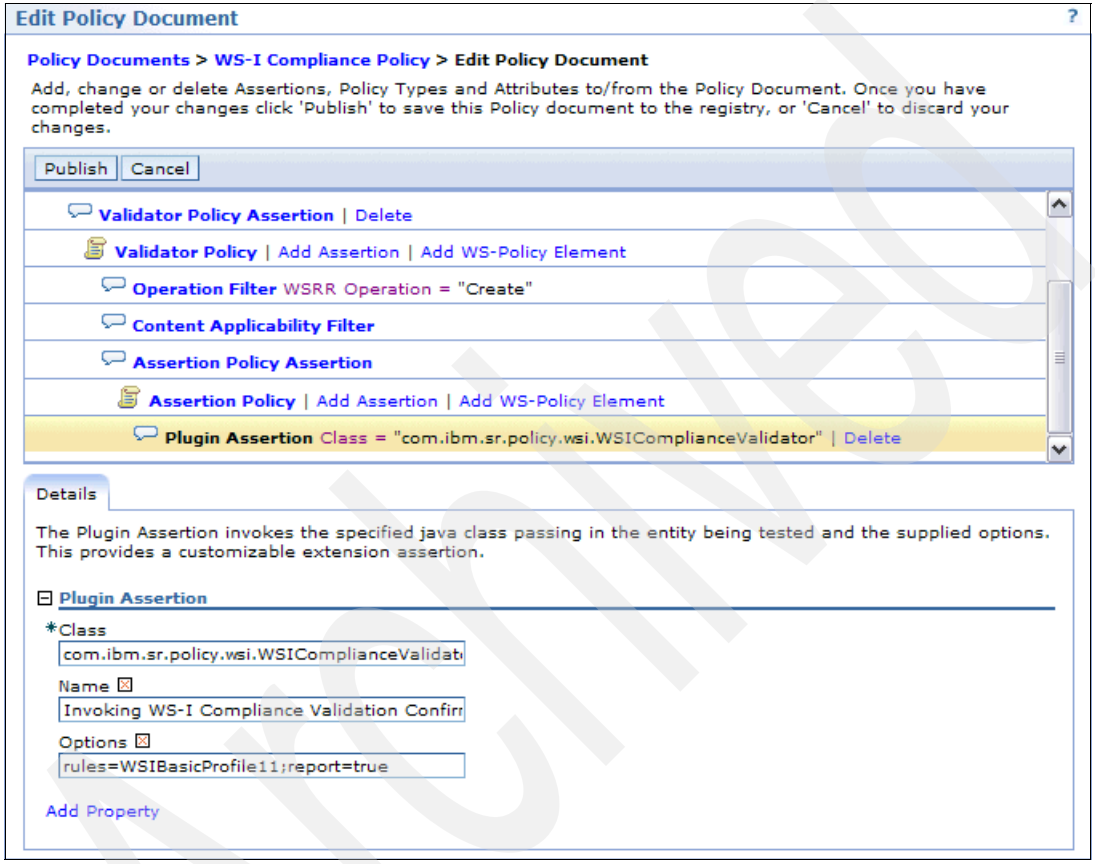


Figure 9-16 Setting Plugin Assertion properties

Archived

15. Click **Publish** to publish the WS-Policy. The WS-I Compliance Policy is now in the WS-Policy collection view, as shown in Figure 9-17.

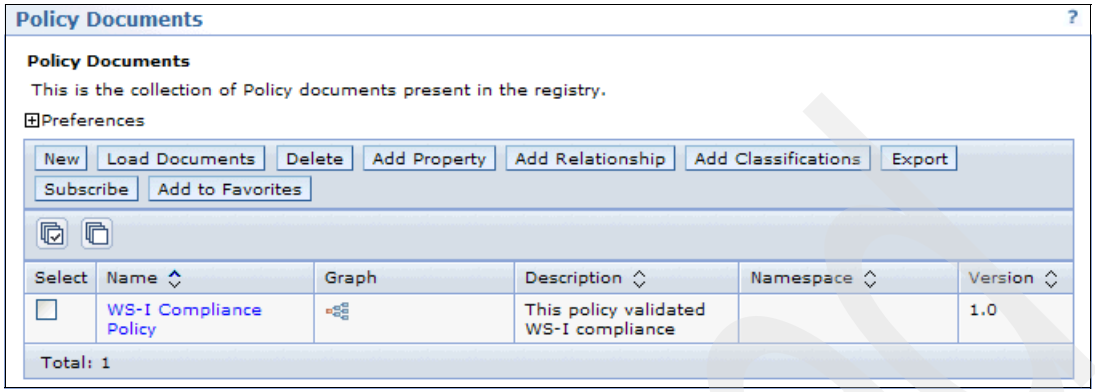


Figure 9-17 WS-Policy Collection View

You can edit the policy by clicking the policy, then going to the Policy tab and clicking **Edit Policy Document** as shown in Figure 9-18.

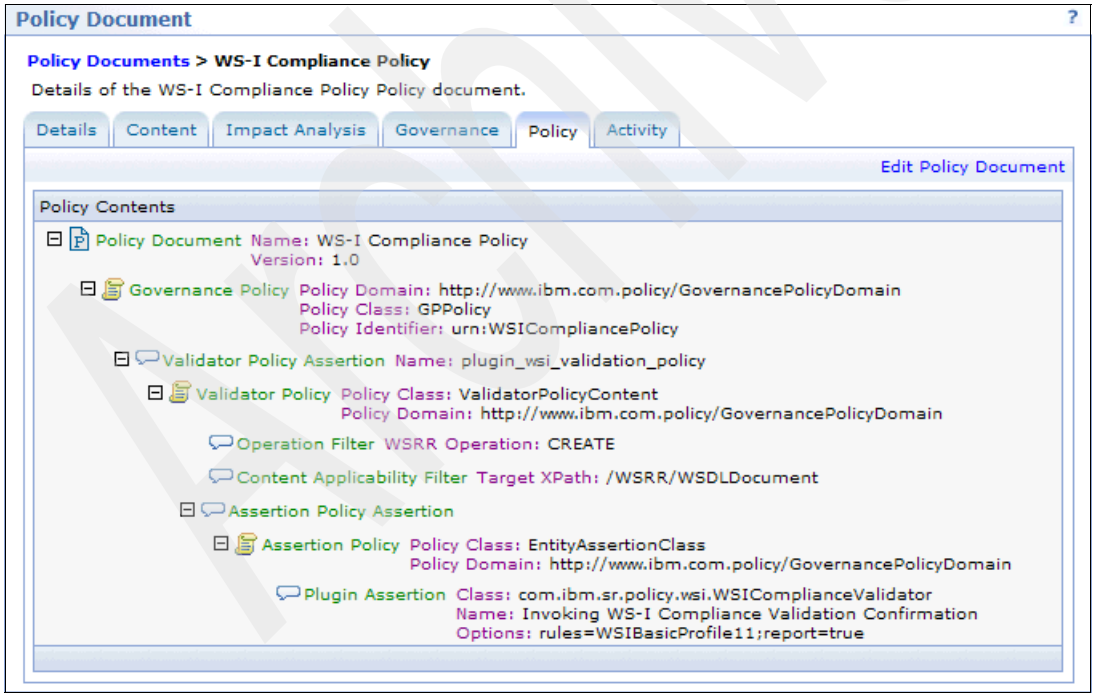


Figure 9-18 Edit Policy Document

Refer to 9.3.3, “Testing and deploying governance policies” on page 341 for information about how to deploy and test the policy.

9.3.2 Creating an updated property enforcement policy

In this section, we explain how to create a WSRR service metadata governance policy that enforces mandatory non-blank values for the user-defined properties `updatedBy` and `updatingPersonLocation` on the Business Service entity when it is updated. In other words, when updating a Business Service if the properties `updateBy` and `updatingPersonLocation` do not exist or exist but do not have a value, the update fails.

To create this policy:

1. Change to the SOA Governance perspective. On the home page, in the Service Document box, click **Policy Documents** to open the Policy Documents view. From the Policy Documents view, click **New**, as shown in Figure 9-6.

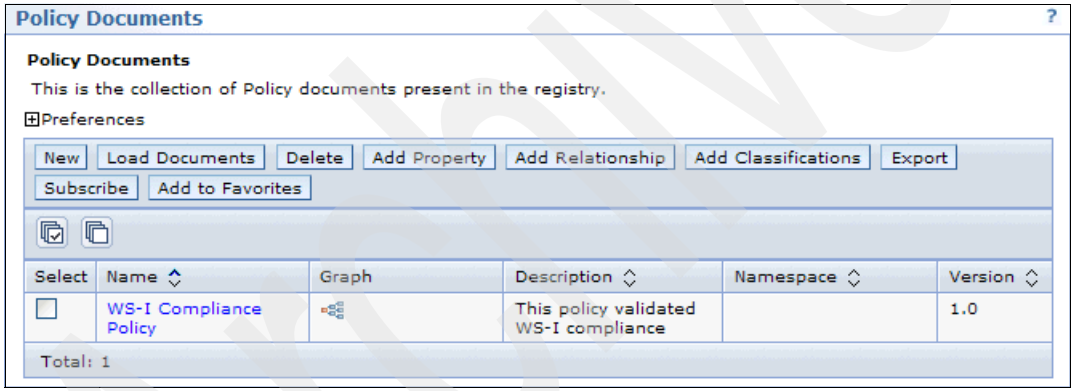


Figure 9-19 Opening the policy collection view

To create a WS-Policy using the WSRR policy editor, you need to select one of the following WS policy frameworks:

- WS Policy Framework 1.2
- WS Policy Framework 1.5
- WS Policy Framework 1.5 2006

WS Policy Framework 1.5 is the most current policy framework. You should use it when creating new policies unless the policy enforcement points require the previous policy frameworks.

2. Select **WS Policy Framework 1.5**, and click **Next**.

3. Then, enter the following information, as shown in Figure 9-20:
   - Name: `Sample Updated Property Enforcement`
   - Version: `1.0`
   - Description: `This policy is an example of a property enforcement`

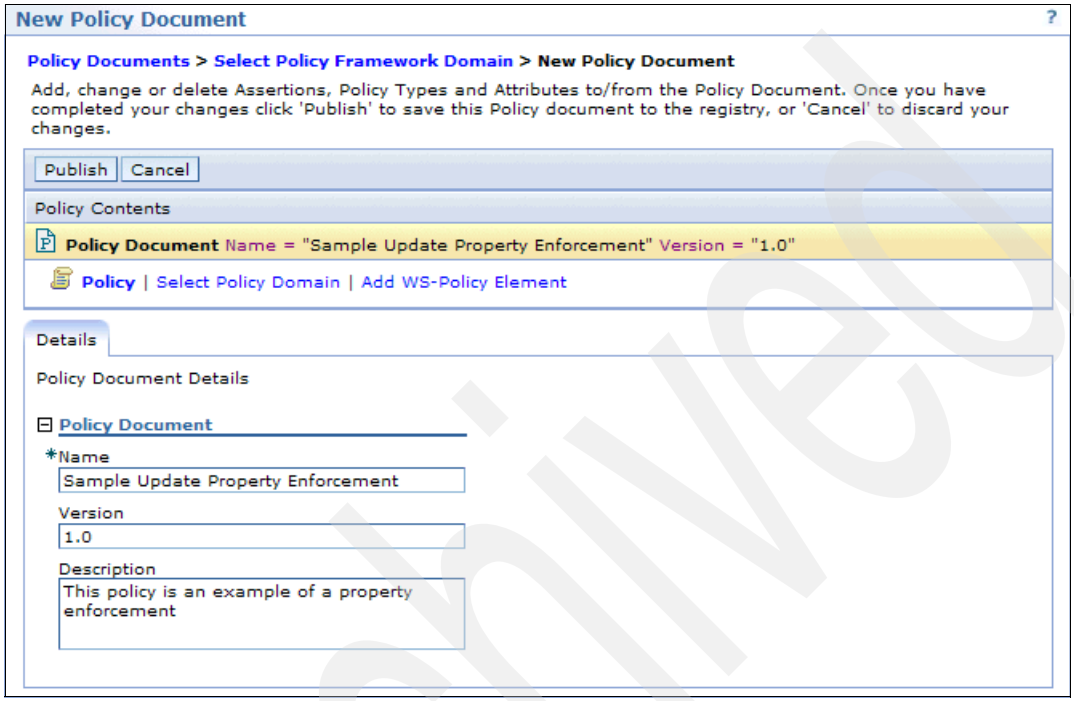


Figure 9-20 Enter policy details

Archived

4. Select the policy domain for creating Governance Policies. As mentioned in 9.1.2, “Policy domains” on page 304, WSRR supports 12 policy domains.
5. Click **Select Policy Domain**. Then, from the **Policy Domains** drop-down menu, select **WSRR Service Metadata Governance Policy 6.2 Domain** and click **Apply**, as shown in Figure 9-21.

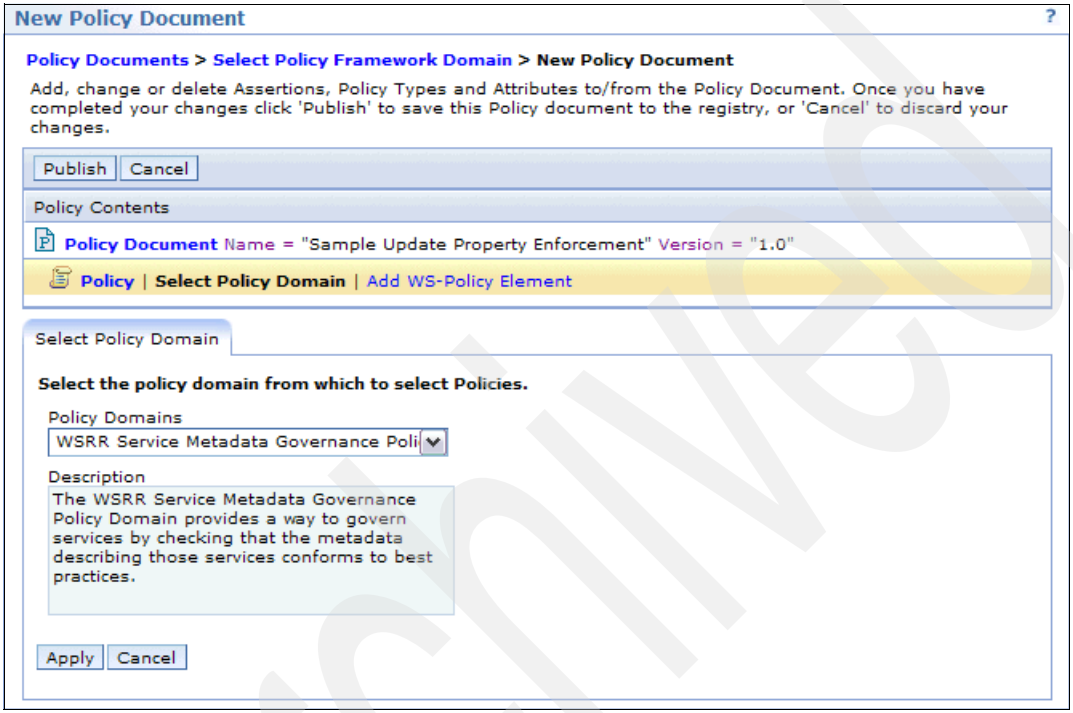


Figure 9-21 Selecting the policy domain

Archived

6. Next, you need to specify the identifier of the Policy Expression, which determines how this policy is referenced in policy attachments or policy references. In the Details section, click **Add Property**. Then, from the **Optional Properties** drop-down menu, select **Policy Identifier** and click **Add**. Enter `urn:propertiesEnforcingGovernancePolicy` in the **Policy Identifier** field, as shown in Figure 9-22.

Figure 9-22 Add Property - Policy Identifier

Archived

7. Now, select the type of policy from the domain's list of policy. In the WSRR Service Metadata Governance Policy 6.2 domain, only one policy type is available. Click **Select Policy Type**. Then, from the **Policy Type** drop-down menu, select **Governance Policy**, and click **Select**. The Policy Class property with the value of GPPolicy is added to the Governance Policy, as shown in Figure 9-23.

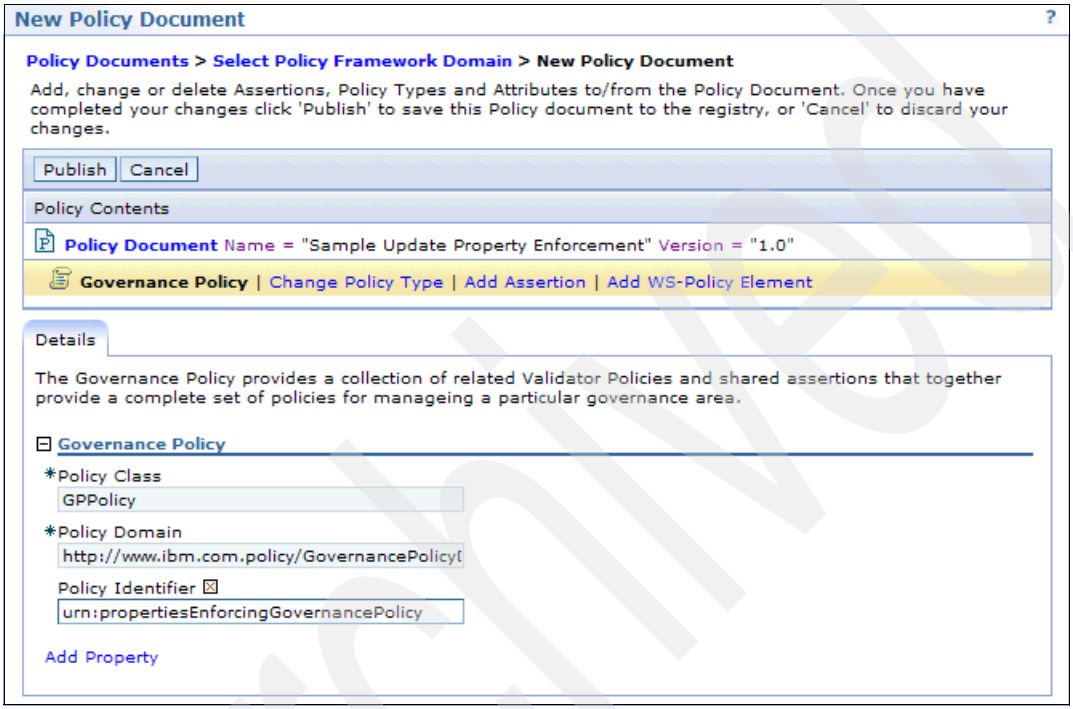


Figure 9-23 Results of selecting policy type of Governance Policy

Archived

8. Next, you add the Validator Policy assertion to the Governance Policy. The Validator Policy assertion is required for the Governance Policy to work properly. You can add the other assertions outside of the Validator Policy element and reference them from within the Validator Policy using the AssertionRef assertion. Click **Add Assertion**. Then, from the Assertion Options drop-down menu, select **Validator Policy Assertion**, and click Add, as shown in Figure 9-24.

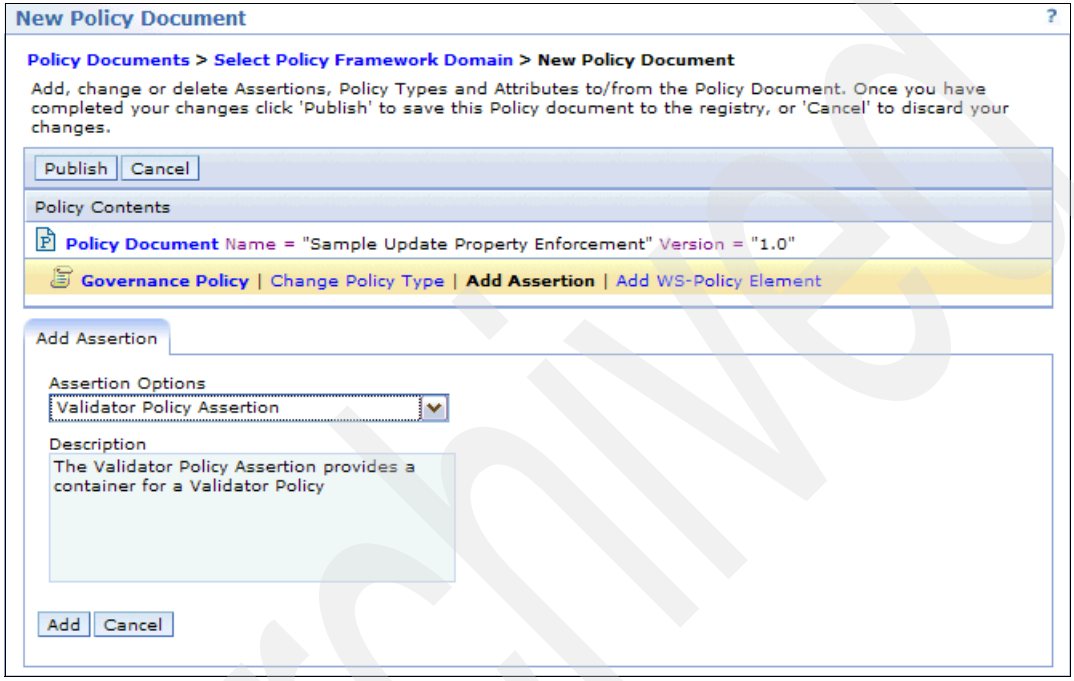


Figure 9-24 Add Validator Policy Assertion

Archived

Now, the policy editor has the Validator Policy Assertion highlighted in the editor, as shown in Figure 9-25, which allows you to add properties to it.

Figure 9-25 Adding properties to Validator Policy Assertion

9. Add a description name to the Validator Policy. Click **Add Property**. Then, from the **Optional Properties** drop-down menu, select **Name**, and click **Add**. Enter `updatedPropertiesEnforcement` in the Name field. The policy editor saves automatically.

   The next steps specify the operation and the content on which this Validator Policy is enforced. The operation field takes the following valid entries:

   - `CREATE`
   - `UPDATE`
   - `DELETE`
   - `VALIDATE`
   - `TRANSITION`
   - `MAKE_GOVERNABLE`
   - `REMOVE_GOVERNANCE`

Archived

10. In the editor pane, click **Operation Filter**. Then, in the Details section, enter `UPDATE` in the WSRR Operation field, as shown in Figure 9-26.

Figure 9-26 Specify UPDATE in the WSRR Operation field

11. In the editor pane, click **Content Applicability Filter**. In the Details section, click **Add Property**. Then, from the **Optional Properties** drop-down menu, select **Target XPath**, and click **Add**.

    The Target XPath property specifies the entities to which the Assertion Policy should apply. It is an XPath expression. In this sample policy, the target entity is the governance enablement profile Business Service entity.

Archived

12. In the Target XPath field, enter the following information, as shown in Figure 9-14:

```
/WSRR/GenericObject[classifiedByAnyOf('http://www.ibm.com/xmlns/prod
/serviceregistry/profile/v6r3/GovernanceEnablementModel#BusinessServ
ice')]
```

This XPath specifies that the policy is applied to all Business Service entities.

Figure 9-27 Specify Content Applicability Filter

13. Next, you add an assertion policy to the validator policy. An assertion allows you to restrict the set of entities on which an operation can be performed. If an entity matches against the Validator Policy containing an assertion, then the assertion determines whether the operation that is associated with the ValidatorPolicy is allowed. The Governance Policy Validator has 11 configurable assertions available.

For the sample metadata enforcement policy, you need to enforce two Property Assertions. A Property Assertion checks that a property exists on the entity and allows the type of the property to be checked. If the assertion is marked as read only, any attempts to update this property also fail. The

Archived

Assertion provides embedded assertions that put constraints on the value of the property.

Because of the enforcement of two Property Assertion, the use of an All Of Assertion is required. An All of Assertion requires all embedded assertions to pass before it will pass. If any embedded assertions fail, this assertion will fail.

From the editor pane, in the Assertion Policy row, click **Add Assertion**. Then, from the **Assertion Options** drop-down menu, select **All Of Assertion**, and click **Add**, as shown in Figure 9-28.

Figure 9-28 Add All Of Assertion

14. Click **Add Property**. Then, from the **Optional Properties** drop-down menu, select **Name**, and click **Add**. Enter `Updated properties cannot be NULL` in the Name field. This provides a human readable error message if the assertion fails. The policy editor saves automatically.

15. Next, add and configure two property assertions within the All Of Assertion. From the editor pane, in the All Of Assertion row, click **Add Assertion**. Then,

from the **Assertion Options** drop-down menu, select **Property Assertion**, and click **Add**. Enter `updatedBy` in the **Property Name** field. This name is the name of the property in which the assertion is acting.

16. Now, embed an assertion in the Property Assertion that adds a Property Not Null Constraint. From the editor pane, in the PropertyAssertion row, click **Add Assertion**. Then, from the **Assertion Options** drop-down menu, select **Property Not Null Constraint**, and click **Add**, as shown in Figure 9-29.

Figure 9-29 Property Assertion with a Property Not Null Constraint

Archived

17. Configure the second Property Assertion using `updatingPersonLocation` as the property name. The result displays as shown in Figure 9-30.

Figure 9-30 Final Policy In the WSRR Policy Editor

Archived

18. Click **Publish**. When published, you will see a list of policy documents (Figure 9-31).

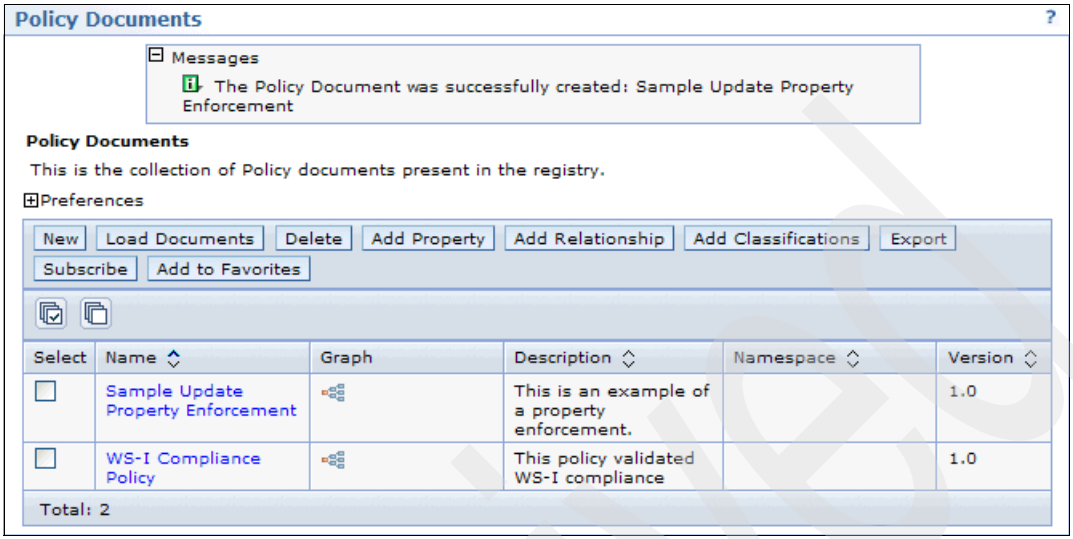


Figure 9-31 Policy Document Collection View

9.3.3 Testing and deploying governance policies

After publishing the document in the policy editor, you export the WS-Policy document from WSRR and then import the policy document into the profile configuration.

Exporting the WS-Policy document from WSRR

To export a WS-Policy file:

1. From the Policy Documents view, click the WS-Policy's name to open the Details view. Then, click the Content tab.
2. Click **Download Documents**, and save the file to a directory.

Deploying policy documents to a WSRR profile

You import the policy document into the active profile configuration by following these steps:

1. Click **Active Profile** → **Governance Policies** → **Load Governance Policy**, and then click **Browse** to locate the governance policy.
2. Provide a name for the policy. Click **OK**, as shown in Figure 9-32 on page 342.

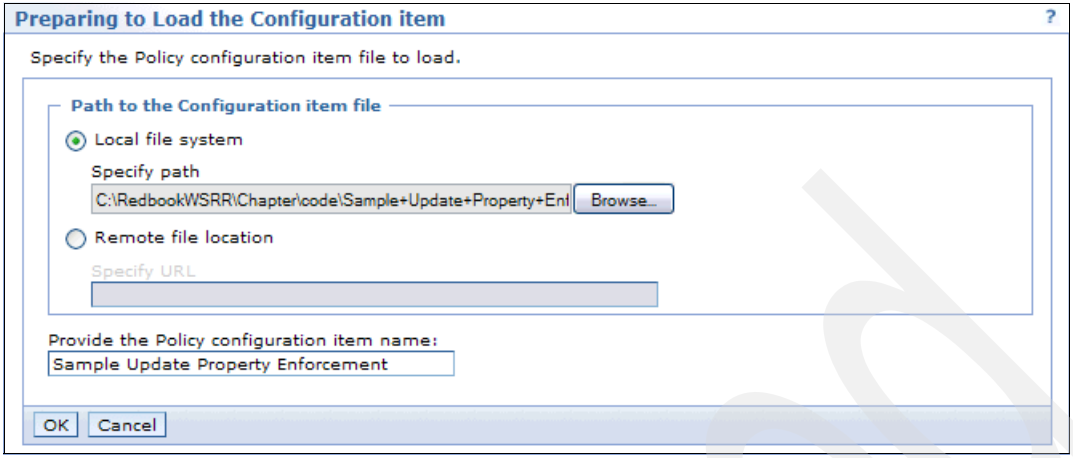


Figure 9-32 Load a Governance Validator Policy

Archived

Figure 9-33 shows of a list of the deployed policies.

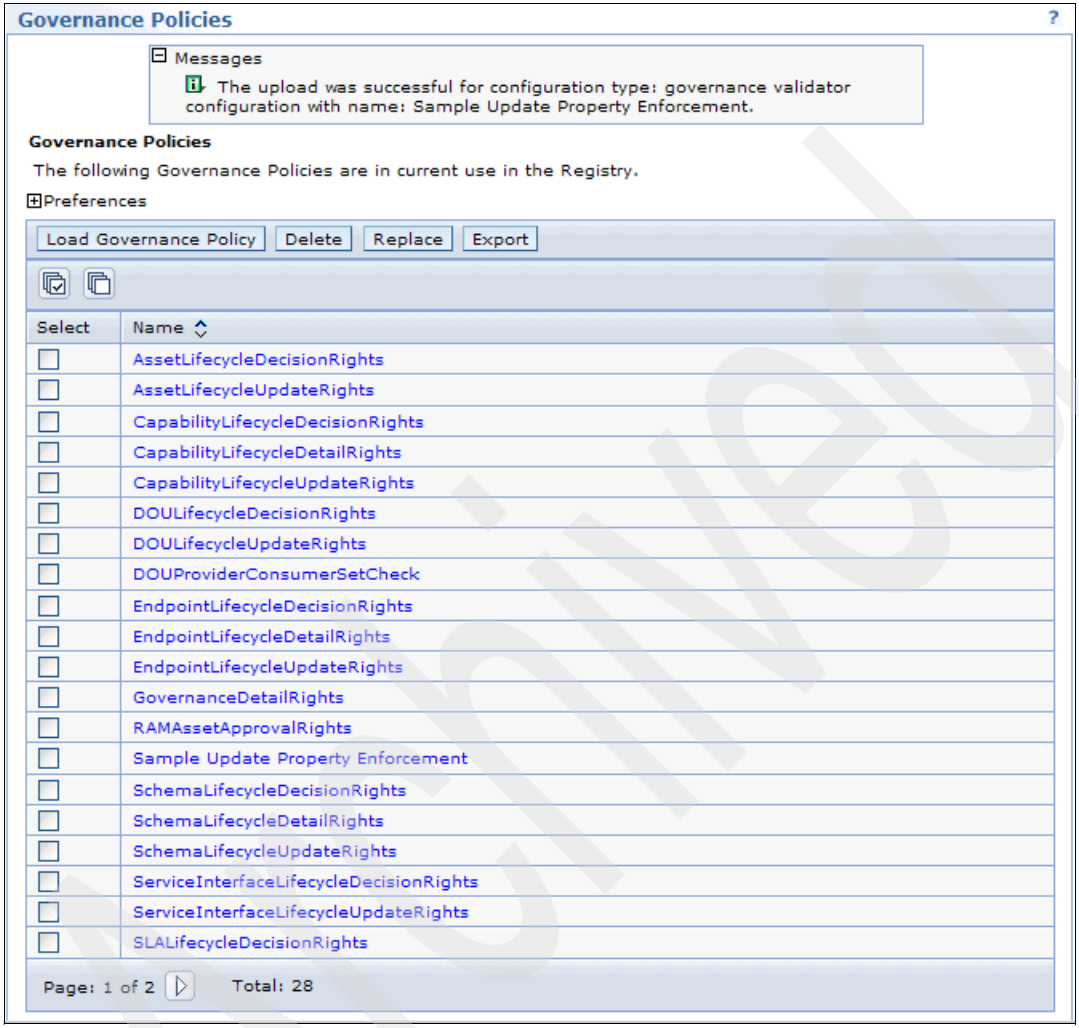


Figure 9-33 List of deployed policies

Archived

Testing the policies

This section describes how test the two governance policies that we created in previous sections of this chapter.

WS-I compliance policy

In 9.3.1, “Creating a WS-I compliance policy” on page 316, we created a WS-I compliance policy file that generates a compliance report when the WSDL file is created for the first time. The operation succeeds even if compliance fails. To test this policy, load a WSDL and view the compliance report as follows:

1. Change to the **SOA Governance** perspective. Click **Actions** → **Load Documents**, and then click **Browse** to locate the document, as shown in Figure 9-34.

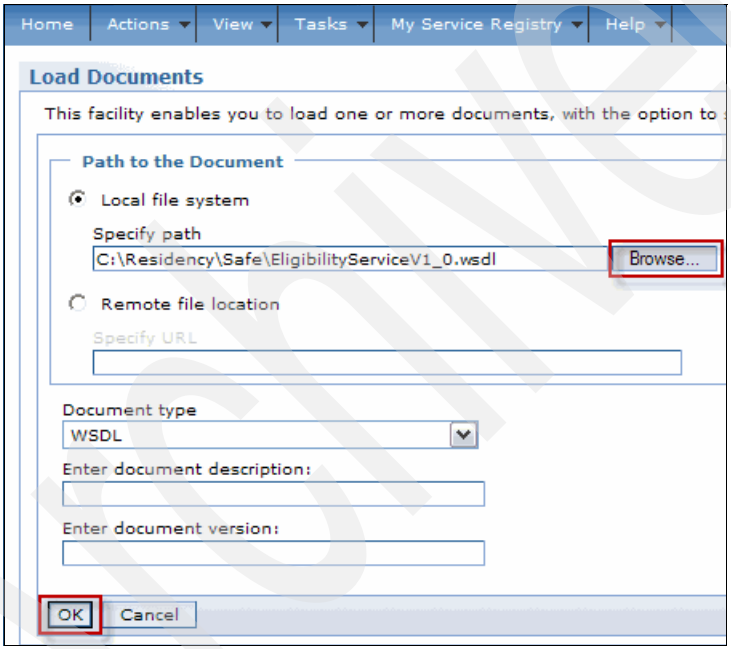


Figure 9-34 Loading documents

Archived

2. After you load the WSDL, click link for the WSDL to view the WSDLDocuments Details view. Note that `_WSIComplianceReport` is in the right column, as shown in Figure 9-35.

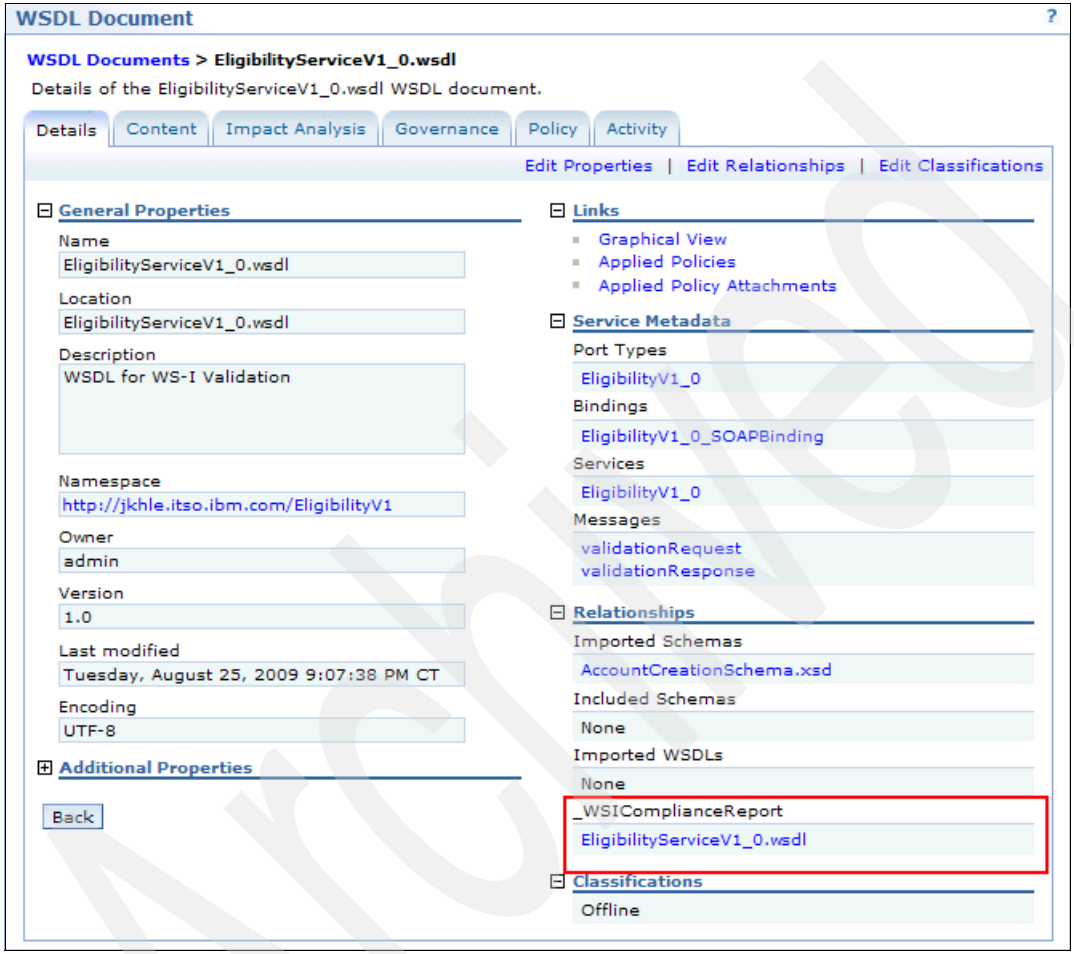


Figure 9-35 WSDLDocument Details View with WSIComplianceReport Link

Archived

3. Click **_WSIComplianceReport**. Then go to the Content tab to view report, as shown in Figure 9-36.

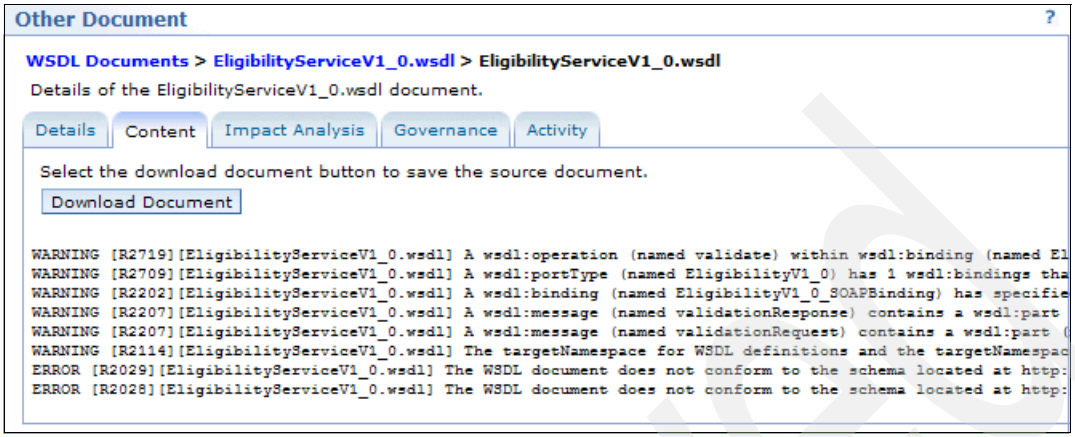


Figure 9-36 WS-I Compliance Report

You can download the report document.

Updated property enforcement policy

In 9.3.2, “Creating an updated property enforcement policy” on page 329, we created a WSRR service metadata governance policy that enforces mandatory non-blank values for the user-defined properties `updatedBy` and `updatingPersonLocation` on then Business Service entity when it is updated. To test this policy, create a new Business Service, and then edit the Business Service properties by modifying the description field. The edit should fail because the `updatedBy` and `updatingPersonLocation` properties do not exist on the Business Service entity.

Figure 9-37 shows the error message return. Note the message contains the text that was entered into the Policy Editor when creating this policy.

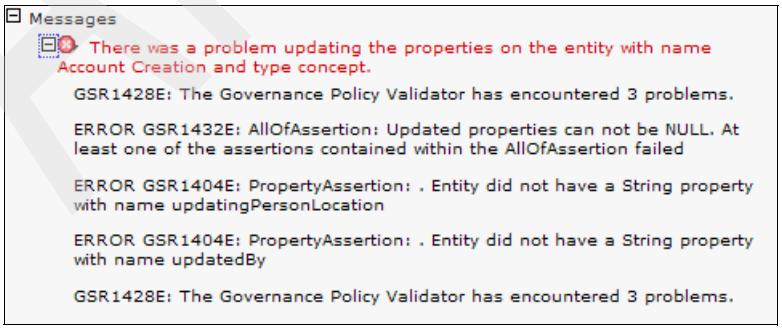


Figure 9-37 Governance policy validator error messages

Edit the Business Service properties again. Add the `updatedBy` and `updatingPersonLocation` properties and give each a value. The edit should then be successful.

9.4 References

Consult the following resources for more information:

- WS-Policy

  http://www.w3.org/Submission/WS-Policy/

- WS-Policy 1.5 Framework

  http://www.w3.org/TR/ws-policy

- WS-Policy Attachment

  http://www.w3.org/Submission/WS-PolicyAttachment/

Archived

Archived

10

Reports

This chapter provides step-by-step details about how to implement reports on various entities in WebSphere Service Registry and Repository (WSRR) using WebSphere Service Registry and Repository Studio (WSRR Studio). It includes the following topics:

- Customizing reports
- Configuring a WSRR location in WSRR Studio
- Creating a report project
- Using sample reports in WSRR Studio
- Configuring the sample reports

© Copyright IBM Corp. 2009. All rights reserved.

10.1 Customizing reports

Various users of WebSphere Service Registry and Repository (WSRR) can design and run reports on the different entities in WSRR using WSRR Studio. WSRR Studio takes advantage of features provided by the Business Intelligence and Reporting Tool (BIRT) for reporting on WSRR entities. BIRT is an open source, Eclipse-based reporting framework for Web applications, particularly those based on Java and J2EE.

For more information about BIRT, see:

http://www.eclipse.org/birt/phoenix

WSRR Studio uses the BIRT Report Designer (Figure 10-1) and the Web Viewer that comes with the Report Engine run time. You can use the BIRT Report Designer perspective to design the report, configure Data Sources, and view Report projects.

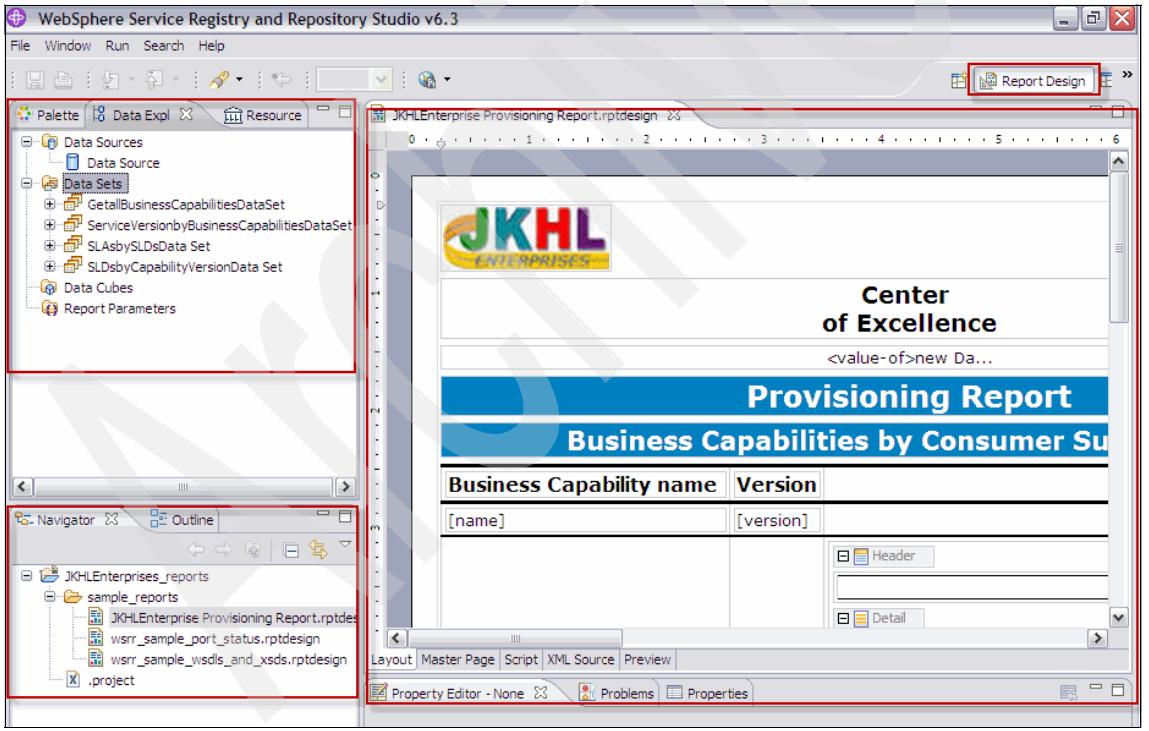


Figure 10-1 BIRT Report Designer perspective in WSRR Studio

The Web Viewer runs in the Report Engine run time, which allows you to view the report from WSRR Studio. Figure 10-2 shows the various reports available from WSRR Studio.

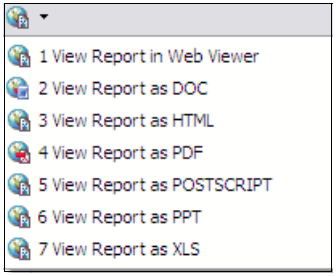


Figure 10-2 Render a report from WSRR Studio

There are two key components to a report:

- A data source
- A data set

A *data source* is the connection information to the reporting data. WSRR Studio provides the following data sources:

- Classic Models Inc. Sample Database
- JDBC Data Source
- Scripted Data Source
- WebSphere Service Registry and Repository (WSRR) Data Source
- XML Data Source

You can create reports for WSRR as you do for any other BIRT report, but you can use a WSRR data source to connect to the data in WSRR.

After you configure a data source, you use a dependent mechanism to contain a description of the reporting data. This mechanism is a *data set*. WSRR uses two types of Data Sets:

- WSRR Name Property Query Data Set

  This data set retrieves information using a stored named property query in WSRR. Before you can use this type of data set, you must first define the property query within WSRR. If additional properties from the entities that are queried need to be returned in the query, then after the query is saved, you need to add those properties to the query. We explain how to implement this type of data set in “Creating named property queries and user-defined property” on page 365.

Archived

- WSRR Free Form Graph Query Data Set

  This data set does not retrieve information using stored named property queries. Therefore, you do not have to create stored named property queries in WSRR to use this data set. The queries used to retrieve information for the WSRR Free Form Graph Query Data Set are defined directly in the data set using XPath. We explain how to implement this type of data set in “Free form graph query data sets” on page 383.

Reports can be deployed to the Tivoli Common Reporting Tool. For more information, see:

http://publib.boulder.ibm.com/infocenter/sr/v6r3/topic/com.ibm.sr.studio.doc/cwsr_mansrvce_studio_tcr.html

10.2 Configuring a WSRR location in WSRR Studio

For WSRR Studio to access resources in a WSRR server, you *must* configure a WSRR location in WSRR Studio before creating a data source for reports as follows:

1. Start WSRR Studio and select a workspace.
2. From the menu bar select **Window** → **Preferences**, as shown in Figure 10-3.

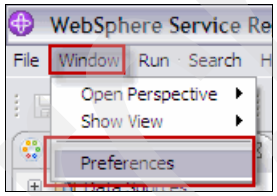


Figure 10-3 Setting preferences

Archived

3. In the Preferences dialog box, select **WebSphere Service Registry and Repository** from the left navigation, as shown in Figure 10-4.

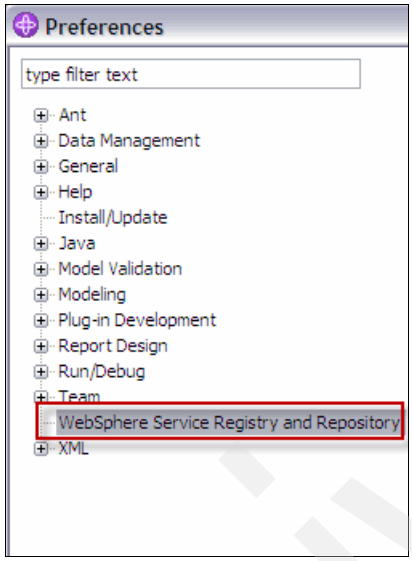


Figure 10-4 Select WebSphere Service Registry and Repository

Archived

4. In the right panel, the **WSRR Locations** pane displays. Click **Add** to add a new WSRR location, as shown in Figure 10-5.

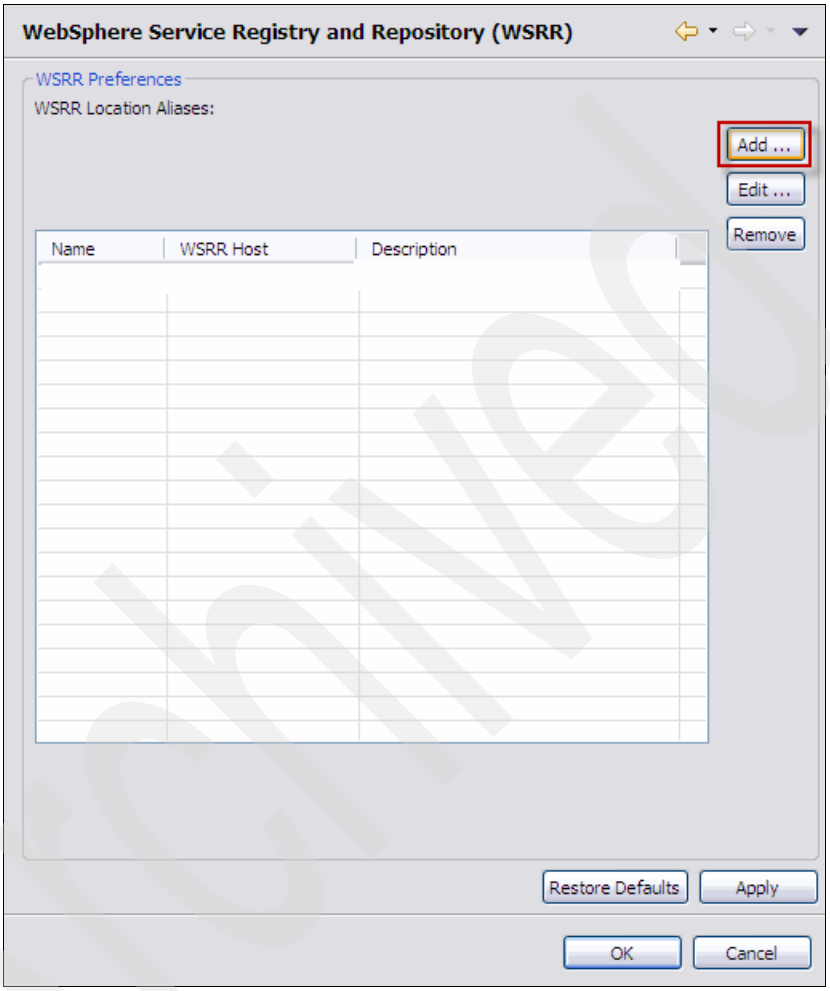


Figure 10-5 Add to a new WSRR location

5. In the Add WSRR location dialog box, enter the connection details on the General Properties tab (as shown in Figure 10-6):
   a. In the “Alias name” field, enter a meaningful name for the connecting WSRR server.
   b. In the “Description” field, enter a description of the connection.
   c. In the “Protocol” field, select the appropriate HTTP protocol. Choose **HTTP** for a non-secured WSRR, or choose **HTTPS** for a secure WSRR.

Archived

d. In the “Hostname” field, enter the fully-qualified host name or IP address (IPv4 or IPv6) of the connecting WSRR server.

> **Note:** If the WebSphere Application Server node name is different from the system name on which WebSphere Application Server is running, WSRR Studio will fail to resolve the IP address.
>
> To check the node name, compare the system name to the name of the node in WebSphere Application Server using the WebSphere Application Server administrative console.
>
> If you need to change the node name, edit the hosts file (`/etc/hosts` or `C:\windows\system32\drivers\etc\hosts`) on the computer that is running WSRR Studio and map the IP address of the WSRR server to the WebSphere Application Server node name. For example, if the IP address of the WSRR server is `10.20.198.52` and the WebSphere Application Server node name is `wasserver`, then append the following information to the hosts file:
>
> ```
> 10.20.198.52   wasserver
> ```

e. In the “Port” field, enter the port number of the WSRR listening port for the connecting WSRR server. By default, the port is 9080 for a non-secured WSRR configuration and 9443 for a secure WSRR configuration.

f. If security is enabled, provide security credentials so that WSRR Studio can communicate with a secure WSRR server. A default empty JKS trust store and key store are supplied that you can use with WebSphere Application Server certification.

> **Note:** For more information about using an alternative JKS trust store and key store, see the WebSphere Service Registry and Repository Information Center at:
>
> http://publib.boulder.ibm.com/infocenter/sr/v6r3/topic/com.ibm.sr.studio.doc/twsr_install_studio_config.html

g. Click **Test Connection** to check the connection to the WSRR server. The new connection is shown as `Successful` or `Failed`.

h. Click **Finish** to save the setting.

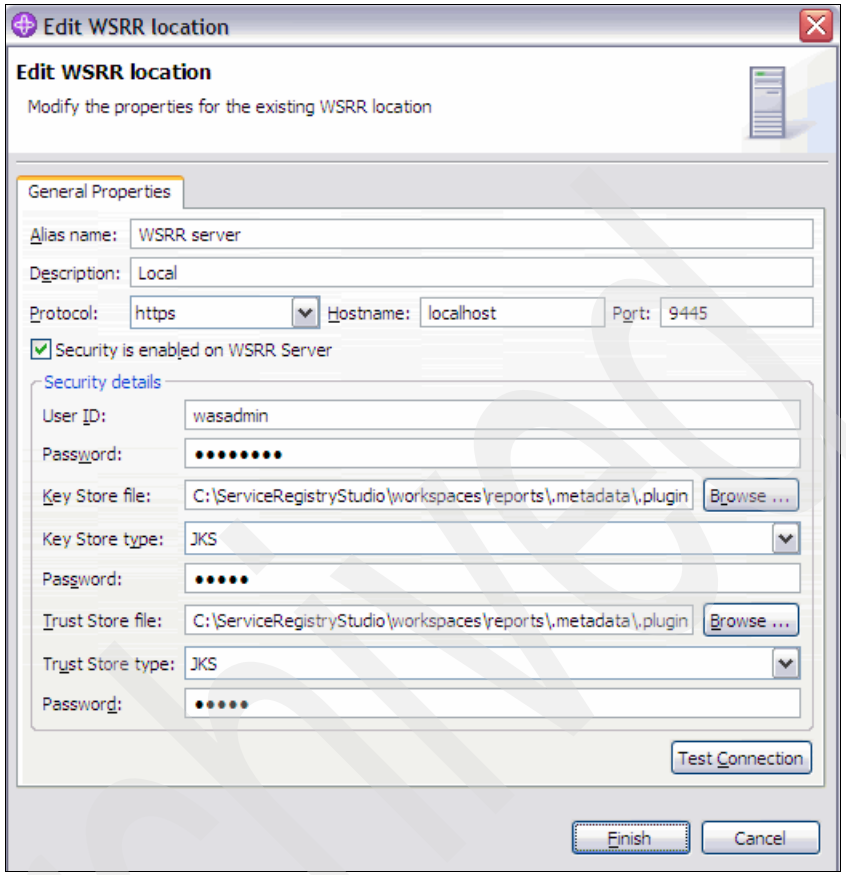


Figure 10-6 WSRR location properties

Archived

6. The new connection is listed on the WSRR locations page of the Preferences window, as shown in Figure 10-7. Click **OK**.

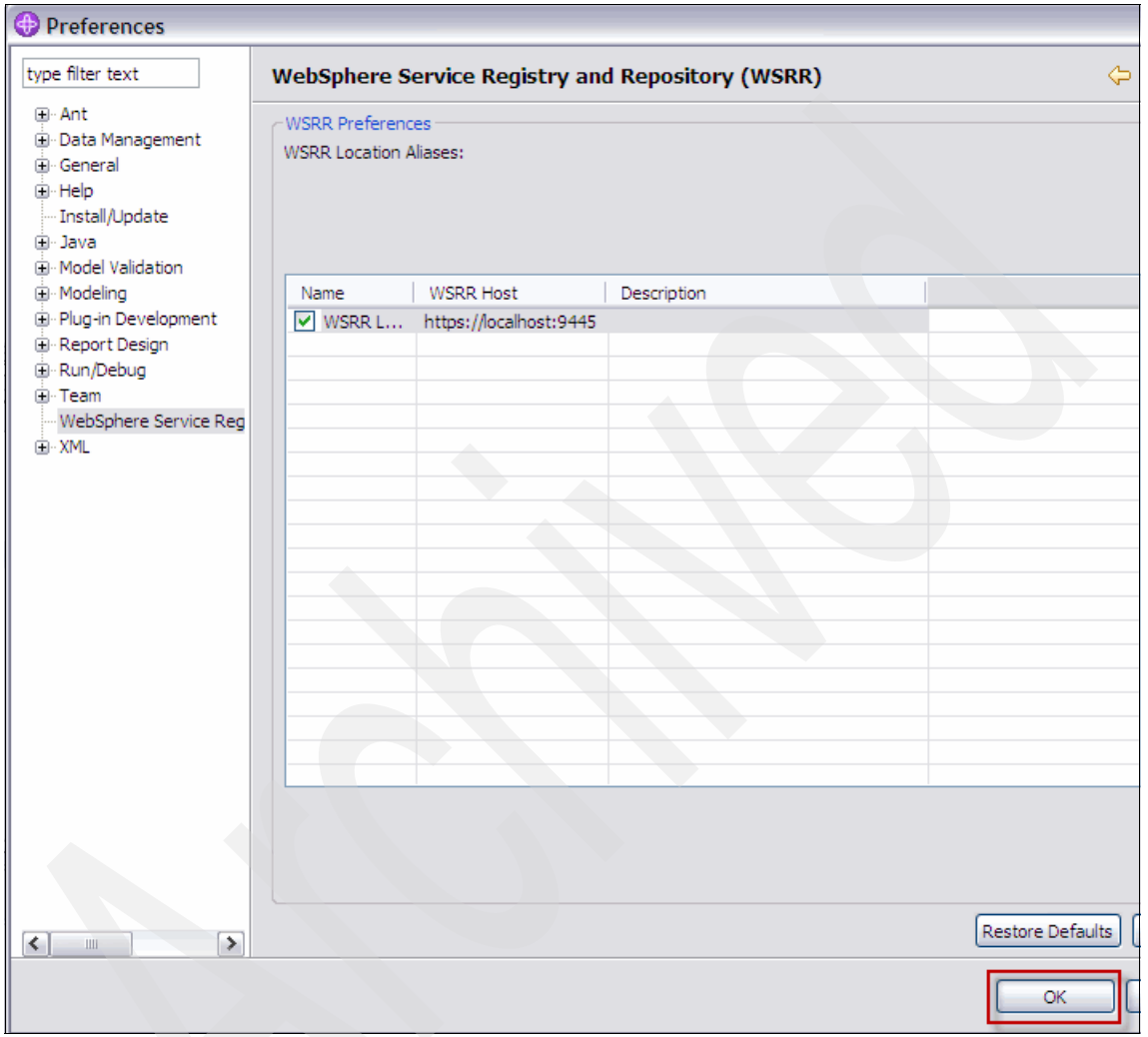


Figure 10-7 Listed WSRR location

Archived

10.3 Creating a report project

Reports and the resources related to those reports are contained in a *report project*. To create a report project, follow these steps:

1. In WSRR, click to open the Perspectives window. Then, select **Report Design**, and click **OK** as shown in Figure 10-8.

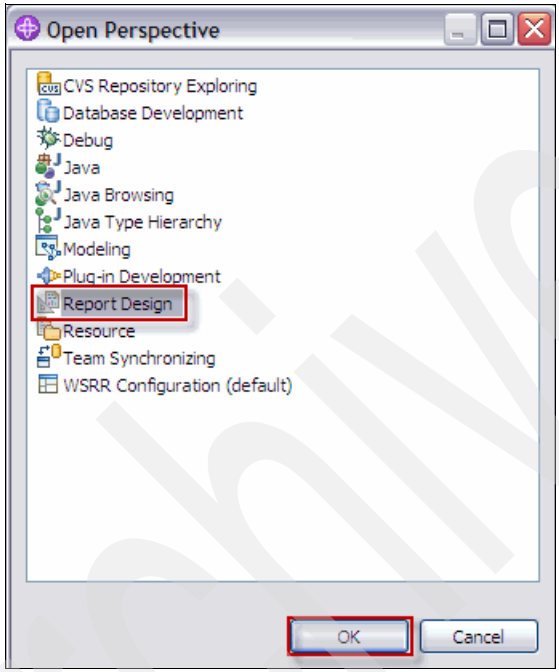


Figure 10-8 Selecting the Report Design perspective

2. Select **File** → **New** → **Other** as shown in Figure 10-9.

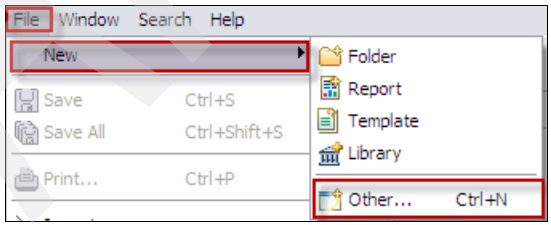


Figure 10-9 Creating a report project

Archived

3. Select **Business Intelligence and Reporting Tools** → **Report Project**, and click **Next** as shown in Figure 10-10.

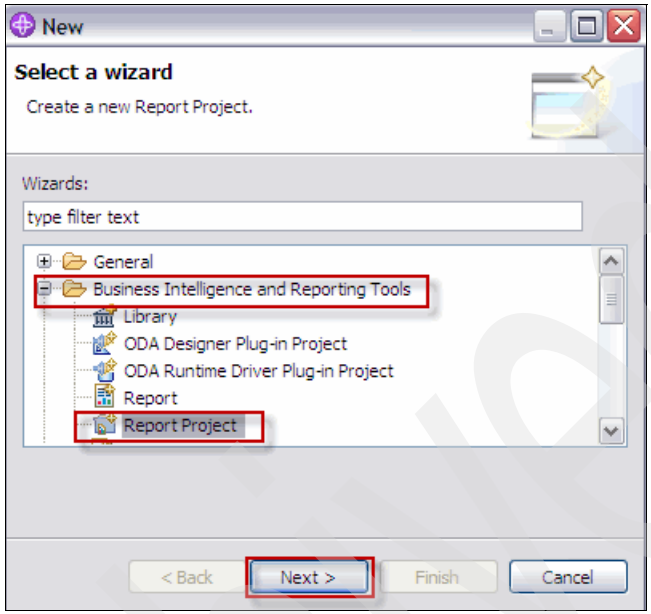


Figure 10-10 Selecting Report Project

Archived

4. Enter `JKHLEnterprises_reports` as the project name, and click **Finish** as shown in Figure 10-11.

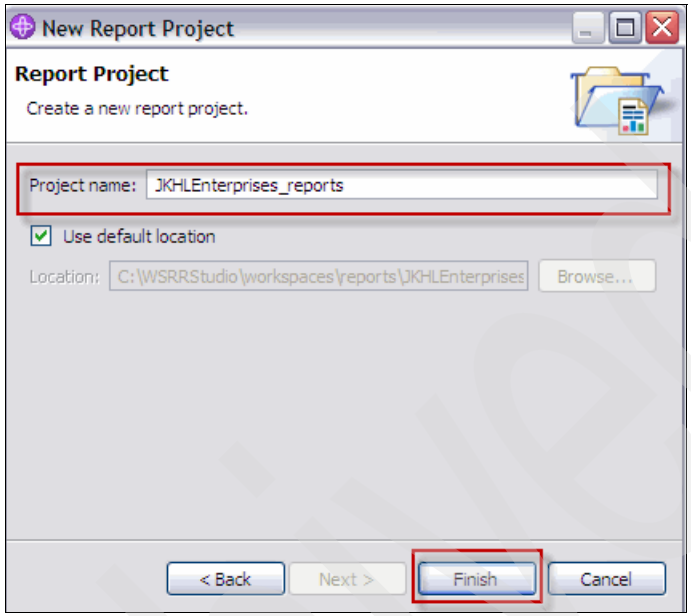


Figure 10-11 Entering the report project name

After the project is created successfully, it is listed in the Navigator view, as shown in Figure 10-12.

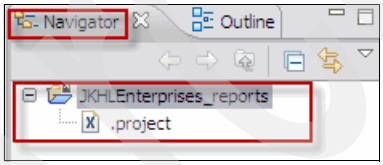


Figure 10-12 JKHLEnterprises project listed

10.4 Using sample reports in WSRR Studio

WSRR Studio provides sample reports as a starting point for reporting information in WSRR server. To use the sample reports, follow these steps:

1. In WSRR Studio, click to open the Perspectives window. Then, select **Report Design**, and click **OK** as shown in Figure 10-13.

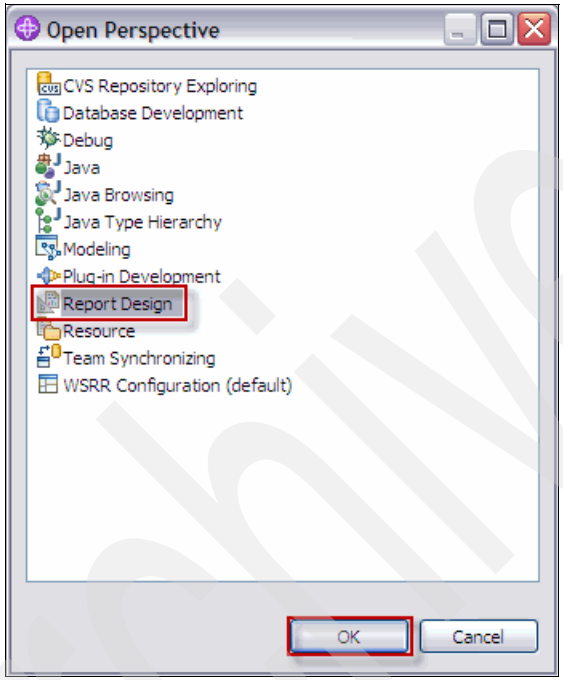


Figure 10-13 Selecting the Report Design perspective

2. In the Navigator view, right-click **JKHLEnterprises_reports**, and select **Import** as shown in Figure 10-14.

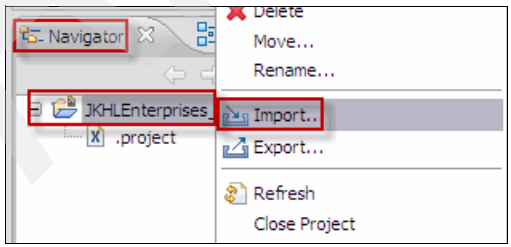


Figure 10-14 Selecting Import

Archived

3. Expand **General**, select **File System**, and click **Next** as shown in Figure 10-15.

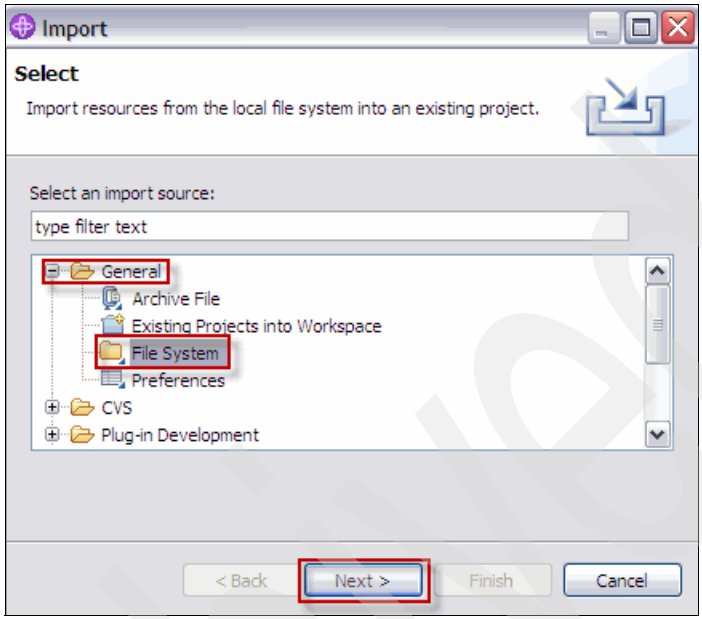


Figure 10-15 Selecting File System for sample reports

Archived

4. Click **Browse** to locate the `sample_reports` directory. Select **sample_reports**, and click **Finish**, as shown in Figure 10-16.

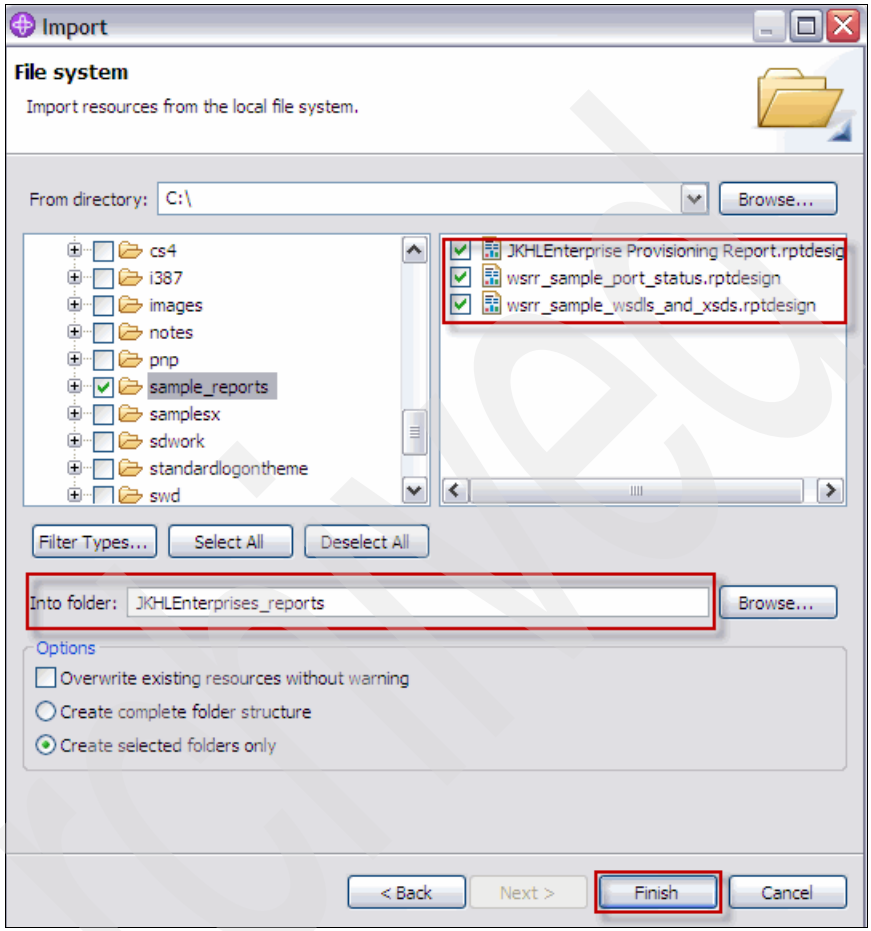


Figure 10-16 Selecting the sample reports

After the files import, they are listed under the JKHLEnterprises_reports project in the Navigator view as shown in Figure 10-17.

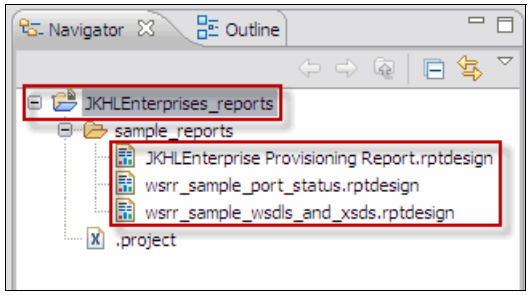


Figure 10-17 Sample reports listed in the JKHLEnteprises_reports project

10.5 Configuring the sample reports

WSRR Studio provides two sample reports:

- `wsrr_sample_port_status.rptdesign`

  Contains the *named property query sample* report
- `wsrr_sample_wsdls_and_xsds.rptdesign`

  Contains the *free form graph query sample* report

This section describes how to create three reports:

- WSDL Port Availability Report sample report
- List of WSDLs and their associated XSDs sample report
- JKHLE provisioning report

10.5.1 WSDL Port Availability Report sample report

This sample report shows the status of WSDL ports for any installed services, using a simplified means of detailing the current usage level of a WSDL port. To use this report, you need to meet the following requirements:

- WSDL files in WSRR
- Named property query called `getWSDLPortAvailability` saved in WSRR
- User-defined property called `pAvailability` with value of either green, amber, or red on each entity that displays on the report
- A WSRR data source used to access WSRR data using a connection to the WSRR location

Creating named property queries and user-defined property

Because the WSRR reporting plug-in retrieves information using stored named property queries, you must first define property queries within WSRR in order to use this sample report. The actual queries that you define are specific to the information that displays in the report. A WSRR administrator usually creates these queries, but anyone who has access to the entities that are queried can create these queries.

To create the necessary queries, follow these steps:

1. Using a Web browser, open the WSRR user interface console.
2. To change to the **Administrator perspective**, select **Administrator** in the **Perspective** list as shown in Figure 10-18.

Figure 10-18 Selecting the Adminstrator perspective

3. Then, select **Actions** → **Query Wizard** as shown in Figure 10-19.

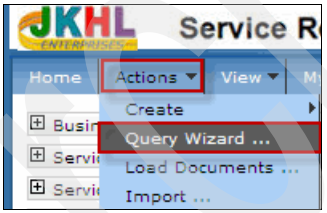


Figure 10-19 Selecting Query Wizard

Archived

4. Select **WSDL Ports** as the entity type, as shown in Figure 10-20, and then click **Next**.

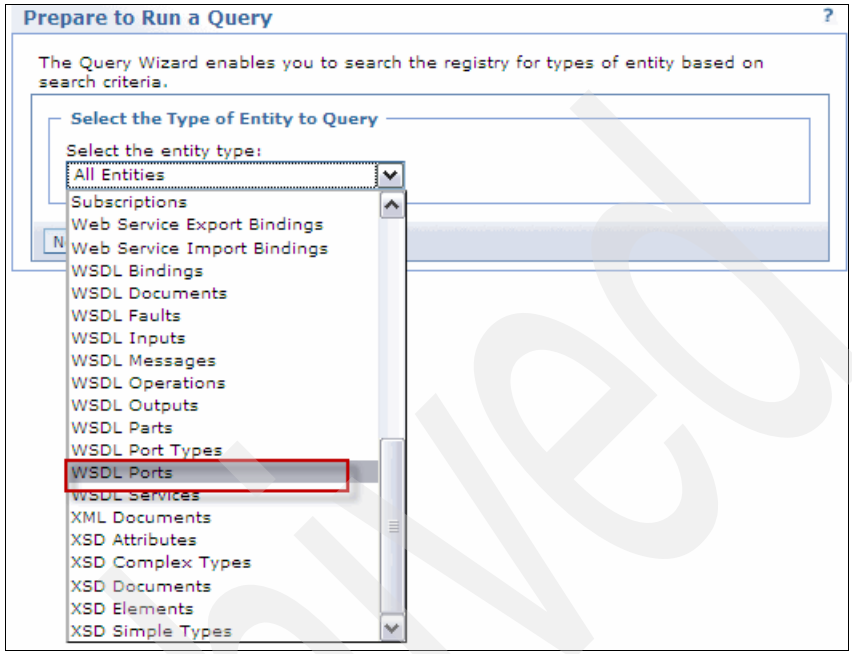


Figure 10-20 Selecting WSDL Ports for property query

Archived

5. In the ‘Name” field, enter an asterisk (*). In the “Property name” field, enter `availability`. Then, click **Next**, as shown in Figure 10-21.

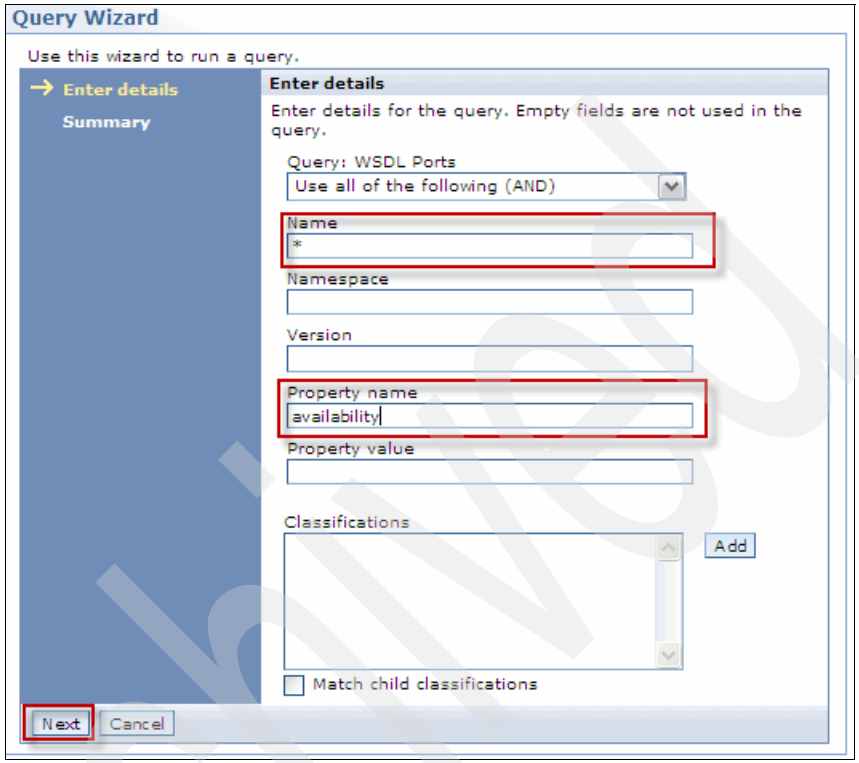


Figure 10-21 Entering search criteria for query

6. In the Summary page, click **Finish** as shown in Figure 10-22.

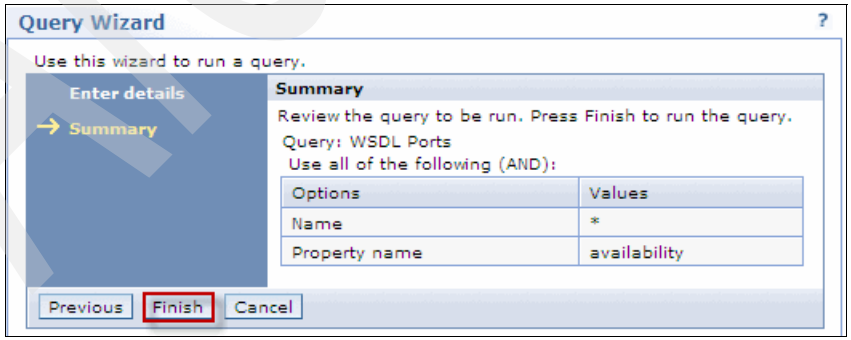


Figure 10-22 Query summary

Archived

7. In the “Name” field, enter `getWSDLPortAvailability`, and click **Save** as shown in Figure 10-23.

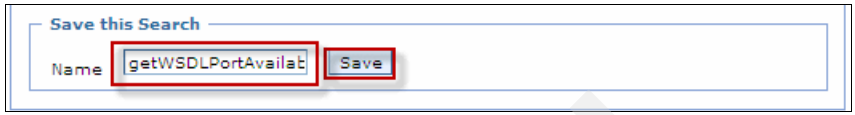


Figure 10-23 Saving the getWSDLPortAvailability query

8. Click **My Service Registry**, and select either **My Saved Searches** or **All Saved** Searches (as shown in Figure 10-24).

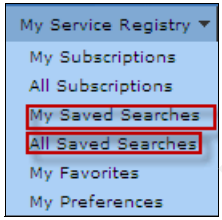


Figure 10-24 Viewing saved queries

9. Select **getWSDLPortAvailability** as shown in Figure 10-25.

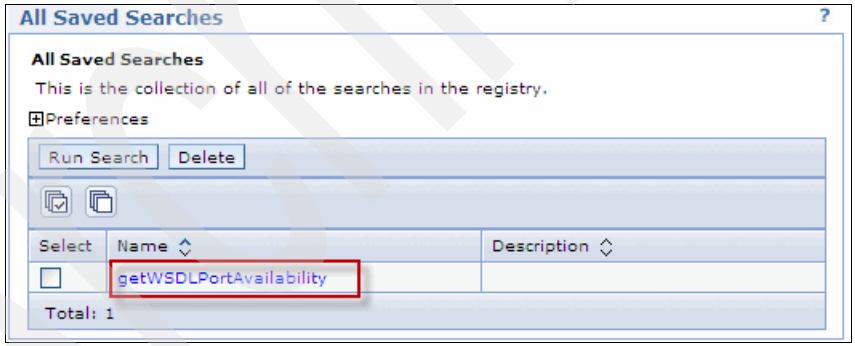


Figure 10-25 List of saved queries

Archived

10. Click **Edit Properties** on the Details tab, as shown in Figure 10-26.

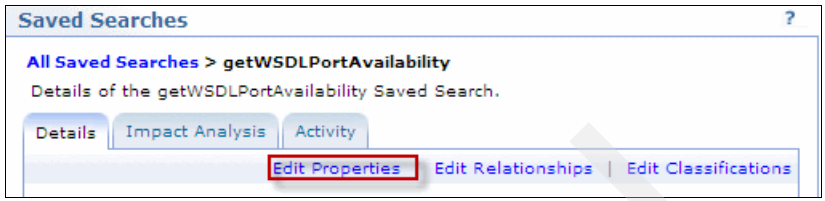


Figure 10-26 Editing saved query properties

11. Expand **Additional Properties**, and click **Add Property** as shown in Figure 10-27.

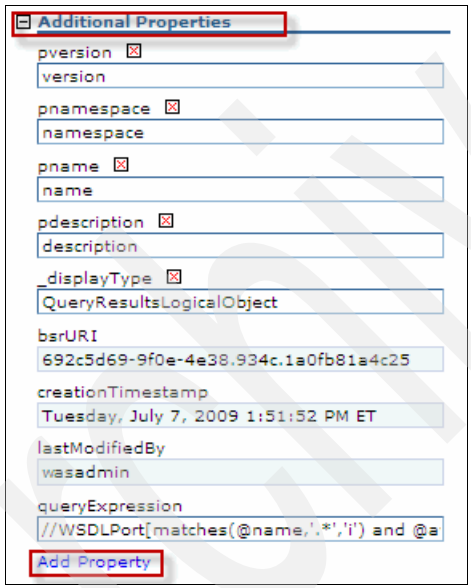


Figure 10-27 Adding an additional property

12. In the “Name” field, enter any value to identify the type of property that is returned. For this example, enter `pAvailability`, and click **Add** as shown in Figure 10-28.

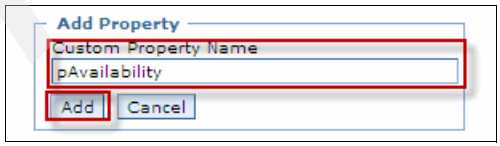


Figure 10-28 Entering pAvailability in the Name field

Archived

13. In the “Value” field, enter the name of the property that you want to be returned. In this case, enter `availability`, and click **OK** as shown in Figure 10-29.

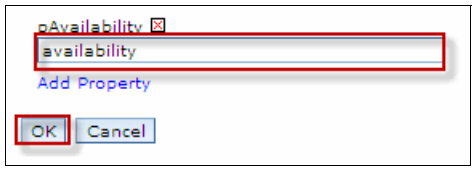


Figure 10-29 Entering availability in the Value field

Creating a user-defined property for each WSDL

For the query to return a status on the availability of WSDL ports, each WSDL Port must have a property called *availability*. Because this property is not part of the Business Model properties, you need to create a user-defined property for each WSDL port.

For each WSDL that you want included in the report, complete these steps:

1. In the WSRR user interface console click **View** → **Service Metadata** → **WSDL** → **Ports**, as shown in Figure 10-30.

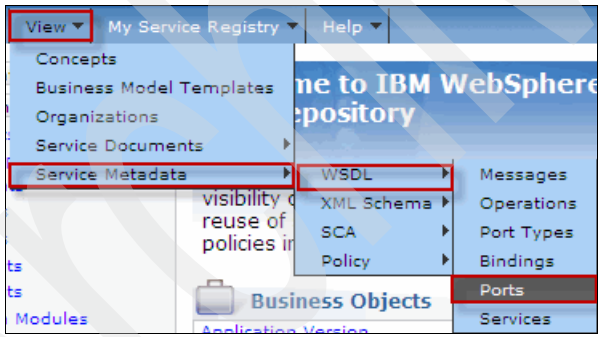


Figure 10-30 Selecting ports

Archived

2. In the Ports window, select **AccountCreationServiceV1_0_ProductionPort** as shown in Figure 10-31.

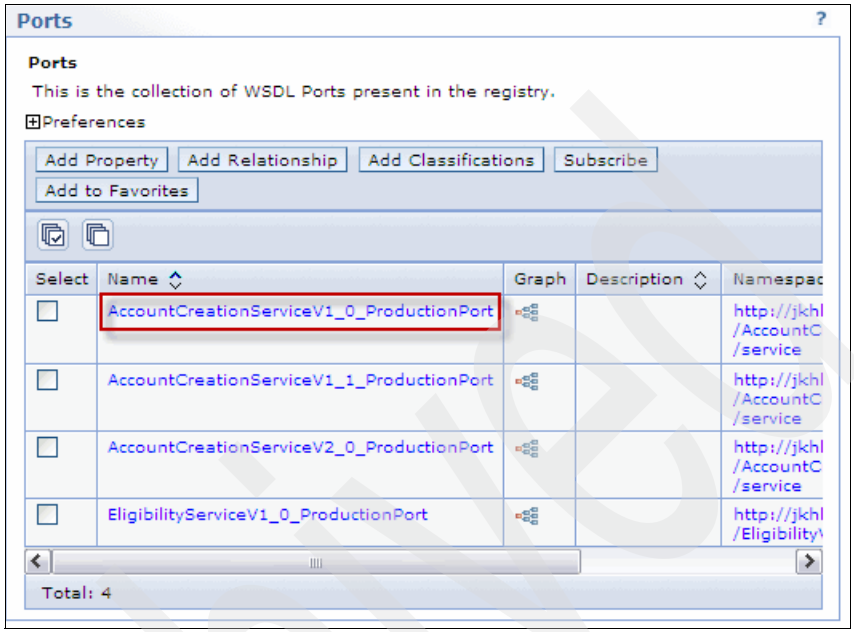


Figure 10-31 Selecting AccountCreationServiceV1_0_ProductionPort

3. Click **Edit Properties** on the Details tab, as shown in Figure 10-32.

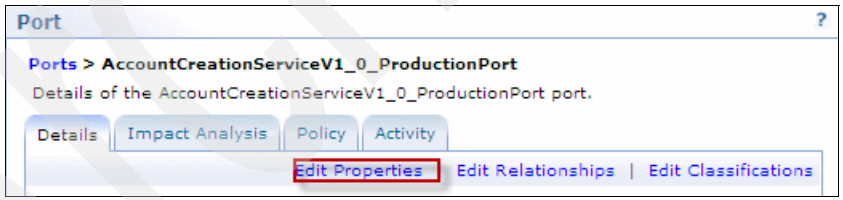


Figure 10-32 Editing properties

Archived

4. Expand **Additional Properties**, and select **Add Property** as shown in Figure 10-33.

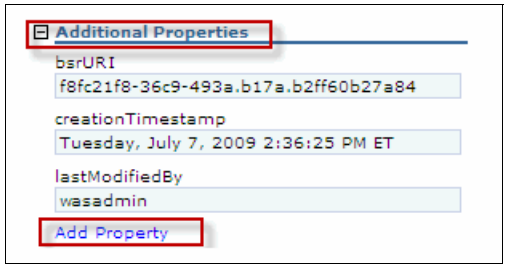


Figure 10-33 Adding a property

5. Enter pAvailability in the “Name” field, as shown in Figure 10-34.

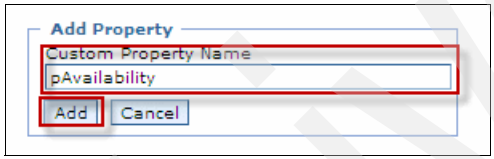


Figure 10-34 Entering the property name

6. In the “Value” field, provide a status. The sample report provides information about ports that have an availability of either red, amber, or green as a value. Click **OK**.

To provide the report with more than one document on which to report, repeat these steps for multiple WSDL documents and ports.

Configuring a WSRR data source

To configure a WSRR data source, follow these steps:

1. Double-click **wsrr_sample_port_status.rptdesign** to open the report. Then, in the Data Explorer view, right-click **Data Sources**, and select **New Data Source**, as shown in Figure 10-35.

> **Note:** The file wsrr_sample_port_status.rptdesign is supplied in the additional materials accompanying this book. To obtain the additional materials see Appendix B, “Additional material” on page 621.

Archived

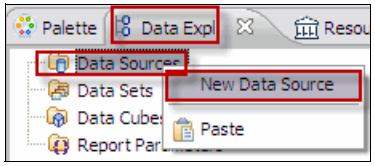


Figure 10-35 New Data Source

2. Select the “Create from a data source type in the following list” option. Then, click **WSRR Data Source**, and leave the default name of *Data Source*, as shown in Figure 10-36. Click **Next**.

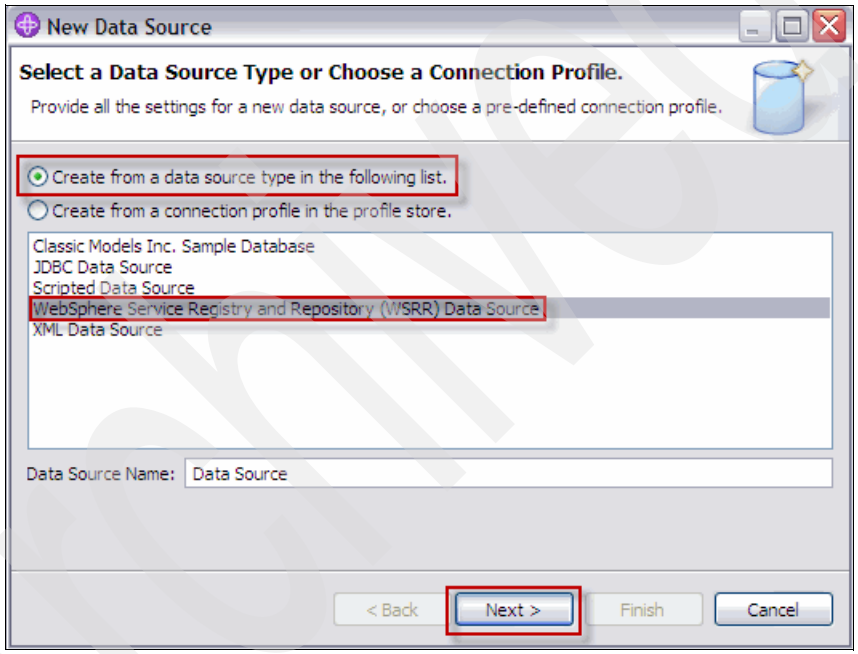


Figure 10-36 Selecting the WSRR Data Source type

Archived

3. Select the previously configured WSRR Location, and click **Finish** as shown in Figure 10-37.

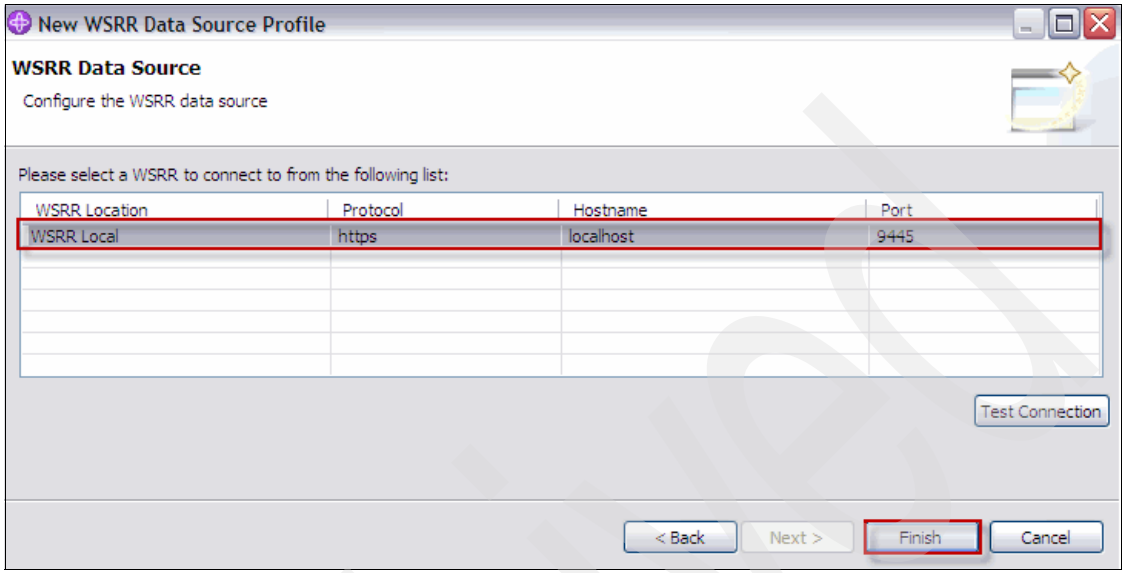


Figure 10-37 Selecting the WSRR Location

Creating the data set

You use this data set to pull data for WSRR using the named property query (`getWSDLPortAvailability`) that you created previously. This data set already exists in the imported reports. To create the data set, follow these steps:

1. In the Report Design perspective, right-click **Data Sets**, and select **New Data Set** as shown in Figure 10-38.

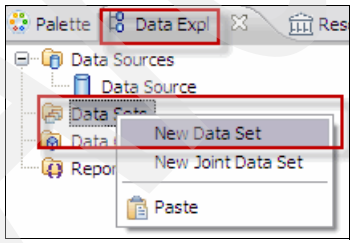


Figure 10-38 Selecting New Data Set

2. Set the Data Set Name to *DataSet*. Select the previously created WSRR data source, **WSRR Named Property Query Data Set**, as the Data Set Type, and then click **Next** (Figure 10-39).

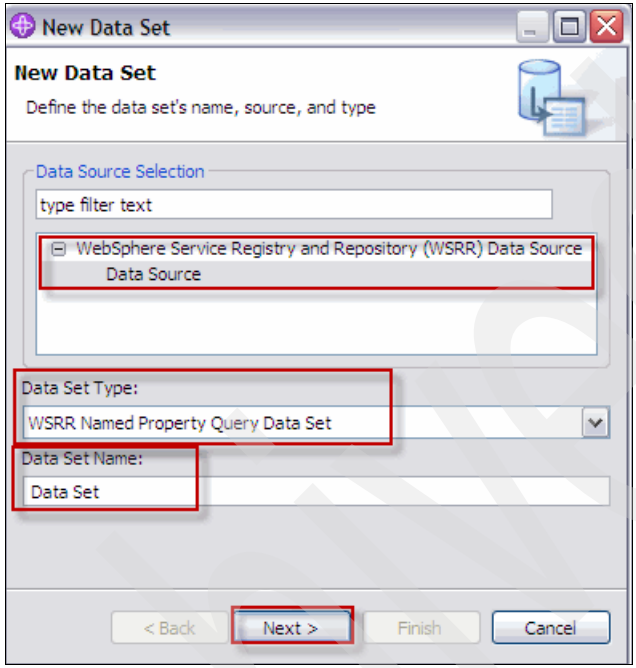


Figure 10-39 Entering the data set properties

Archived

3. Select the previously defined `getWSDLPortAvailability` query, and click **Finish** as shown in Figure 10-40.

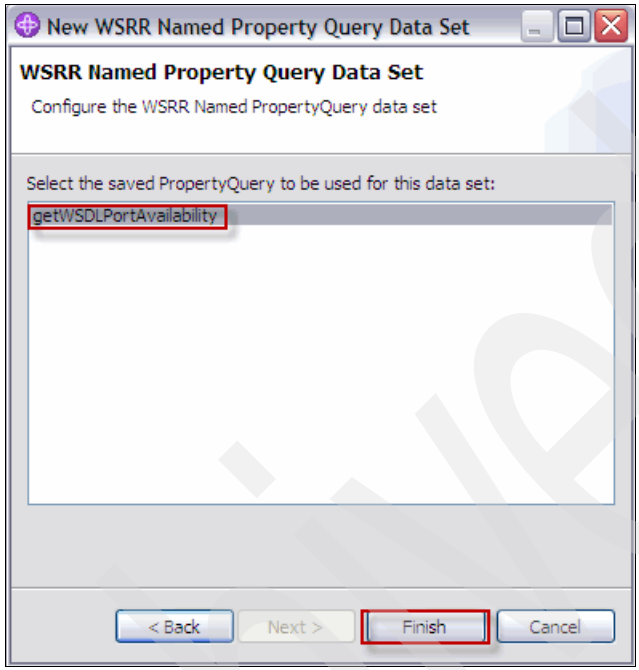


Figure 10-40 Saving the data set

Archived

4. In the Edit Data Set window, select **Preview Results** to ensure that the previously defined query (`getWSDLPortAvailability`) is working correctly and returning data as expected, as shown in Figure 10-41.

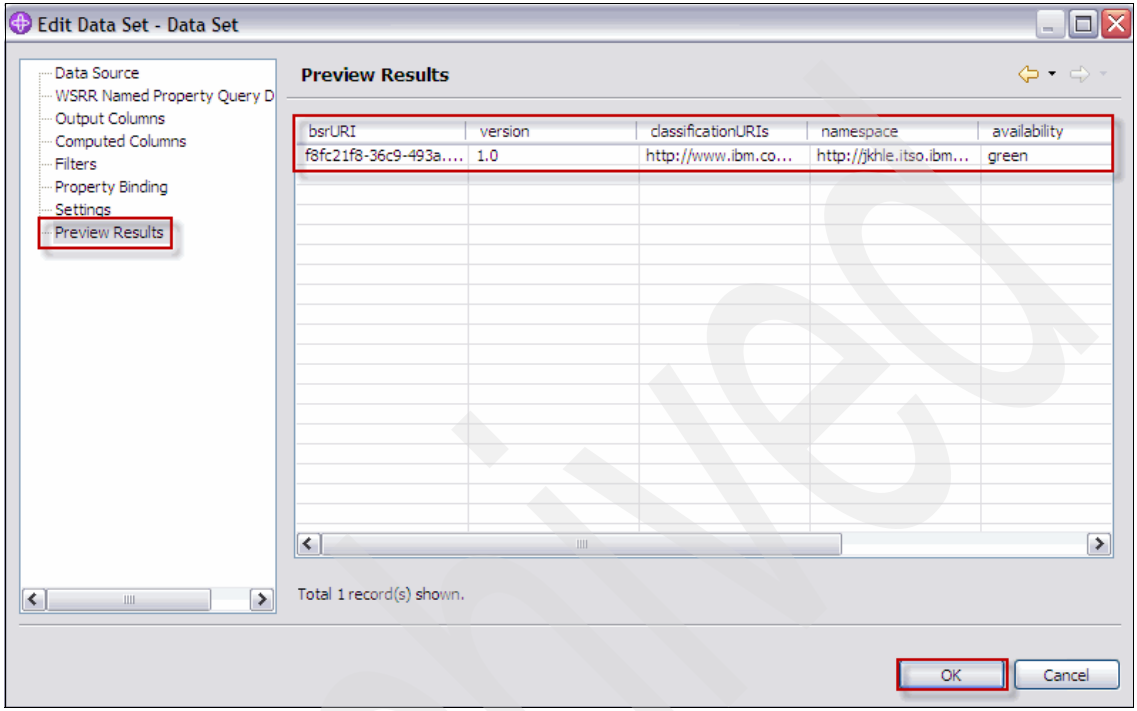


Figure 10-41 Selecting Preview Results

Archived

Figure 10-42 depicts the query results as a generated pie-chart.

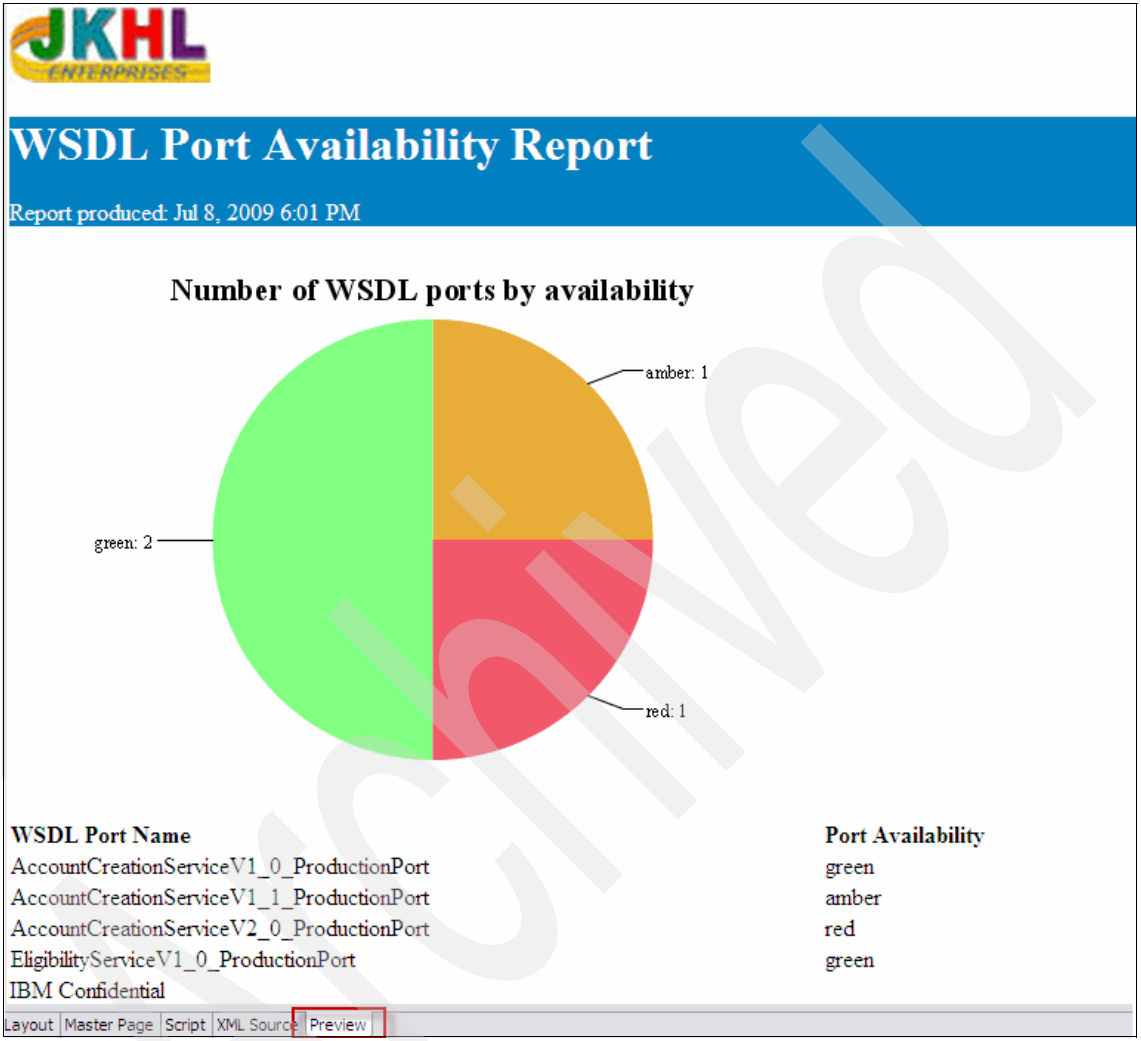


Figure 10-42 Results in a generated pie-chart

10.5.2 List of WSDLs and their associated XSDs sample report

This sample report produces a basic table list of all WSDL documents and all the associated XSD documents in a WSRR instance. The sample demonstrates how you can use free form graph query data sets. The sample is configured with data sets and a partially-configured data source, but it does not come with any data

against which you can run the report. You need to create this data manually. Before you can use this sample report, you must have WSDL documents and XSD documents in WSRR.

Creating a new data source

To create a data source, follow these steps:

1. In WSRR Studio, click to open the Perspectives window. Then, select **Report Design**, and click **OK** as shown in Figure 10-43.

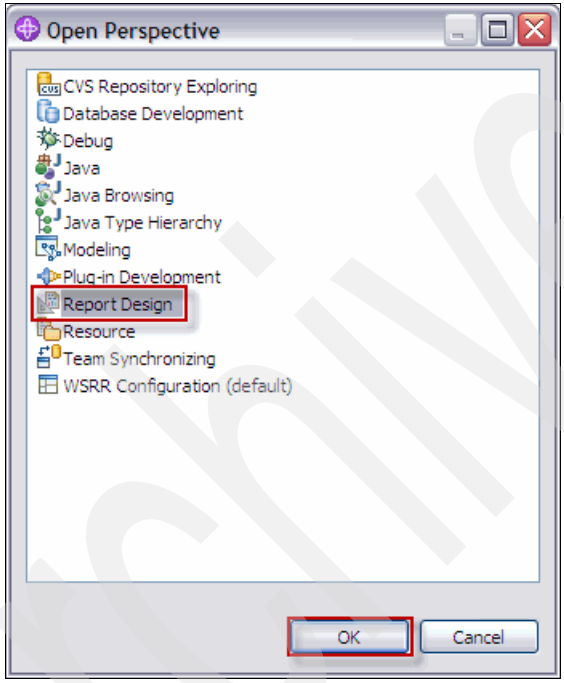


Figure 10-43 Selecting Report Design

2. Double-click **wsrr_sample_wsdls_and_xsds.rptdesign** to open this report. Then, in the Data Explorer view, right-click **Data Sources** and select **New Data Source** as shown in Figure 10-44.

> **Note:** The file wsrr_sample_wsdls_and_xsds.rptdesign is supplied in the additional materials accompanying this book. To obtain the additional materials see Appendix B, “Additional material” on page 621.

Archived

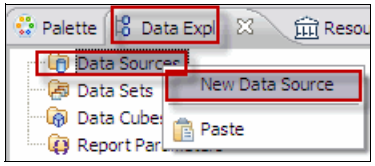


Figure 10-44 Selecting New Data Source

3. Select the “Create from a data source type in the following list” option. Then, click **WebSphere Service Registry and Repository (WSRR) Data Source**, and leave the default name of *Data Source*. Click **Next**, as shown in Figure 10-45.

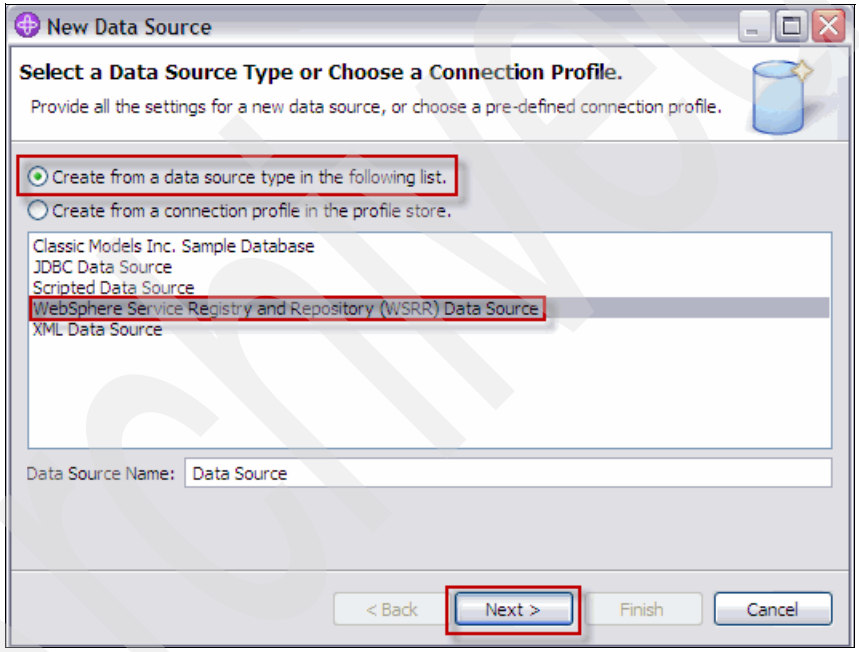


Figure 10-45 Selecting Data Source properties

Archived

4. Select **WSRR Local**, and click **Finish** as shown in Figure 10-46.

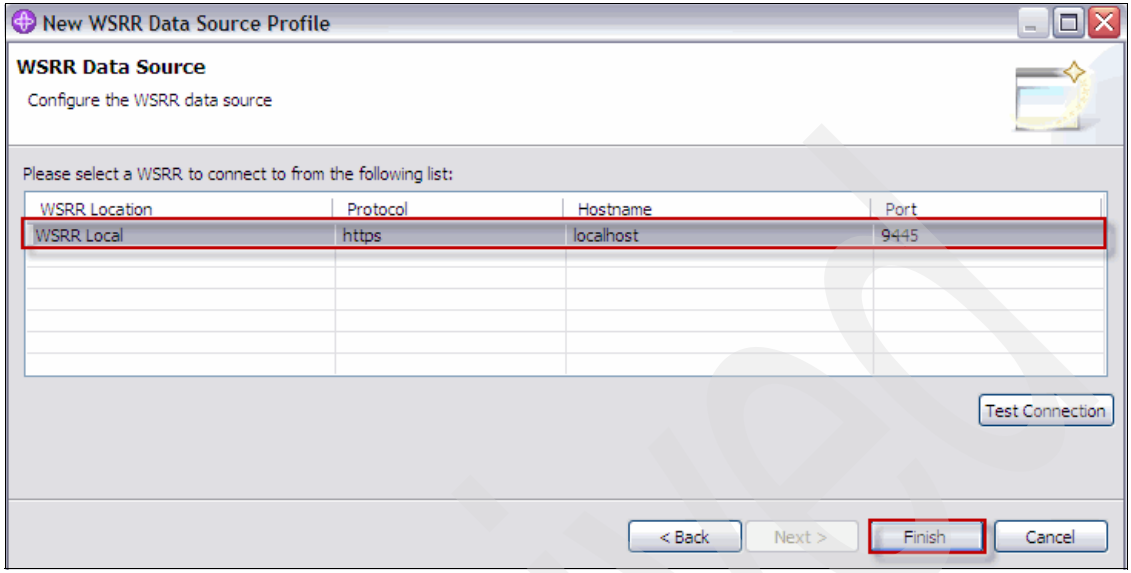
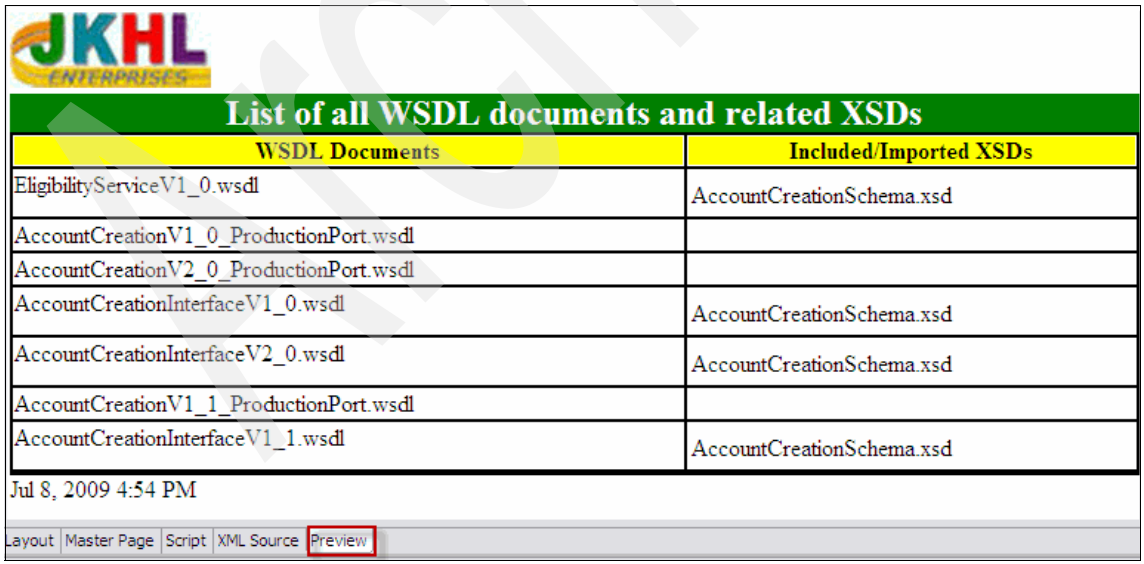


Figure 10-46 Selecting WSRR Location

5. When the data source is configured correctly, click **Preview** to view the report of a list of all WSDLs and their related XSDs in the WSRR Location, as shown in Figure 10-47.



List of all WSDL documents and related XSDs

WSDL Documents	Included/Imported XSDs
EligibilityServiceV1_0.wsdl	AccountCreationSchema.xsd
AccountCreationV1_0_ProductionPort.wsdl	
AccountCreationV2_0_ProductionPort.wsdl	
AccountCreationInterfaceV1_0.wsdl	AccountCreationSchema.xsd
AccountCreationInterfaceV2_0.wsdl	AccountCreationSchema.xsd
AccountCreationV1_1_ProductionPort.wsdl	
AccountCreationInterfaceV1_1.wsdl	AccountCreationSchema.xsd

Jul 8, 2009 4:54 PM

Layout | Master Page | Script | XML Source | Preview

Figure 10-47 Results from the sample report

Note: If the report fails, click **Layout**, double-click **Data Source**, and click **Test Connection**, as shown in Figure 10-48.

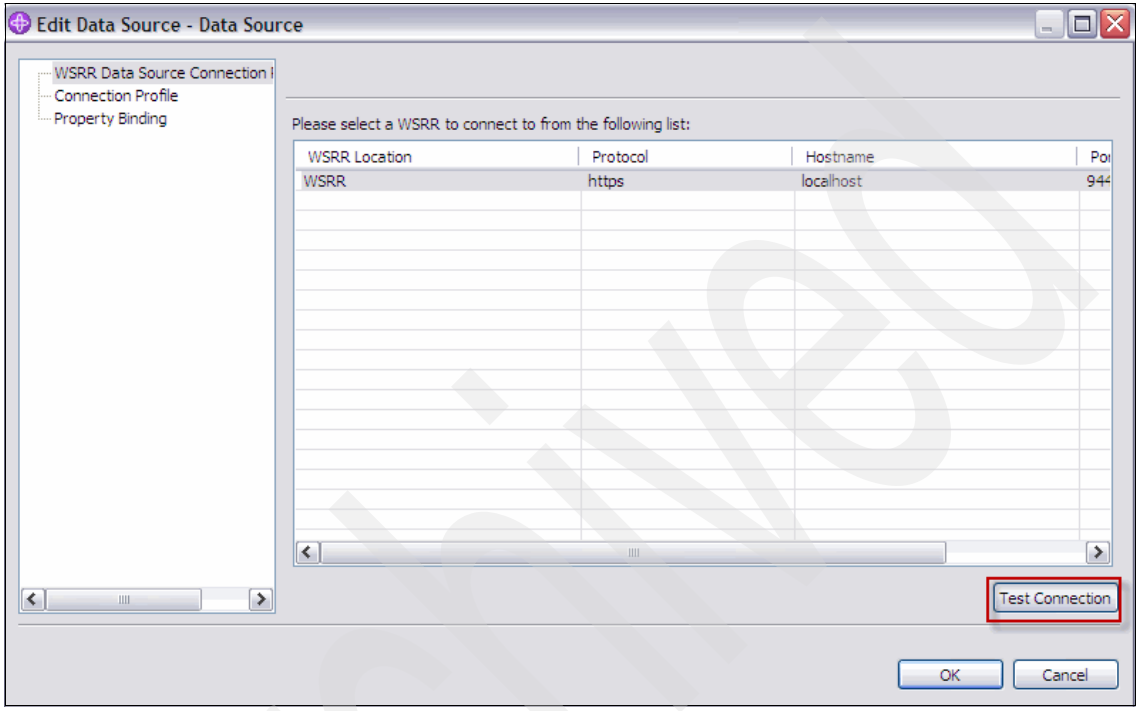


Figure 10-48 Verifying the test connection

10.5.3 JKHLE provisioning report

The provisioning report displays information about a service's available service level definition (SLD) and the service level agreement (SLA) which are subscripted to it. The report alerts in red if the total of the SLA's Maximum Message per day exceeds the Certified Maximum Messages per day, which alerts the operations team that more resources need to be provisioned for this service.

This section discusses some of the concepts used to create the JKHLEnterprises provisioning report.

Free form graph query data sets

The sample provisioning report, uses WSRR free form graph data sets to generate the data in the final report.

To create the correct queries to retrieve the desired data, it is important to understand the entities and their properties and relationships in the governance enablement and service models in the governance enablement profiles. This report uses the business capability, capability version, SLD, and SLA entities, as shown in Figure 10-49.

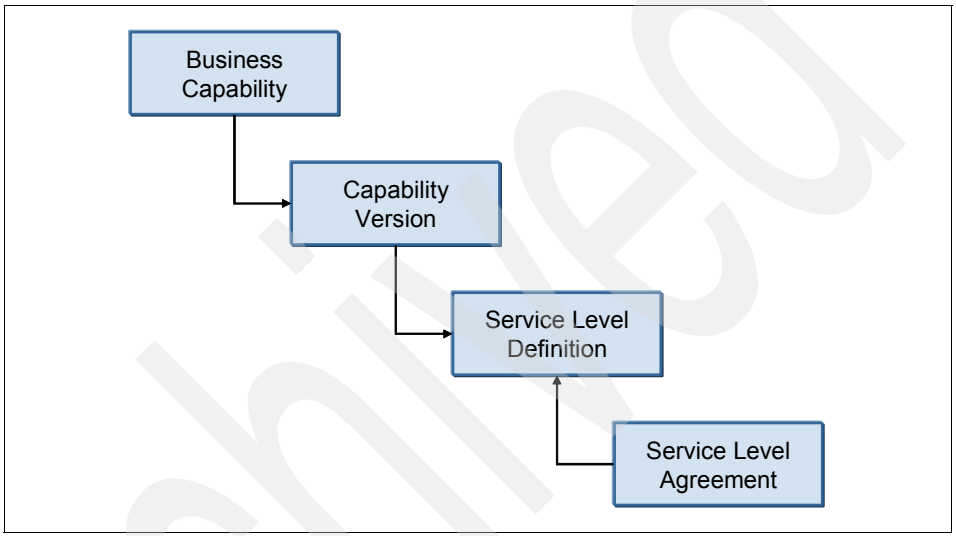


Figure 10-49 JKHLE provisioning report entities

Table 10-1 describes how to use the entities in the WSRR free form graph data sets in order for them to display in this report.

Table 10-1 JKHLEnterprise provisioning report data Sets

Data set name	Description	XPath
`GetallBusinessCapabilities DataSet`	Free form graph query that retrieves all business capability entities with a classification of business capability	`/WSRR/GenericObject[classifiedByAllOf('http://www.ibm.com/xmlns/prod/serviceregistry/profile/v6r3/GovernanceEnablementModel#BusinessCapability')]`
`ServiceVersionbyBusinessCapabilities DataSet`	Free form graph query that retrieves all capability version entities for a specific business capability entity	`/WSRR/GenericObject[@bsrURI={bsrUri}]/gep63_capabilityVersions(.)`

Data set name	Description	XPath
SLDsbyCapabilityVersion DataSet	Free form graph query to retrieve all the SLDs entity for a specific capability version entity	/WSRR/GenericObject[@bsrURI={bsrUri}]/gep63_provides(.)
SLAsbySLDsDataSet	Free form graph query to retrieve all the SLAs entities for a specific SLD entity	/WSRR/GenericObject[classifiedByAllOf('http://www.ibm.com/xmlns/prod/serviceregistry/profile/v6r3/GovernanceEnablementModel#ServiceLevelAgreement') and gep63_agreedEndpoints(.)/@bsrURI={bsrUri}]

To display the Expected Production Date, Peak Message Rate, and Maximum Messages Per Day for SLAs on the report, the query must return property values.

1. Using WSRR Studio, double-click **cascading_parameter.rptdesign**. Then, double-click **SLAsbySLDsDataSet** to edit it.

> **Note:** The file cascading_parameter.rptdesign is supplied in the additional materials accompanying this book. To obtain the additional materials see Appendix B, “Additional material” on page 621.

2. Select the WSRR free form graph query data set as shown in Figure 10-50.

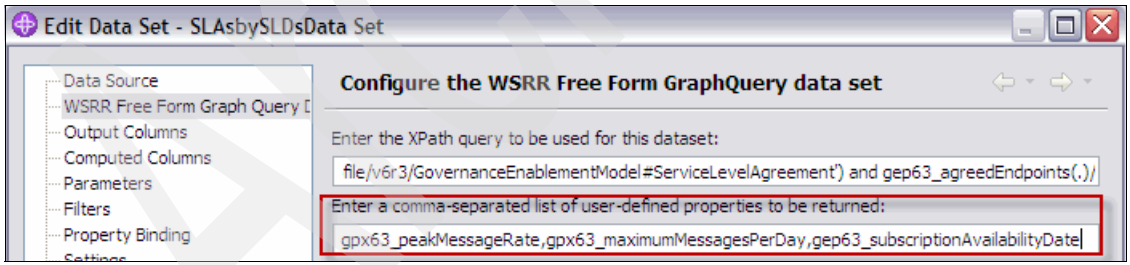


Figure 10-50 Returned properties for SLAsbySLDsDataSet

You can use this methodology to display the Certified Peak Message Rate and Certified Maximum Message Per Day for SLDs, as shown in Figure 10-51.

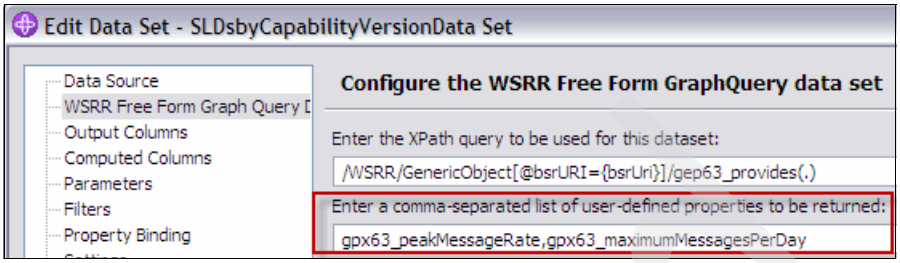


Figure 10-51 Returned properties for SLDsbyCapabilityVersionDataSet

Cascading parameter

To retrieve all the capability version entities for a specific business capability entity, you can use a *cascading parameter*. A cascading parameter allows you to interlink data sets. You can use the value from one data set to determine the output of another data set. In this report, a query is run to retrieve all the capability versions for each business capability.

You define the cascading parameter first when creating the data set. Then, you reference the cascading parameter during the layout of the nested tables by editing the data binding for a parameter.

1. Using WSRR Studio, double-click **cascading_parameter.rptdesign**. Then, double-click **ServiceVersionbyBusinessCapabilityDataSet** to open the data set.
2. Click **Parameter**, which is the capability version cascading parameter that is bound to the business capability's `bsrURI`, as shown in Figure 10-52.

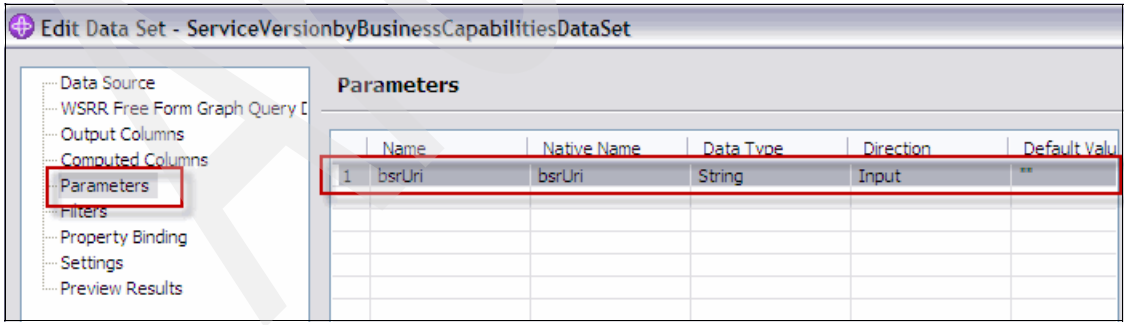


Figure 10-52 Cascading parameter to receive business capability's bsrURI

When laying out the design of the report, we used nested tables to expose the `ServiceVersionbyBusinessCapabilitiesDataSet` (inner or child table) to the data

returned in the `GetallBusinessCapabilitiesDataSet` (outer or parent table), as shown in Figure 10-53.

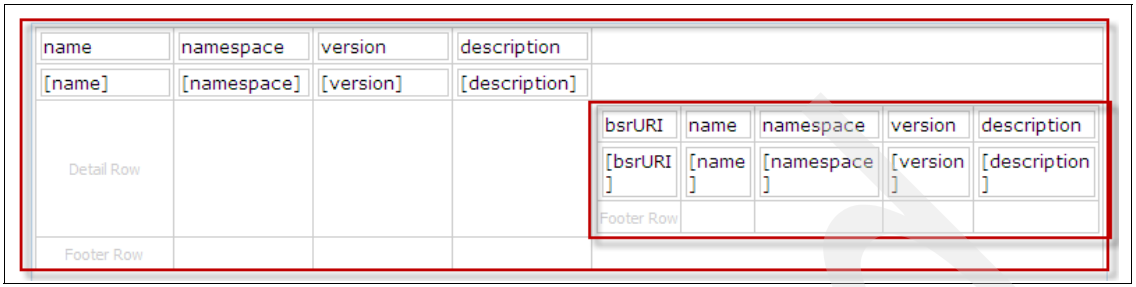


Figure 10-53 Nested tables

Archived

After the tables are nested, you need to bind the business capability's `bsrURI` to the inner table. In the following example, Table-BC is the business capability parent table. Naming the tables helps to ensure that the tables are nested properly.

1. In WSRR Studio, right-click **Table** and select **Edit Data Binding** as shown in Figure 10-54.

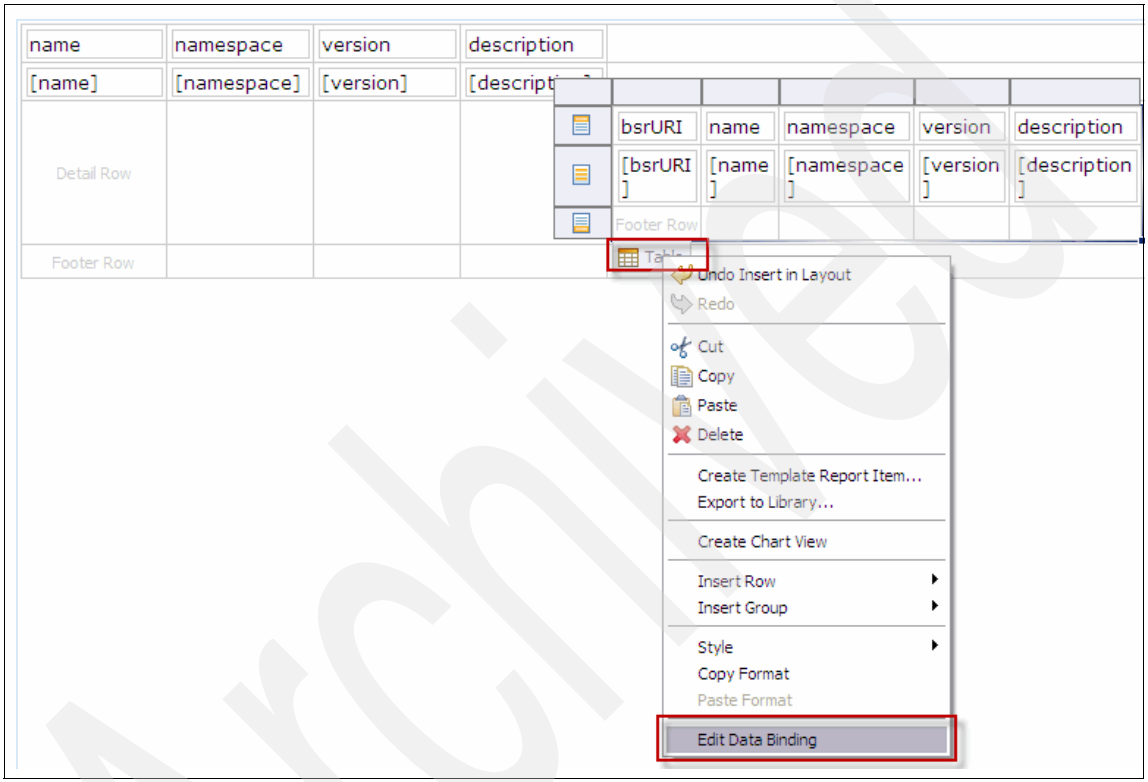


Figure 10-54 Selecting Edit Data Binding on inner table

Archived

2. Select **Data Set**, and click **Dataset Parameter Binding** as shown in Figure 10-55.

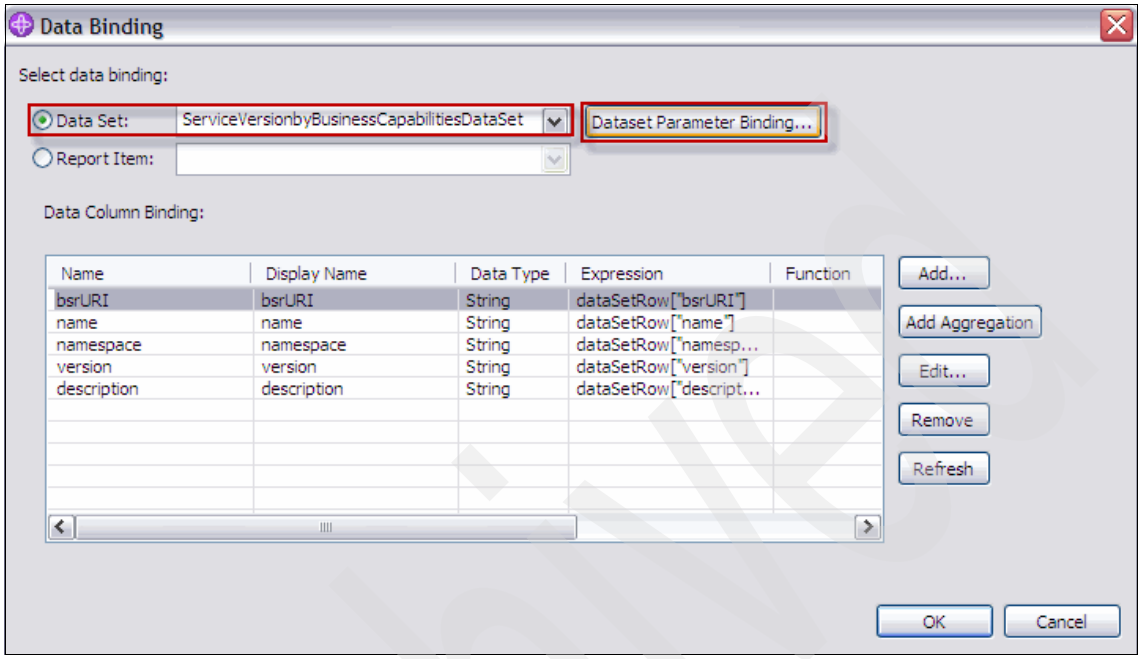


Figure 10-55 Clicking Dataset Parameter Binding

Archived

3. Select **bsrUri**, and click **Edit**, as shown in Figure 10-56).

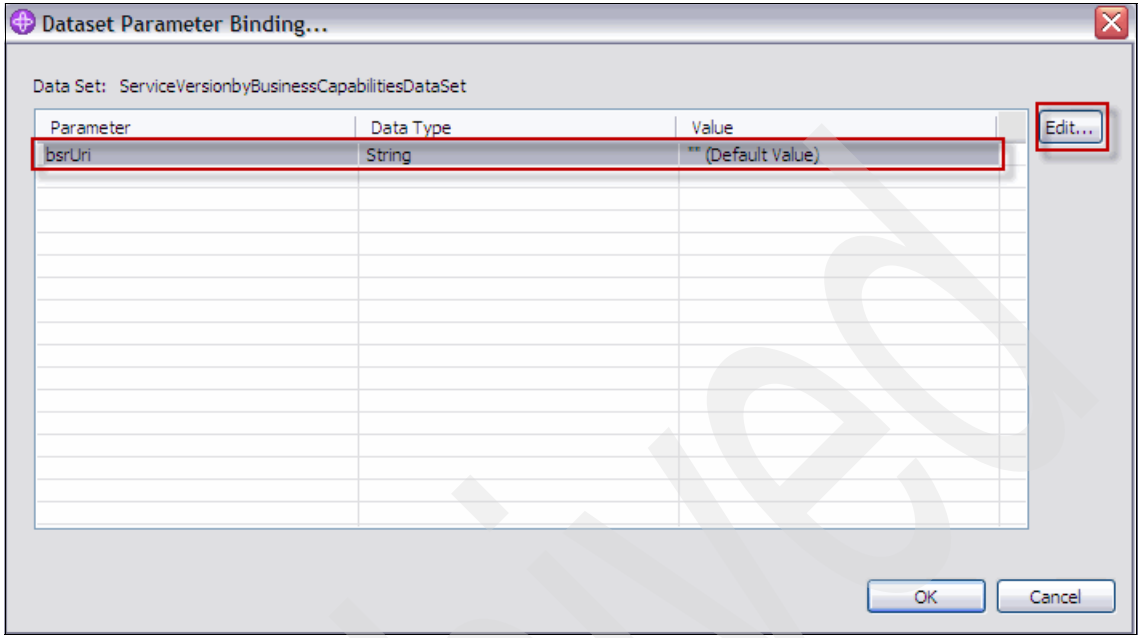


Figure 10-56 Selecting Edit for bsrUri Data Set Parameter Data Binding

Archived

4. Click fx to open the **Expression Builder**. Then, select **Available Columns Binding** → **Table-BC**, and double-click **bsrURI** to add to expression, as shown in Figure 10-57.

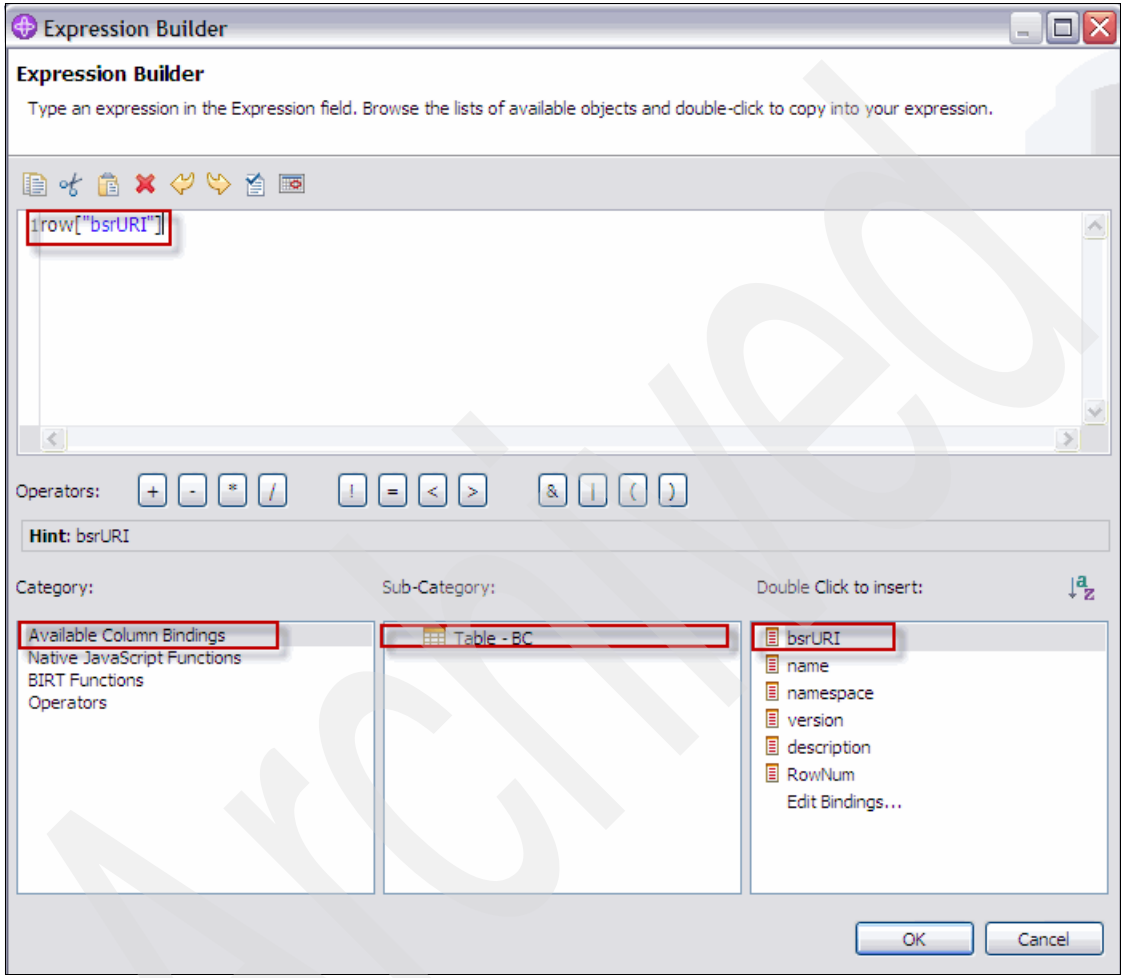


Figure 10-57 Selecting Business capability's bsrURi in Expression Builder

5. Click OK four times to get back to the report layout.

To total the Maximum Message Per Day for each SLA, we used a BIRT function, `Total.Sum()`. This function is not in the list, but you can enter it in the expression field as follows:

1. Using WSRR Studio, double-click **Total.Sum(row["gpx63_maximumMessagesPerDay"]** in the layout view to open the Expression editor.

2. Select **Available Column Bindings** → **Table - SLAsbySLDs**. Then, double-click **gps63_maximumMessagesPerDay** to add to the expression builder, as shown in Figure 10-58.

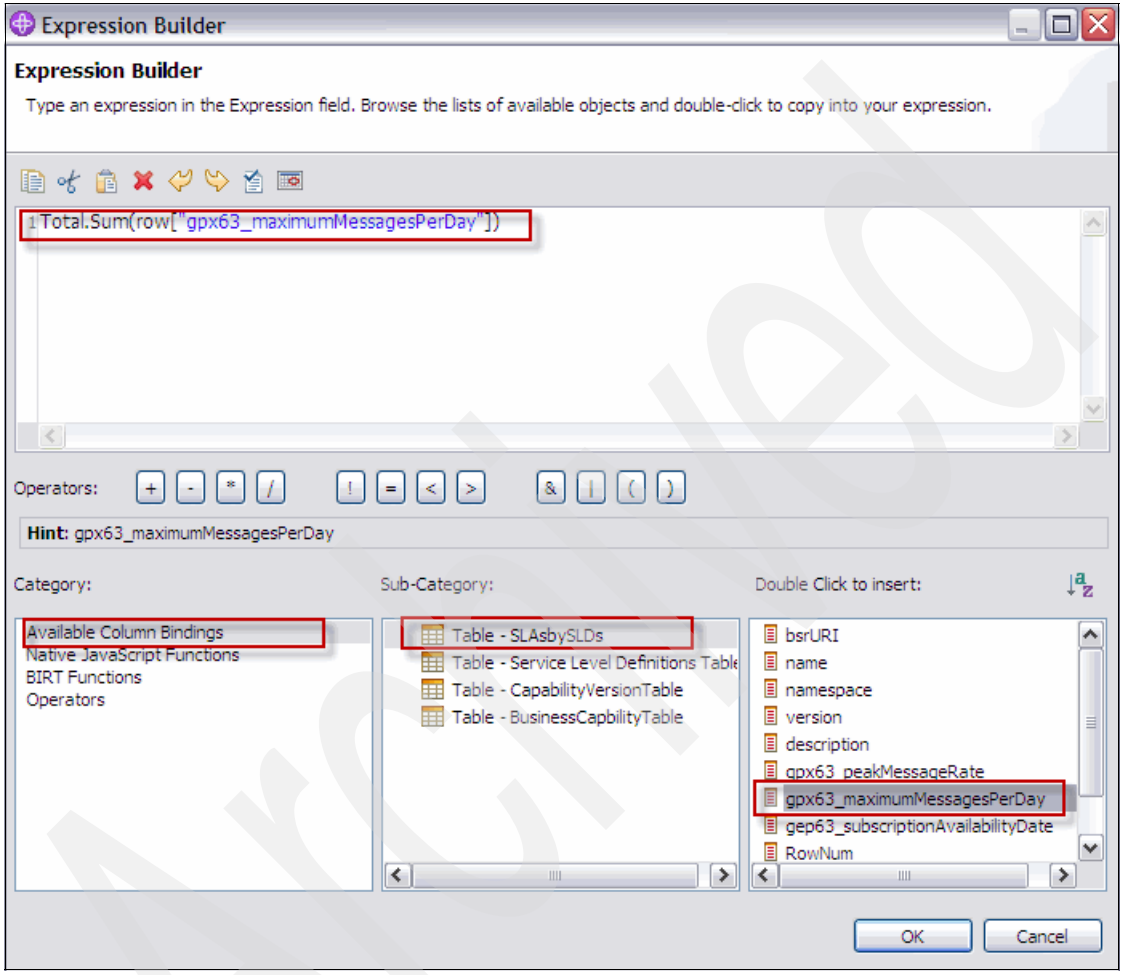


Figure 10-58 Selecting gps63_maximumMessagesPerDay in the Expression Builder

3. Change the following text row:

   `["gpx63_maximumMessagesPerDay"]`

   to

   `Total.Sum(row["gpx63_maximumMessagesPerDay"])`

4. Click **OK**.

JKHLE has a requirement that if the total of the SLA's Maximum Message Per Day exceed the Certified Maximum Messages Per Day, then show the total Maximum Messages Per Day for the SLA in red. This status alerts the operations team that more resources are needed for this service.

5. Using WSRR Studio, click **Total.Sum(row["gpx63_maximumMessagesPerDay"],** and in the Property Editor click **Highlights** as shown in Figure 10-59.

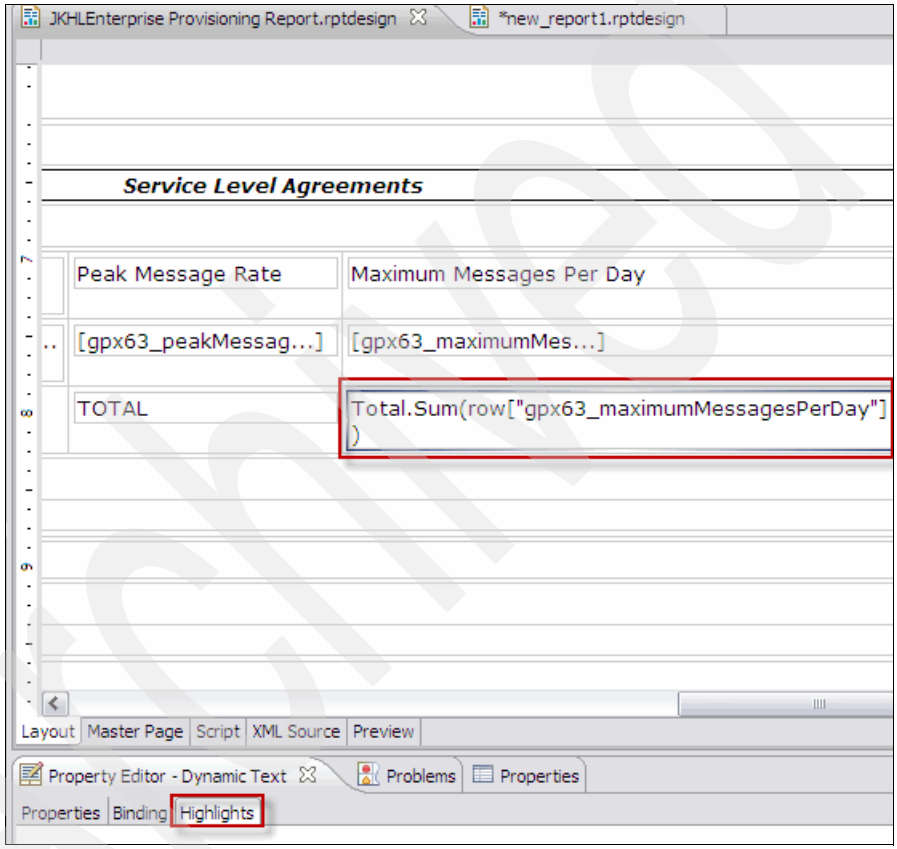


Figure 10-59 Selecting Highlights for SLAs

Archived

6. Select **Greater than** and the comparison value is built using an expression. Select **<Build Expression...>** as shown in Figure 10-60.

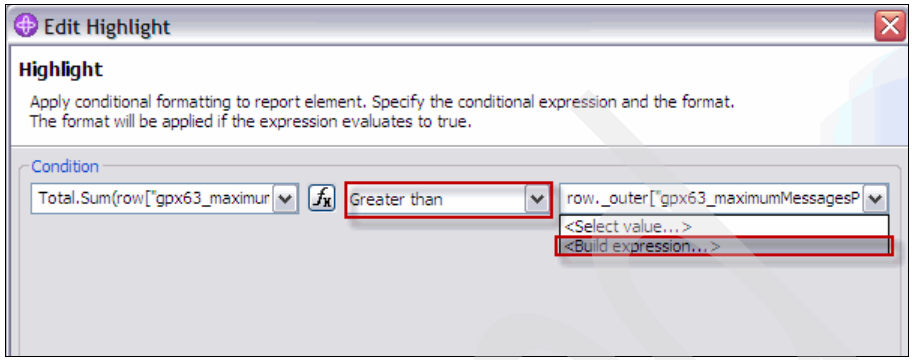


Figure 10-60 Highlight Editor

Archived

7. In the Expression Builder, select **Available Column Bindings** → **Table - Service Level Definitions Table**, double-click **gpx63_maximumMessagePerDay** to insert expression, as shown in Figure 10-61. Click **OK**.

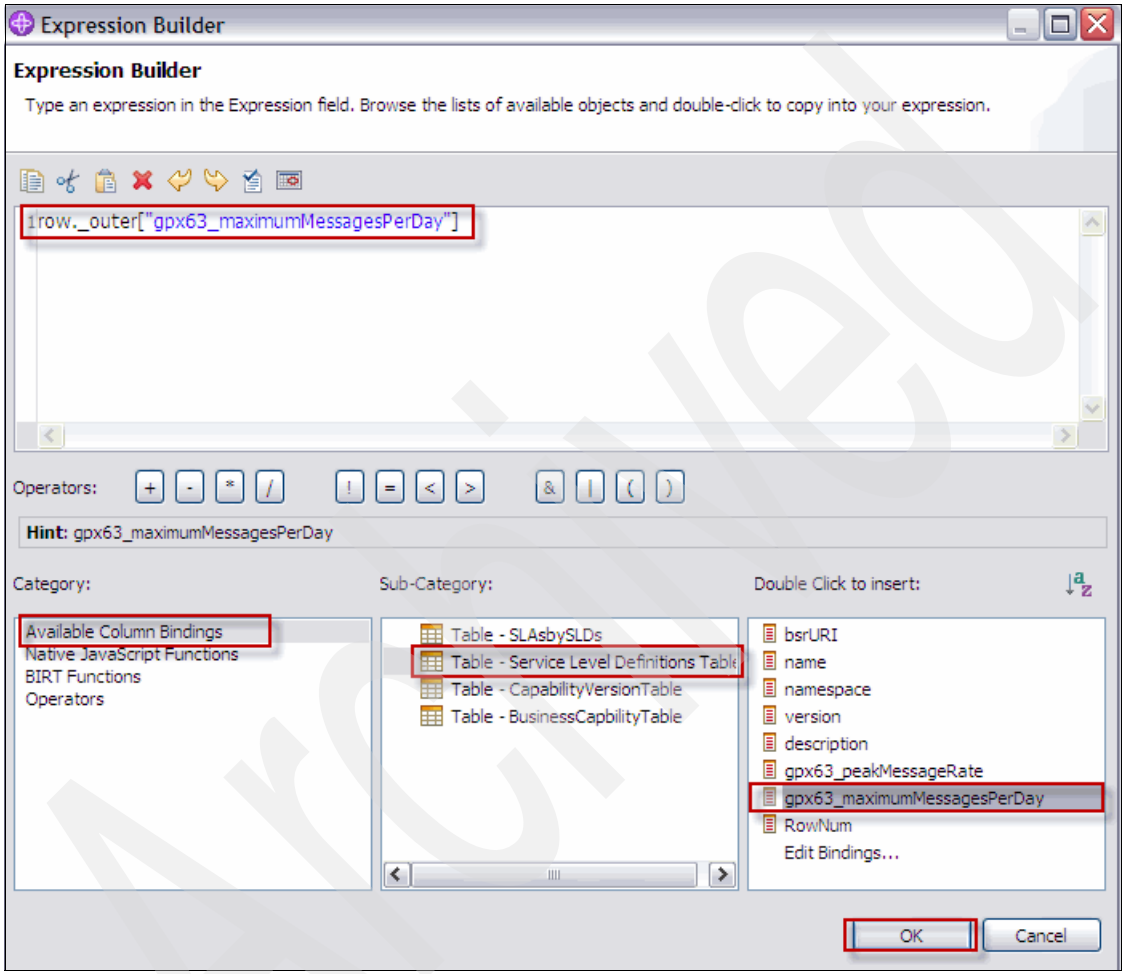


Figure 10-61 Selecting Maximum Messages Per Day from SLD table

8. Now, to get this value to turn red, in the Format section, select **Red** for the color, and click **OK** as shown in Figure 10-62.

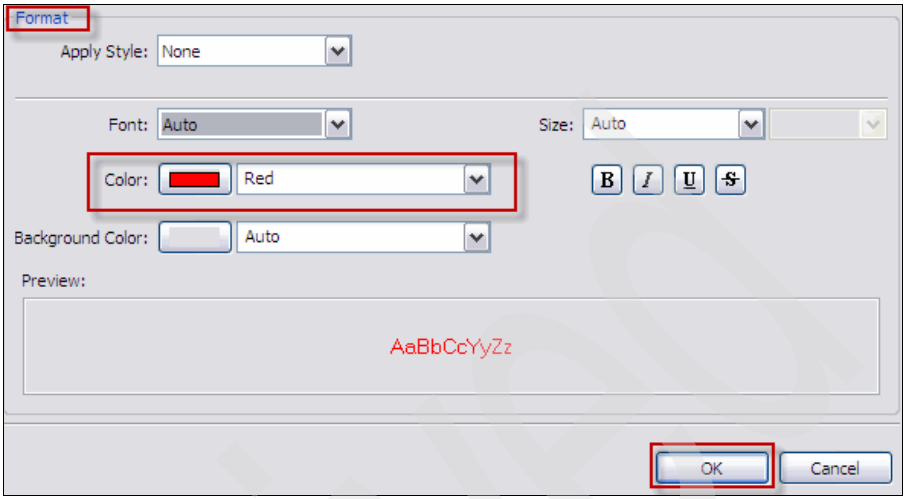


Figure 10-62 Format value red

Archived

After the Data Source is configured, review the report as follows:

1. In WSRR Studio, click to open the Perspectives window. Select **Report Design**, and click **OK** as shown in Figure 10-63.

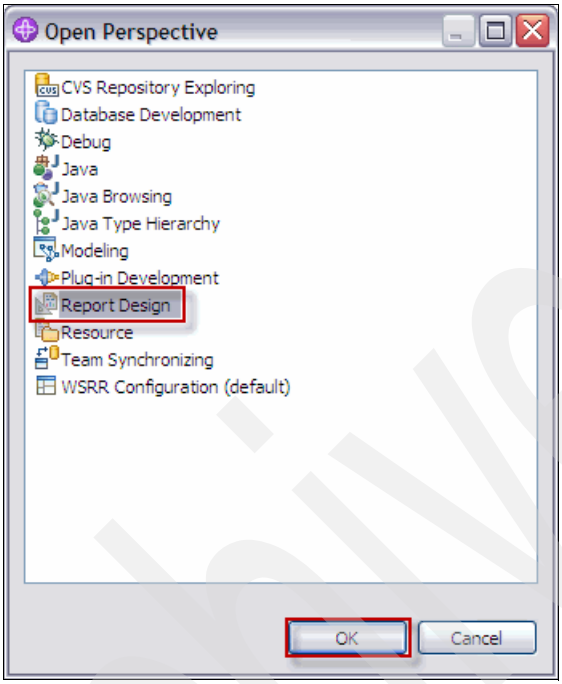


Figure 10-63 Selecting Report Design perspective

2. Double-click **JKHLE Provisioning Report.rptdesign** to open this report.

> **Note:** The file JKHLE Provisioning Report.rptdesign is supplied in the additional materials accompanying this book. To obtain the additional materials see Appendix B, “Additional material” on page 621.

3. In the Data Explorer view, right-click **Data Sources**, and select **New Data Source** as shown in Figure 10-64.

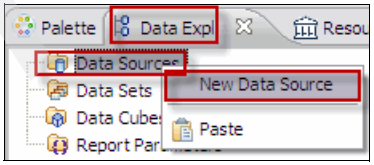


Figure 10-64 Selecting New Data Source

4. Select **WSRR Data Source**, and leave the default name of *Data Source* as shown in Figure 10-65.

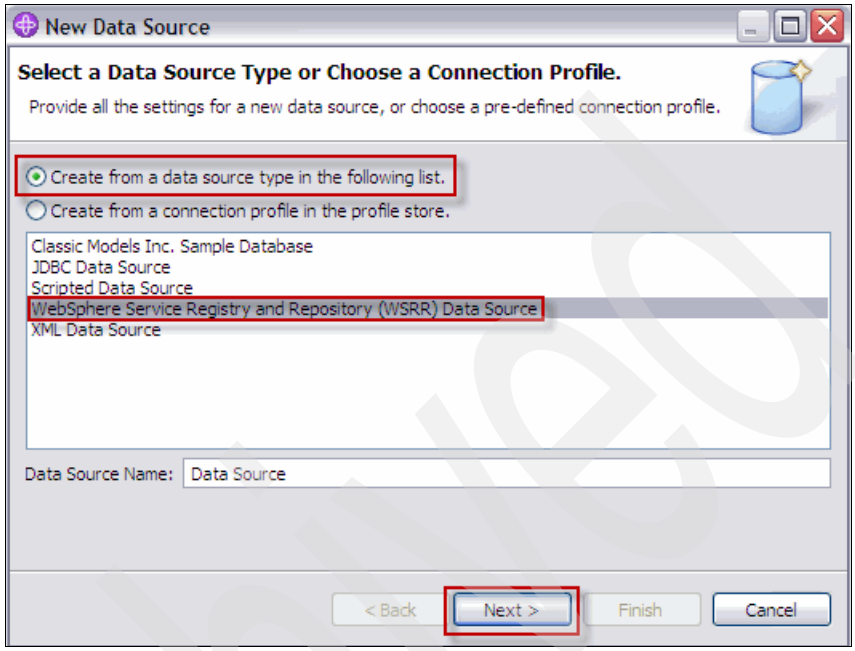


Figure 10-65 Selecting WSRR Data Source

Archived

5. Select the **WSRR Location**, and click **Finish** as shown in Figure 10-67.

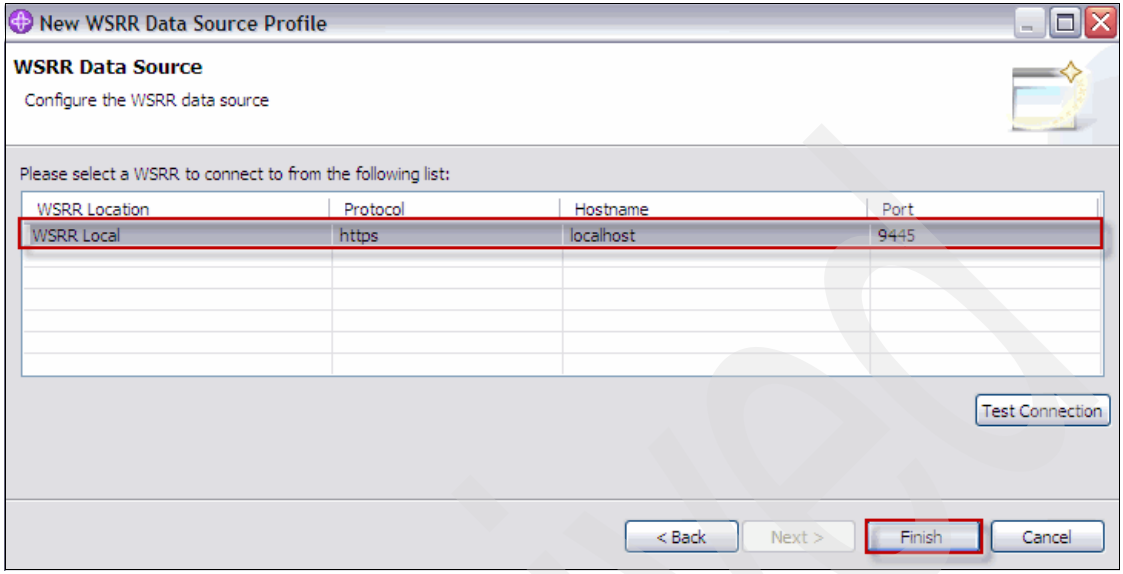


Figure 10-66 Selecting a data source

6. When the data source is configured correctly, click **Preview** to view the report (Figure 10-67).

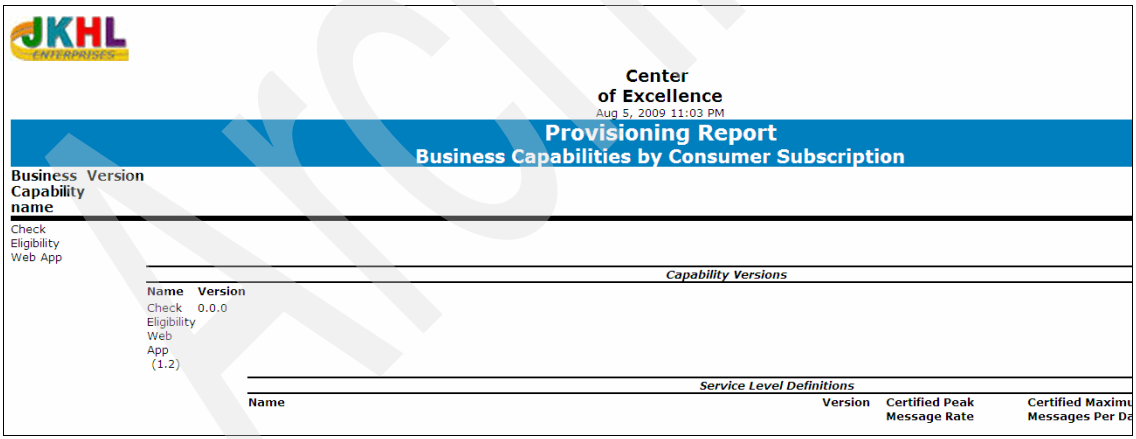


Figure 10-67 Results from Previewing the JKHLE Provisioning Report

Part 3

Scenarios

Archived

© Copyright IBM Corp. 2009. All rights reserved.

Archived

11

Creating an organizational structure

This chapter describes the first step in a set of service governance scenarios at JKHL Enterprises and explains how to create the company's organizational structure.

Note: In the examples in this book, we use a case study about a fictional company named *JKHL Enterprises* (JKHLE). For information about this case study, see Chapter 4, "JKHL Enterprises case study" on page 153. For information about the roles used throughout this scenario, refer to 3.3, "Roles in the governance enablement profile" on page 103.

This chapter includes the following topics:

- Creating the JKHLE organizational structure
- Creating an organization
- Adding child organizations

© Copyright IBM Corp. 2009. All rights reserved.

11.1 Creating the JKHLE organizational structure

When multiple composite applications use a service, it is vital for effective governance to determine who is responsible for that service.

Often an enterprise organizes its staff reporting structure and finances around business operations. The department that is responsible for certain business operations might also be responsible for the development of services and the runtime IT for these operations. In a service-oriented architecture (SOA), however, another organization might be responsible for the development of the services and runtime IT for given operations. Therefore, services and composite applications in an SOA often do not follow an enterprise's strict hierarchical reporting and financial structure, creating gaps and overlap in IT responsibilities.

You can use an organizational structure defined in WSRR to assign ownership of the SOA services that are governed. WSRR also supports identification of those departments within the organization that subscribe to or use specific services.

Figure 11-1 illustrates the process for creating an organization.

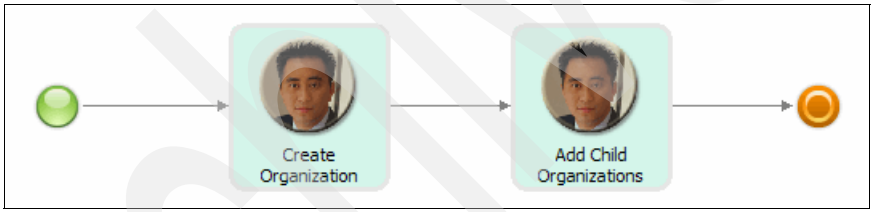


Figure 11-1 Create organization process

As part of the administration setup, an administrator creates organizations that are represented in the registry. These organizations are required to allow relationships to be built between services and organizations to show service ownership and consumption.

Ian, the administrator at JKHLE creates the following organizations:

- *JKHL Enterprises*, a top-level organization that represents the complete enterprise.
- *Common services*, a child organization of JKHL Enterprises that represents the departmental team that is responsible for the development and delivery of services that are shared throughout the enterprise.
- *Commercial*, an organization that represents the commercial line of business (LOB).

11.2 Creating an organization

To create the top-level organization, log on to WSRR, and select the **Administrator** perspective. Then, follow these steps:

1. Click **Actions** → **Create** → **Organization**, as shown in Figure 11-2.

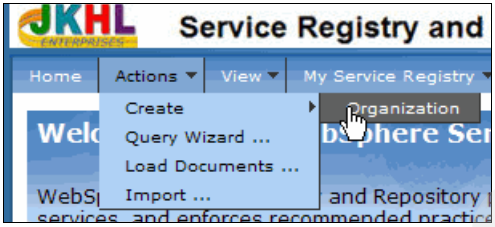


Figure 11-2 Create organization

2. Enter `JKHL Enterprises` in the Name field, as shown in Figure 11-3.

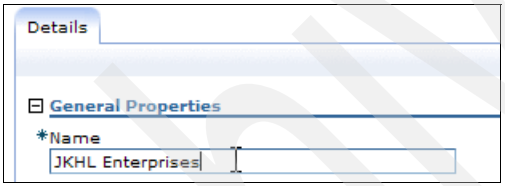


Figure 11-3 Entering the organization name

Archived

11.3 Adding child organizations

To add a child organization, follow these steps:

1. Click **Add Organization** to the right of the Child Organizations relationship, as shown in Figure 11-4.

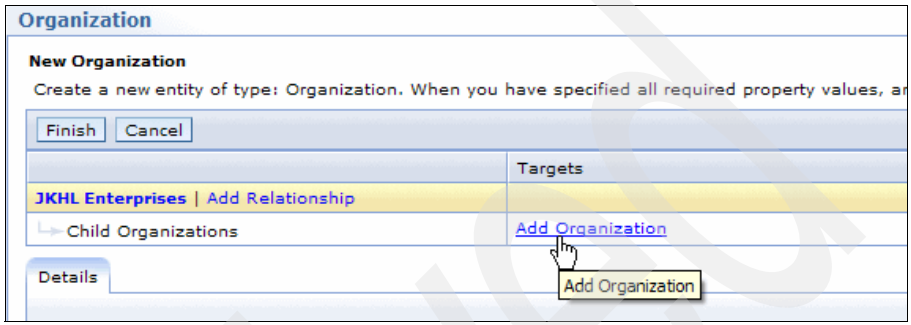


Figure 11-4 Adding a child organization

2. In the New Entity section, click **Create**, as shown in Figure 11-5.

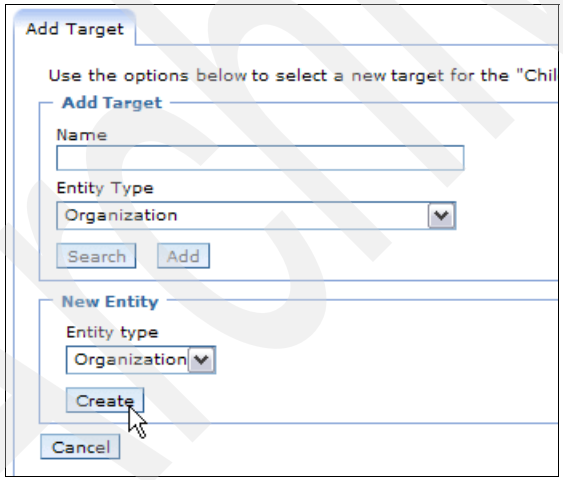


Figure 11-5 Creating the child organization

3. Enter `Common services` in the Name field, and click **OK**. Common services is added as a child organization of JKHL Enterprises.
4. Click **Add Organization** to the right of the Child Organizations relationship.
5. In the **New Entity** section, click **Create**.

Archived

6. Enter `Commercial` in the Name field, and click **OK**. Commercial is added as a child organization of JKHL Enterprises.
7. Click **Finish** to save your changes.

Figure 11-6 illustrates the organizational structure in the graphical view.

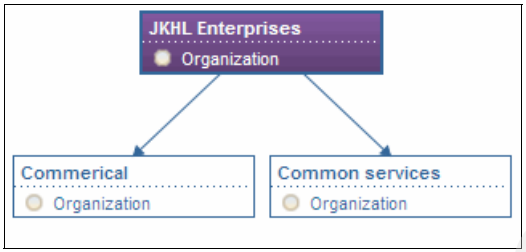


Figure 11-6 JKHL Enterprises organizational structure

Archived

Archived

12

Governing a schema

This chapter provides step-by-step instructions about how to create and govern a schema specification at JKHL Enterprises.

Note: In the examples in this book, we use a case study about a fictional company named *JKHL Enterprises* (JKHLE). For information about this case study, see Chapter 4, “JKHL Enterprises case study” on page 153. For information about the roles used throughout this scenario refer to 3.3, “Roles in the governance enablement profile” on page 103.

This chapter includes the following topics:

- Governing a schema specification
- Creating the schema specification
- Proposing the schema specification
- Approving the schema specification

© Copyright IBM Corp. 2009. All rights reserved.

12.1 Governing a schema specification

At JKHL Enterprises (JKHLE), a schema specification is required to describe the business objects that are required to establish customer and account records. This schema is intended as a re-usable asset within JKHLE. For example JKHLE wants to standardize the way customer information, including postal addresses, are defined. This standardization aids interoperability between services and operations. If a specific business operation requires more information than is already described in the schema, that operation can define its own schema that extends this schema, or the business can request that a new version of the schema.

Figure 12-1 illustrates the process for governing a schema specification.

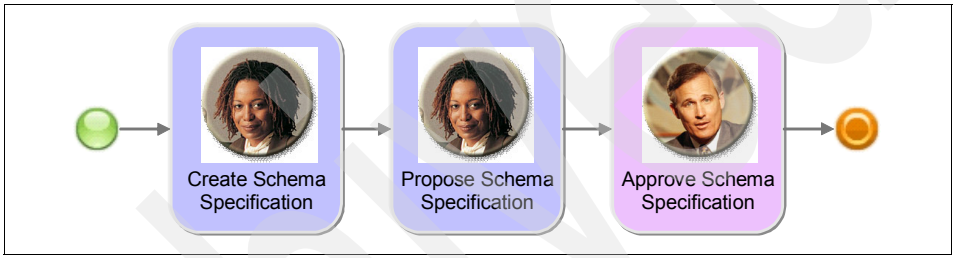


Figure 12-1 Governing a schema specification

12.2 Creating the schema specification

At JKHLE, Connie is the Development Manager for the Common services area that guides the creation of a new schema specification to represent customer and account records.

To create the new schema specification, log on to WebSphere Service Registry and Repository (WSSR), and select the **Development** perspective. Then, follow these steps:

1. Click **Actions** → **Create** → **Schema Specification**, as shown in Figure 12-2.

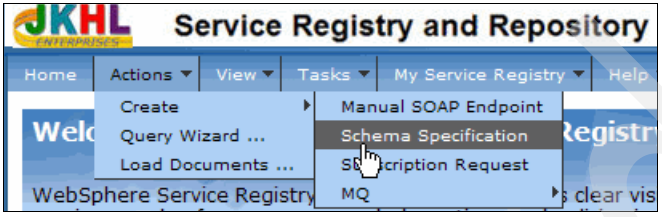


Figure 12-2 Navigating to create a schema specification

2. Enter the following property values:
   - Name: `JKHLE global schema`
   - Description: `The schema definition for customer and account records.`
   - Namespace: `http://jkhle.itso.ibm.com/Account`
   - Version: `1.0`
   - Requirements Link: `http://requirements.jkhle.com/requirements.jsp&id=5682`

     The Requirements Link, in this case a fictitious link, is the relevant item in JKHLE's requirements tracking tool.
3. Assign an owning organization to the schema specification, as follows:
   a. Click **Add Organization** in the Targets columns to the right of the Owning Organization relationship, as shown in Figure 12-3 on page 410.
   b. Enter `C` in the **Name** field, select Common services from the auto suggest list, and click **Add**. The Common services organization is added as a target of the Owning organization relationship.
   c. Click **Finish** to save the schema specification. The details page for the schema specification displays.

Archived

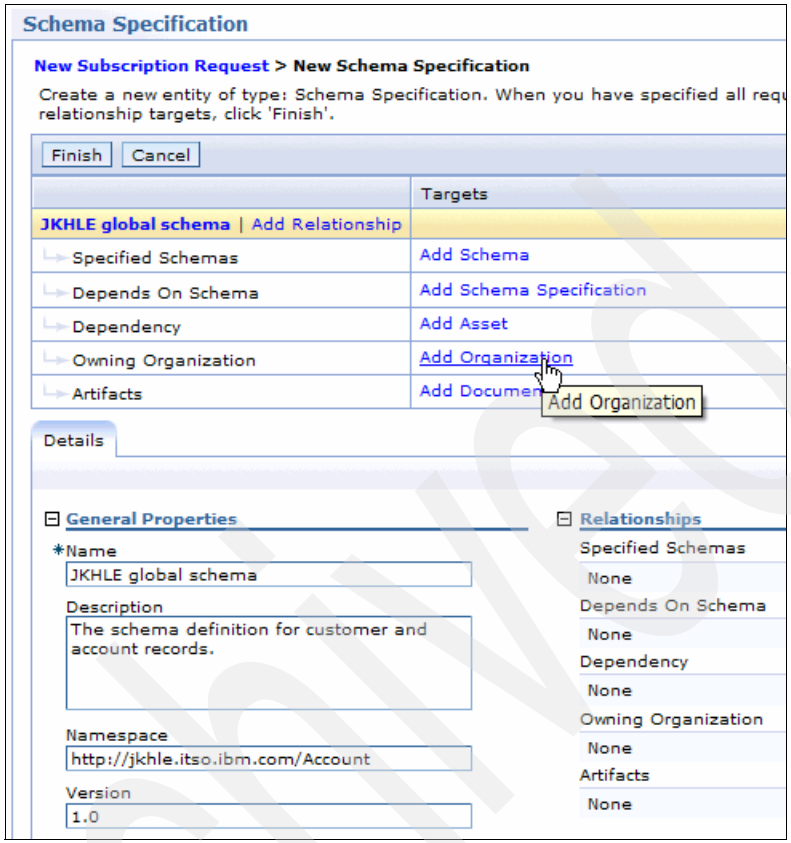


Figure 12-3 Add an owning organization

The new governance state is Asset Identified as shown in Figure 12-4. This is the initial state in the Asset life cycle. A new schema specification is entered into the Asset life cycle automatically.

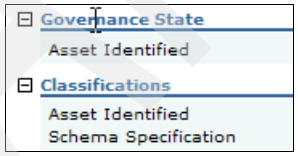


Figure 12-4 Asset identified governance state

Archived

4. Add the schema XSD file as follows:
   a. Click **Edit Relationships**.
   b. Click **Add Document** to the right of the Artifacts relationship, as shown in Figure 12-5.

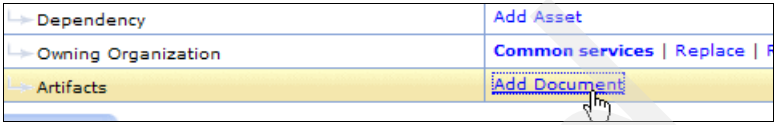


Figure 12-5 Adding an artifact

   c. In the Load Document window, select **XSD Document** from the Document Type list, and click **Load Document**, as shown in Figure 12-6.

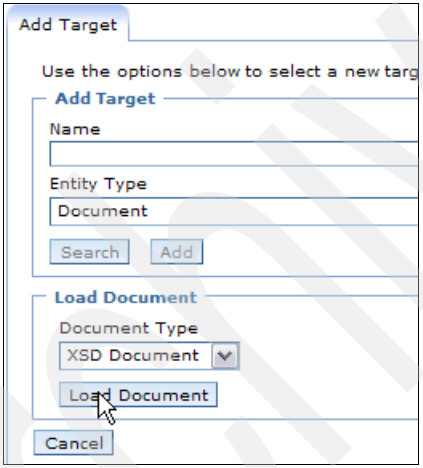


Figure 12-6 Loading the XSD

   d. Click **Browse** and navigate to the directory where the XSD file is located. Select the `JKHLEGlobalSchema.xsd` file, and click **Open**.
   e. Enter the document version as `1.0` and click **OK**.
   f. Click **Finish** to add the XSD. Then, click **Finish** again to save these changes to the schema specification.

> **Note:** You are required to save your changes at this stage because the schema artefact is created only after the schema specification document is committed to the registry.

Archived

5. To add the schema to the schema specification:
   a. Click **Edit Relationships**.
   b. Click **Add Schema**, and enter `*Account*` in the **Name** field.
   c. Select **http://jkhle.itso.ibm.com/Account (1.0)** from the auto suggest list, and click **Add**.
   d. Click **Finish**.

12.3 Proposing the schema specification

At this stage of the process, the scope and requirements for the schema specification have been agreed upon. An XML schema document has also been loaded into WSRR. This schema document describes the proposed schema definition. Click **Propose**, as shown in Figure 12-7, to transition the schema specification to the Asset Scope Review governance state.

Archived

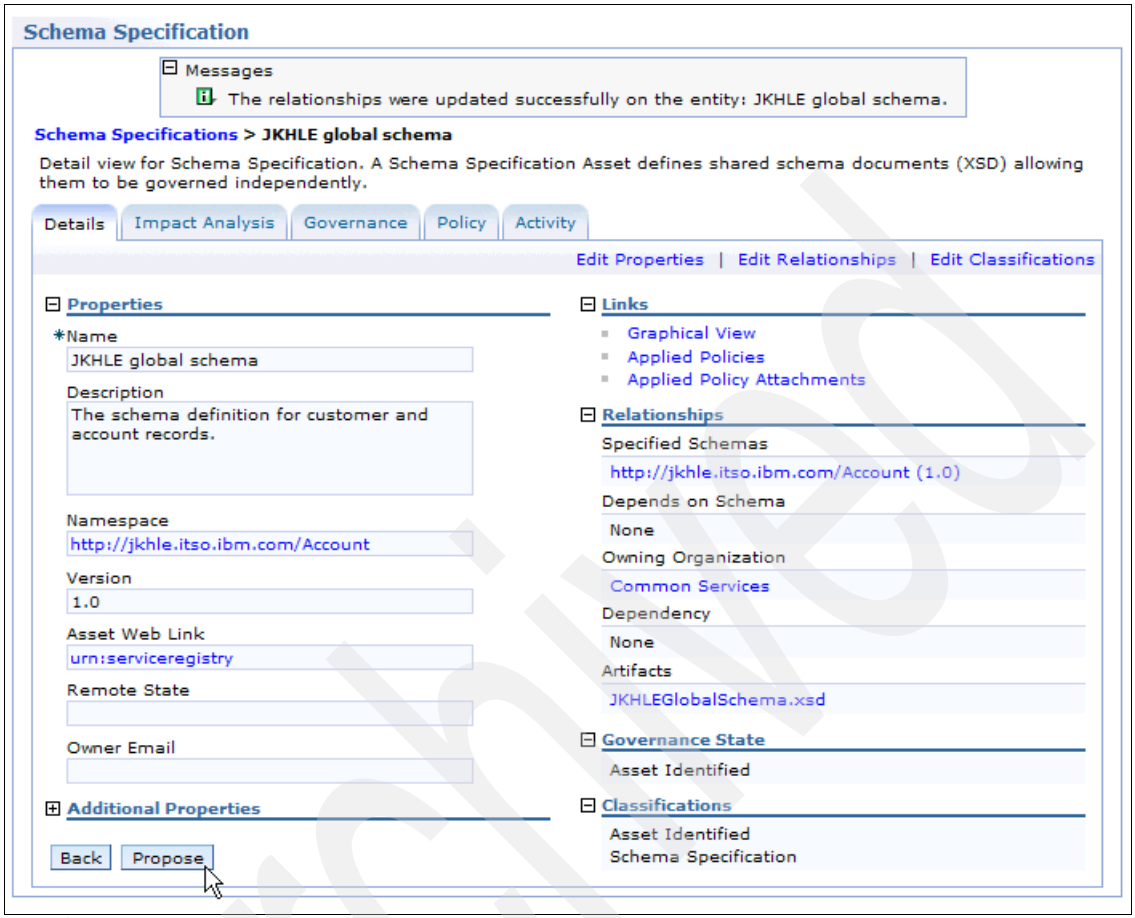


Figure 12-7 JKHLE Global Schema schema specification

The governance state for the schema specification is now *Asset Scope Review*, as shown in Figure 12-8.

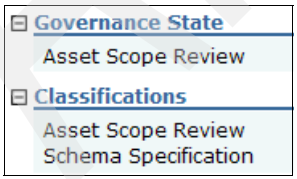


Figure 12-8 Asset Scope Review governance state

Archived

12.4 Approving the schema specification

In the Asset Scope Review state, the SOA governance team must now review the schema specification. If required the SOA governance team can involve other stakeholders, such as the organizations that will consume the service in this decision process. The SOA governance team review the specification to ensure that it meets their requirements and that the business objects that it utilizes are specified in a manner that promotes future reuse. After the team agrees that all aspects of the design meet the requirements, the schema is approved by David who is a member of the SOA governance team.

To transition the schema specification to the Approved state, log on to WSSR, and select the **SOA Governance** perspective. Then, complete the following steps:

1. Click **Tasks** → **Schema Specification Tasks** → **Schema Specifications for Scope Review**, as shown in Figure 12-9.

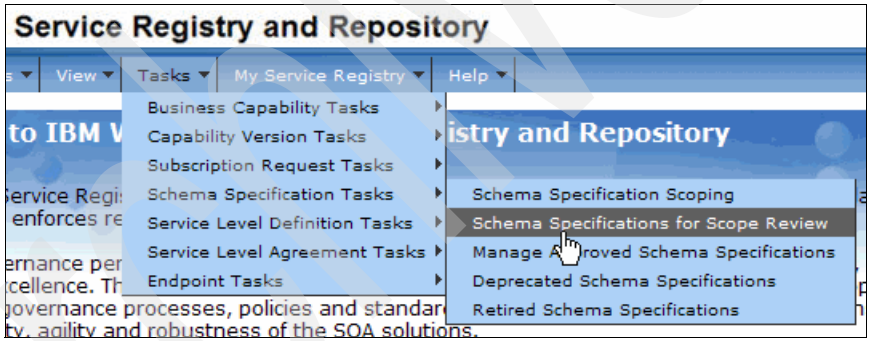


Figure 12-9 Navigating to the schema specifications for scope review

2. Click **JKHLE global schema** to display the schema specification details.
3. Click **Approve** and note that the new governance state is Asset Approved, as shown in Figure 12-10.

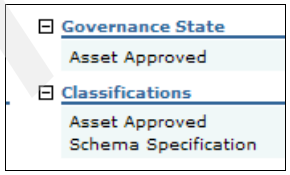


Figure 12-10 Asset approved schema specification governance state

Figure 12-11 illustrates the completed schema specification.

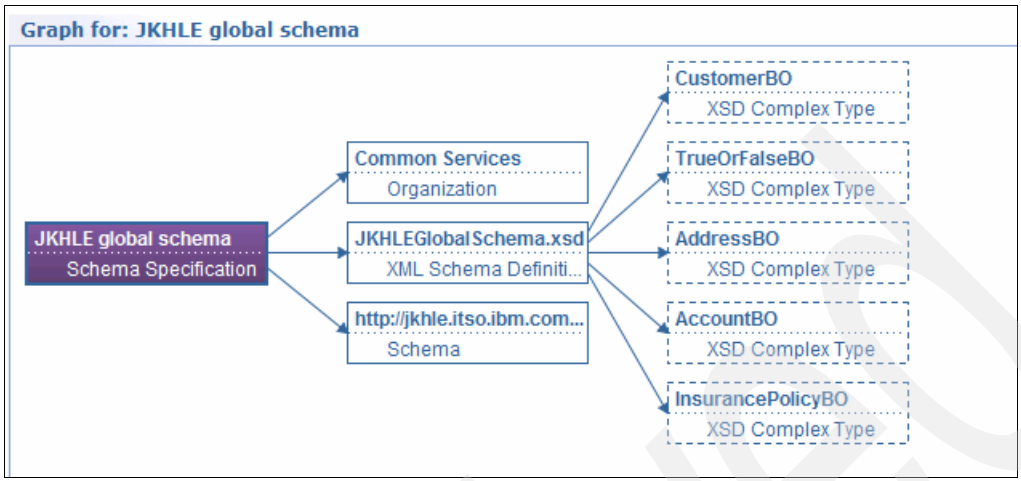


Figure 12-11 Graph of completed schema specification

Archived

Figure 12-12 illustrates the life cycle for a schema specification and other objects of type *Asset*.

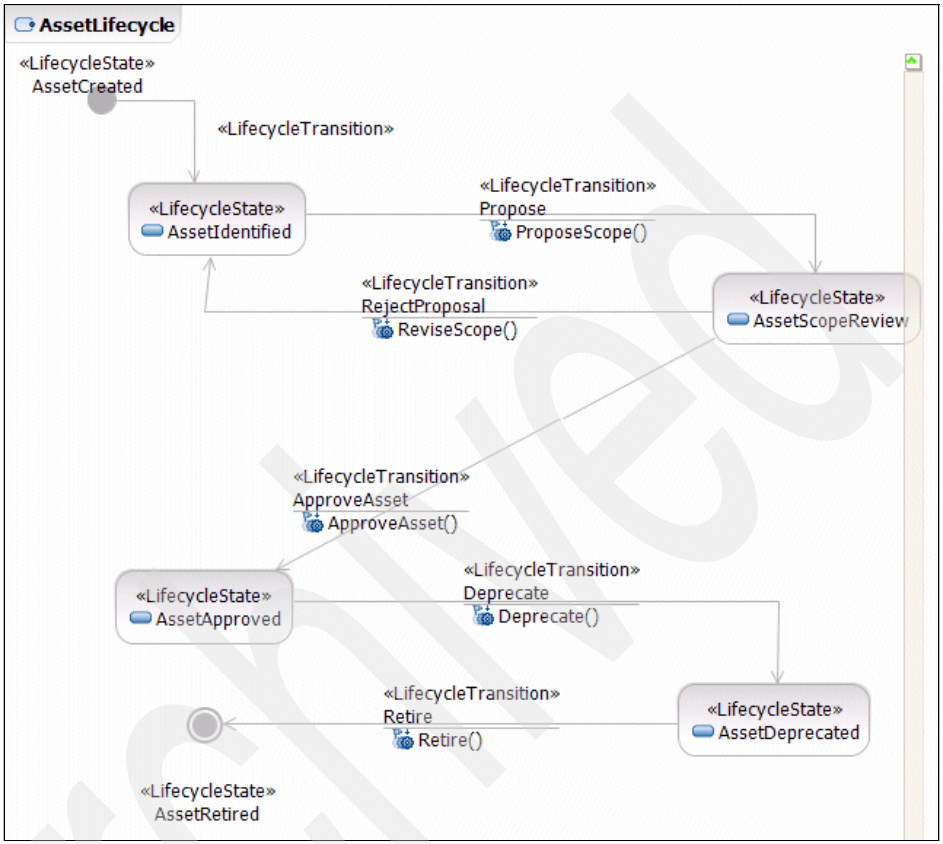


Figure 12-12 Asset life cycle

Archived

13

Governing a service defined in a monolithic WSDL

This chapter provides step-by-step instructions about how to register a service in WebSphere Service Registry and Repository (WSRR) that is defined in a monolithic WSDL.

Note: In the examples in this book, we use a case study about a fictional company named *JKHL Enterprises* (JKHLE). For information about this case study, see Chapter 4, “JKHL Enterprises case study” on page 153. For information about the roles used throughout this scenario refer to 3.3, “Roles in the governance enablement profile” on page 103.

This chapter includes the following topics:

- Governing a new service at JKHLE
- Creating a business capability
- Reviewing and approving the business service
- Scoping a service version
- Planning a service version
- Creating a service level definition
- Deploying a service version to staging
- Deploying a service version to pre-production
- Deploying a service version to production

© Copyright IBM Corp. 2009. All rights reserved.

13.1 Governing a new service at JKHLE

JKHLE recently implemented WSRR for the governance of service-oriented architecture (SOA) services. They created the eligibility service using tooling that produces a single monolithic WSDL. That is, the service includes the interface, binding, and endpoint information.

JKHLE decided to keep the WSDL as is rather than to split it out manually. There are other scenarios where it might be more practical to store monolithic WSDLs in WSRR, such as when discovering services that re already deployed, storing services that are developed by a third party, and so forth.

Note: A monolithic WSDL that contains the interface, binding, and endpoint definitions is not recommended; however, as noted, circumstances might exist where design this is impractical to avoid.

This scenario describes how to add the eligibility service to the registry. Figure 13-1 illustrates the main steps required to complete this process.

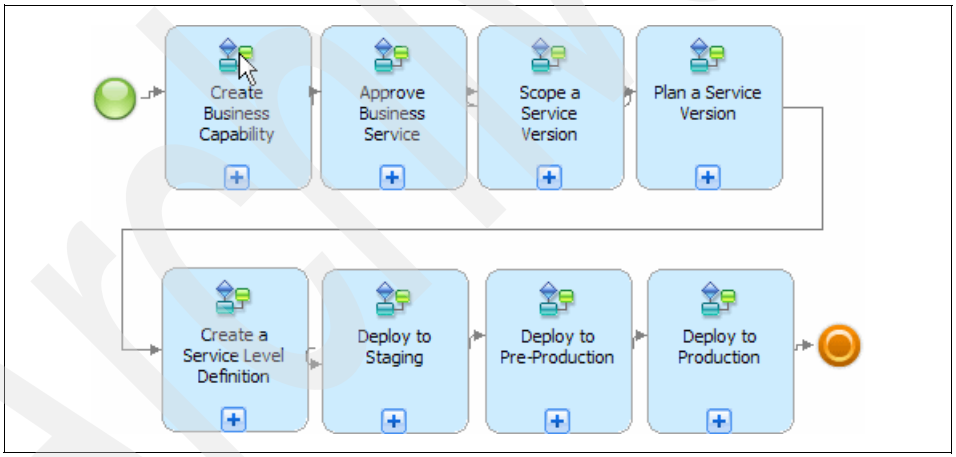


Figure 13-1 Governing a service with a monolithic WSDL process

Archived

In this chapter, we show the progress that is made in building the entities in the governance model using a depiction of the model as shown in Figure 13-2.

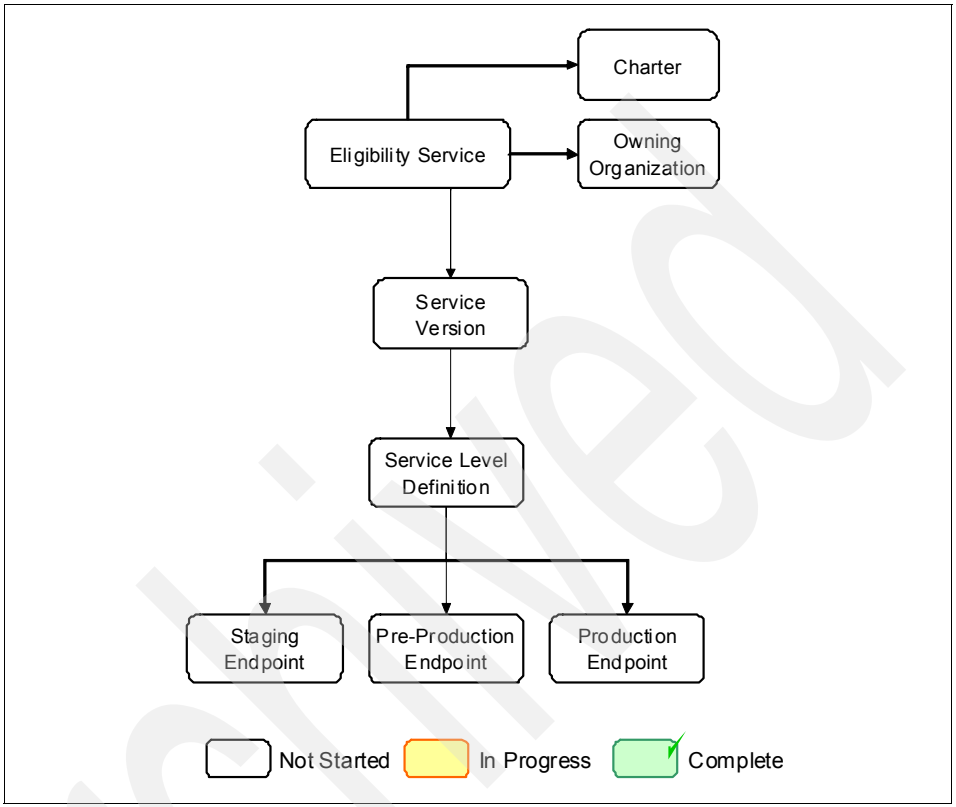


Figure 13-2 WSRR entities for service governance

Archived

13.2 Creating a business capability

The first phase in governing a new service is creating the business capability as illustrated in Figure 13-3. We describe the steps involved in this process in this section.

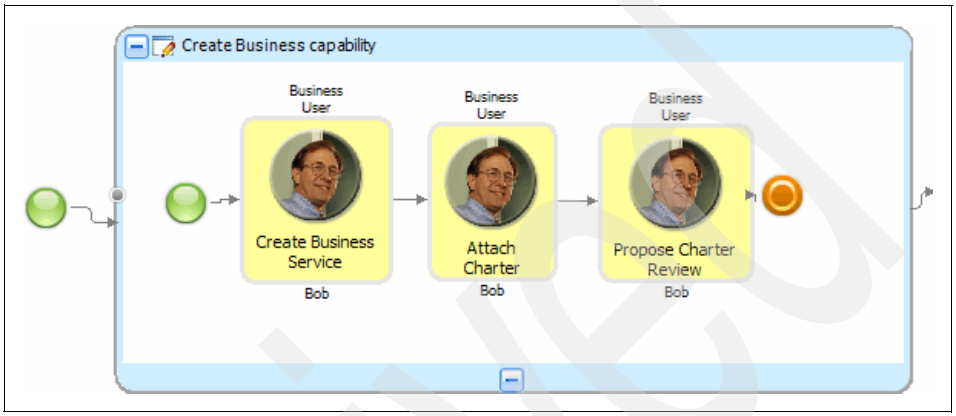


Figure 13-3 Creating the business capability process

13.2.1 Creating a new business service

Bob, the business analyst for the commercial line of business at JKHLE, is tasked with defining a new Eligibility service, which verifies that a customer satisfies the commercial credentials to be an account holder.

Having used the search facilities in the WSRR Web UI to confirm that there is no suitable existing service, Bob creates a new business service object to identify the requirement for such a capability.

Archived

To create a new business service, log on to WSRR, and select the **Business** perspective. Then, follow these steps:

1. Click **Actions** → **Create** → **Business Service**, as shown in Figure 13-4.

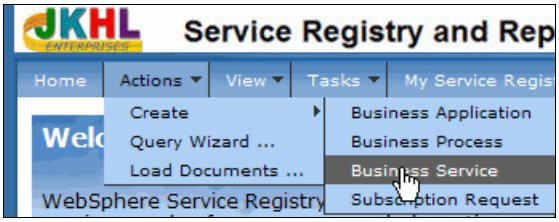


Figure 13-4 Creating a business service

2. Enter the following information, as shown in Figure 13-5:
   - `Name: Eligibility service`
   - Description: `Validate customer information is complete and can be verified before opening an account.`

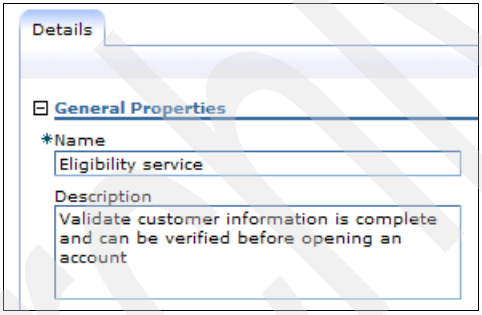


Figure 13-5 Entering the business service properties

3. Click **Finish**. The details page for the new business service displays.

Archived

The Governance State of the new business service is *Business Capability Identified*, as shown in Figure 13-6. This state is the initial state in the capability life cycle; a new business service is entered into this life cycle automatically.

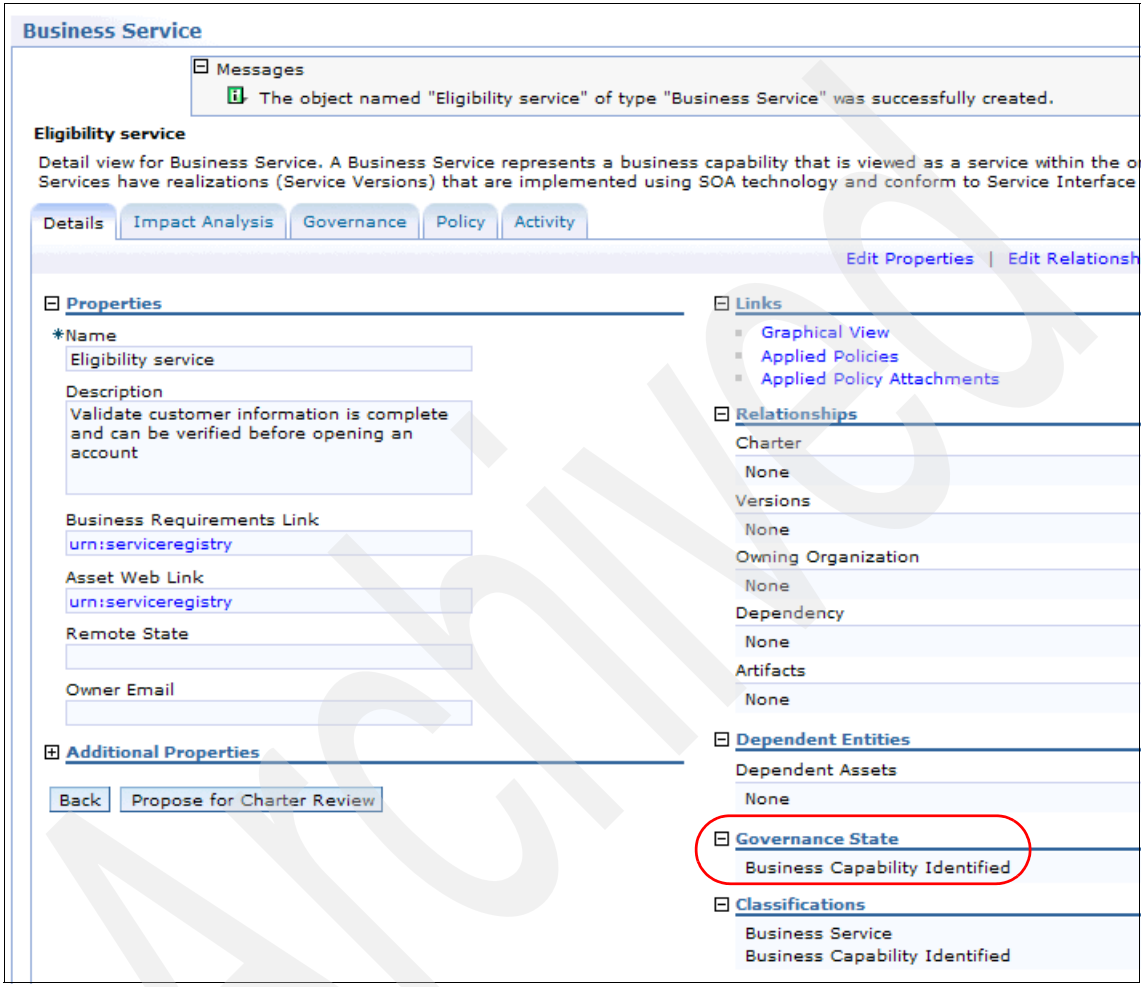


Figure 13-6 New business service

Archived

13.2.2 Attaching a charter

The *charter* is a separate document that describes the required capability in detail, including all functional and non-functional requirements. The charter is authored externally and then loaded into WSRR and attached to the business service.

To attach a charter, load the document that contains the charter, and attach it to the business service by using the Charter relationship. Log in to WSRR and then follow these steps:

1. Click **Edit Relationships**.
2. Click **Add Other Document** to the right of the **Charter** relationship, as shown in Figure 13-7.

Figure 13-7 Adding the charter document

Archived

3. Click **Load Document** (Figure 13-8).

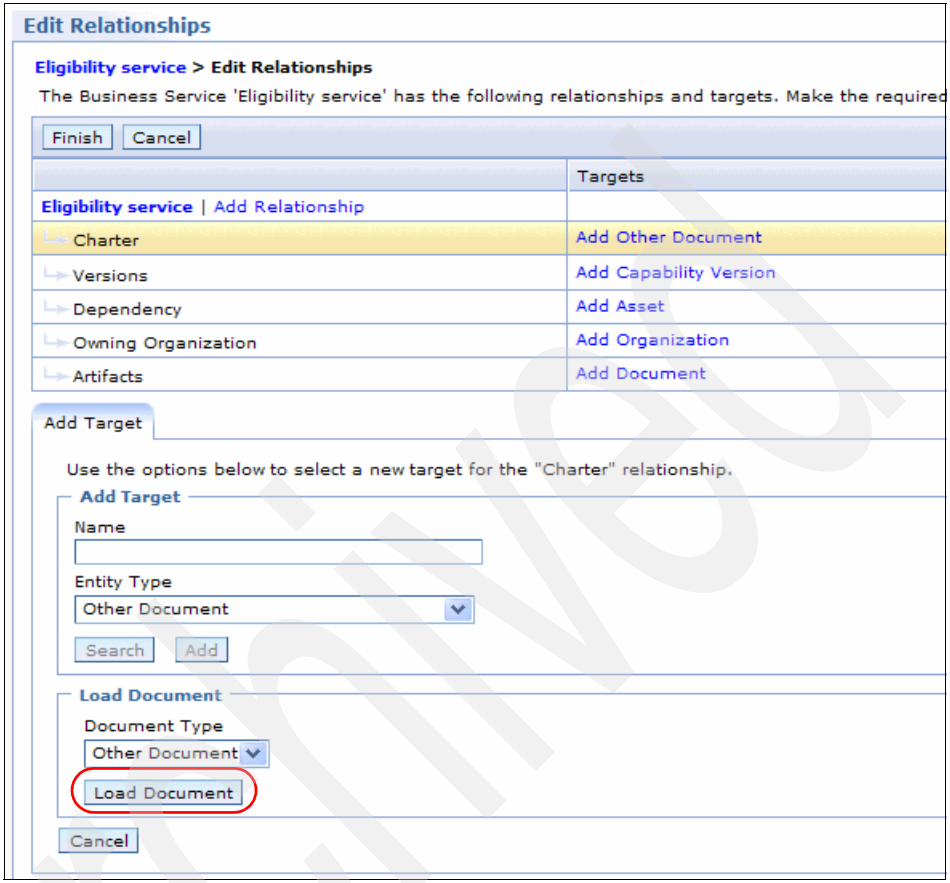


Figure 13-8 Loading the document

4. Click **Browse**, and navigate to the directory where the charter document is located.
5. Select **EligibilityServiceCharter.doc**, click **Open**, and then click **OK**.
6. Click **Finish** to load the charter document. The charter document is added as a target of the Charter relationship.
7. Click **Finish** to save your changes.

Archived

The charter document is loaded for the Eligibility business service as shown in Figure 13-9.

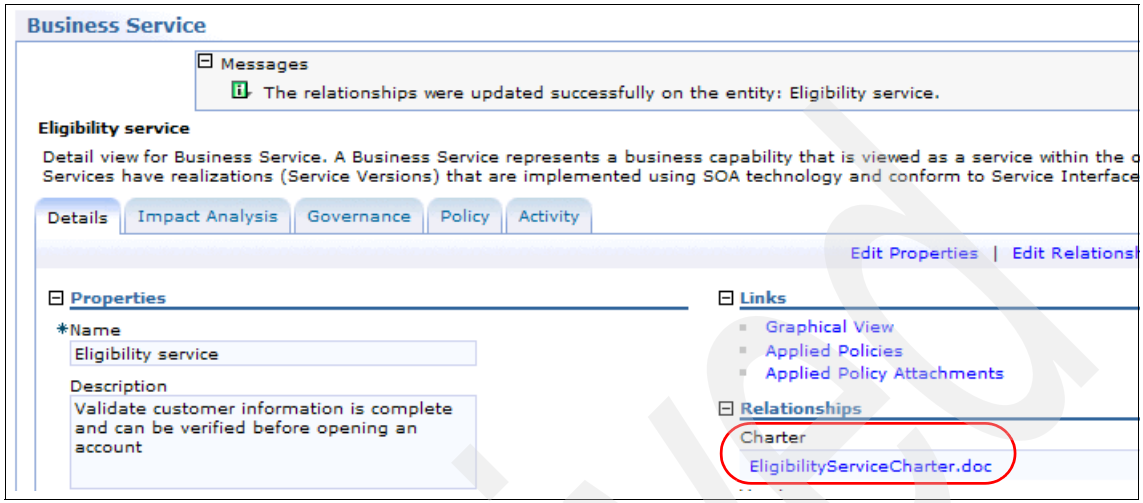


Figure 13-9 Business service charter

13.2.3 Proposing the business service

The business user has created the initial definition of the service, and it is now ready for review by the SOA governance team. To indicate the service's readiness for review and to make it available to the reviewers, the charter must be *proposed*.

Archived

To transition the business service to the Charter Review state, click **Propose for Charter Review**. Note that the new governance state is Charter Review (Figure 13-10).

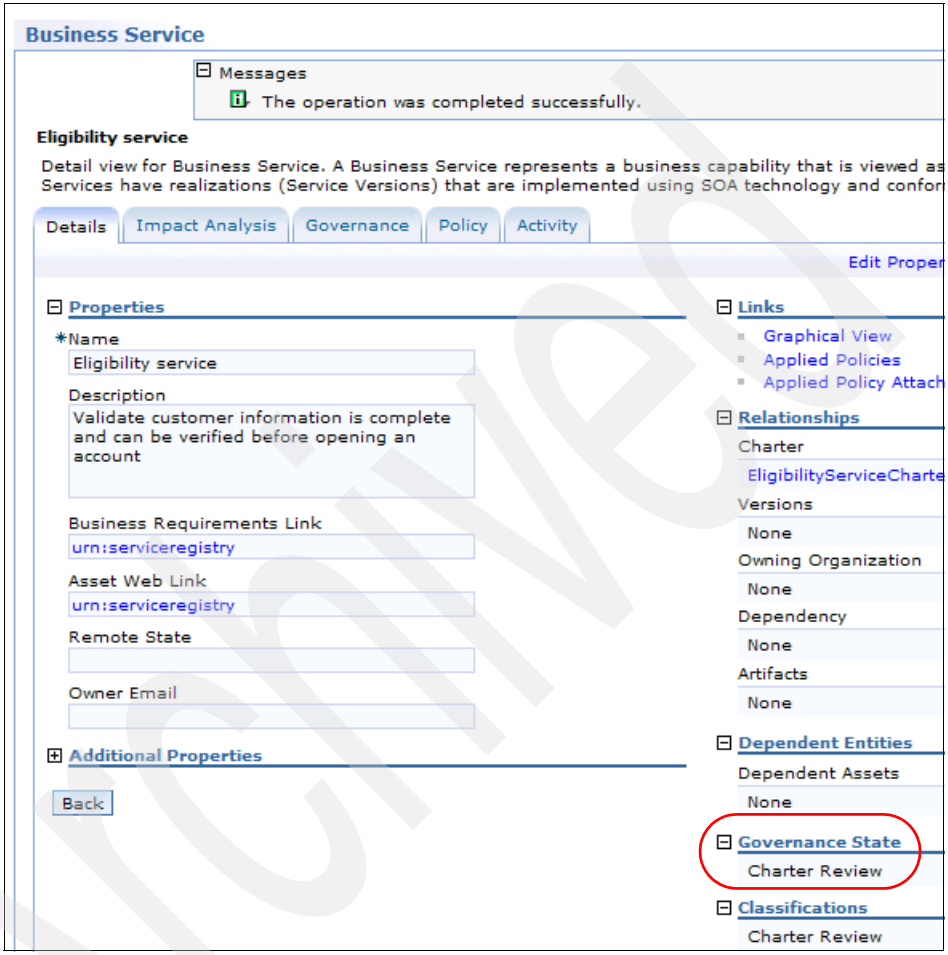


Figure 13-10 Charter review governance state

The business service and charter entities shown in Figure 13-11 are now ready for review.

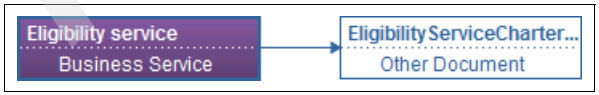


Figure 13-11 Graphical view of business service and charter entities

Figure 13-12 shows the status of the WSRR entities after the business capability is created.

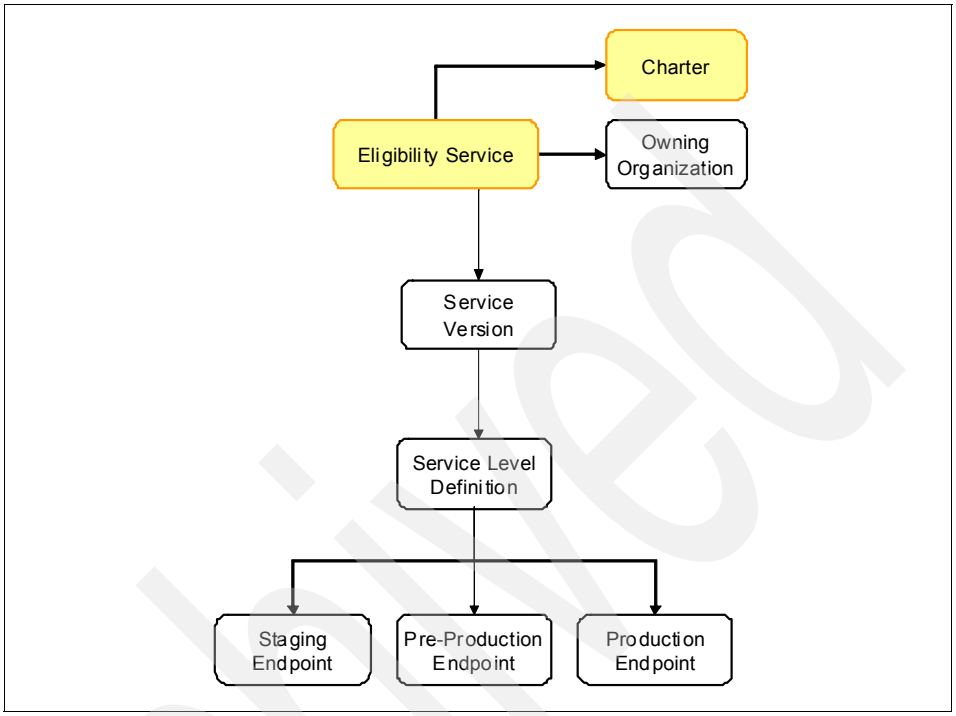


Figure 13-12 Status after business capability created

13.3 Reviewing and approving the business service

The SOA governance team ensures that governance processes are enforced. Before the business service can be transitioned to the next stage in the life cycle, the team must ensure that the proposed capability does not duplicate other services in the registry and that an owning organization is assigned that will be responsible for all versions of this capability and for managing requirements for this service.

Archived

The next phase in governing a new service at JKHLE is approving the business service capability as illustrated in Figure 13-13. We describe this process in this section.

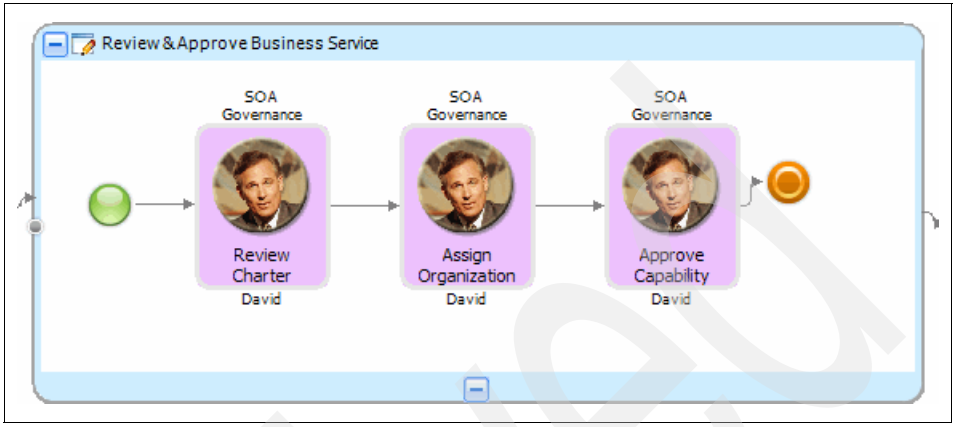


Figure 13-13 Approving the business service process

13.3.1 Reviewing the new business service

David, chairman of the SOA governance team reviews the business service definition and charter.

To review the business service definition and charter, log on to WSRR and select the **SOA Governance** perspective.

1. Click **Tasks** → **Business Capability Tasks** → **Business Capabilities for Approval**, as shown in Figure 13-14, to display all business capabilities in the Charter Review state.

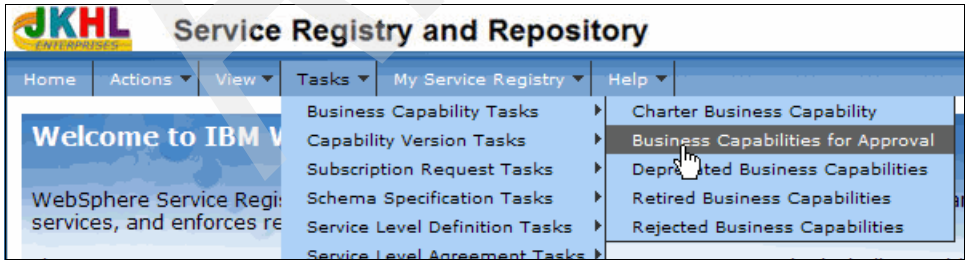


Figure 13-14 Navigating to business capabilities

2. Click **Eligibility service** to display the business service details, and review the basic information in the business service definition.
3. Click **EligibilityServiceCharter.doc** under the Charter relationship, as shown in Figure 13-15.

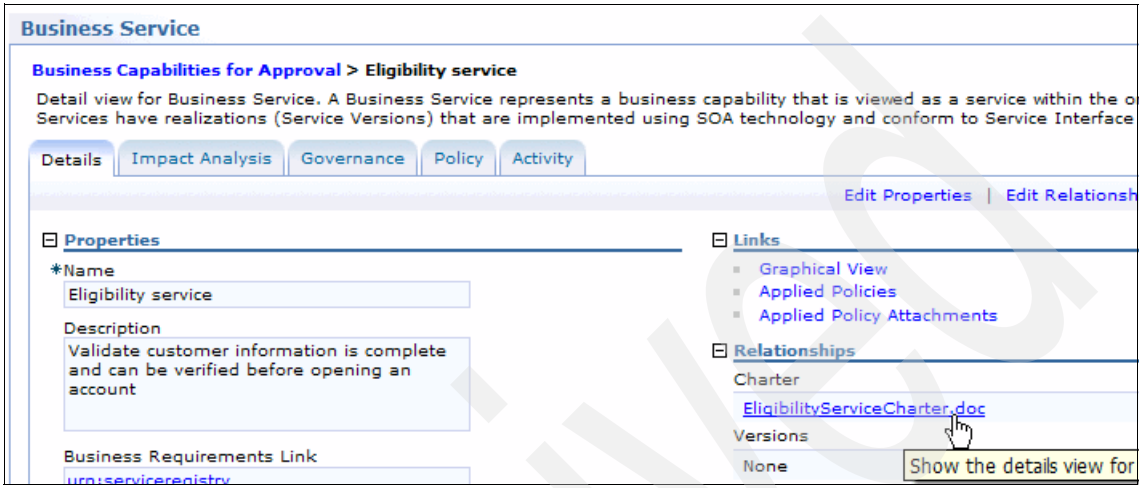


Figure 13-15 Selecting the charter document

4. Click **Content** (Figure 13-16), and then click **Download Document**.

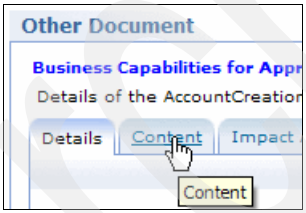


Figure 13-16 Viewing the content

5. Click **OK** to open the charter document and review the charter.
6. Close the charter document and return to the WSRR Web UI.

Archived

13.3.2 Assigning an owning organization to the business service

Now, you need to assign the organization that is responsible for this service as follows:

1. Click **Eligibility service** in the breadcrumb trail to display the business service details.
2. Click **Edit Relationships**.
3. Click **Add Organization** to the right of the Owning Organization relationship, as shown in Figure 13-17.

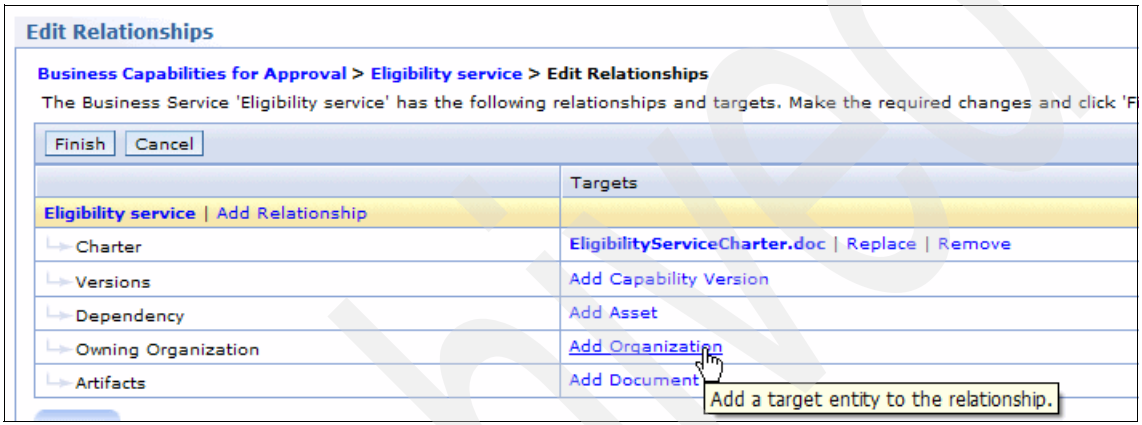


Figure 13-17 Adding an owning organization

4. Enter `C` in the **Name** field. Then, select **Common Services** from the auto suggest list, and click **Add**. The Common Services organization is added as a target of the Owning organization relationship.
5. Click **Finish** to save your changes.

13.3.3 Approving the business service

Having ensured that all of the governance requirements are satisfied and having used the search facilities in the WSRR Web UI to confirm that the proposed capability does not duplicate other services in the registry, the SOA governance team member approves the business service.

Archived

To transition the business service to the Business Capability Approved state, click **Approve Capability** (Figure 13-18).

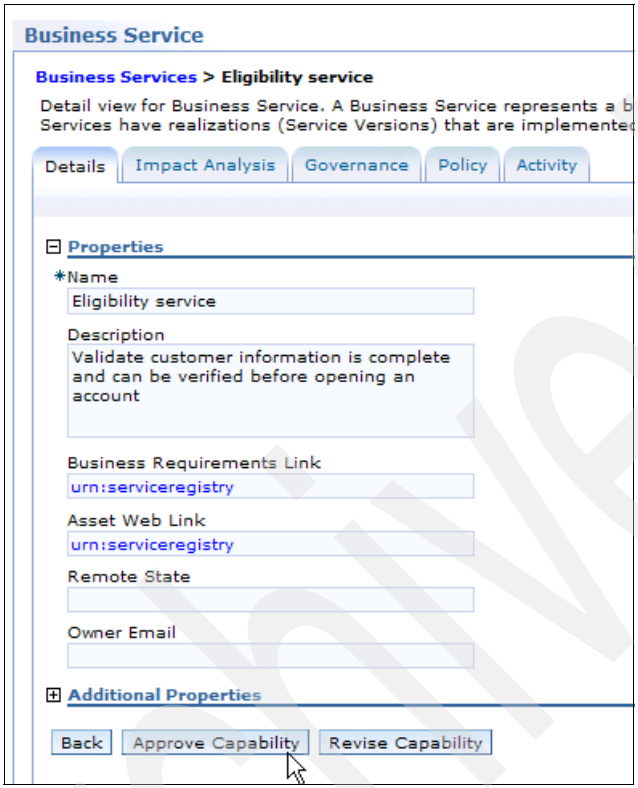


Figure 13-18 Approving the business service capability

The new governance state is Business Capability Approved (Figure 13-19).

Figure 13-19 Business capability approved business service governance state

Archived

Figure 13-12 shows the status of the WSRR entities after the business service has been approved.

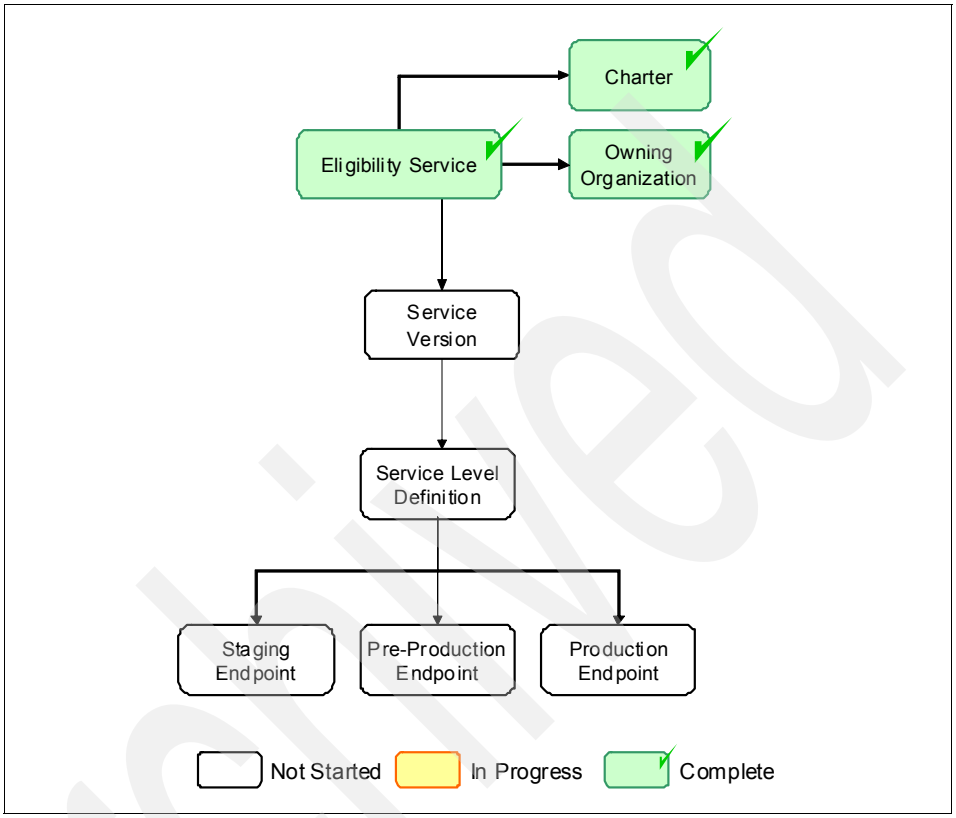


Figure 13-20 Status after business service approved

For more information about the capability life cycle, see 3.5, “Capability life cycle” on page 108.

13.4 Scoping a service version

A service (or capability) version represents a specific version or release of a service, and includes a range of functional and non-functional specifications for that version of the service. Over time multiple versions of the same business capability may be created, as the service functionality is enhanced and extended.

Archived

The next phase in the governing a new service scenario at JKHLE is scoping the service version as illustrated in Figure 13-21. We describe this process in this section.

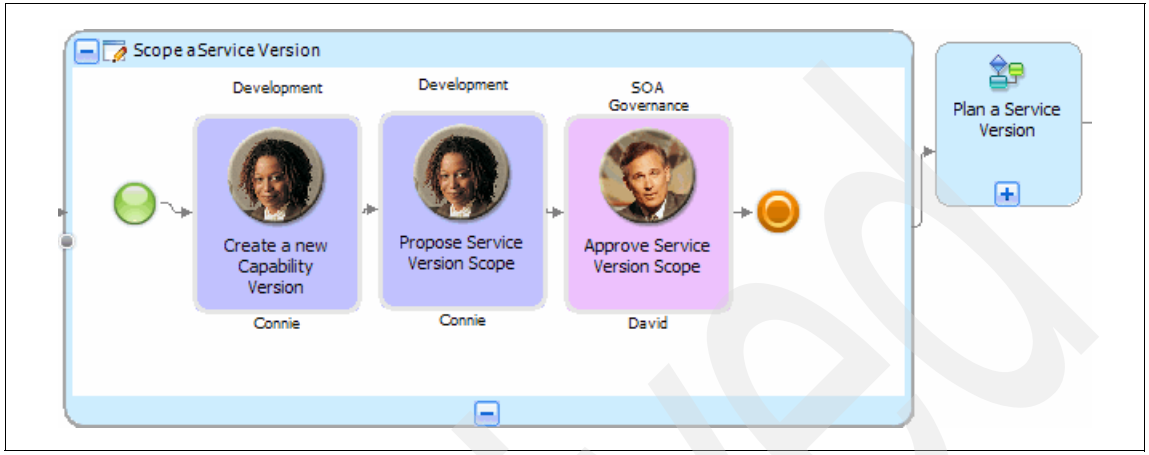


Figure 13-21 Scoping the service version process

13.4.1 Creating a new service version

After the business capability is defined, reviewed, and approved, a new service version is defined.

It is now the responsibility primarily of the development organization to create an implementation of the service. Connie, the Development Manager for Common services, has the responsibility for developing the new eligibility service for JKHLE.

To create the new service version, log on to WSRR, and select the **Development** perspective. Then, follow these steps:

1. Click **View** → **Business Governance** → **Business Capabilities** → **Business Services**. Then, click **Eligibility service** to display the business service details.
2. Assign a new service version to the business service by clicking **New Capability Version**, as shown in Figure 13-22.

Archived

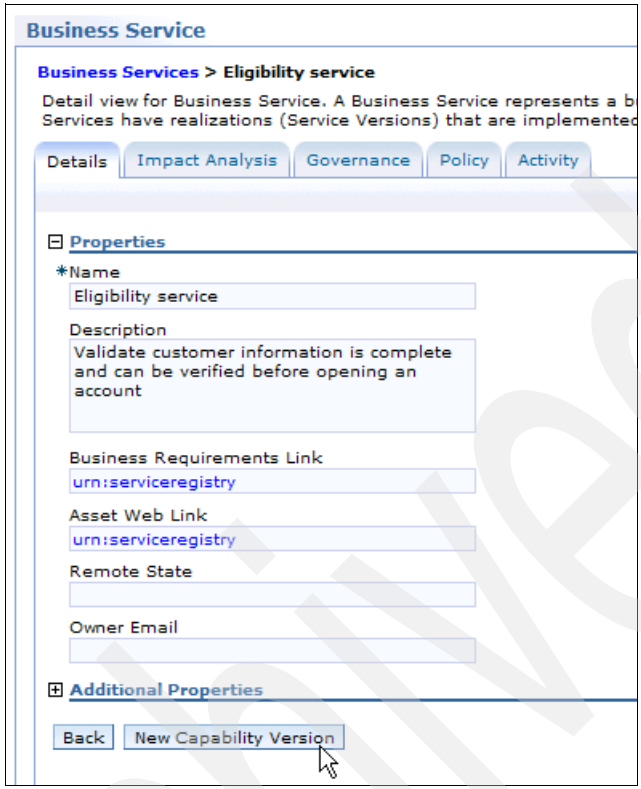


Figure 13-22 Creating a new capability version

3. Under the relationship called *Versions*, click **Eligibility service** (Figure 13-23) to display the details of the new service version.

> **Note:** A Chartered Business Capability relationship is provided, which you can use to navigate back to the business service.

Archived

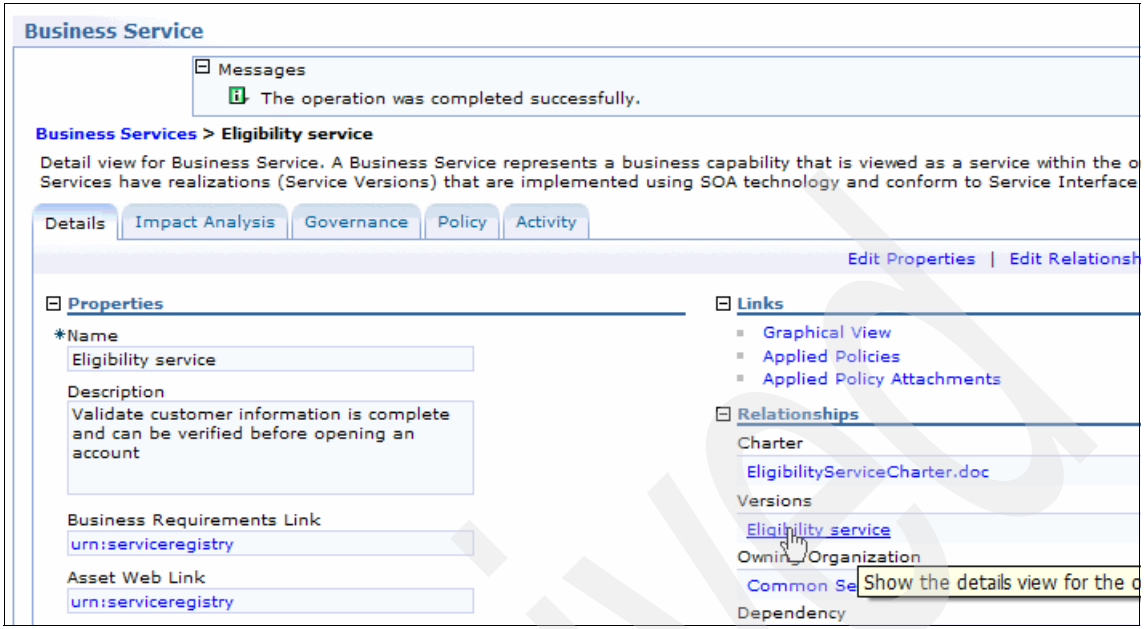


Figure 13-23 Updating the new service version

The owning organization for the new service version is set automatically to Common Services, the same as the business service. You can change this owning organization if necessary.

4. Click **Edit Properties**, as shown in Figure 13-24.

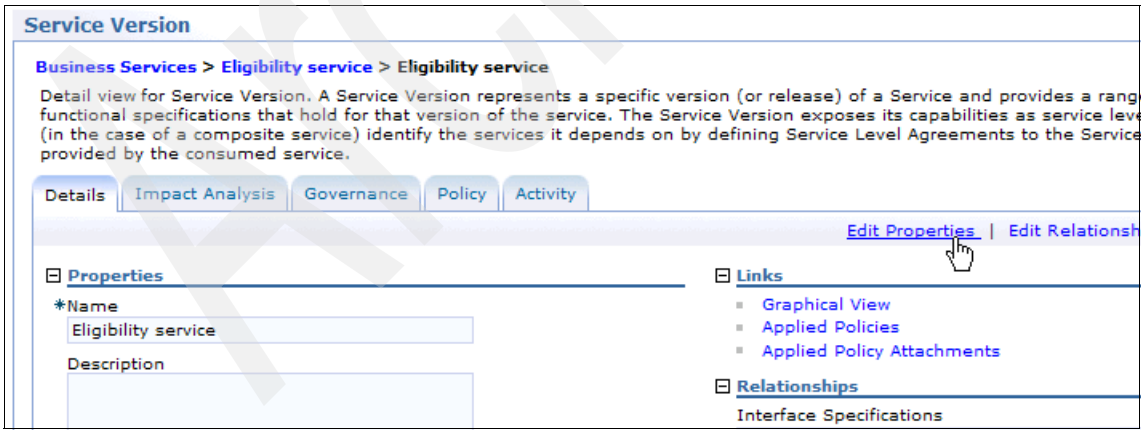


Figure 13-24 Editing service version properties

Archived

5. Change the following property values:
   - Name: `Eligibility service (1.0)`
   - Version: `1.0`
   - Description: `Service to determine customer account eligibility.`
   - Version Requirements Link:
     `http://requirements.jkhle.com/requirements.jsp?id=1210`

     This is a link, fictitious in this case, to the relevant item in JKHLE's requirements tracking tool.
6. Click **OK** to save the changes.

The governance state is *Identified*. This state is the initial state in the model phase of the SOA life cycle; a new service version is entered into the SOA life cycle automatically. Figure 13-25 illustrates the full SOA life cycle.

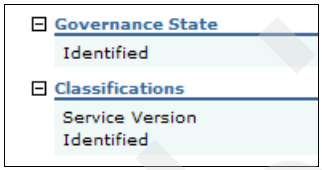


Figure 13-25 Service version governance identified state

13.4.2 Proposing the service version scope

Now that the scope of the service version is defined, it must be circulated for review so that all potential consumers of the service can verify that the requirements are within the proposed scope from the development team.

You transition the service version to the Scope Review state by clicking **Propose Scope**. Note that the new governance state is Scope Review, as shown in Figure 13-26.

Figure 13-26 Scope review service version governance state

13.4.3 Approving the service version scope

In the Scope Review state, David from the SOA governance team reviews the service version requirements.

David carries out the following verification activities:

- Checks that this service version is warranted throughout the organization
- Checks that the requirements and stakeholders are in agreement
- Checks that the owning organization that is responsible for delivering the requirements is identified and is assigned to the service version

When the service version scope review is complete, the scope can be approved.

To transition the service version to the Scoped state, log on to WSRR, and select the **SOA Governance** perspective. Then, complete these steps:

1. Click **Tasks** → **Capability Version Tasks** → **Versions for Scope Review**, as shown in Figure 13-27.

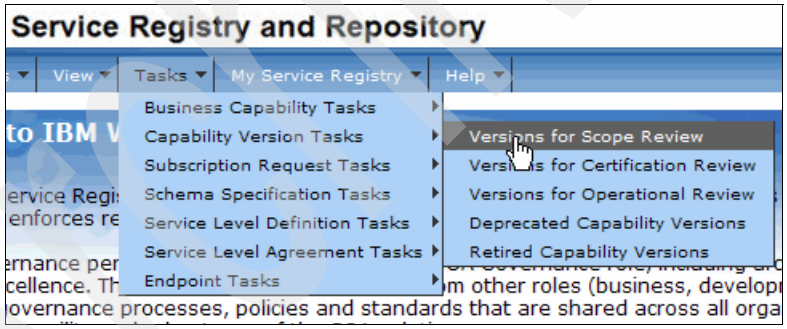


Figure 13-27 Navigating to versions for scope review

2. Click **Eligibility service (1.0)** to display the service version details.

   As part of the review activity, a member of the SOA governance team also opens the business service by following the **Eligibility service** link under the Chartered Business Capability(s) relationship. Review the charter document to ensure that the requirements for this service version are clear, and then return to the service version by following the **Eligibility service (1.0)** link under the Versions relationship.

Archived

3. Click **Approve Scope**. Note that the new governance state is Scoped, as shown in Figure 13-28.

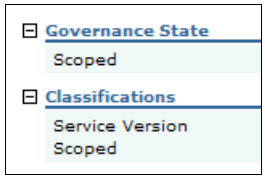


Figure 13-28 Service version governance scoped state

Figure 13-29 shows the business service, charter, and service version.

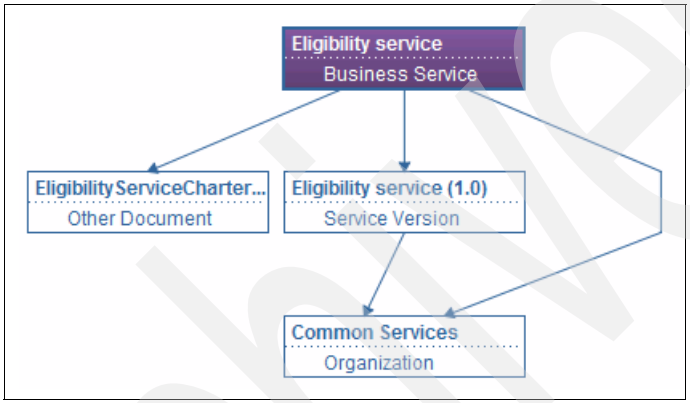


Figure 13-29 Business and service version entities

Archived

Figure 13-12 shows the status of the WSRR entities after the new service version is scoped.

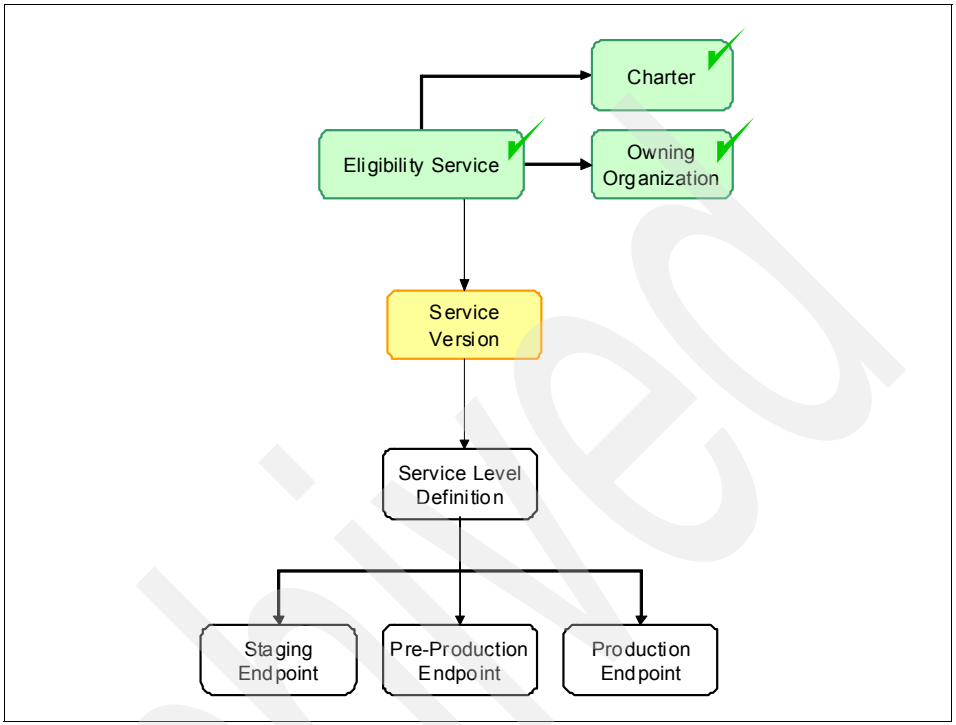


Figure 13-30 Status after service version is scoped

13.5 Planning a service version

The scoped version is now in the planning phase where the development and owning organizations must work together to define the funding and timeframes for the project. In this phase, architecture sizing and funding decisions are made, including establishing dependencies on other assets or services.

Archived

Figure 13-31 illustrates this planning process. We describe this process in this section.

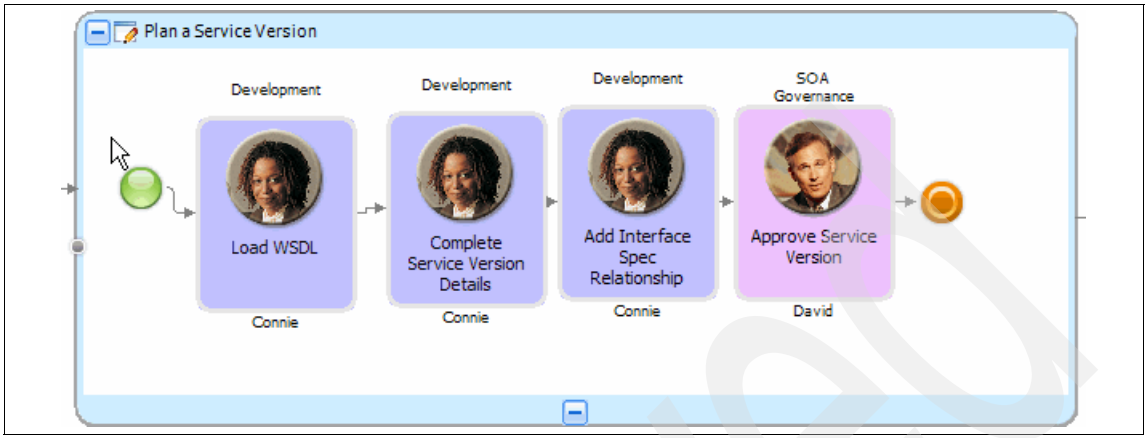


Figure 13-31 Planning a service version process

13.5.1 Loading the WSDL

At this stage in the process, you need to load the service interface information into WSRR, so that the service version can be linked to the service interface object, which is derived from the WSDL. Because the WSDL also contains an endpoint, it is not desirable for the WSDL to be linked to the service version. The WSRR correlator modifier creates a service endpoint object in WSRR automatically and relates it to the WSDL. Thus, the WSDL is contained automatically in the governance life cycle of the endpoint, enabling the ESBs to query the life cycle state of the port object in WSRR, which also reflects the state of the service endpoint. the port object is derived from the WSDL and takes on the same governance state as the WSDL itself.

For more information about the correlator modifier, see the WSRR Information Center at:

http://publib.boulder.ibm.com/infocenter/sr/v6r3/topic/com.ibm.sr.doc/cwsr_correlator_modifier_R5.html

Loading the WSDL into registry is the responsibility of the development organization. Connie, the Development Manager for the Common services area, is responsible for this phase of the scenario.

You can now load the monolithic WSDL, which includes the service interface, as follows:

1. Click **Actions** → **Load Documents**. Then, click **Browse**, and navigate to the directory where the WSDL document is located.
2. Select **EligibilityServiceV1_0_StagingPort.wsdl**, and click **Open**.
3. Enter `1.0` in the document version field.
4. Ensure that the Document Type in the Load Document pane is set to **WSDL**, as shown in Figure 13-32**.** Click **OK**.

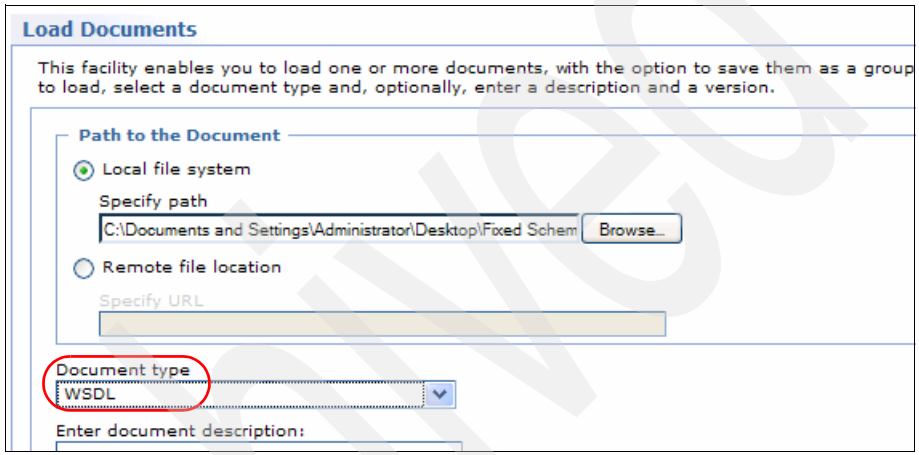


Figure 13-32 Loading a WSDL document

5. Click **Finish** to load the WSDL document. Note that the dependent XSDDocument JKHLEGlobalSchema is detected and identified as in repository, as shown in Figure 13-33.

Figure 13-33 Dependent XSD

6. Click **Finish** to save your changes.

Archived

13.5.2 Completing service version details

The service version must be updated to include proposed availability and termination dates.

Updating the service version details is the responsibility of the development organization. Connie, the Development Manager for the Common services area, is responsible for this phase of the scenario.

To complete the service version details, log on to WSRR, and select the **Development** perspective. Then, follow these steps:

1. Click **Tasks** → **Capability Version Tasks** → **Version Planning**, as shown in Figure 13-34.

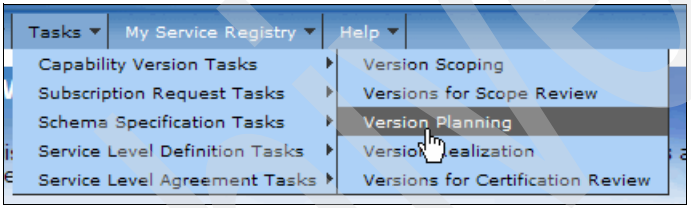


Figure 13-34 Navigating to version planning tasks

2. Click **Eligibility service (1.0)** to display the service version details, as shown in Figure 13-35.

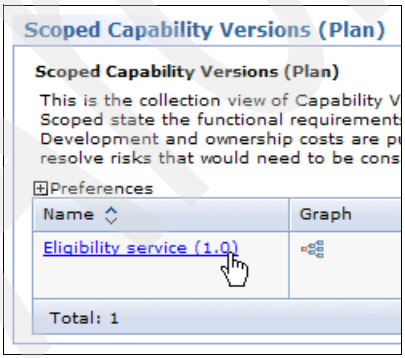


Figure 13-35 Selecting the service version

3. Then click **Edit Properties**. Enter values of your choice in the **Version Availability Date** and **Version Termination Date** fields.

4. Click **OK** to save your changes.

13.5.3 Adding the interface specification relationship

A relationship must now be created between the service version and the interface specification. Follow these steps:

1. Click **Edit Relationships**.
2. Click **Add Service Interface** to the right of the Interface Specifications relationship, as shown in Figure 13-36.

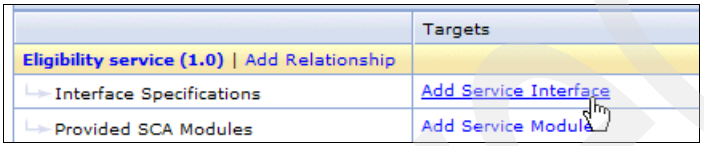


Figure 13-36 Adding the service interface relationship

3. Enter E in the Name field, and select **EligibilityV1_0** from the auto suggest list as shown in Figure 13-37. Then, click **Add**. The Eligibility Service Interface is added as a target of the interface specification.

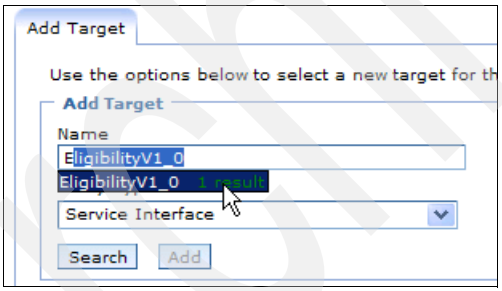


Figure 13-37 Selecting the WSDL

4. Click **Finish** to save your changes.

Archived

13.5.4 Approving the service version

After all parties have agreed the details in the plan, David from the SOA governance team can approve the service version plan.

To transition the service version to the Planned state, log on to WSRR, and select the **SOA Governance** perspective. Then, follow these steps:

1. Click **Tasks** → **Capability Version Tasks** → **Version Planning**, as shown in Figure 13-27.

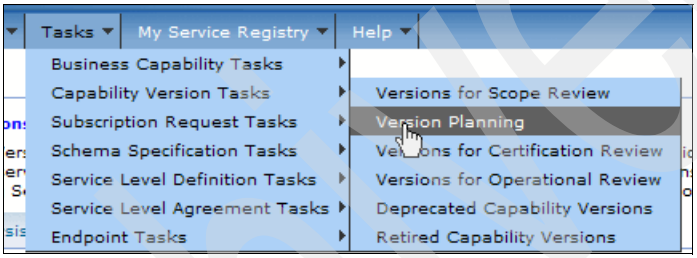


Figure 13-38 Navigating to version planning

2. Click **Eligibility service (1.0)** to display the service version details.
3. Click **Approve Specification**. Note that the new governance state is Specified, as shown in Figure 13-39.

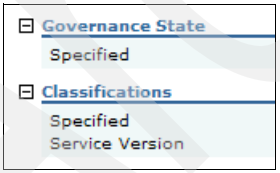


Figure 13-39 Specified service version governance state

Archived

Figure 13-12 shows the status of the WSRR entities after the new service version is planned.

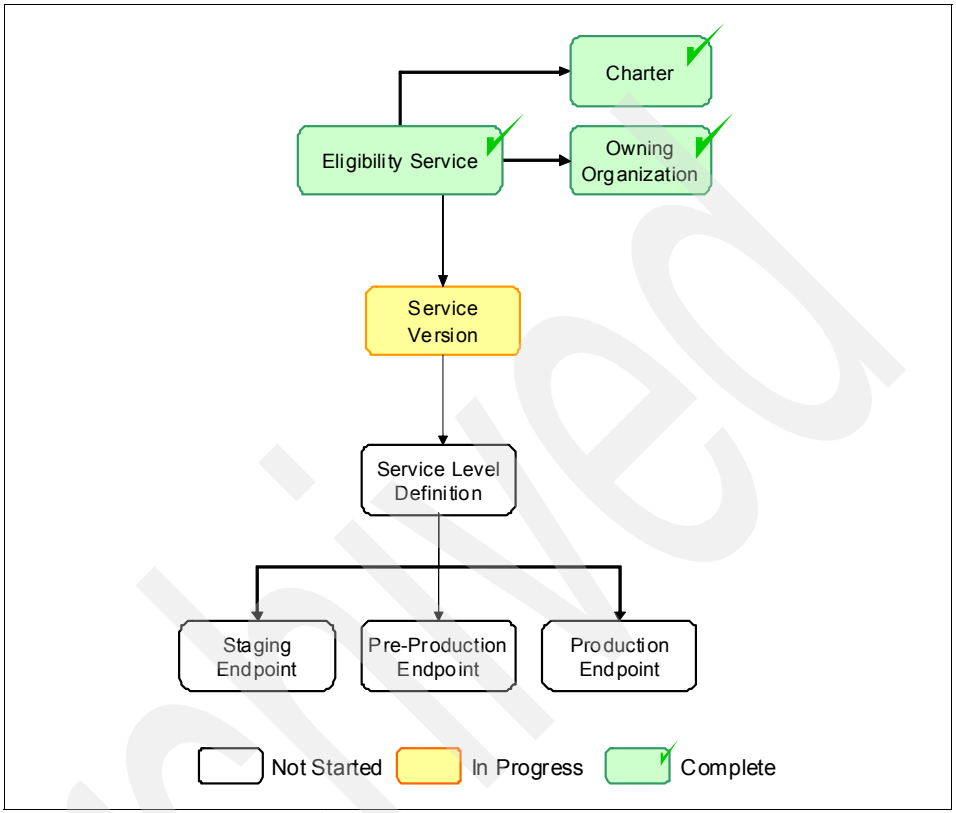


Figure 13-40 Status after service version is planned

Archived

13.6 Creating a service level definition

Now that the interface, schema, and business objects are defined, the development team must define the service level definition to which the service version will adhere. The service level definition specifies non-functional or quality of service characteristics to be provided by the service.

Figure 13-41 illustrates the process for creating a service level definition. We describe this process in this section.

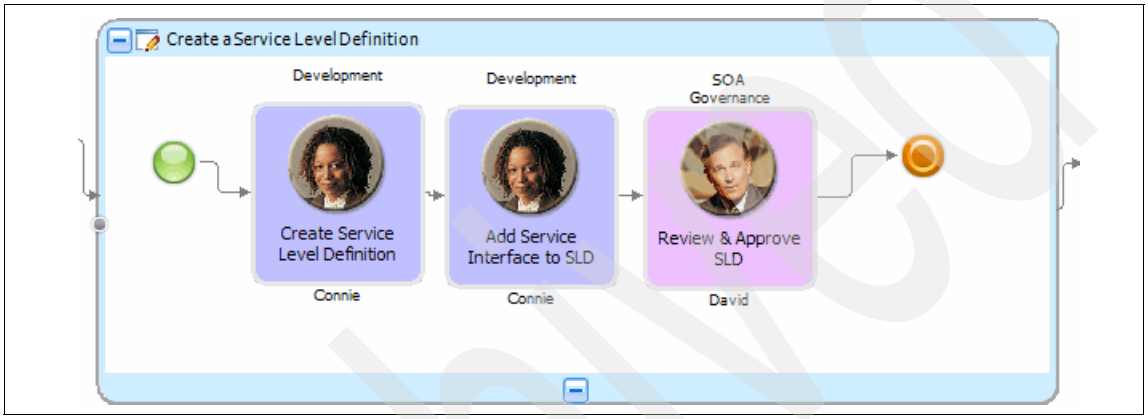


Figure 13-41 Creating a service level definition process

13.6.1 Creating a service level definition

Creating the service level definition (SLD) is the responsibility of the development organization. Connie, the Development Manager for the Common services area, is responsible for this phase of the scenario.

To create the new service level definition, log on to WSRR, and select the **Development** perspective. Then, follow these steps:

1. Click **Tasks** → **Capability Version Tasks** → **Define SLD & prepare for staging**, as shown in Figure 13-42.

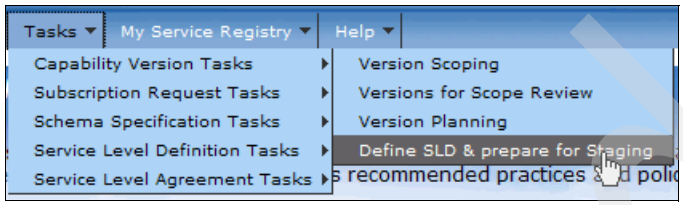


Figure 13-42 Navigate to versions for staging

2. Click **Eligibility service (1.0)** to display the service version details.
3. Click **New SLD**.
4. Click **SLD - Eligibility service (1.0)** under the Provides relationship to display the SLD details, as shown in Figure 13-43.

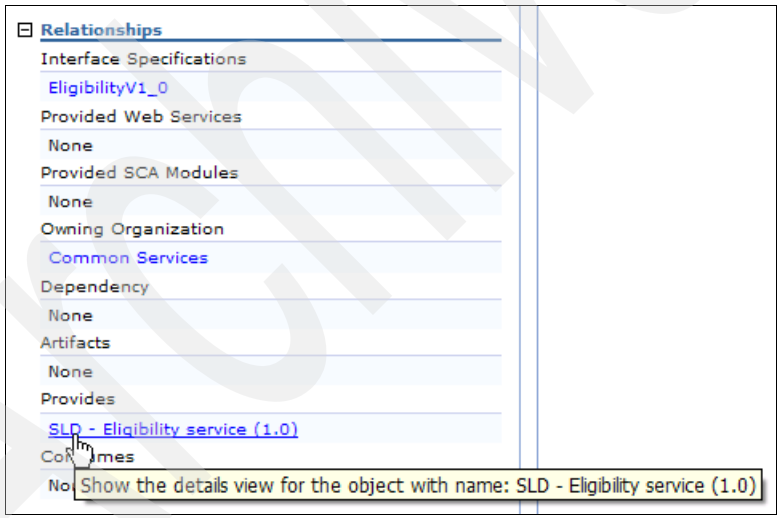


Figure 13-43 Selecting the SLD

Archived

5. Click **Edit Properties**, and enter the following information, as shown in Figure 13-44:
   - Average Response Time: `100`
   - Availability: `Working Hours Only`

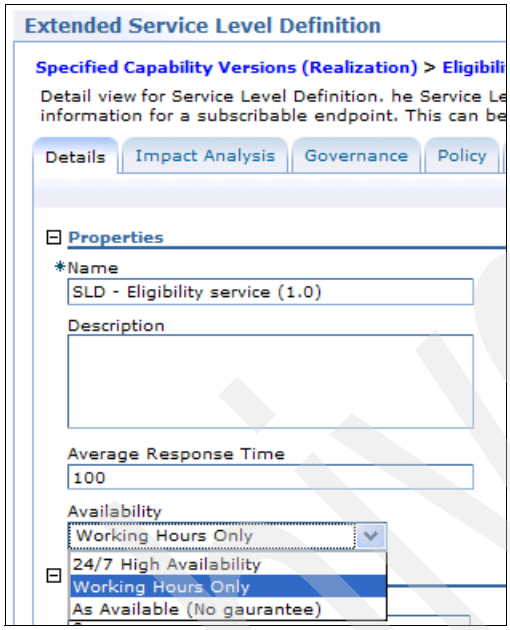


Figure 13-44 Entering the SLD details

6. Click **OK** to save your changes.

Archived

13.6.2 Adding the service interface

You now need to create a relationship between the service level definition and the service interface as follows:

1. Click **Edit Relationships**.
2. Click **Add Service Interface** to the right of the Service Interface relationship, as shown in Figure 13-45.

Figure 13-45 Adding the service interface to the SLD

3. Enter E in the Name field, select **EligibilityV1_0** (Figure 13-46) from the auto suggest list, and click **Add**. The service interface is added as a target of the service interface relationship.

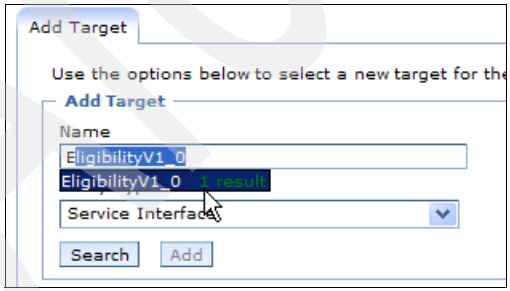


Figure 13-46 Selecting the service interface

4. Click **Finish** to save the relationship change.

Archived

5. Click **Propose Scope** and note that the new governance state is SLD Scope Review (Figure 13-47).

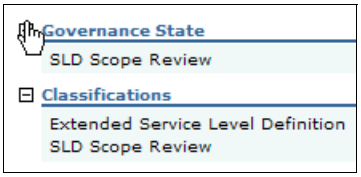


Figure 13-47 Extended SLD scope review governance state

13.6.3 Approving the service level definition

The SOA governance team confirm that the service level definition meets the non-functional requirements. David from the SOA governance team can then approve the service level definition scope.

To transition the service version to the Subscribable state, log on to WSRR, and select the **SOA Governance** perspective. Then, follow these steps:

1. Click **Tasks** → **Service Level Definition Tasks** → **Service Level Definition for Scope Review**, as shown in Figure 13-48.

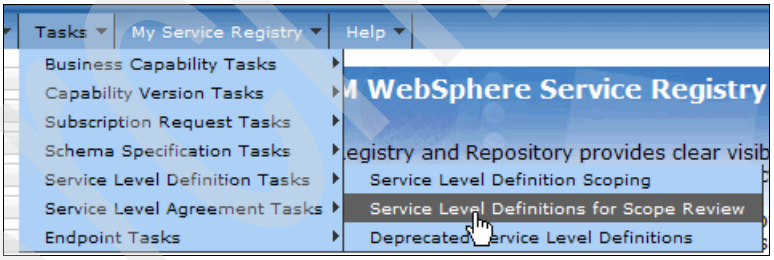


Figure 13-48 Navigating to SLD for scope review

2. Click **SLD - Eligibility service (1.0)** to display the SLD details.

Archived

3. Click **Approve Scope**. Note that the new governance state is SLD Subscribable, as shown in Figure 13-49.

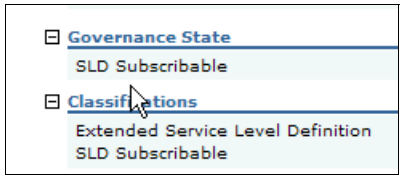


Figure 13-49 Extended SLD subcribable governance state

The modified SLD life cycle is shown in Figure 5-28 on page 198.

Figure 13-50 shows the status of the WSRR entities after the new service level definition is created.

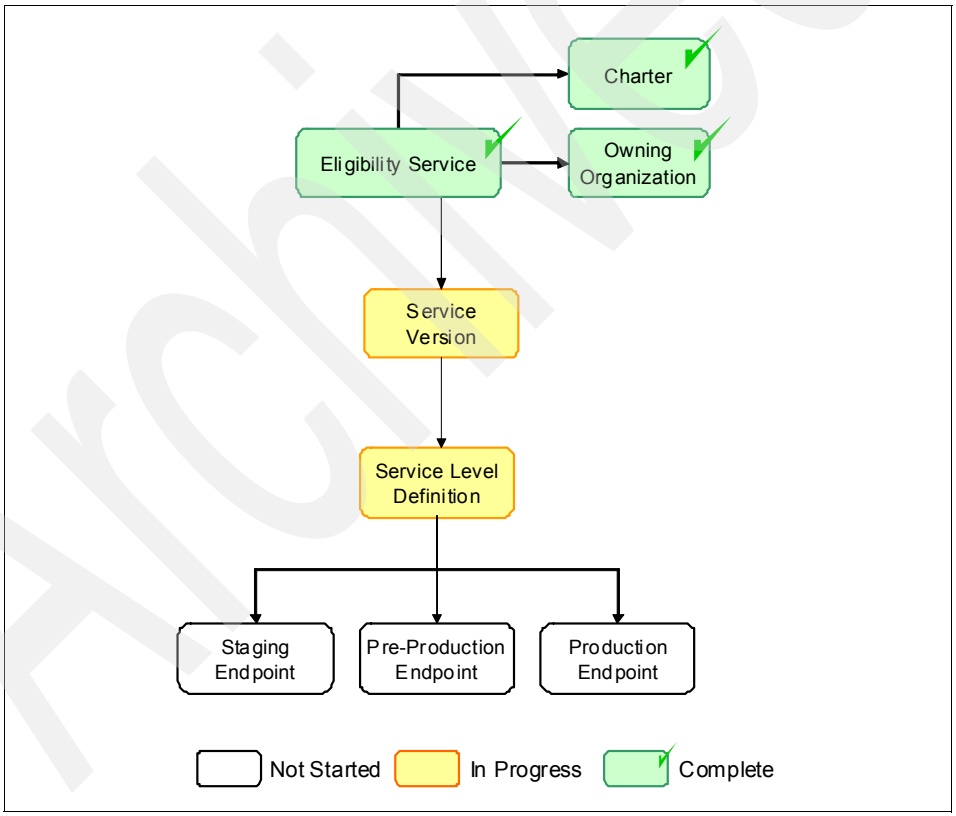


Figure 13-50 Status after creation of a service level definition

Archived

13.7 Deploying a service version to staging

Having created and unit tested an implementation of the service, the Development team is ready to pass the service to the Operations team to deploy to a staging environment for testing.

Refer to Chapter 5, “Modeling in WebSphere Service Registry and Repository Studio” on page 163 for details of the SOA life cycle, which is used to govern the service version.

Figure 13-51 illustrates the process for deploying a service version.

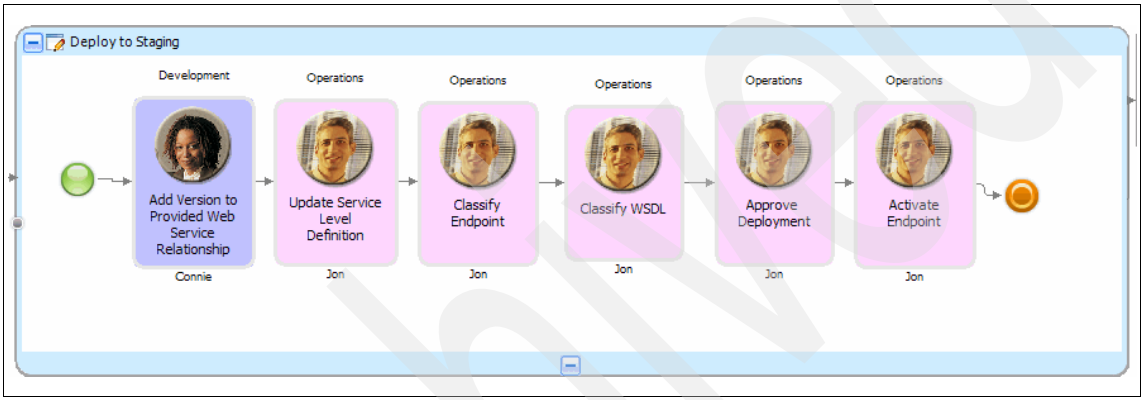


Figure 13-51 Deploying a service version to staging

Archived

Figure 13-52 illustrates the WSRR environment at JKHLE. The new eligibility service is verified and promoted to the staging environment for functional testing.

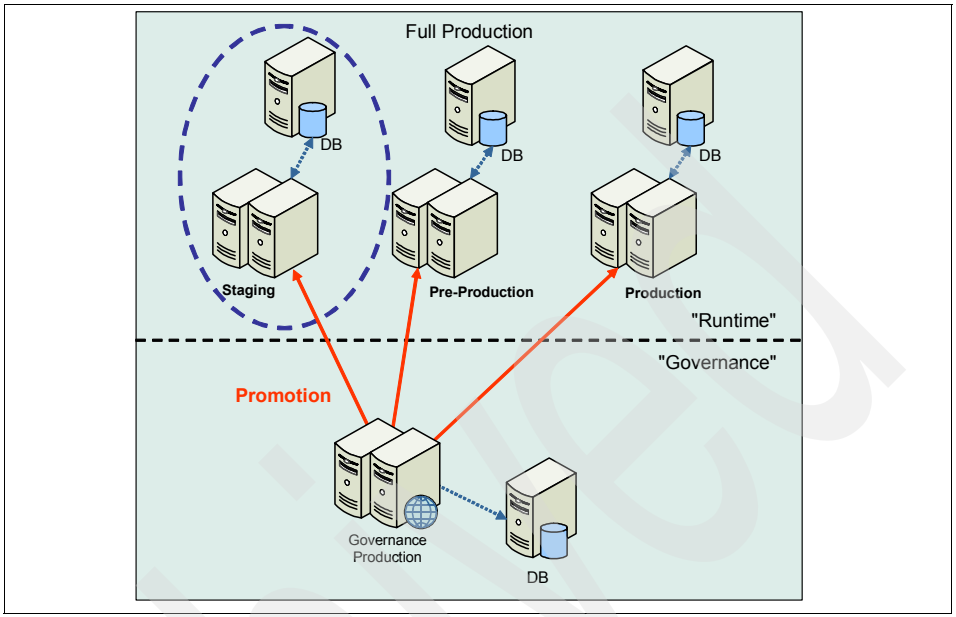


Figure 13-52 Deploying to the staging environment

13.7.1 Adding relationship to provided Web service

The service implementation WSDL containing the staging environment endpoint was loaded previously into WSRR when the service version was planned. A relationship now needs to be created between the service version and the provided Web service endpoint. Connie the Development Manager for the Common services area is responsible for this phase of the scenario.

To add the target of the provided Web service relationship, log on to WSRR, and select the **Development** perspective. Then, to create a relationship between the service version and the provided Web service, follow these steps:

1. Click **Tasks** → **Capability Version Tasks** → **Define SLD & prepare for staging**.
2. Click **Eligibility service (1.0)** to display the service version details.
3. Click **Edit Relationships**.

4. Click **Add Service** to the right of the **Provided Web Services** relationship, as shown in Figure 13-53.

Figure 13-53 Adding the provided Web service relationship

Archived

5. Enter E in the **Name** field, select **EligibilityV1_0** from the auto suggest list, and click **Add**. The endpoint is added as a target of the Provided Web Services relationship (Figure 13-54).

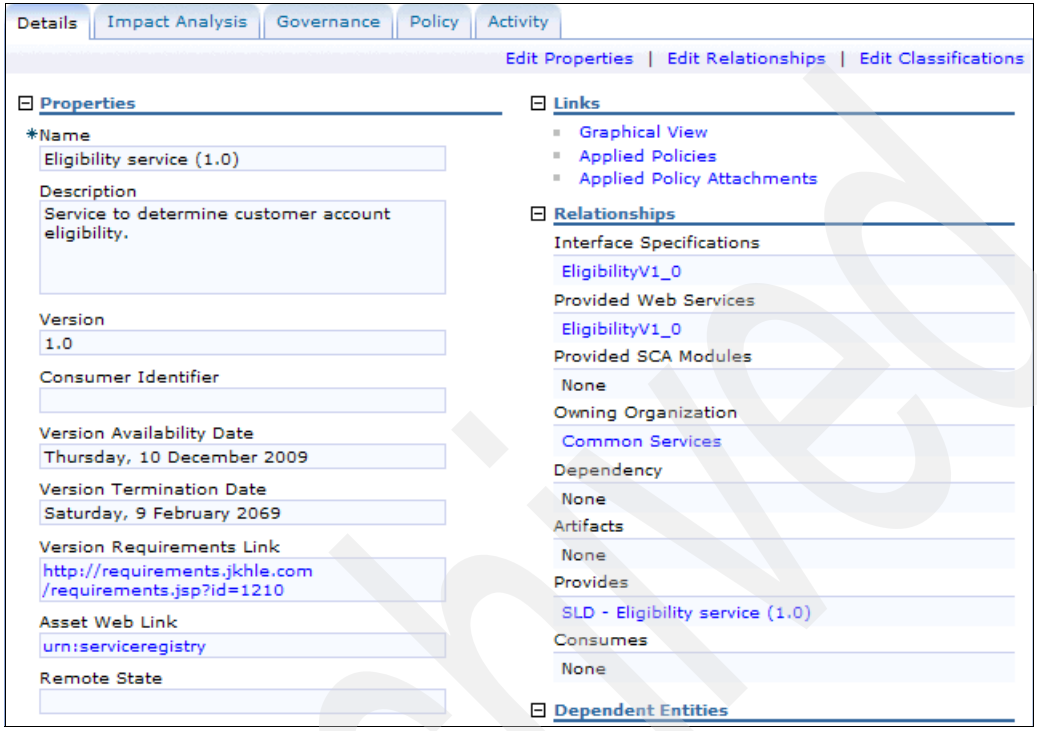


Figure 13-54 Realized service definition

6. Click **Finish** to save changes.

13.7.2 Updating the service level definition

Having loaded the endpoint into the WSRR, the next step in making the service consumable in the staging environment is to associate the staging endpoint with the service level definition.

Jon from the Operations team is responsible for updating the service level definition.

Archived

To update the service level definition, log on to WSRR, and select the **Operations** perspective. Then, follow these steps:

1. Click **Tasks** → **Service Level Definition Tasks** → **Manage Subscribable Service Level Definitions**, as shown in Figure 13-55.

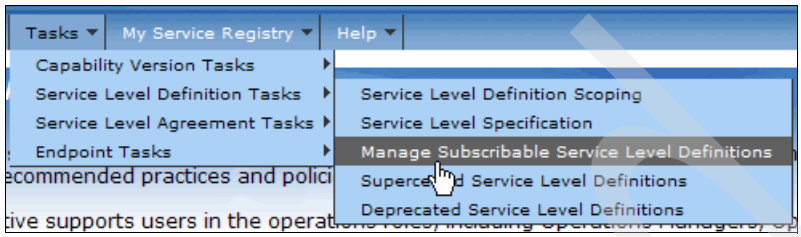


Figure 13-55 Navigating to subscribable service level definitions

2. Click **SLD - Eligibility service (1.0)**.
3. Click **Edit Relationships**.
4. Click **Add Service Endpoint** to the right of the Available Endpoints relationship.
5. Enter an asterisk (*) in the Name field, select **https://localhost:9443/EligibilityV1_0/services/EligibilityServiceV1_0_StagingPort** from the auto suggest list, and click **Add**. The service endpoint is added as a target of the Available Endpoints relationship.
6. Click **Add Service Port** to the right of the Bound Web Service Port relationship.
7. Enter E in the Name field, select **EligibilityServiceV1_0_StagingPort** from the auto suggest list, and click **Add**. The service port is added as a target of the Bound Web Service Port relationship.
8. Click **Finish** to save the changes.

Archived

The service level definition relationships is updated as shown in Figure 13-56.

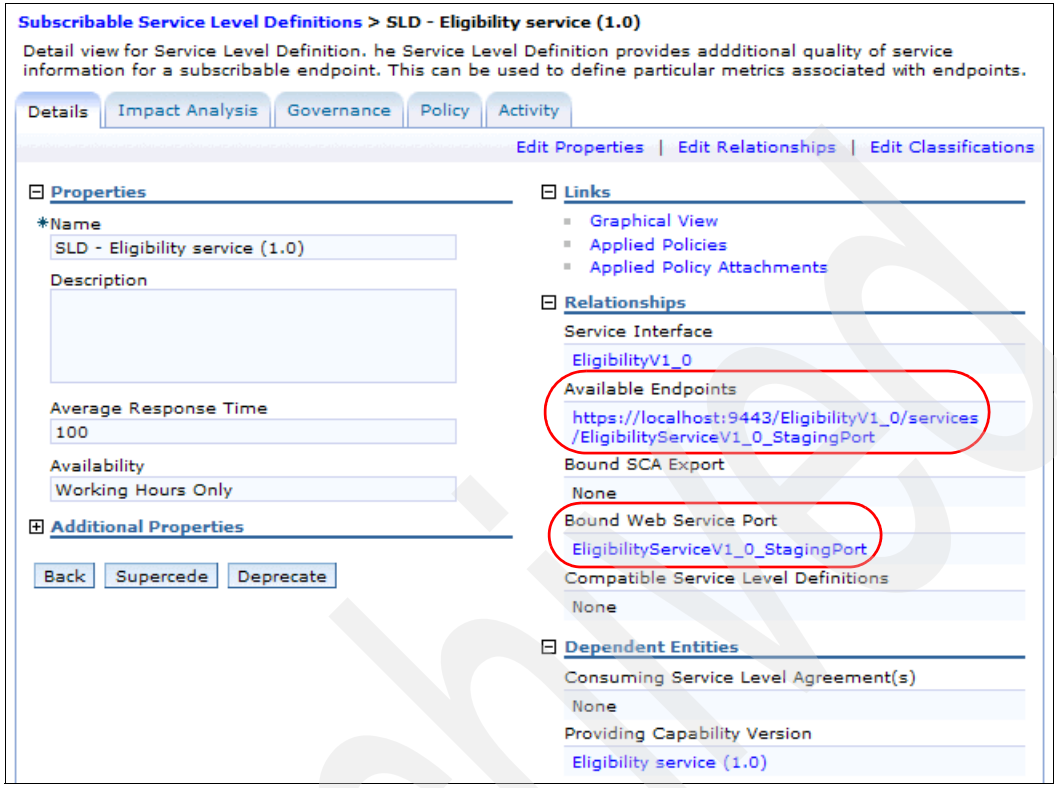


Figure 13-56 Service level definition relationships

13.7.3 Classifying the endpoint

The Operations team approves the staging endpoint to verify that the service is running but not necessarily functioning correctly. To classify the endpoint, log on to WSRR, and select the Operations perspective. Then, follow these steps:

1. Click **Tasks** → **Endpoint Tasks** → **Endpoints For Activation**.

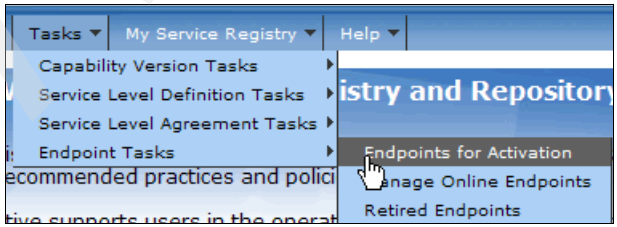


Figure 13-57 Navigate to endpoint for activation

2. Click the offline endpoint **https://localhost:9443/EligibilityV1_0/services/EligibilityServiceV1_0_StagingPort**, as shown in Figure 13-58.

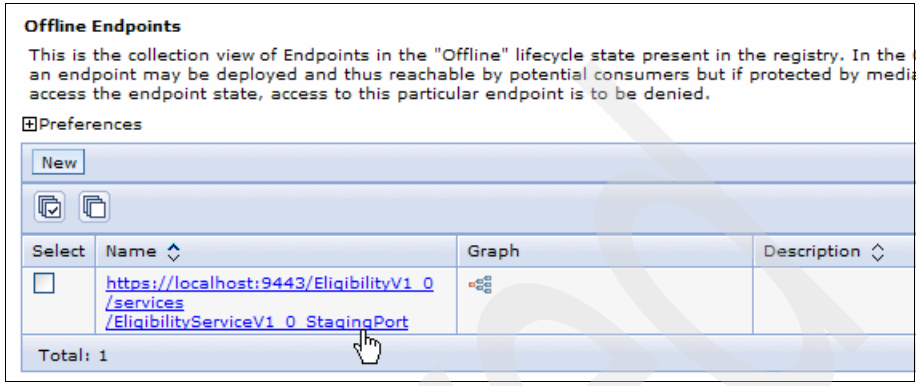


Figure 13-58 Selecting the offline endpoint

3. Click **Edit Classifications**.
4. Expand **Governance Profile Taxonomy** → **Environment**.
5. Select **Staging** as shown in Figure 13-59, and click **Add**. The Staging classification is added to the Classification list.

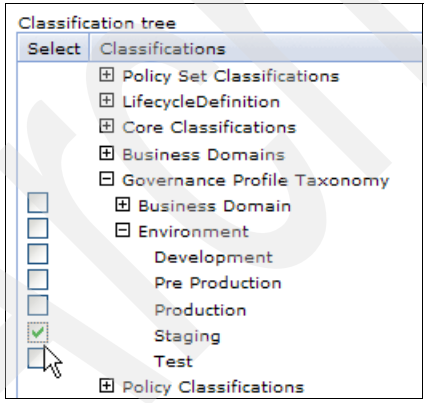


Figure 13-59 Staging

6. Click **OK** to assign the classification.

Archived

13.7.4 Classifying the WSDL

In the current scenario, JKHLE published a monolithic WSDL which includes the staging endpoint, common binding, and common interface into WSRR. Because they are also publishing a monolithic WSDL for the pre-production and production endpoints, which contain the same binding and interface, they need to classify the WSDL as well as the endpoint. This classification is required because the correlated objects, such as the PortType, will be common throughout all of the services. Without this classification, when the service is promoted to pre-production, for example, the PortType would “pull in” both the staging and pre-production WSDL and promote them both to the pre-production environment, which is not desirable. This is one of the limitations of using monolithic WSDL files.

To classify the WSDL:

1. Click the **EligibilityServiceV1_0_StagingPort.wsdl** link under the sm63_sourceDoucments relationship (Figure 13-60 on page 459).
2. Click **Edit Classifications**.
3. Expand **Governance Profile Taxonomy** → **Environment.**
4. Select **Staging** (Figure 13-59), and click **Add**. The Staging classification is added to the Classification list.
5. Click **OK** to assign the classification.

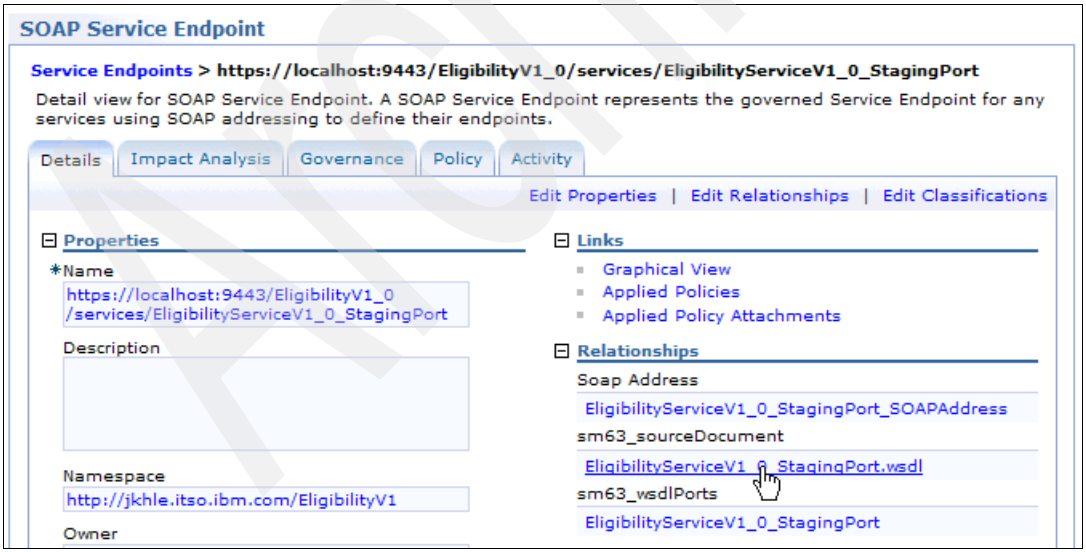


Figure 13-60 Navigating to the source document

13.7.5 Approving staging deployment

At this point, the service is released officially for use in the staging environment as follows:

1. Click **Tasks** → **Capability Version Tasks** → **Define SLD & Prepare for Staging**, as shown in Figure 13-61.

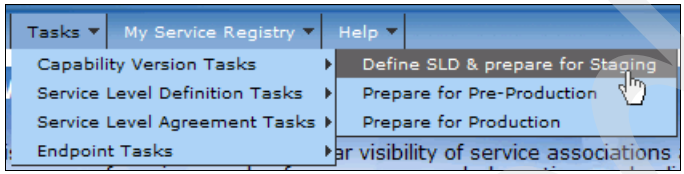


Figure 13-61 Navigate to versions for staging

2. Click **Eligibility service (1.0)** to display the service version details.
3. Click **Approve Staging Deployment**. Note that the new governance state is Staged, as shown in Figure 13-62.

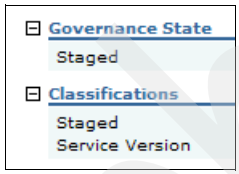


Figure 13-62 Service version staged governance state

13.7.6 Activating the endpoint

Although the service is promoted to the staging environment and approved for use, the endpoint needs to be activated, which updates its status to Online. To activate the endpoint, follow these steps:

1. Click **Tasks** → **Endpoints** → **Endpoints For Activation**.
2. Click the offline endpoint **https://localhost:9443/EligibilityV1_0/services/EligibilityServiceV1_0_StaggingPort**.
3. Click **Approve For Use. N**ote that the new governance state is Online. as shown in Figure 13-63.

Archived

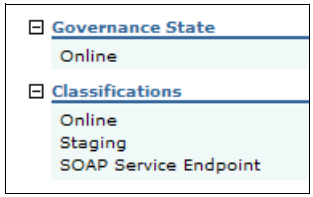


Figure 13-63 Service endpoint online governance state

The eligibility service is now promoted to the staging environment. In the staging environment, initial testing of the eligibility service is carried out before deployment to pre-production for final acceptance testing.

Figure 13-64 shows the status of the WSRR entities after the staging environment endpoint is available.

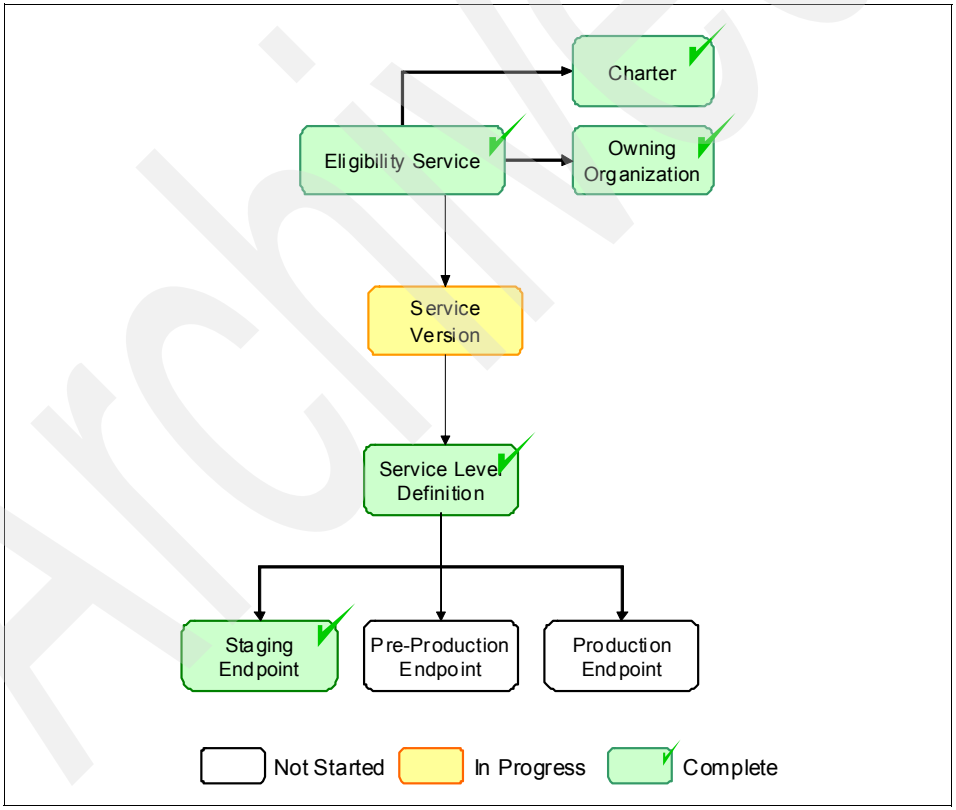


Figure 13-64 Status after deployment to the staging environment

Archived

13.8 Deploying a service version to pre-production

After functional testing concludes successfully, the service is deployed to the pre-production environment in a similar manner, and its state becomes *Certified*.

Figure 13-65 illustrates the process for deploying a service version to pre-production. We describe this process in this section.

Figure 13-65 Deploying a service version to pre-production process

Archived

Figure 13-66 shows the deployment topology including the pre-production environment.

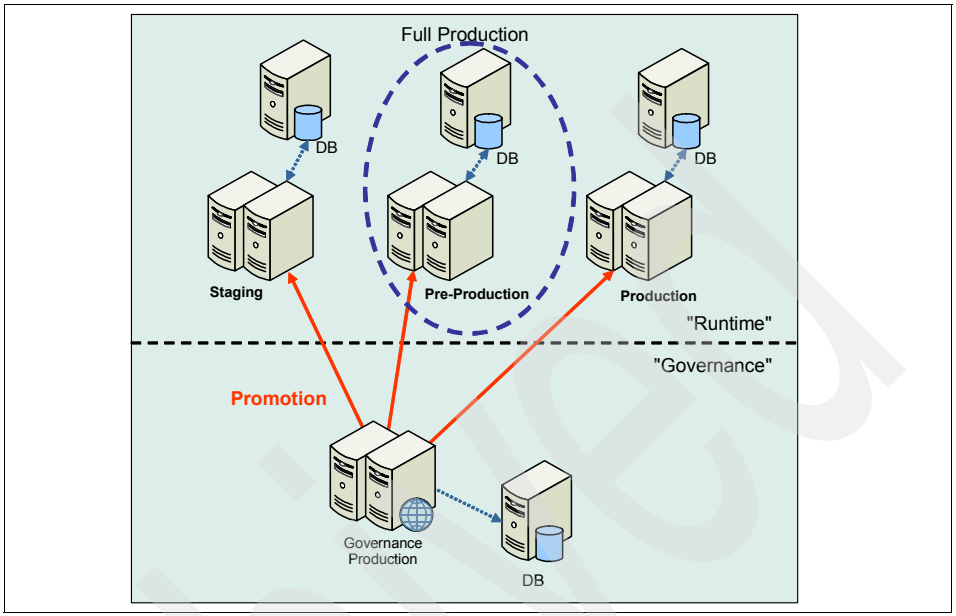


Figure 13-66 Deploying to the pre-production environment

13.8.1 Loading the pre-production endpoint WSDL

The service implementation WSDL is loaded into WSRR. Connie the Development Manager for the Common services area is responsible for this phase of the scenario.

To load the pre-production endpoint WSDL, log on to WSRR, and select the **Development** perspective. Then, follow these steps:

1. Click **Actions** → **Load Documents**.
2. Ensure that the **Document type** is set to **WSDL**.
3. Click **Browse** and navigate to the directory where the pre-production WSDL is located.
4. Select **EligibilityV1_0_PreProductionPort.wsdl**, and click **Open**.
5. Enter **1.0** in the document version field. Then, click **OK**.

6. Click **Finish** to load the WSDL document.

The pre-production endpoint WSDL is loaded, as shown in Figure 13-67.

Figure 13-67 Loading the pre-production endpoint WSDL

13.8.2 Updating the service level definition

Having loaded the pre-production endpoint into WSRR, the next step in making the service consumable in the pre-production environment is to associate the pre-production endpoint with the service level definition.

Jon from the Operations team is responsible for updating the service level definition.

To update the service level definition, log on to WSRR, and select the **Operations** perspective. Then, follow these steps:

1. Click **Tasks** → **Service Level Definition Tasks** → **Manage Subscribable Service Level Definitions**.
2. Click **SLD - Eligibility service (1.0)**.
3. Click **Edit Relationships**.
4. Click **Add Service Endpoint** alongside the **Available Endpoints** relationship.
5. Enter `*Pre` in the **Name** field, select **https://localhost:9443/EligibilityV1_0/services/EligibilityServiceV1_0_PreProductionPort** from the auto suggest list, and click **Add**. The service endpoint is added as a target of the Available Endpoints relationship.
6. Click **Add Service Port** to the right of the **Bound Web Service Port** relationship.
7. Enter `*Pre` in the **Name** field, select **EligibilityServiceV1_0_PreProductionPort** from the auto suggest list, and click **Add**. The service port is added as a target of the Bound Web Service Port relationship.
8. Click **Finish** to save your changes.

Archived

The service level definition relationships is updated to include the pre-production endpoint information, as shown in Figure 13-68.

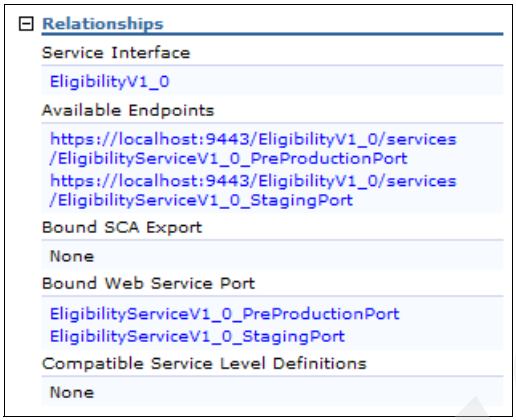


Figure 13-68 Service level definitions with pre-production endpoints

13.8.3 Classifying the endpoint

The Operations team approves the pre-production endpoint. To classify the endpoint, log on to WSRR, and select the **Operations** perspective. Then, follow these steps:

1. Click **Tasks** → **Endpoint Tasks** → **Endpoints For Activation**.
2. Click the development offline endpoint **https://localhost:9443/EligibilityV1_0/services/EligibilityServiceV1_0_PreProductionPort**.
3. Click **Edit Classifications**.
4. Expand **Governance Profile Taxonomy** → **Environment**.
5. Select **Pre-Production**, and click **Add**. The pre-production classification is added to the Classification list.
6. Click **OK** to assign the classification.

13.8.4 Classifying the WSDL

To classify the WSDL, follow these steps:

1. Click the **EligibilityServiceV1_0_PreProductionPort.wsdl** link under the sm63_sourceDoucments relationship.
2. Click **Edit Classifications**.

Archived

3. Expand **Governance Profile Taxonomy** → **Environment**.
4. Select **Pre-Production**, and click **Add**. The Pre-Production classification is added to the Classification list.
5. Click **OK** to assign the classification.

13.8.5 Approving pre-production deployment

At this point the service is released officially for use in the pre-production environment as follows:

1. Click **Tasks** → **Capability Version Tasks** → **Prepare for Pre-Production**, as shown in Figure 13-69.

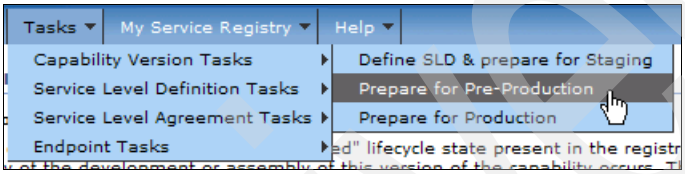


Figure 13-69 Navigate to versions for pre-production

2. Click **Eligibility service (1.0)** to display the service version details.
3. Click **Approve Certification**. Note that the new governance state is Certified as shown in Figure 13-70.

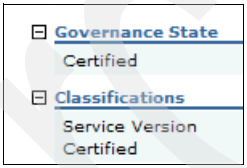


Figure 13-70 Certified service version governance state

13.8.6 Activating the endpoint

Although the service is promoted to the pre-production environment and approved for use, the endpoint needs to be activated, which updates its status to Online. To activate the endpoint:

1. Click **Tasks** → **Endpoints** → **Endpoints For Activation**.
2. Click the offline endpoint **https://localhost:9443/EligibilityV1_0/services/EligibilityServiceV1_0_PreProductionPort**.

3. Click **Approve For Use**. Note that the new governance state is Online.

The eligibility service is now promoted to the pre-production environment.

In the pre-production environment, final acceptance testing of the eligibility service is carried out to ensure that if it is made available in production it meets all of its proposed service level definitions and service level agreements. Figure 13-71 shows the status of the WSRR entities after the pre-production environment endpoint is available.

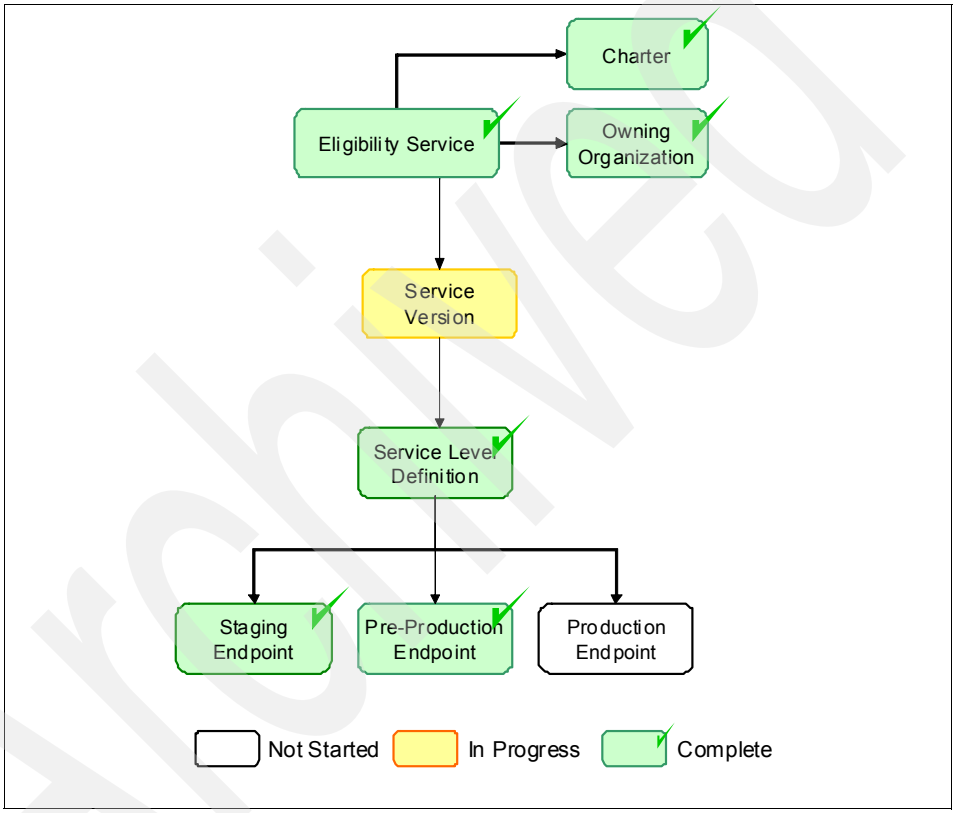


Figure 13-71 Status after deployment to the pre-production environment

Archived

13.9 Deploying a service version to production

After final testing and verification concludes successfully, the service is deployed to the production environment, and its state becomes Operational. Figure 13-72 illustrates the process for deploying a service version to production. We describe this process in this section.

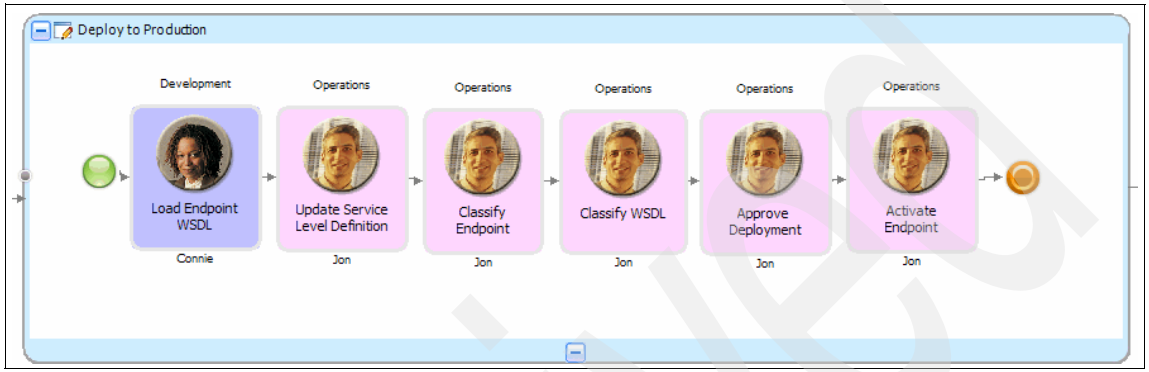


Figure 13-72 Deploying a service version to production process

Figure 13-73 shows the deployment topology including the production environment.

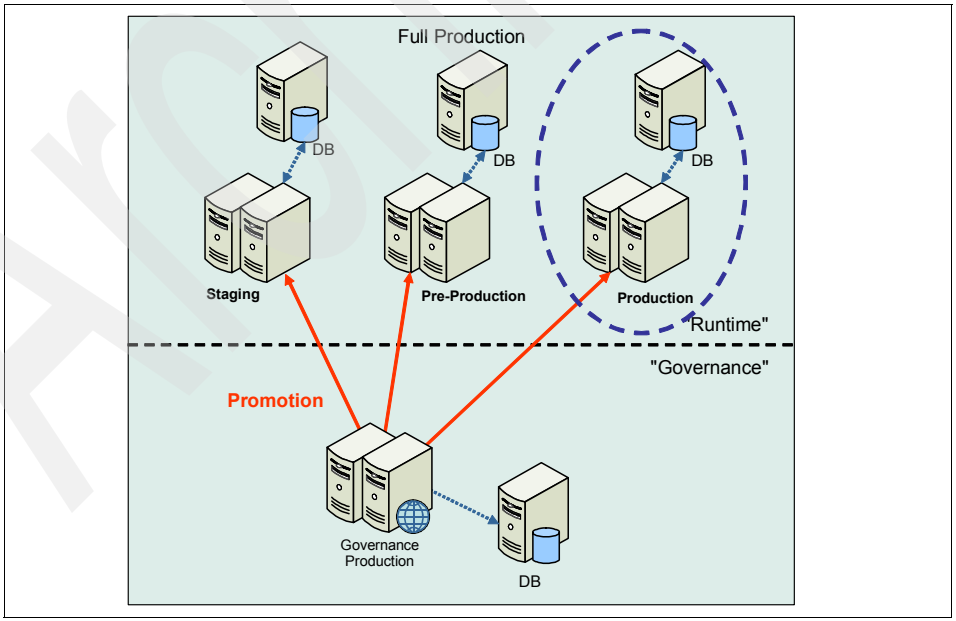


Figure 13-73 Deploying to the production environment

13.9.1 Loading the production endpoint WSDL

The service implementation WSDL is loaded into WSRR. Connie the Development Manager for the Common services area is responsible for this phase of the scenario.

To load the production endpoint WSDL, log on to WSRR, and select the **Development** perspective. Then, follow these steps:

1. Click **Actions** → **Load Documents**.
2. Ensure that the Document type is set to **WSDL**.
3. Click **Browse**, and navigate to the directory where the pre-production WSDL is located.
4. Select **EligibilityServiceV1_0_ProductionPort.wsdl**, and click **Open**.
5. Enter **1.0** in the document version field.
6. Click **OK**.
7. Click **Finish** to load the WSDL document.

The production endpoint WSDL is loaded, as shown in Figure 13-74.

Figure 13-74 Loading the production endpoint WSDL

13.9.2 Updating the service level definition

Having loaded the endpoint into the WSRR, the next step in making the service consumable in the production environment is to associate the production endpoint with the service level definition.

Jon from the Operations team is responsible for updating the service level definition.

Archived

To update the service level definition, log on to WSRR, and select the **Operations** perspective. Then, follow these steps:

1. Click **Tasks** → **Service Level Definition Tasks** → **Manage Subscribable Service Level Definitions**.
2. Click **SLD - Eligibility service (1.0)**.
3. Click **Edit Relationships**.
4. Click **Add Service Endpoint** to the right of the **Available Endpoints** relationship.
5. Enter `*Prod` in the **Name** field, select **https://localhost:9443/EligibilityV1_0/services/EligibilityServiceV1_0_ProductionPort** from the auto suggest list, and click **Add**. The service endpoint is added as a target of the Available Endpoints relationship.
6. Click **Add Service Port** to the right of the **Bound Web Service Port** relationship.
7. Enter `*Prod` in the **Name** field, select **EligibilityServiceV1_0_ProductionPort** from the auto suggest list, and click **Add**. The service port is added as a target of the Bound Web Service Port relationship.
8. Click **Finish** to save your changes.

The SLD relationships are updated to include the Production endpoint information, as shown in Figure 13-68.

Figure 13-75 Service level definitions with production endpoints

13.9.3 Classifying the endpoint

The Operations team approve the production endpoint. To classify the endpoint, log on to WSRR, and select the **Operations** perspective. Then, follow these steps:

1. Click **Tasks** → **Endpoints** → **Endpoints For Activation**.
2. Click the production endpoint **https://localhost:9443/EligibilityV1_0/services/EligibilityServiceV1_0_ProductionPort**.
3. Click **Edit Classifications**.
4. Expand **Governance Profile Taxonomy** → **Environment**.
5. Select **Production**, and click **Add**. The Production classification is added to the Classification list.
6. Click **OK** to assign the classification.

13.9.4 Classifying the WSDL

To classify the WSDL:

1. Click the **EligibilityServiceV1_0_ProductionPort.wsdl** link under the sm63_sourceDoucments relationship.
2. Click **Edit Classifications**.
3. Expand **Governance Profile Taxonomy** → **Environment**.
4. Select **Production**, and click **Add**. The pre-production classification is added to the Classification list.
5. Click **OK** to assign the classification.

Archived

13.9.5 Approving production deployment

At this point, the service is ready to be released officially. Follow these steps:

1. Click **Tasks** → **Capability Version Tasks** → **Prepare for Production**, as shown in Figure 13-76).

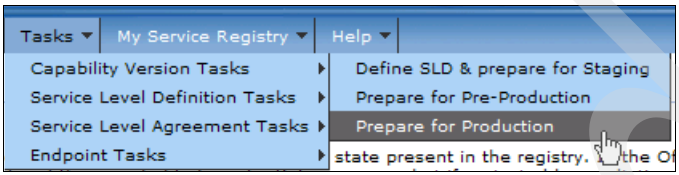


Figure 13-76 Navigate to versions for production

2. Click **Eligibility service (1.0)** to display the service version details.
3. Click the **Approve Production Deployment** button and note that the new governance state is Operational, as shown in Figure 13-77.

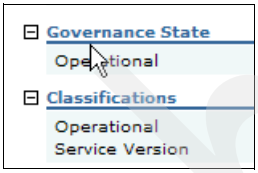


Figure 13-77 Operational service version governance state

13.9.6 Activating the endpoint

Although the service is promoted to the production environment and approved for use, the endpoint needs to be activated, which update its status to Online. To activate the endpoint:

1. Click **Tasks** → **Endpoints** → **Endpoints For Activation**.
2. Click the offline endpoint **https://localhost:9443/EligibilityV1_0/services/EligibilityServiceV1_0_ProductionPort**.
3. Click **Approve For Use**. Note that the new governance state is Online.

The eligibility service is now promoted to the production environment.

Archived

Figure 13-71 shows the status of the WSRR entities after the production environment endpoint is available.

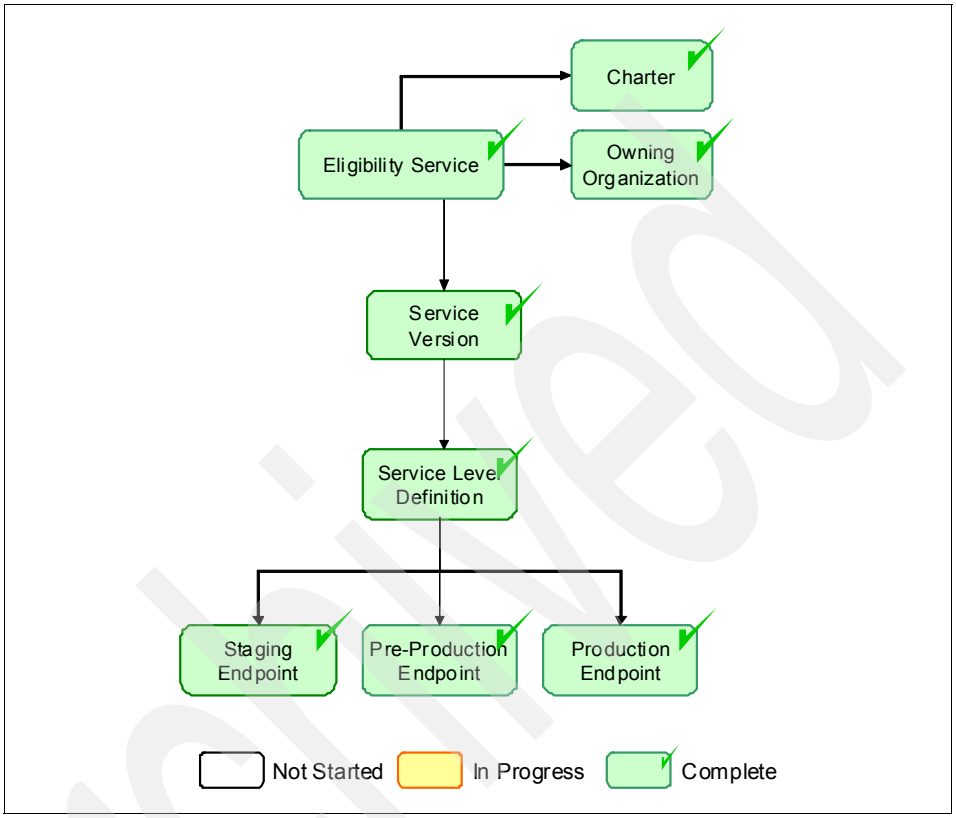


Figure 13-78 Status after deployment to the production environment

Archived

Archived

14

Governing a service that reuses an existing service

This chapter provides step-by step-instructions about how to register a service in WSRR that reuses an existing service.

Note: In the examples in this book, we use a case study about a fictional company named *JKHL Enterprises* (JKHLE). For information about this case study, see Chapter 4, “JKHL Enterprises case study” on page 153. For information about the roles used throughout this scenario refer to 3.3, “Roles in the governance enablement profile” on page 103.

This chapter includes the following topics:

- Governing a new service at JKHLE
- Creating a business capability
- Reviewing and approving business service
- Scoping a service version
- Creating and approving a subscription request
- Planning a service version
- Creating a service level definition
- Creating a service level agreement
- Deploying a service version to staging
- Deploying a service version to pre-production
- Deploying a service version to production

© Copyright IBM Corp. 2009. All rights reserved.

14.1 Governing a new service at JKHLE

In this scenario, we discuss the process for creating and governing a new service that makes use of another service already defined with the Service Registry. Figure 14-1 illustrates the main steps required to complete this process.

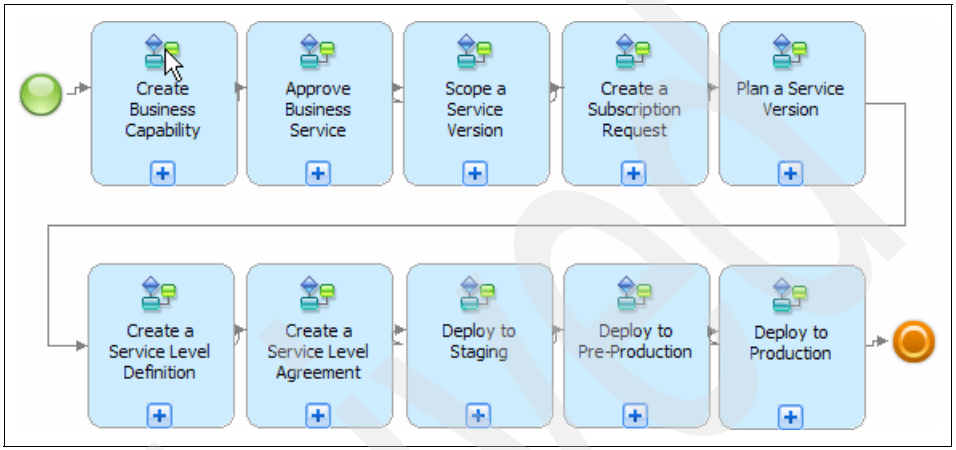


Figure 14-1 Governing a new service process

Archived

Figure 14-2 highlights the major entities that are created in WSRR throughout this process. The eligibility service is registered already in WSRR and is deployed to the staging, pre-production, and production environments as indicated in this figure.

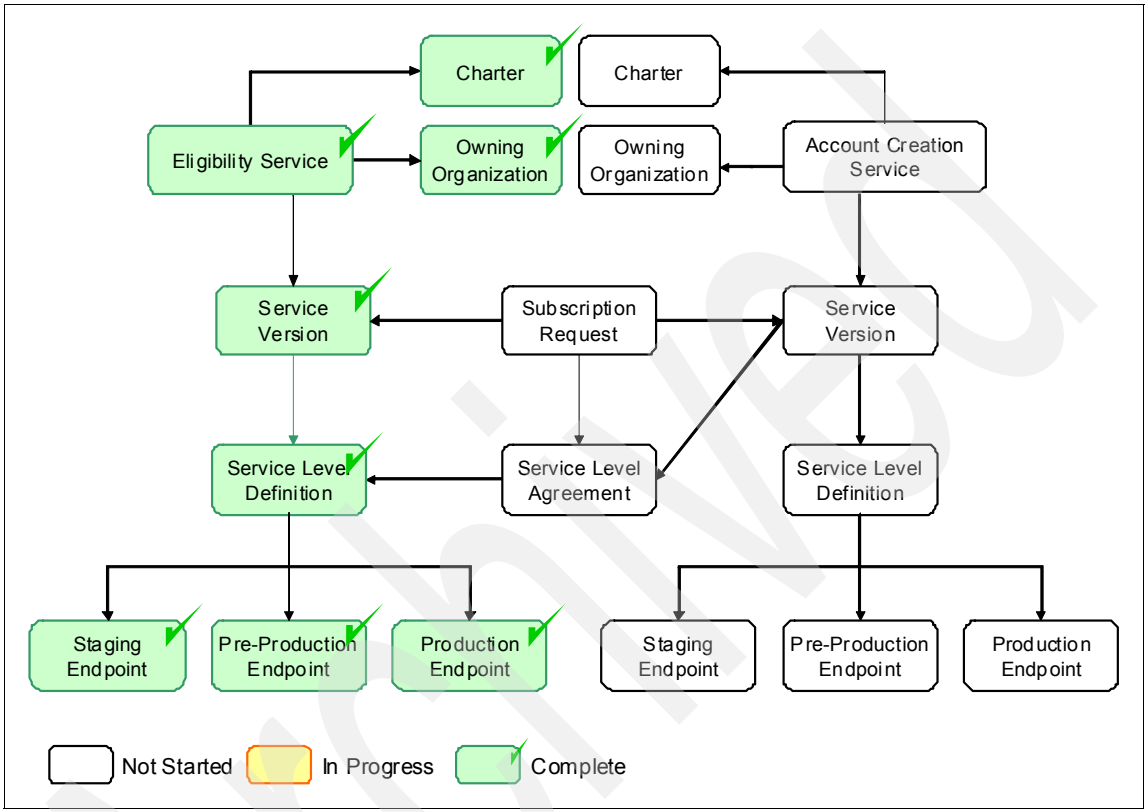


Figure 14-2 WSRR entities for service governance

Archived

14.2 Creating a business capability

The first phase in governing a new service is creating the business capability as illustrated in Figure 14-3. We describe the process to create a business capability in this section.

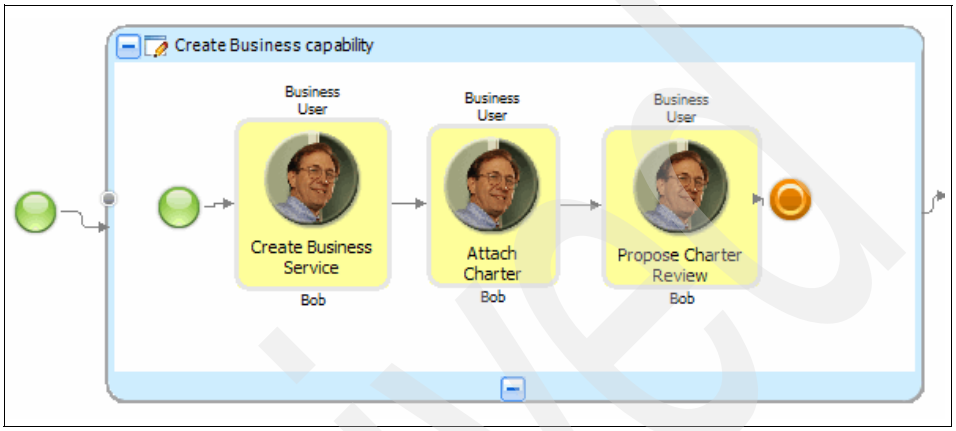


Figure 14-3 Creating the business capability process

14.2.1 Creating a new business service

Bob, the business analyst for the commercial line of business at JKHLE, is responsible for defining a new account opening process. This process allows a customer to create an account so that the customer can purchase products through the company Web site.

Bob first uses the search facilities in the WSRR Web UI to confirm that there is no suitable existing service. Then, Bob creates a new business service object to identify the requirement for such a capability.

Archived

To create a new business service, log on to WSRR, and select the **Business** perspective. Then, follow these steps:

1. Click **Actions** → **Create** → **Business Service**, as shown in Figure 14-4.

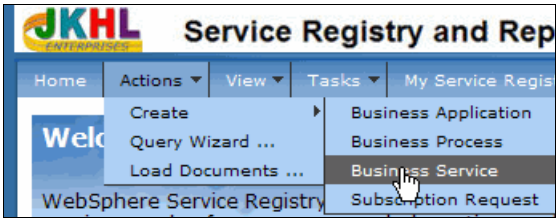


Figure 14-4 Creating a business service

2. Enter the following information, as shown in Figure 14-5:
   - `Name: Account creation business service`
   - Description: `Confirm customer eligibility, perform credit check and create new customer account.`

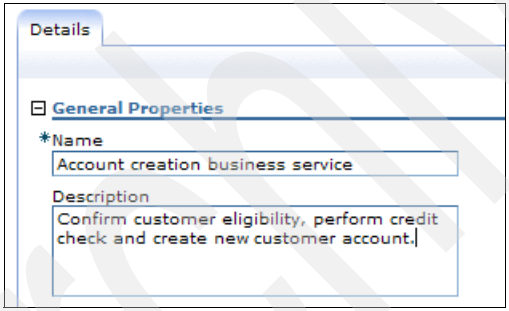


Figure 14-5 Entering the business service properties

3. Click **Finish**. The details page for the new business service displays.

Archived

The governance state of the new business service is business capability identified, as shown in Figure 14-6. This state is the initial state in the capability life cycle; a new business service is entered into this life cycle automatically.

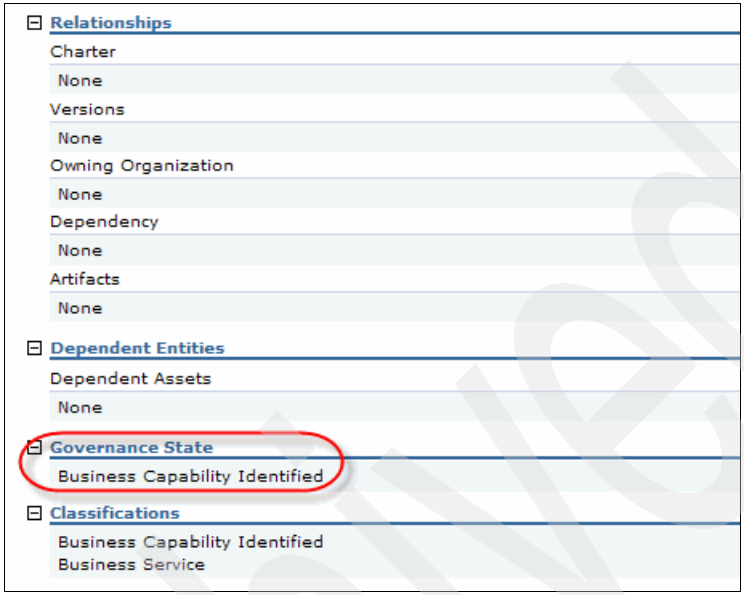


Figure 14-6 New business service

14.2.2 Attaching a charter

The *charter* is a separate document that describes the required capability in detail, including all functional and non-functional requirements. The charter is authored externally and then is loaded into WSRR and attached to the business service.

You need to load the document that contains the charter, and attach it to the business service by using the Charter relationship as follows:

1. Click **View** → **Business Governance** → **Business Capabilities** → **Business Service** as shown in Figure 14-7.

Figure 14-7 Navigating to the business service

2. Click **Account creation business service** to view the business service details.
3. Click **Edit Relationships**.
4. Click **Add Other Document** to the right of the **Charter** relationship, as shown in Figure 14-8.

Figure 14-8 Adding the charter document

Archived

5. Click **Load Document**, as shown in Figure 14-9.

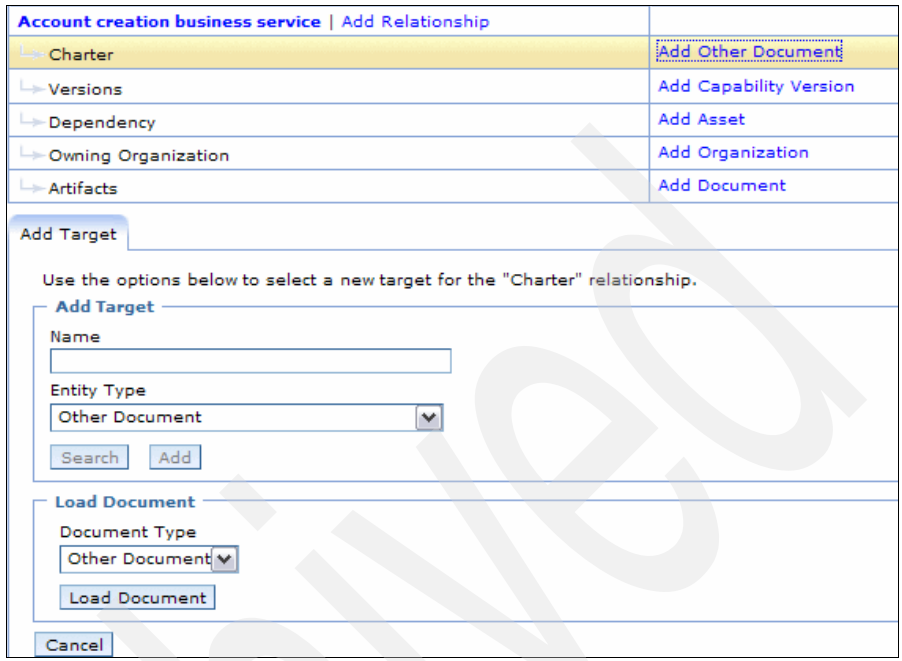


Figure 14-9 Loading the document

6. Click **Browse**, and navigate to the directory where the charter document is located.
7. Select **AccountCreationServiceCharter.doc**, click **Open**, and then click **OK**.
8. Click **Finish** to load the charter document. The charter document is added as a target of the Charter relationship.
9. Click **Finish** to save your changes.

Archived

The charter document is loaded for the Account creation business service as shown in Figure 14-10.

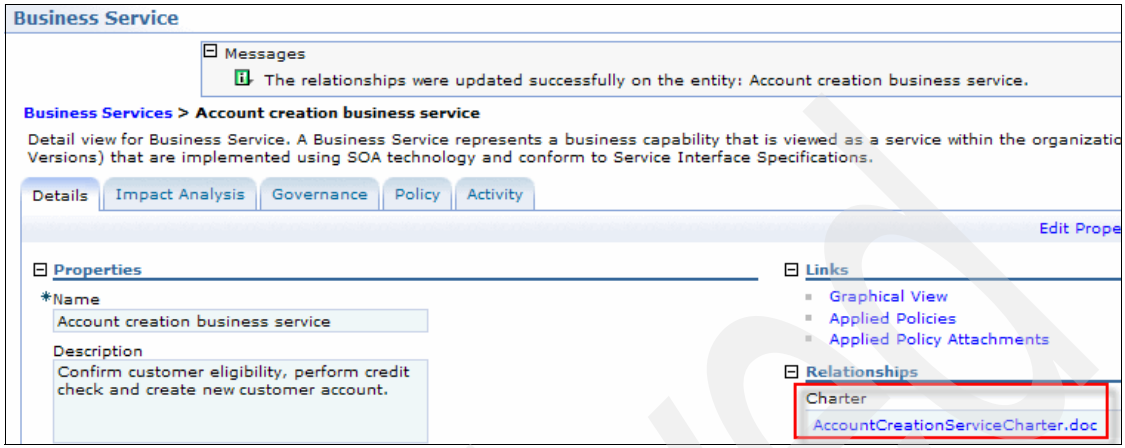


Figure 14-10 Business service charter

14.2.3 Proposing the business service

The business user has created the initial definition of the service, and it is now ready for review by the SOA governance team. To indicate the service's readiness for review and to make it available to the reviewers, the charter must be proposed.

Archived

To transition the business service to the Charter Review state, click **Propose for Charter Review**. Note that the new governance state is Charter Review as shown in Figure 14-11.

Figure 14-11 Charter review governance state

The business service and charter entities shown in Figure 14-12 are now ready for review.

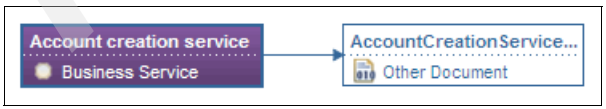


Figure 14-12 Graphical view of business service and charter entities

Figure 14-13 shows the status of the WSRR entities after the business capability is created.

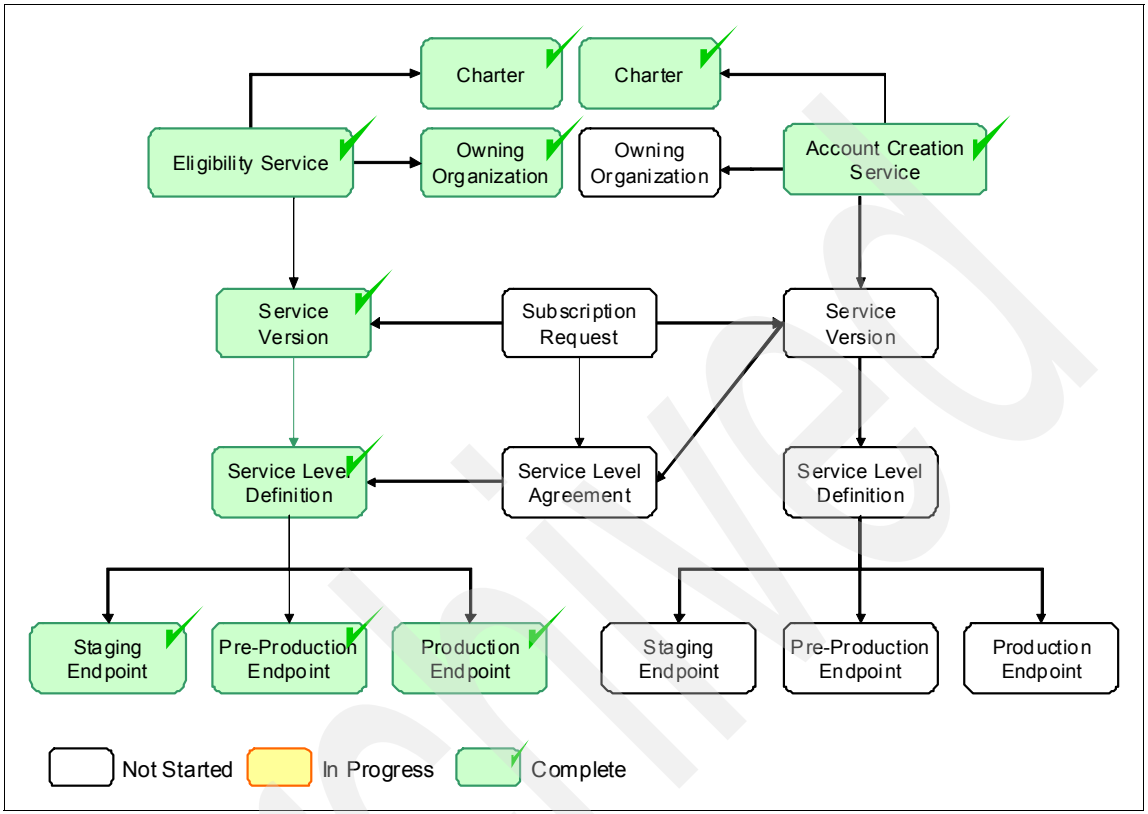


Figure 14-13 Status after business capability created

14.3 Reviewing and approving business service

The SOA governance team ensures that governance processes are enforced. Before the business service can be transitioned to the next stage in the life cycle, the team must ensure that the proposed capability does not duplicate other services in the registry and that an owning organization is assigned that is responsible for all versions of this capability and for managing requirements for this service.

Archived

The next phase in the governing a new service scenario at JKHLE is approving the business service capability as illustrated in Figure 14-14. We describe this process in this section.

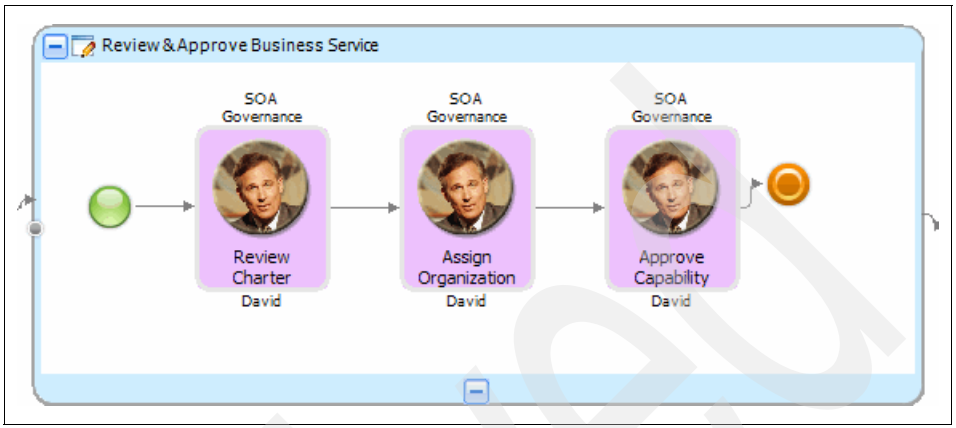


Figure 14-14 Approving the business service process

14.3.1 Reviewing the new business service

David, chairman of the SOA governance team, reviews the business service definition and charter.

To review the business service definition, log on to WSRR, and select the **SOA Governance** perspective. Then, follow these steps:

1. Click **Tasks** → **Business Capability Tasks** → **Business Capabilities for Approval** (Figure 14-15) to display all business capabilities in the Charter Review state.

Figure 14-15 Navigating to business capabilities

2. Click **Account creation business service** to display the business service details, and review the basic information in the business service definition.
3. Click **AccountCreationServiceCharter.doc** under the **Charter** relationship.

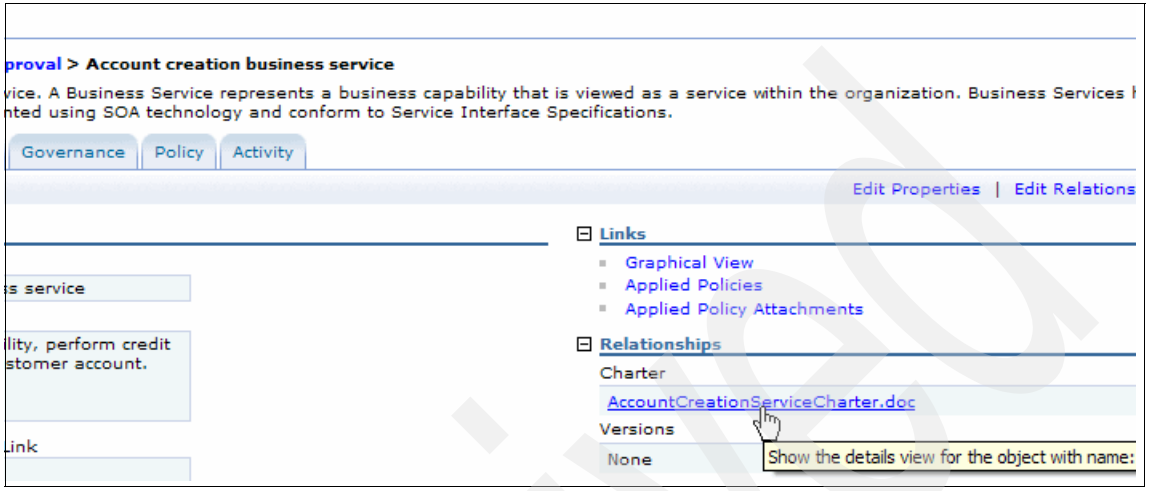


Figure 14-16 Selecting the charter document

4. Click **Content** (as shown in Figure 14-17), and then click **Download Document**.

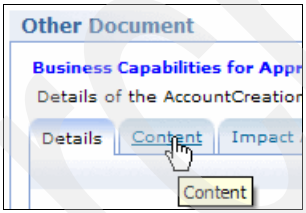


Figure 14-17 Viewing the content

5. Click **OK** to open the charter document and review the charter.
6. Close the charter document and return to the WSRR Web UI.

Archived

14.3.2 Assigning an owning organization to the business service

Now, you need to assign the organization that is responsible for this service as follows:

1. Click **Account creation business service** in the breadcrumb trail to display the business service details.
2. Click **Edit Relationships**.
3. Click **Add Organization** to the right of the **Owning Organization** relationship as shown in Figure 14-18.

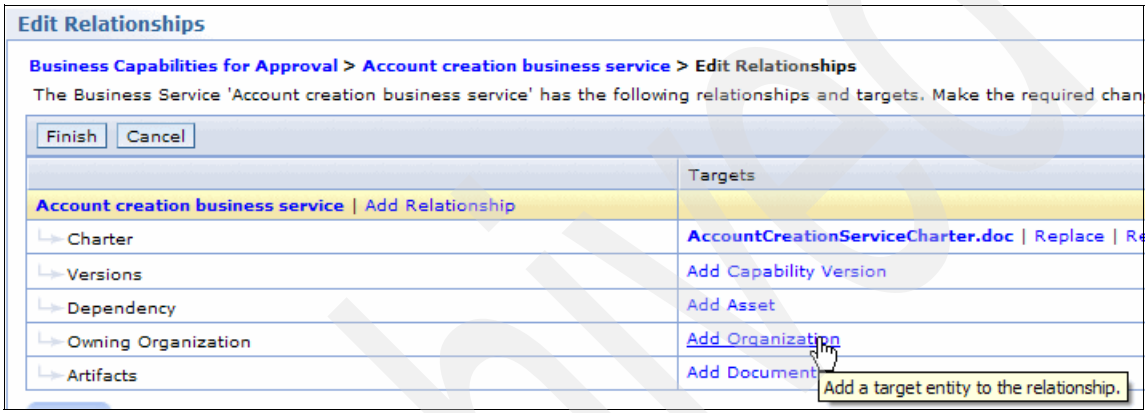


Figure 14-18 Adding an owning organization

4. Enter `C` in the **Name** field, select **Commercial** from the auto suggest list, and click **Add**. The Commercial organization is added as a target of the Owning organization relationship.
5. Click **Finish** to save your changes.

14.3.3 Approving the business service

Having ensured that all of the governance requirements are satisfied and having used the search facilities in the WSRR Web UI to confirm that the proposed capability does not duplicate other services in the registry, the SOA governance team member approves the business service.

Archived

To transition the business service to the Business Capability Approved state, Click **Approve Capability** as shown in Figure 14-19.

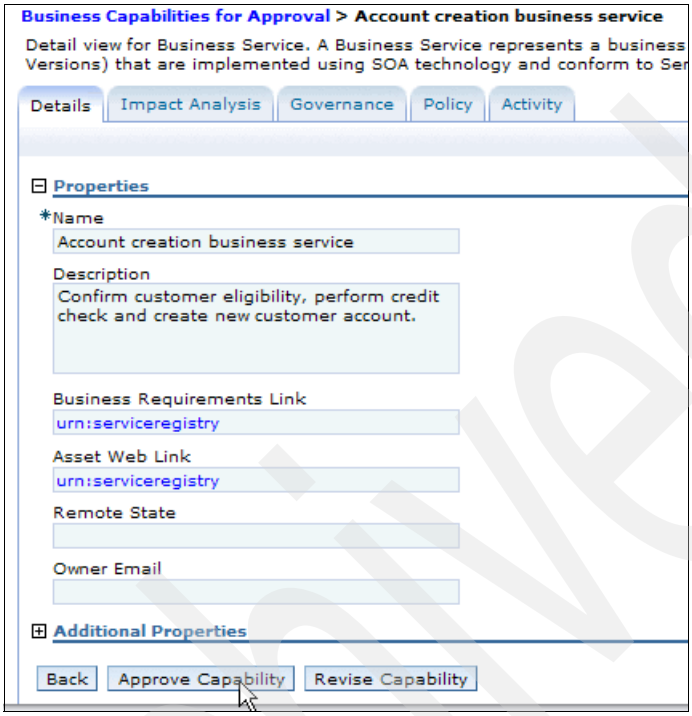


Figure 14-19 Approving the business service capability

The new governance state is Business Capability Approved, as shown in Figure 14-20.

Figure 14-20 Business capability approved business service governance state

Archived

Figure 14-13 shows the status of the WSRR entities after the business service is approved.

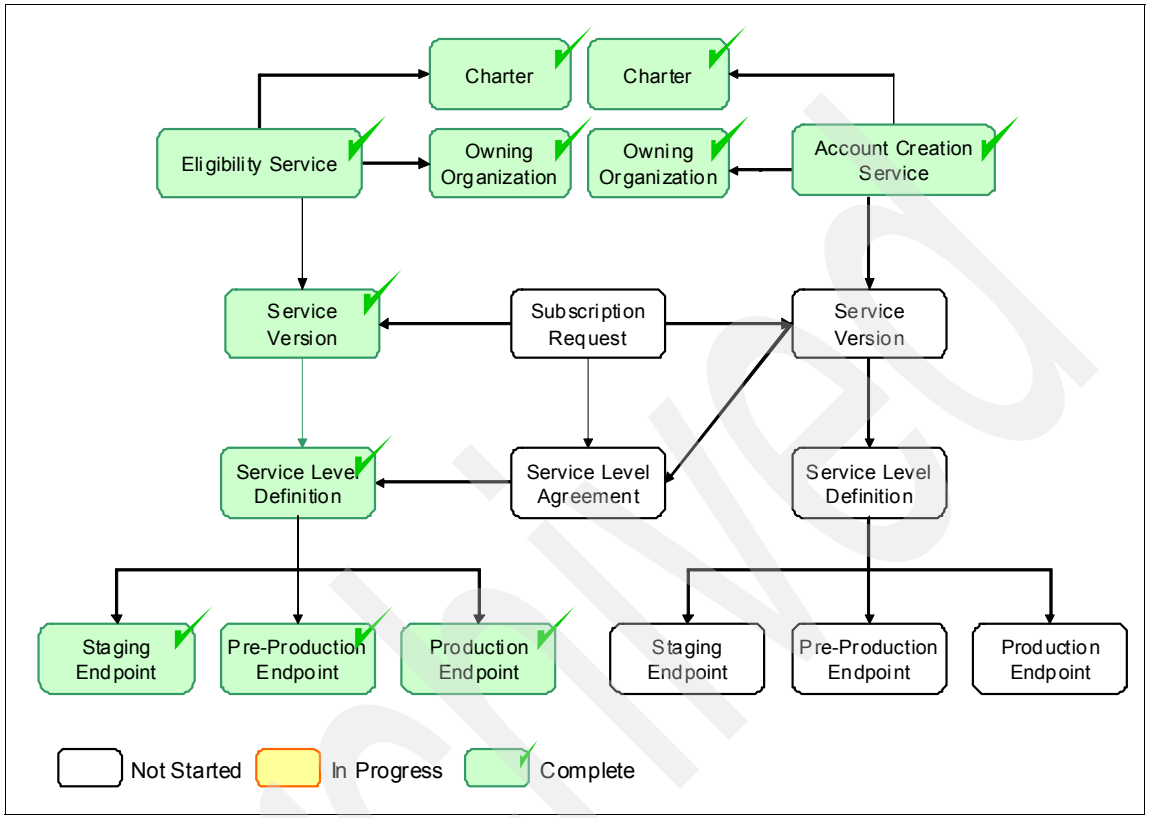


Figure 14-21 Status after business service approved

For information about the capability life cycle, see 3.5, “Capability life cycle” on page 108.

14.4 Scoping a service version

A *service* (or capability) version represents a specific version or release of a service, and includes a range of functional and non-functional specifications for that version of the service. Over time multiple versions of the same business capability can be created, as the service functionality is enhanced and extended.

The next phase in the governing a new service scenario at JKHLE is scoping the service version as illustrated in Figure 14-22. We describe this process in this section.

Figure 14-22 Scoping the service version process

14.4.1 Creating a new service version

After the business capability is defined, reviewed, and approved, a new service version is defined.

It is now the responsibility primarily of the development organization to create an implementation of the service. Debra, the Development Manager for the commercial line of business, is responsible for developing the new account creation service for JKHLE.

To create the new service version, log on to WSRR, and select the **Development** perspective. Then, follow these steps:

1. Click **View** → **Business Governance** → **Business Capabilities** → **Business Services**.
2. Click **Account creation business service** to display the business service details.

Archived

3. Assign a new service version to the business service by clicking **New Capability Version** as shown in Figure 14-23.

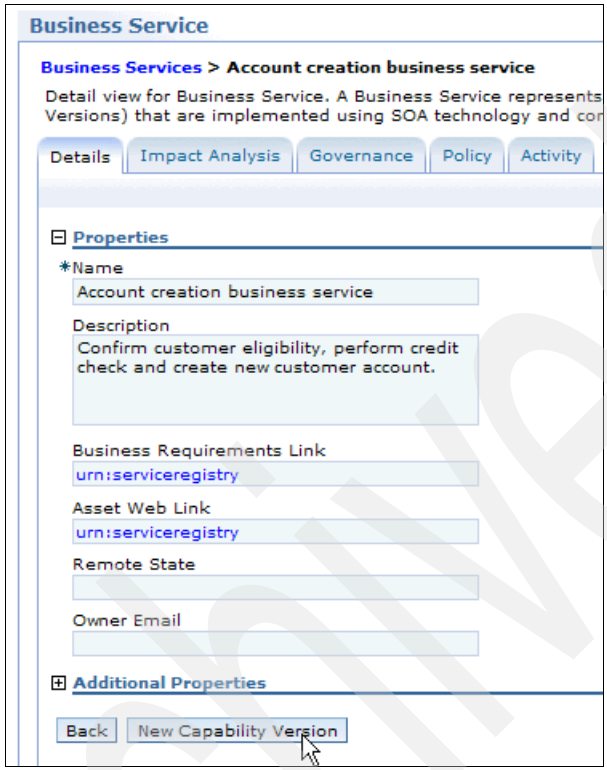


Figure 14-23 Creating a new capability version

Archived

4. Under the relationship called Versions, click **Account creation business service**, as shown in Figure 14-24, to display the details of the new service version.

> **Note:** A Chartered Business Capability relationship is provided, which you can use to navigate back to the business service.

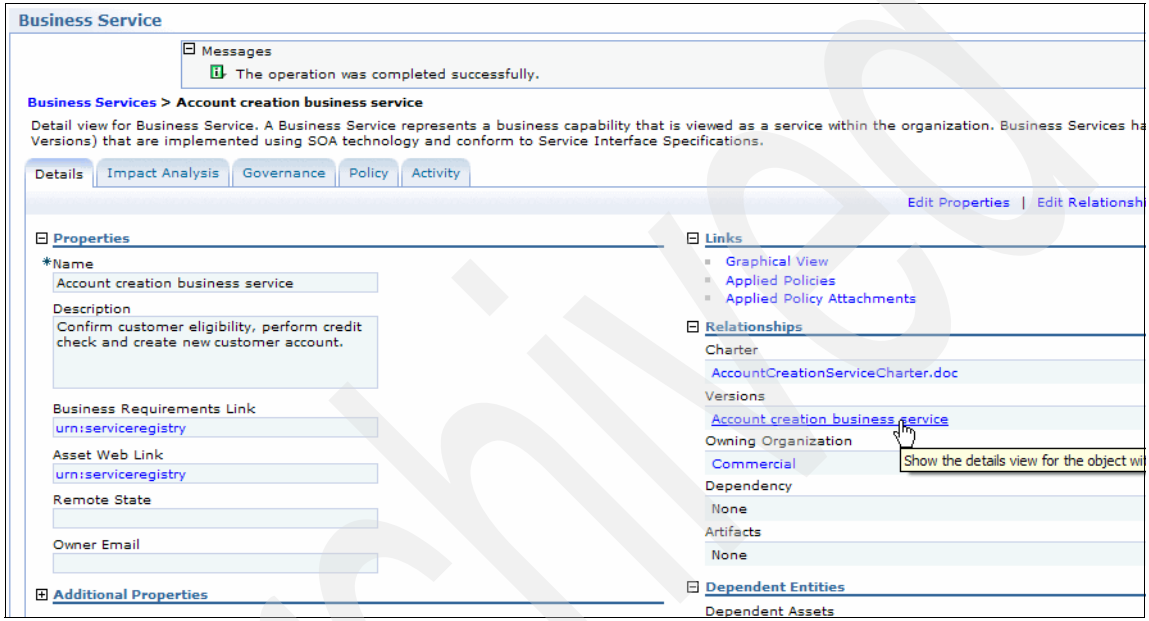


Figure 14-24 Updating the new service version

The owning organization for the new service version is set automatically to be Commercial, the same as the business service. You can change this owning organization if necessary.

Archived

5. Click **Edit Properties**, as shown in Figure 14-25.

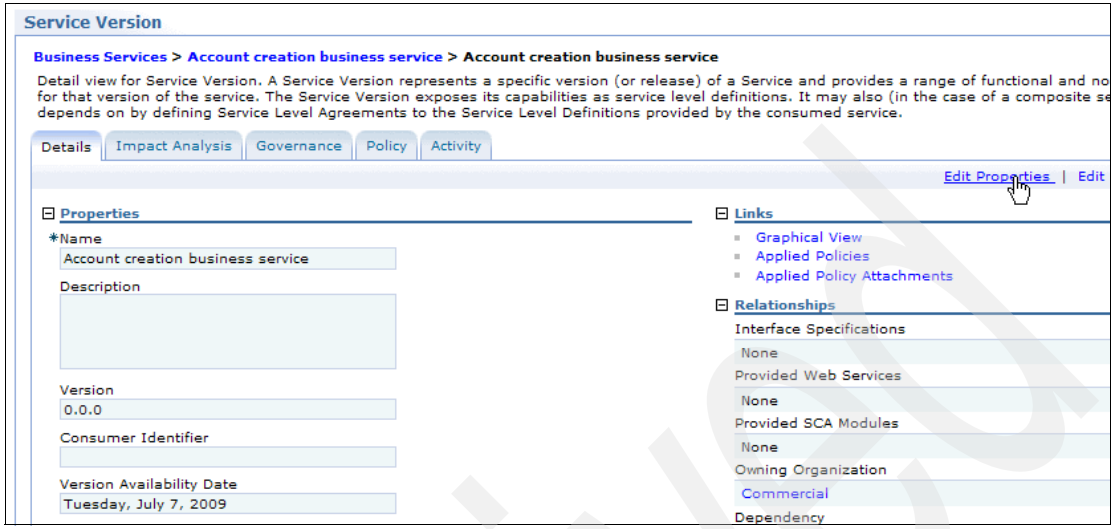


Figure 14-25 Editing service version properties

6. Change the following property values:
   - Name: `Account creation service (1.0)`
   - Version: `1.0`
   - Description: `This service version provides the capabilities for the account creation service`
   - Consumer Identifier: `ACSV000`

     The Consumer Identifier is an identifier that the service passes in the header of all service invocations it attempts. The invoked services use this identifier to determine whether the consumer has the appropriate service level agreements in place to make the invocation.
   - Version Requirements Link: `http://requirements.jkhle.com/requirements.jsp?id=8820`

     This is a link, fictitious in this case, to the relevant item in JKHLE's requirements tracking tool.
7. Click **OK** to save the changes.

Archived

The governance state is Identified, as shown in Figure 14-26. This state is the initial state in the model phase of the SOA life cycle. A new service version is entered into the SOA life cycle automatically. Figure 5-40 on page 208 shows the full SOA life cycle used by JKHLE.

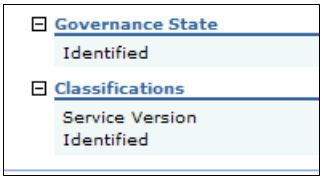


Figure 14-26 Service version governance identified state

14.4.2 Proposing the service version scope

Now that the scope of the service version is defined, it must be circulated for review so that all potential consumers of the service can verify that the requirements are within the proposed scope by the development team.

You transition the service version to the Scope Review state by clicking **Propose Scope**. Note that the new governance state is Scope Review as shown in Figure 14-27.

Figure 14-27 Scope review service version governance state

Archived

14.4.3 Approving the service version scope

In the Scope Review state, David from the SOA governance team reviews the service version requirements.

He carries out the following verification activities:

- Checks that this service version is warranted across the organization
- Checks that the requirements and stakeholders have been agreed
- Checks that the owning organization that is responsible for delivering the requirements is identified and is assigned to the service version

When the service version scope review is complete, the scope can be approved.

To transition the service version to the Scoped state, log on to WSRR, and select the **SOA Governance** perspective. Then, follow these steps:

1. Click **Tasks** → **Capability Version Tasks** → **Versions for Scope Review** as shown in Figure 14-28.

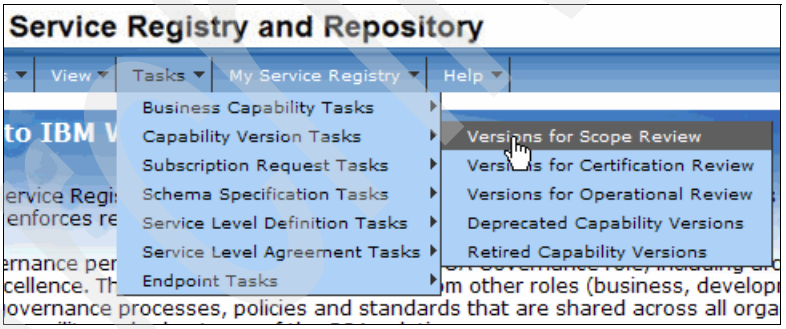


Figure 14-28 Navigating to versions for scope review

2. Click **Account creation service (1.0)** to display the service version details.

   As part of the review activity, a member of the SOA governance team also opens the business service using the **Account creation business service** link under the Chartered Business Capability(s) relationship. Then, that member reviews the charter document to ensure that the requirements for this service version are clear and then returns to the service version using the **Account Creation Service (1.0)** link under the Versions relationship.

Archived

3. Click **Approve Scope**. Note that the new governance state is Scoped as shown in Figure 14-29.

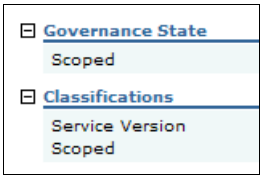


Figure 14-29 Service version governance scoped state

Figure 14-30 shows the business service, charter, and service version.

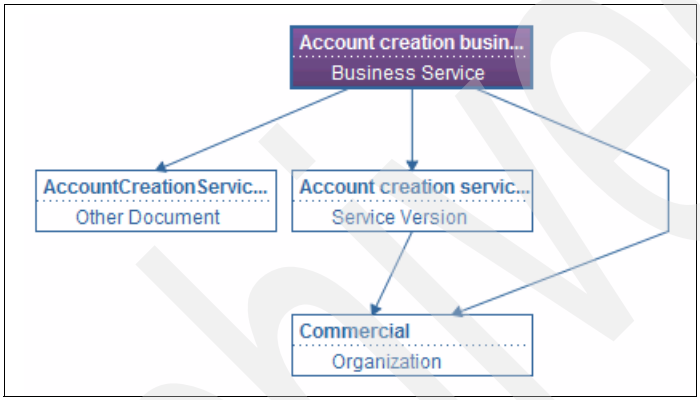


Figure 14-30 Business and service version entities

Archived

Figure 14-13 shows the status of the WSRR entities after the new service version is scoped.

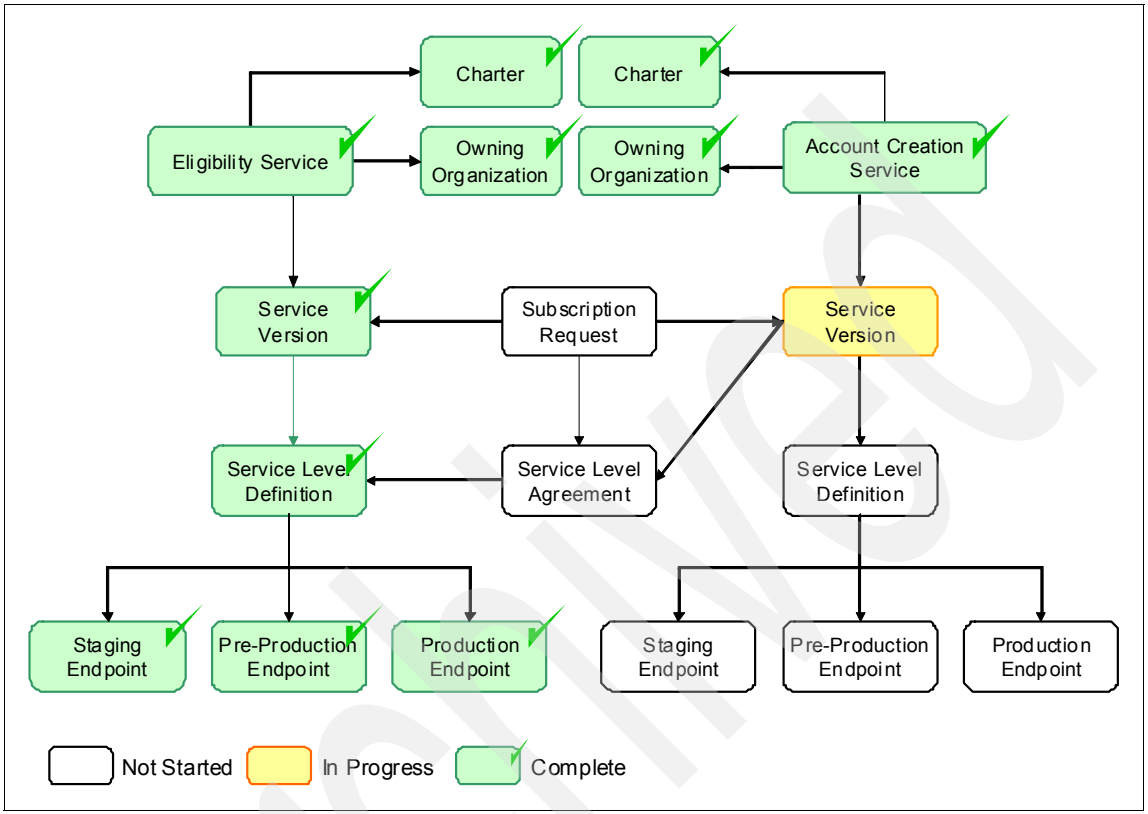


Figure 14-31 Status after service version is scoped

14.5 Creating and approving a subscription request

The new account creation service includes requirements to verify the eligibility of a customer. As the Development Manager for the commercial line of business, Debra uses the JKHLE service registry and repository to identify that an existing service will provide the required functionality. A subscription request is created to formalize the reuse of the eligibility service.

The next phase in the governing a new service scenario at JKHLE is creating the subscription request as illustrated in Figure 14-32. We describe this process in this section.

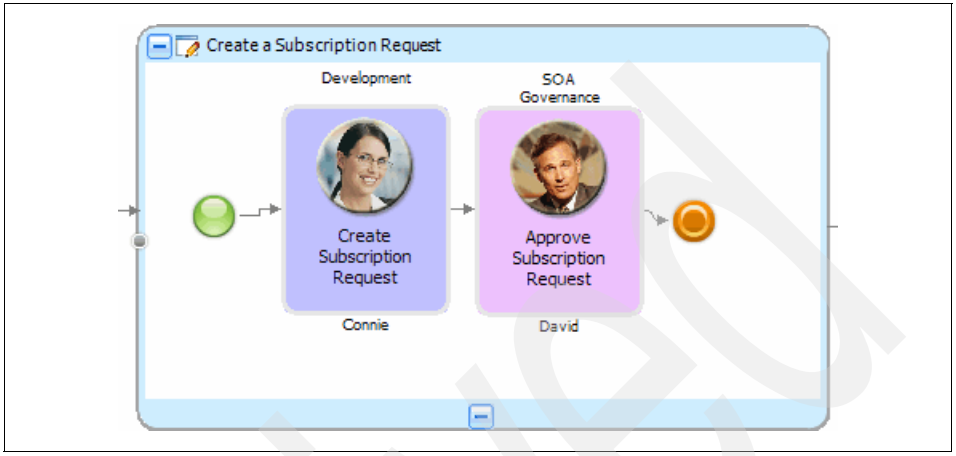


Figure 14-32 Creating the subscription request process

14.5.1 Creating a subscription request

Debra, the Development Manager for the commercial line of business, is responsible for creating the subscription request to use the eligibility service.

To create the new subscription request, log on to WSRR, and select the **Development** perspective. Then, follow these steps:

1. Click **Actions** → **Create** → **Subscription Request** as shown in Figure 14-33.

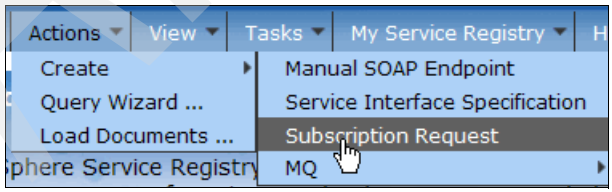


Figure 14-33 Navigate to create subscription request

Archived

2. Enter the following information:
   - Name: `Account creation consumption of eligibility service`
   - Requirements Link: `http://requirements.jkhle.com/requirements.jsp?id=7785`

     This is a link, fictitious in this case, to the relevant Subscription Request item in JKHLE's requirements tracking tool. Note that if the requirements are specified in a document rather than in a requirements tracking tool, you add the document as a target of the Artifacts relationship.
3. Click **Add Capability Version** to the right of the **Consumer** relationship as shown in Figure 14-34.

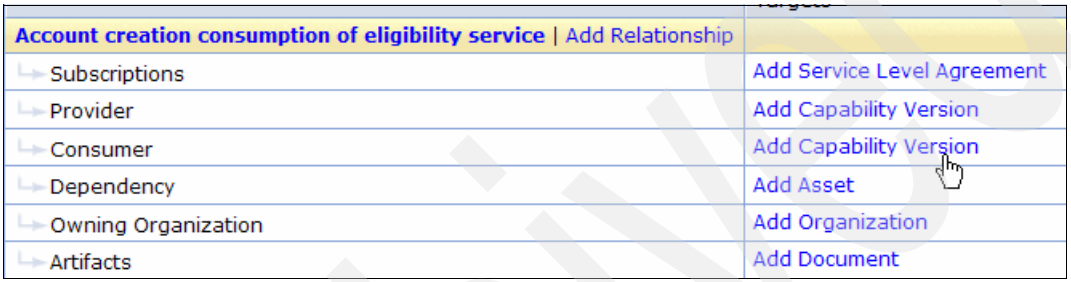


Figure 14-34 Adding the consumer relationship

4. Enter `A` in the **Name** field, select **Account creation service (1.0)** from the auto suggest list, and click **Add**. The service version is added as a target of the Consumer relationship.
5. Click **Add Capability Version** to the right of the **Provider** relationship.
6. Enter `E` in the **Name** field, select **Eligibility service (1.0)** from the auto suggest list, and click **Add**. The service version is added as a target of the Provider relationship.
7. Click **Finish** to save your changes.
8. Click **Propose**. Note that the new governance state is Asset Scope Review as shown in Figure 14-35.

Figure 14-35 Asset scope review governance state

14.5.2 Approving the subscription request

The SOA governance team reviews and approves the subscription request. This team ensures that the following commitments are made:

- The service provider commits to providing the necessary resources to meet the service level that the account creation service requires according to the project schedule.
- The consumer agrees that the capabilities detailed in the eligibility service version specifications will meet the requirements of the account creation service. Additionally, this confirmation of acceptance of the subscription request indicates that they have all of the detailed interface, binding, and policy information that is required in order for them to continue the development of the account creation service.

After the subscription request is approved by all of the consumer and provider stakeholders, David, the chairman of the SOA governance team, approves the subscription request. This seal of approval from the governance team means that the relationship between the consumer and provider can be entered into.

To transition the subscription request to the Asset Approved state, log on to WSRR, and select the **SOA Governance** perspective. Then, follow these steps:

1. Click **Tasks** → **Subscription Request Tasks** → **Subscription Requests for Scope Review** as shown in Figure 14-36.

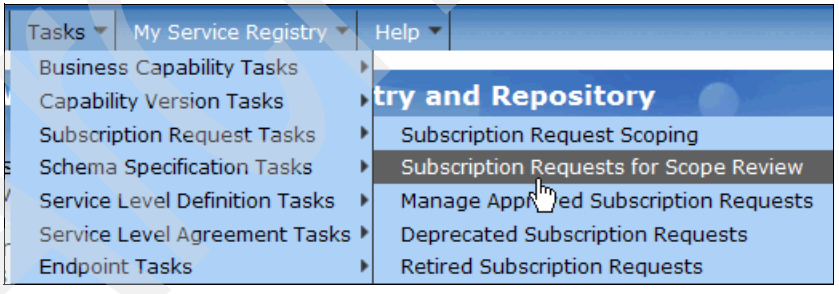


Figure 14-36 Navigating to subscription requests for scope review

2. Click **Account creation consumption of eligibility service** to display the subscription request details.

3. Click **Approve**. Note that the new governance state is Asset Approved as shown in Figure 14-37.

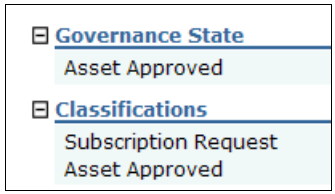


Figure 14-37 Subscription request governance asset approved state

Figure 14-38 shows the status of the WSRR entities after the new subscription request is approved.

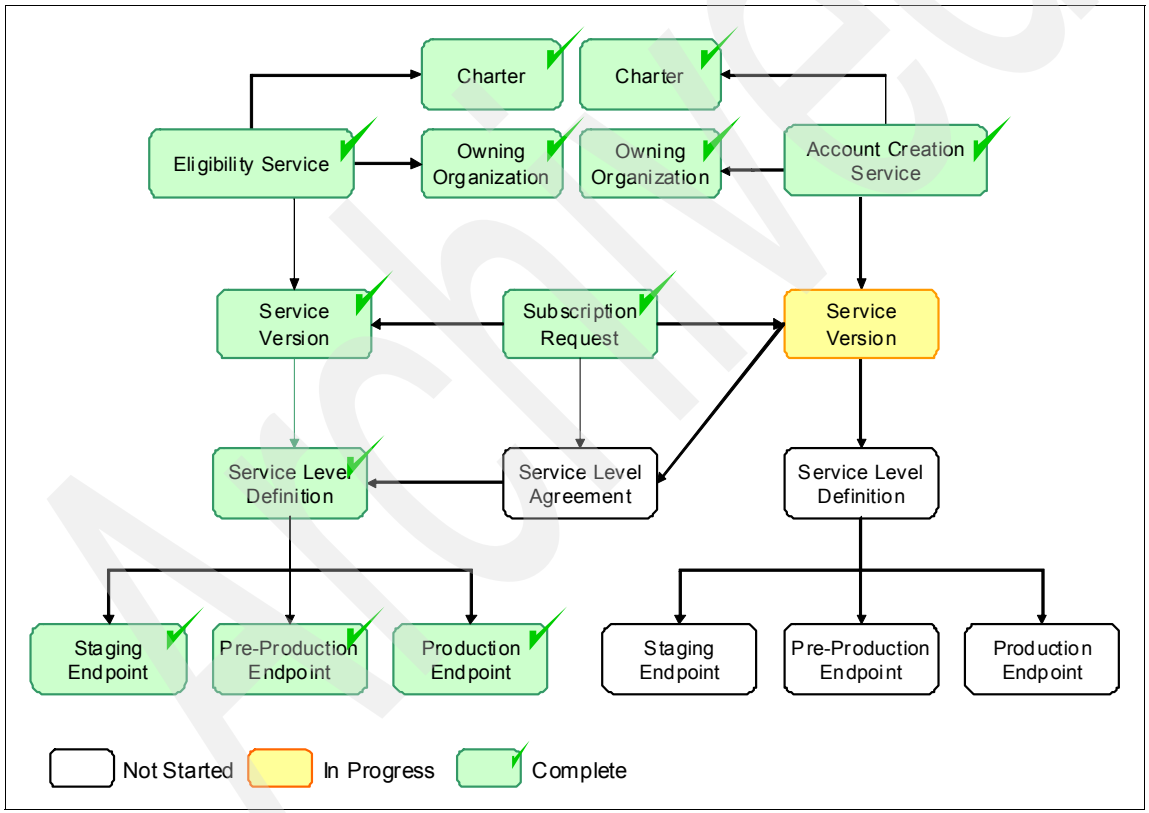


Figure 14-38 Status after subscription request is approved

Archived

14.6 Planning a service version

The scoped version is now in the planning phase where the development and owning organizations must work together to define the funding and timeframes for the project. In this phase, architecture sizing and funding decisions are made, including establishing dependencies on other assets or services. Figure 14-39 illustrates this planning process. We describe this process in this section.

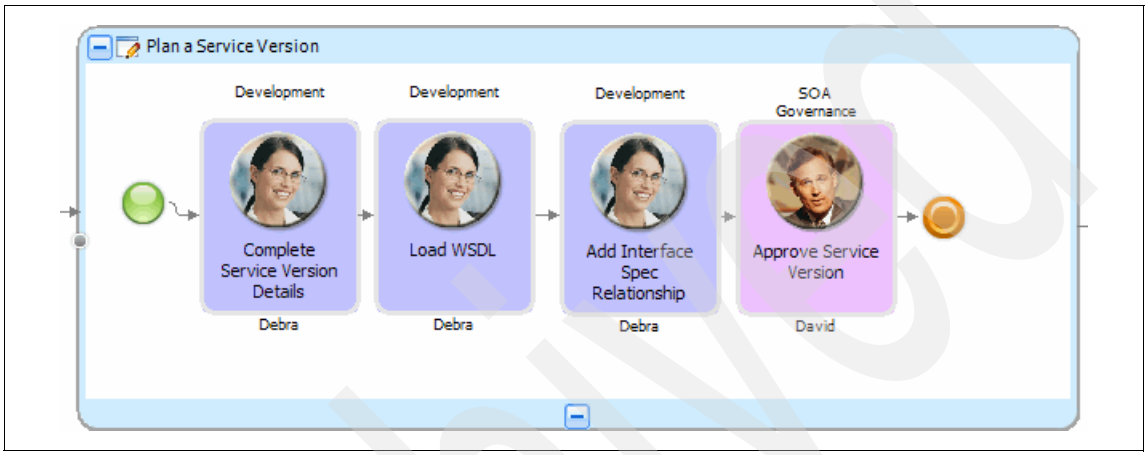


Figure 14-39 Planning a service version process

14.6.1 Completing service version details

The service version must be updated to include proposed availability and termination dates. The plan is then proposed, which makes the plan available for review.

Updating the service version details is the responsibility of the development organization. Debra, the Development Manager for the commercial line of business, is responsible for this phase of the scenario.

Archived

To complete the service version details, log on to WSRR, and select the **Development** perspective. Then, follow these steps:

1. Click **Tasks** → **Capability Version Tasks** → **Version Planning** as shown in Figure 14-40.

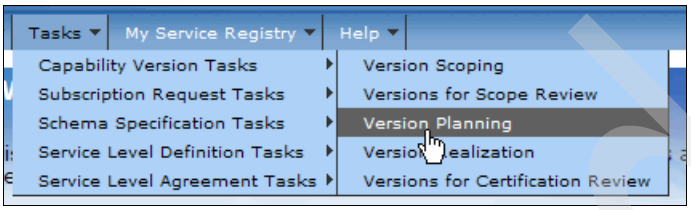


Figure 14-40 Navigating to version planning tasks

2. Click **Account creation service (1.0)** to display the service version details as shown in Figure 14-41.

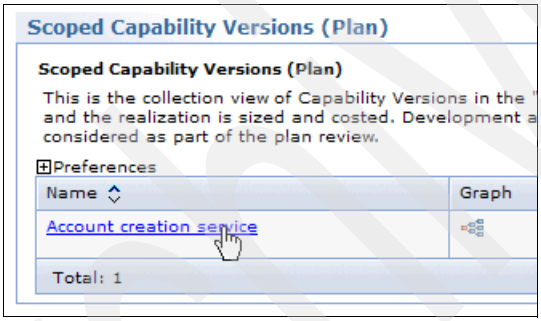


Figure 14-41 Selecting the service version

3. Then, click **Edit Properties**.
4. Enter the values of your choice in the Version Availability Date and Version Termination Date fields.
5. Click **OK** to save your changes.

Archived

14.6.2 Loading the WSDL

You can now load the abstract WSDL that defines the service interface as follows:

1. Click **Edit Relationships**.
2. Click **Add Document** to the right of the **Artifacts** relationship as shown in Figure 14-42. You might have to scroll down in the relationships pane.

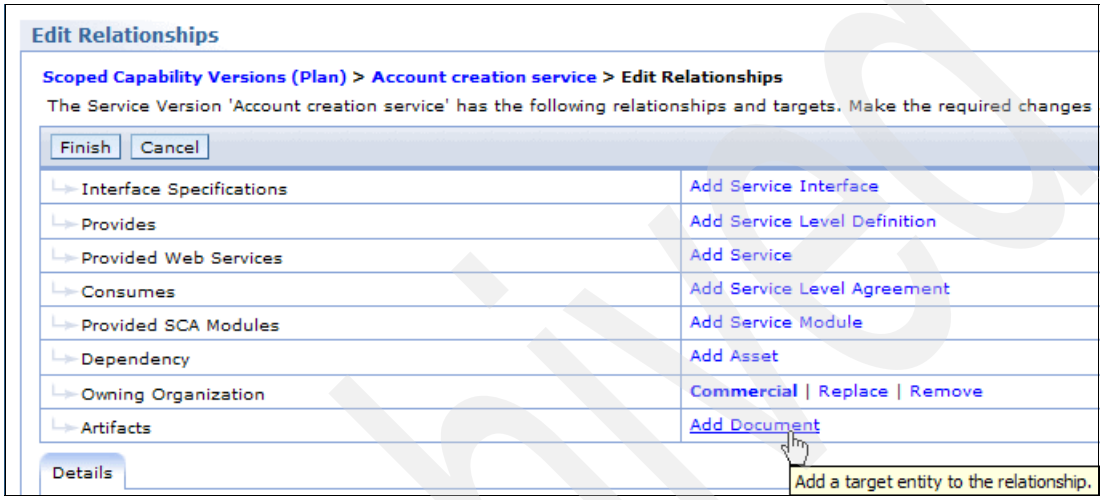


Figure 14-42 Adding an artifact document

3. Ensure that the Document Type in the Load Document pane is set to **WSDL Document**. Click **Load Document** (Figure 14-43).

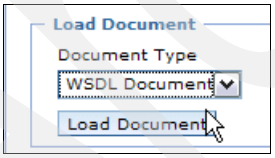


Figure 14-43 Loading a WSDL document

4. Click **Browse**, and navigate to the directory where the WSDL document is located.
5. Select **AccountCreationInterfaceV1_0.wsdl**, and click **Open**. Enter a document version of `1.0`, and then click **OK**.

Archived

6. Click **Finish** to load the WSDL document. It is added as a target of the Artifacts relationship. Note that the dependent XSDDocument loaded previously is detected and identified as in repository as shown in Figure 14-44.

Figure 14-44 Dependent XSD

7. Click **Finish** to save your changes.

> **Note:** You must save the Service Version at this stage to persist the `AccountCreationInterfaceV1_0.wsdl` in the Service Registry. The next steps use some of the objects that are created when the WSDL is saved into WSRR.

14.6.3 Adding the interface specification relationship

Now, you need to create a relationship between the service version and the interface specification as follows:

1. Click **Edit Relationships**.
2. Click **Add Service Interface** to the right of the **Interface Specifications** relationship as shown in Figure 14-45.

Figure 14-45 Adding the service interface relationship

3. Enter `A` in the **Name** field, select **AccountCreationV1_0** from the auto suggest list as shown in Figure 14-46, and click **Add**. The Account Creation Service Interface is added as a target of the Interface Specification.

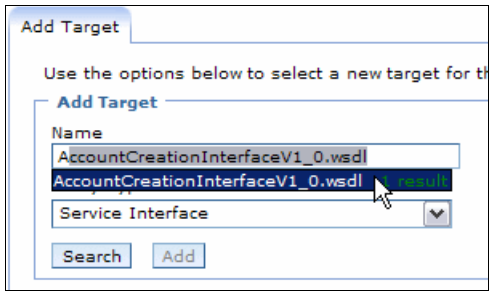


Figure 14-46 Selecting the WSDL

4. Click **Finish** to save your changes.

14.6.4 Approving the service version

After all parties have agreed the details in the plan, David from the SOA governance team can approve the service version plan.

To transition the service version to the Specified state, log on to WSRR and select the **SOA Governance** perspective. Then, follow these steps:

1. Click **Tasks** → **Capability Version Tasks** → **Version Planning** as shown in Figure 14-28.

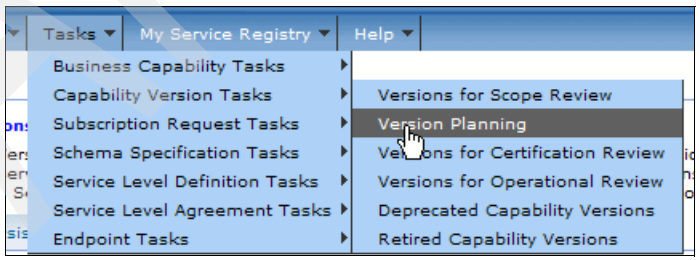


Figure 14-47 Navigating to version planning

2. Click **Account creation service (1.0)** to display the service version details.

3. Click **Approve Specification**. Note that the new governance state is Specified as shown in Figure 14-48.

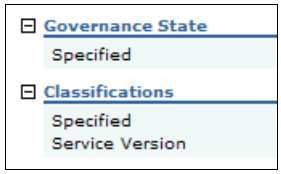


Figure 14-48 Specified service version governance state

Figure 14-49 shows the status of the WSRR entities after the new service version is planned.

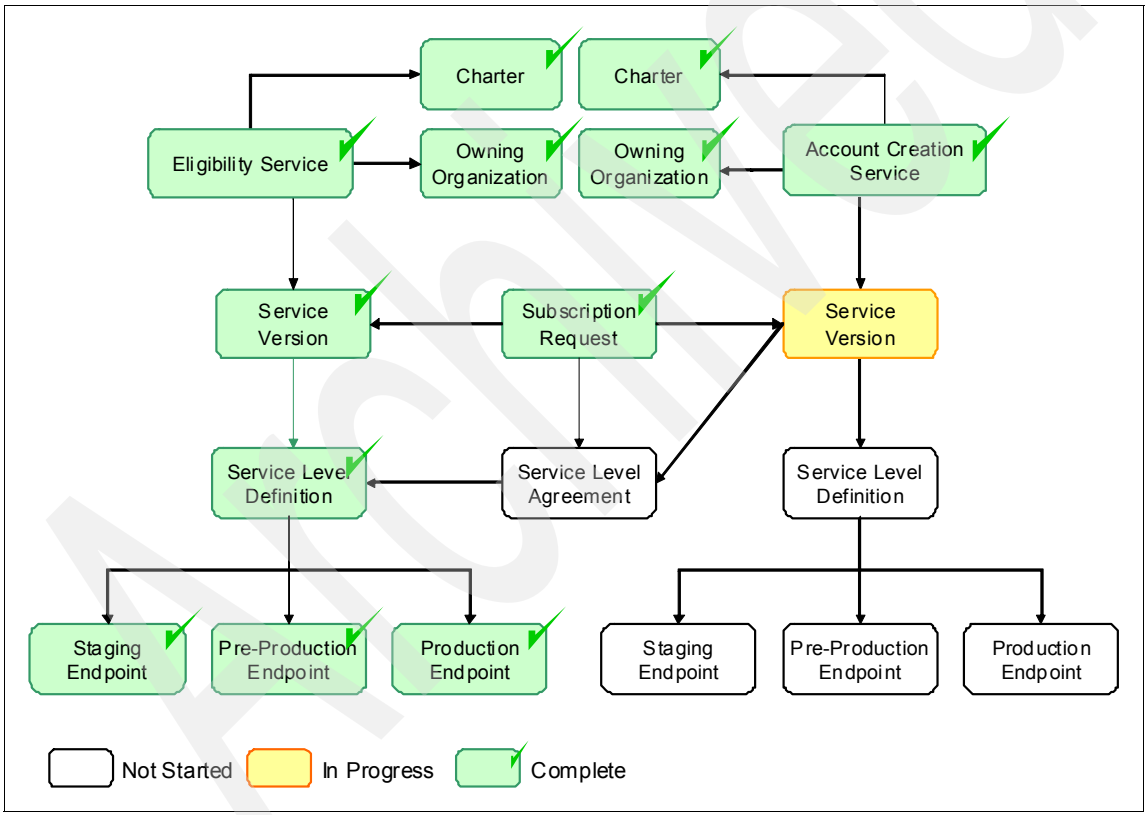


Figure 14-49 Status after service version is planned

Archived

14.7 Creating a service level definition

Now that the interface, schema, and business objects are defined, the development team must define the service level definition to which the service version will adhere. The service level definition specifies non-functional or quality of service characteristics to be provided by the service. Figure 14-50 illustrates the process for creating a service level definition. We describe this process in this section.

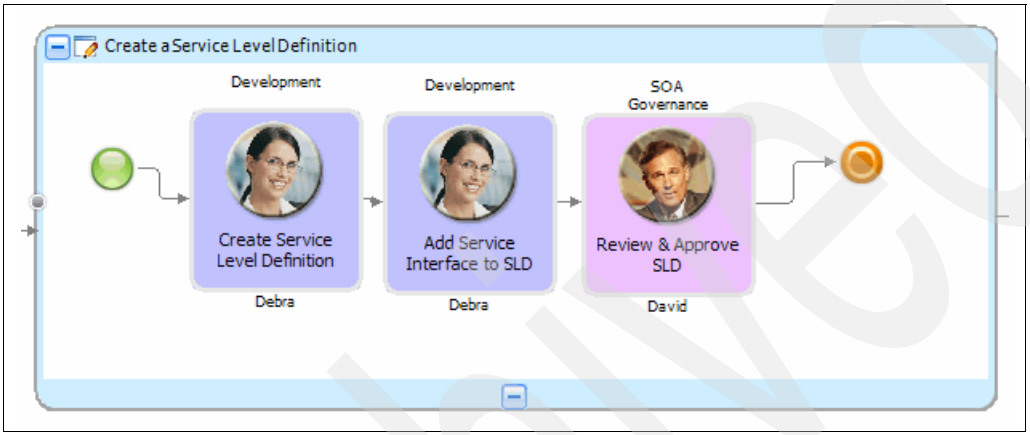


Figure 14-50 Creating a service level definition process

14.7.1 Creating the service level definition

Creating the service level definition (SLD) is the responsibility of the development organization. Debra, the Development Manager for the commercial line of business, is responsible for this phase of the scenario.

Archived

To create the new service level definition, log on to WSRR, and select the **Development** perspective. Then, follow these steps:

1. Click **Tasks** → **Capability Version Tasks** → **Define SLD & prepare for staging** as shown in Figure 14-51.

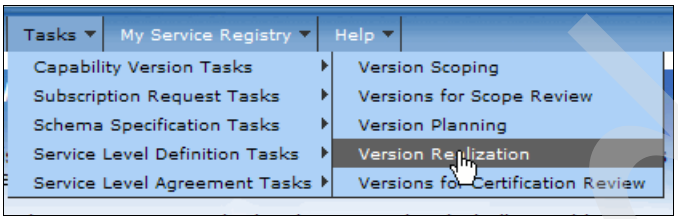


Figure 14-51 Navigating to versions for staging

2. Click **Account creation service (1.0)** to display the SLD details.
3. Click **New SLD**.
4. Click **SLD - Account creation service (1.0)** under the **Provides** relationship to display the SLD details (Figure 14-52).

Figure 14-52 Selecting the SLD

Archived

5. Click **Edit Properties**, shown in Figure 14-53 on page 511, and enter the following information:
   - Average Response Time: `100`
   - Availability: `Working Hours Only`

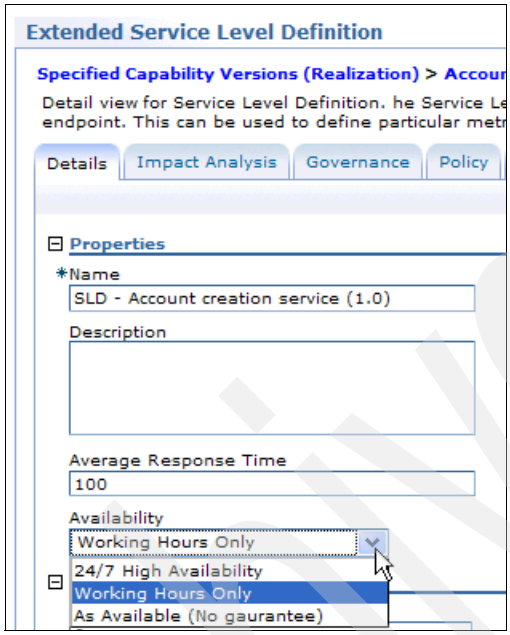


Figure 14-53 Entering the SLD details

6. Click **OK** to save your changes.

Archived

14.7.2 Adding the service interface

Now, you need to create a relationship between the service level definition and the service interface as follows:

1. Click **Edit Relationships**.
2. Click **Add Service Interface** to the right of the **Service Interface** relationship as shown in Figure 14-54.

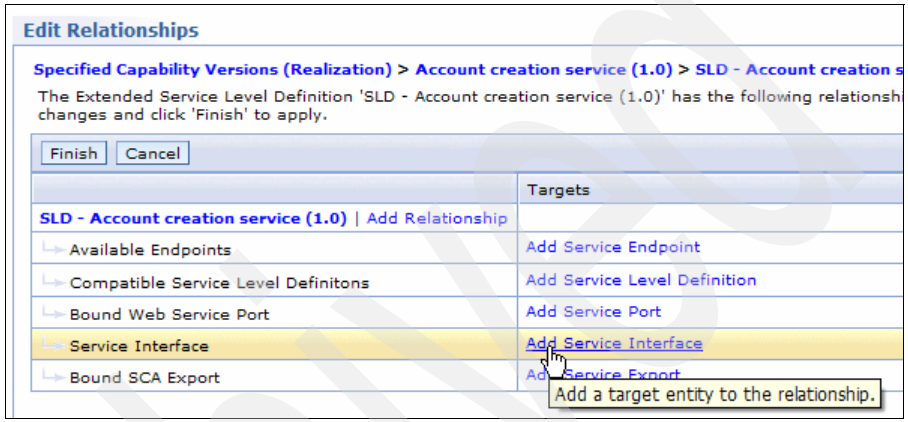


Figure 14-54 Adding the service interface to the SLD

3. Enter `A` in the Name field, select **AccountCreationV1_0** (Figure 14-55) from the auto suggest list, and click **Add**. The service interface is added as a target of the Service Interface relationship.

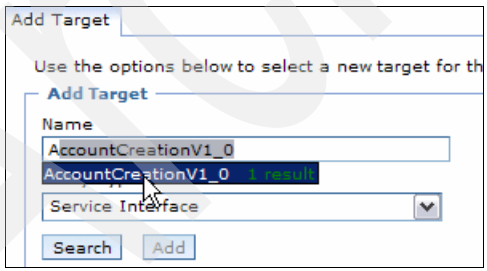


Figure 14-55 Selecting the service interface

4. Click **Finish** to save the relationship change.

5. Click **Propose Scope**. Note that the new governance state is SLD Scope Review as shown in Figure 14-56.

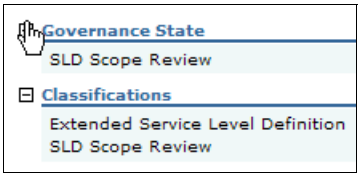


Figure 14-56 Extended SLD scope review governance state

14.7.3 Approving the service level definition

The SOA governance team confirms that the service level definition meets the non-functional requirements. David from the SOA governance team can then approve the service level definition scope.

To transition the service version to the Subscribable state, log on to WSRR, and select the **SOA Governance** perspective. Then, follow these steps:

1. Click **Tasks** → **Service Level Definition Tasks** → **Service Level Definition for Scope Review** as shown in Figure 14-57.

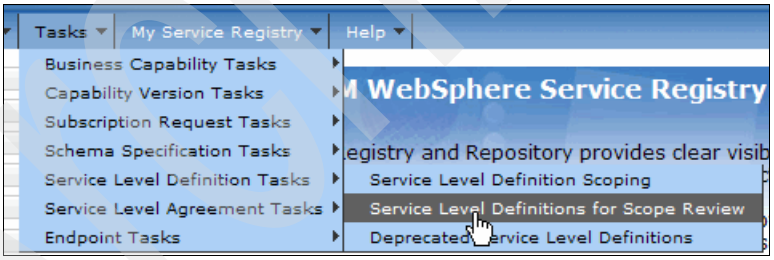


Figure 14-57 Navigating to SLD for scope review

2. Click **SLD - Account creation service (1.0)** to display the SLD details.

Archived

3. Click **Approve Scope**. Note that the new governance state is SLD Subscribable as shown in Figure 14-58.

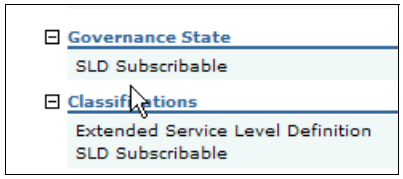


Figure 14-58 Extended SLD subcribable governance state

Figure 5-28 on page 198 shows the modified SLD life cycle used by JKHLE.

Figure 14-59 shows the status of the WSRR entities after the new service level definition is created.

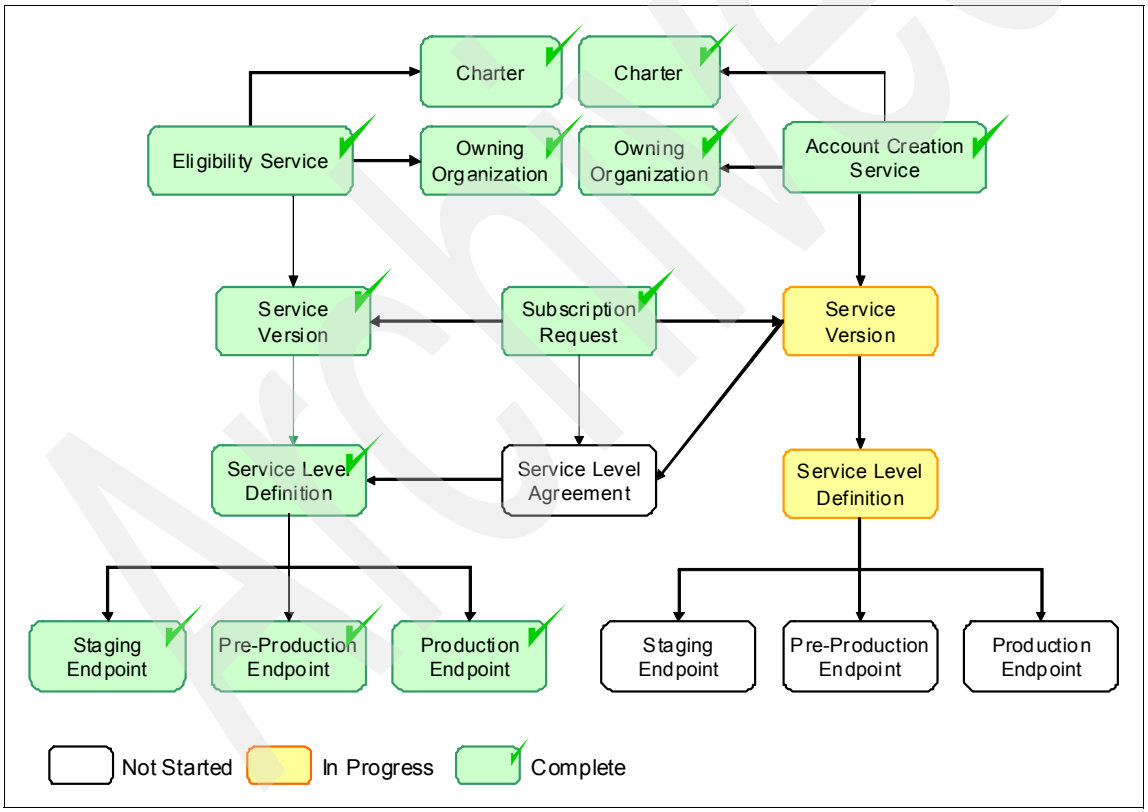


Figure 14-59 Status after creation of a service level definition

Archived

14.8 Creating a service level agreement

The service level agreement (SLA) is the technical implementation of the subscription request. It allows the context and actual interaction patterns to be selected and refined for:

- Service level definition
- Interface
- Bindings
- Ports
- Endpoints
- Policies

The service version consumer account creation, in this case, selects from the subscribable service level definitions that the provider has made available. The consumer then defines the specific quality of service and resource requirements for the service subscription to use. At this time, the provider can negotiate, and thus approve, reject, or refine the request.

Figure 14-60 illustrates the process for creating a service level agreement. We describe this process in this section.

Figure 14-60 Creating the service level agreement process

Archived

14.8.1 Creating the service level agreement

Creating the service level agreement (SLA) is the responsibility of the development organization. Debra, the Development Manager for the commercial line of business, is responsible for creating the service level agreement.

To create the new service level agreement, log on to WSRR and select the **Development** perspective. Then, follow these steps:

1. Click **Tasks** → **Subscription Request Tasks** → **Manage Approved Subscription Requests** as shown in Figure 14-61.

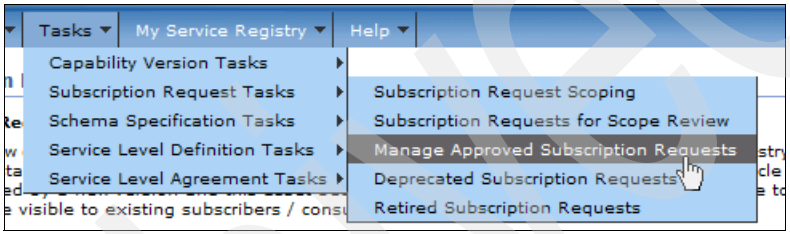


Figure 14-61 Navigating to service level agreements

2. Click **Account creation consumption of eligibility service**.
3. Click **New SLA**.

Archived

4. Under the relationship called **Subscribed Service Level Agreements**, click **SLA - Account creation consumption of eligibility service** (Figure 14-62) to display the details of the new service version.

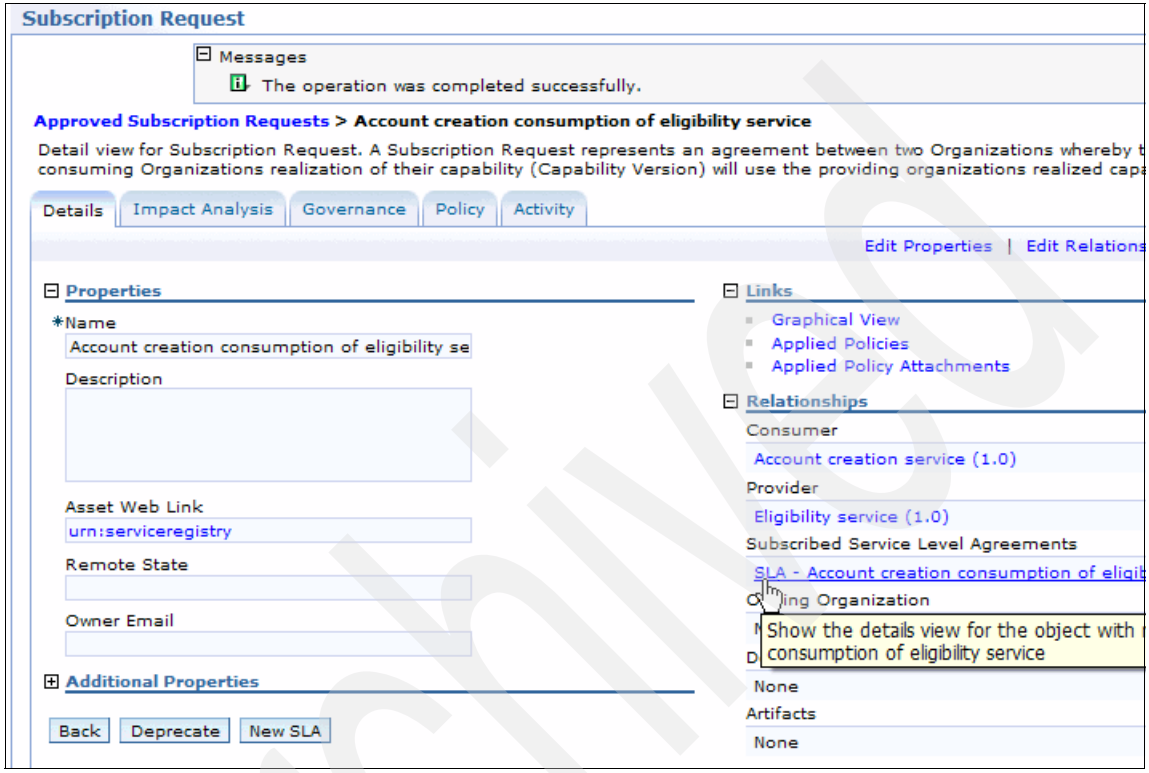


Figure 14-62 Details of the new service version

Archived

5. Click **Edit Properties**, and enter the following values (as shown in Figure 14-63):
   - Name: SLA - Account creation consumption of eligibility service
   - Description: This is the service level agreement between the Eligibility Service (provider) and the Account Creation Service (consumer)
   - Service Level Agreement Availability Date: A date of your choosing
   - Service Level Agreement Termination Date: A date of your choosing

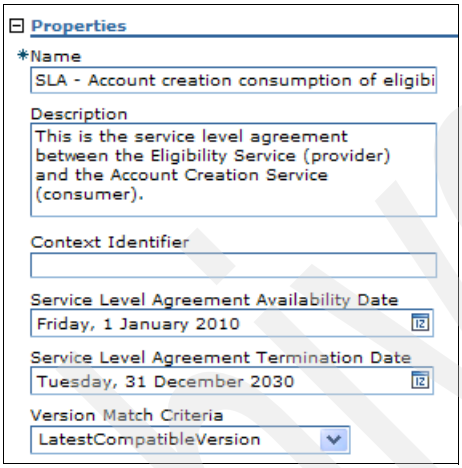


Figure 14-63 Entering the SLA properties

6. Click **OK** to save your changes.

Archived

14.8.2 Adding the agreed endpoints relationship

Now, you need to create a relationship between the service level agreement and the service level definition of the service that is being reused (agreed endpoints). Follow these steps:

1. Click **Edit Relationships**.
2. Click **Add Service Level Definition** to the right of the **Agreed Endpoints** relationship as shown in Figure 14-64.

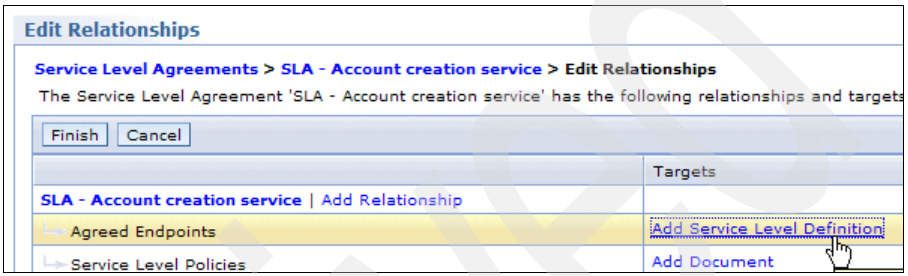


Figure 14-64 Adding the agreed endpoints to the SLA

3. Enter an asterisk (*) in the Name field, select **SLD - Eligibility service (1.0)** (Figure 14-65) from the auto suggest list, and click **Add**. The service level definition is added as a target of the Agreed Endpoints relationship.

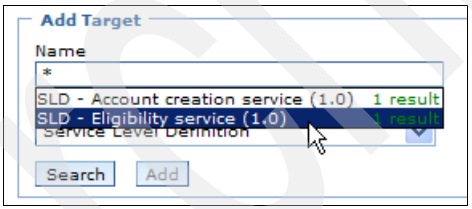


Figure 14-65 Searching for the SLD

4. Click **Finish** to save the relationship change.

Archived

5. Click **Request SLA**. Note that the new governance state is SLA Requested as shown in Figure 14-66.

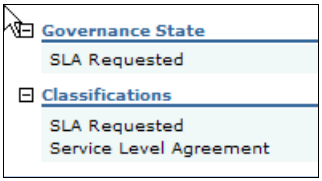


Figure 14-66 Service level agreement requested governance state

14.8.3 Approving the service level agreement

The provider of the service has the option to approve or reject the request or to ask for it to be revised. Here, the request is approved, which moves it to the Inactive state. Thus, the development team that want to consume the service can continue development based on the consumption of this specific service level definition, but they do not yet have authorization to access any endpoints.

The Operations team has the responsibility of approving the service level agreement. Jon, the Operations release manager for Common services, approves the service level agreement.

To transition the service level agreement to the SLA Inactive state, log on to WSRR, and select the **Operations** perspective. Then, follow these steps:

1. Click **Tasks** → **Service Level Agreement Tasks** → **SLA Requests for Approval** as shown in Figure 14-67.

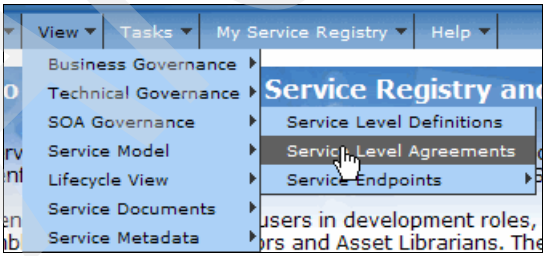


Figure 14-67 Navigating to service level agreements

2. Click **SLA - Account creation consumption of eligibility service** to display the SLA details.

3. Click **Approve SLA Request**. Note that the new governance state is SLA Inactive as shown in Figure 14-68.

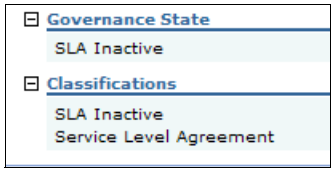


Figure 14-68 Service level agreement Inactive governance state

14.8.4 Activating the service level agreement

When there is a suitable endpoint available for the consumer to invoke, the operations manager for the provided eligibility service activates the SLA. Because the endpoints for the eligibility service are already in the service registry, the SLA can be activated as follows:

1. Click **Tasks** → **Service Level Agreement Tasks** → **SLAs For Activation** as shown in Figure 14-69.

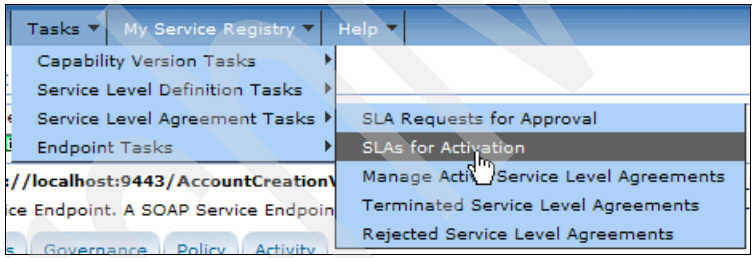


Figure 14-69 Navigating to endpoints for activation

2. Click **SLA - Account creation consumption of eligibility service**.
3. Click **Activate SLA**. Note that the new governance state is SLA Active as shown in Figure 14-70.

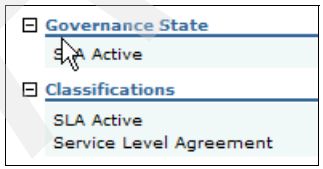


Figure 14-70 SLA Active governance state

Figure 14-71 shows the status of the WSRR entities after the new service level agreement is created.

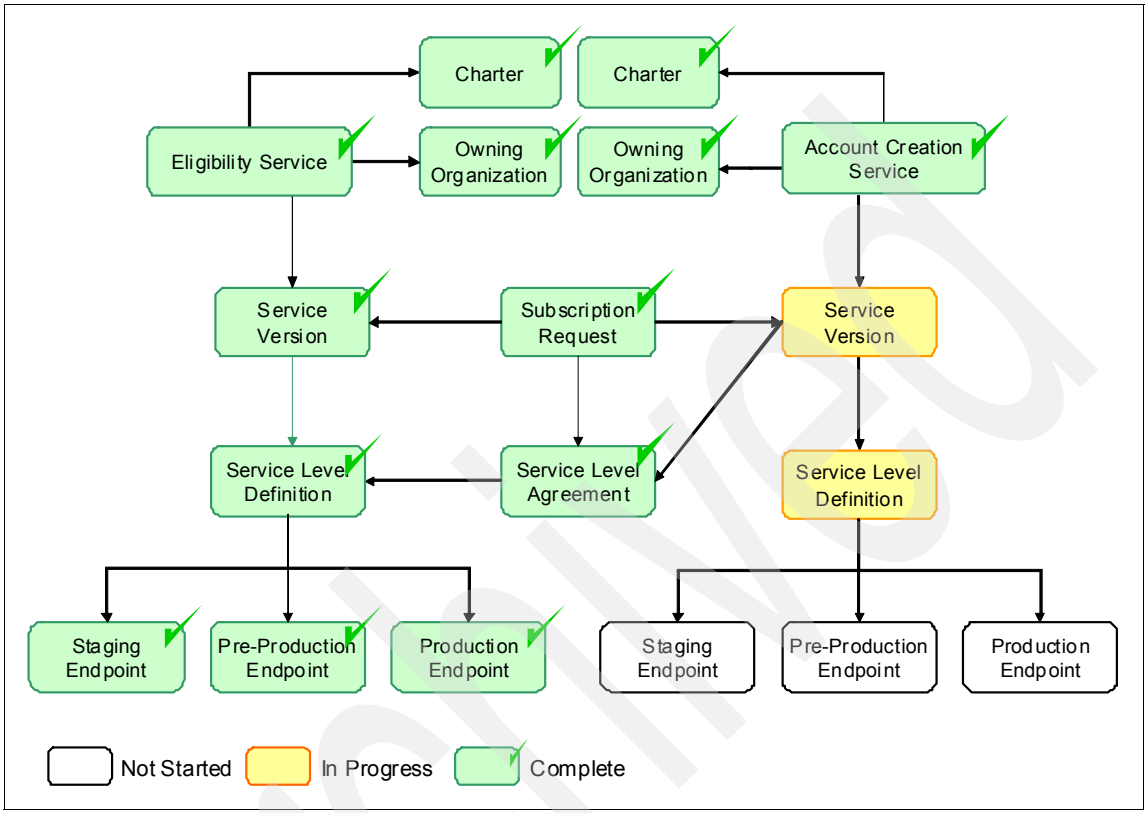


Figure 14-71 Status after creation of a service level agreement

14.9 Deploying a service version to staging

Having created and unit tested an implementation of the service, Development is ready to pass the service to the Operations team for deployment to a staging environment for testing.

Refer to Figure 5-40 on page 208 for details of the SOA life cycle that is used to govern the service version.

Figure 14-72 illustrates the process for deploying a service version. We describe this process in this section.

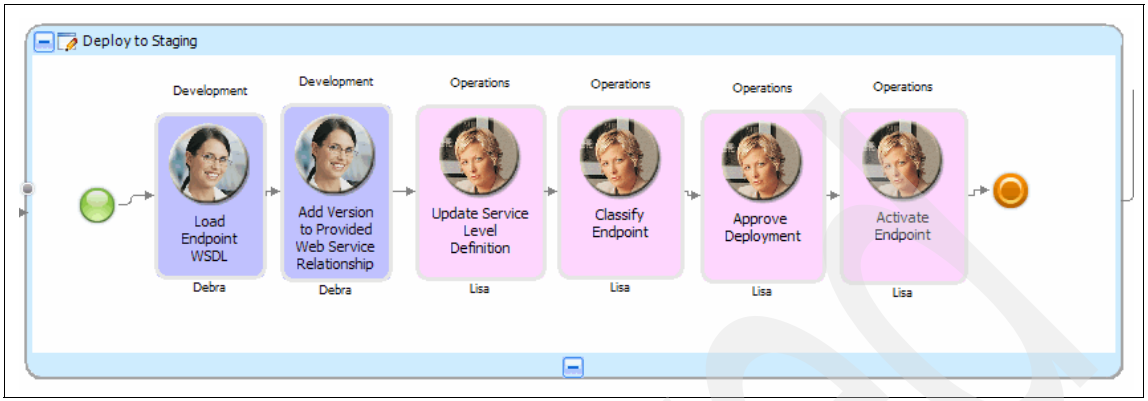


Figure 14-72 Deploying a service version to staging process

Figure 14-73) illustrates the WSRR environment at JKHLE. The new account creation service is verified and promoted to the staging environment for functional testing.

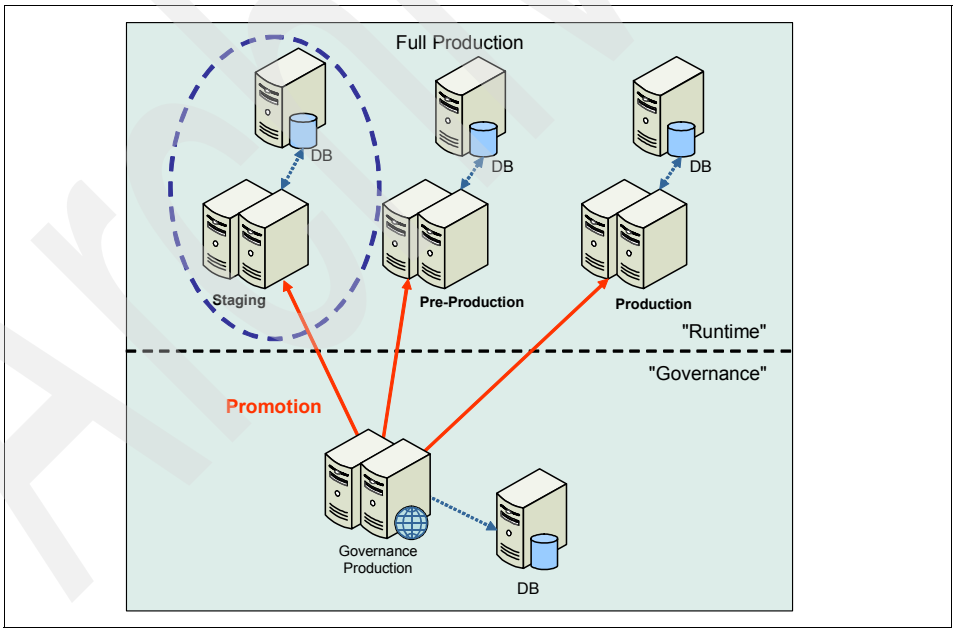


Figure 14-73 Deploying to the staging environment

14.9.1 Loading staging endpoint WSDL

The service implementation WSDL is loaded into the WSRR. Debra, the Development Manager for the commercial line of business, is responsible for this phase of the scenario.

To load the endpoint WSDL, log on to WSRR, and select the **Development** perspective. Then, follow these steps:

1. Click **Actions** → **Load Documents**.
2. Ensure that the Document type is set to **WSDL**.
3. Click **Browse**, and navigate to the directory where the development WSDL is located.
4. Select **AccountCreationV1_0_StagingPort.wsdl**, and click **Open**. Enter a document version of `1.0`, and then click **OK**.
5. Click **Finish** to load the WSDL document.

The staging endpoint WSDL is loaded (Figure 14-74).

Figure 14-74 Loading the staging endpoint WSDL

14.9.2 Adding relationship to the provided Web service

Next, create a relationship between the service version and the provided Web service as follows:

1. Click **Tasks** → **Capability Version Tasks** → **Define SLD & prepare for staging**.
2. Click **Account Creation Service (1.0)** to display the service version details.
3. Click **Edit Relationships**.

Archived

4. Click **Add Service** to the right of the **Provided Web Services** relationship as shown in Figure 14-75.

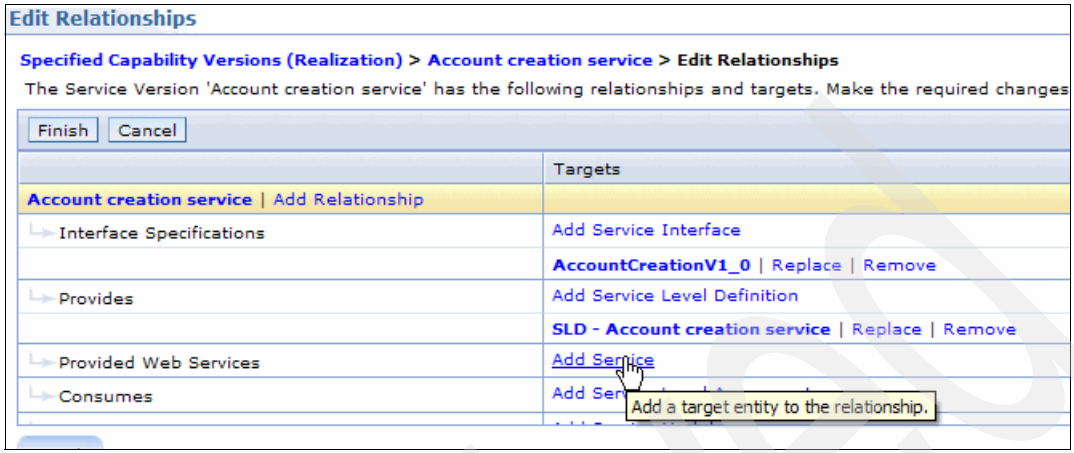


Figure 14-75 Adding the provided Web service relationship

5. Enter `A` in the **Name** field, select **AccountCreationV1_0** from the auto suggest list, and click **Add**. The service is added as a target of the Provided Web Services relationship (Figure 14-76).
6. Click **Finish** to save your changes.

Archived

Specified Capability Versions (Realization) > Account creation service (1.0)

Detail view for Service Version. A Service Version represents a specific version (or release) of a Service and provides a range c functional and non functional specifications that hold for that version of the service. The Service Version exposes its capabiliti service level definitions. It may also (in the case of a composite service) identify the services it depends on by defining Servi Agreements to the Service Level Definitions provided by the consumed service.

Details | Impact Analysis | Governance | Policy | Activity

Edit Properties | Edit Relationships | Edit Cla

Properties

*Name
Account creation service (1.0)

Description
This service version provides the capabilities for the account creation service

Version
1.0

Consumer Identifier
ACSV000

Version Availability Date
Friday, 1 January 2010

Version Termination Date
Tuesday, 31 December 2030

Version Requirements Link
http://requirements.jkhle.com /requirements.jsp?id=8820

Asset Web Link
urn:serviceregistry

Remote State

Links

- Graphical View
- Applied Policies
- Applied Policy Attachments

Relationships

Interface Specifications
AccountCreationV1_0

Provided Web Services
AccountCreationV1_0

Provided SCA Modules
None

Owning Organization
Commercial

Dependency
None

Artifacts
AccountCreationInterfaceV1_0.wsdl

Provides
SLD - Account creation service (1.0)

Consumes
SLA - Account creation consumption of eligibility servi

Figure 14-76 Realized service definition

14.9.3 Updating the service level definition

Having loaded the endpoint into the WSRR, the final step in making the service consumable in the staging environment is to associate the staging endpoint with the service level definition.

The Operations team has the responsibility of updating the service level definition. Lisa, the Operations Release Manager for the commercial line of business, approves the service level definition.

To update the service level definition, log on to WSRR, and select the **Operations** perspective. Then, follow these steps:

1. Click **Tasks** → **Service Level Definition Tasks** → **Manage Subscribable Service Level Definitions** as shown in Figure 14-77.

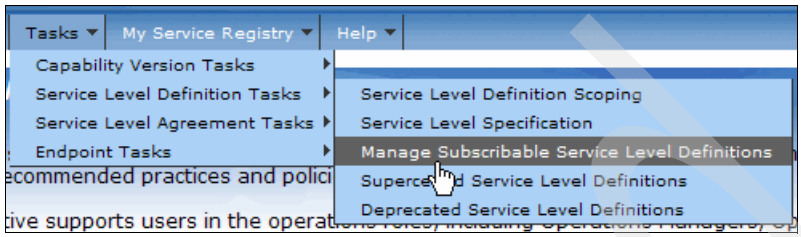


Figure 14-77 Navigating to subscribable service level definitions

2. Click **SLD - Account creation service (1.0)**.
3. Click **Edit Relationships**.
4. Click **Add Service Endpoint** to the right of the **Available Endpoints** relationship.
5. Enter `*Acc` in the Name field, select **https://localhost:9443/AccountCreationV1_0/services/AccountCreationServiceV1_0_StagingPort** from the auto suggest list, and click **Add**. The service endpoint is added as a target of the Available Endpoints relationship.
6. Click **Add Service Port** to the right of the **Bound Web Service Port** relationship.
7. Enter `A` in the Name field, select **AccountCreationServiceV1_0_StagingPort** from the auto suggest list, and click **Add**. The service port is added as a target of the Bound Web Service Port relationship.
8. Click **Finish** to save your changes.

Archived

The service level definition relationships are updated as shown in Figure 14-78.

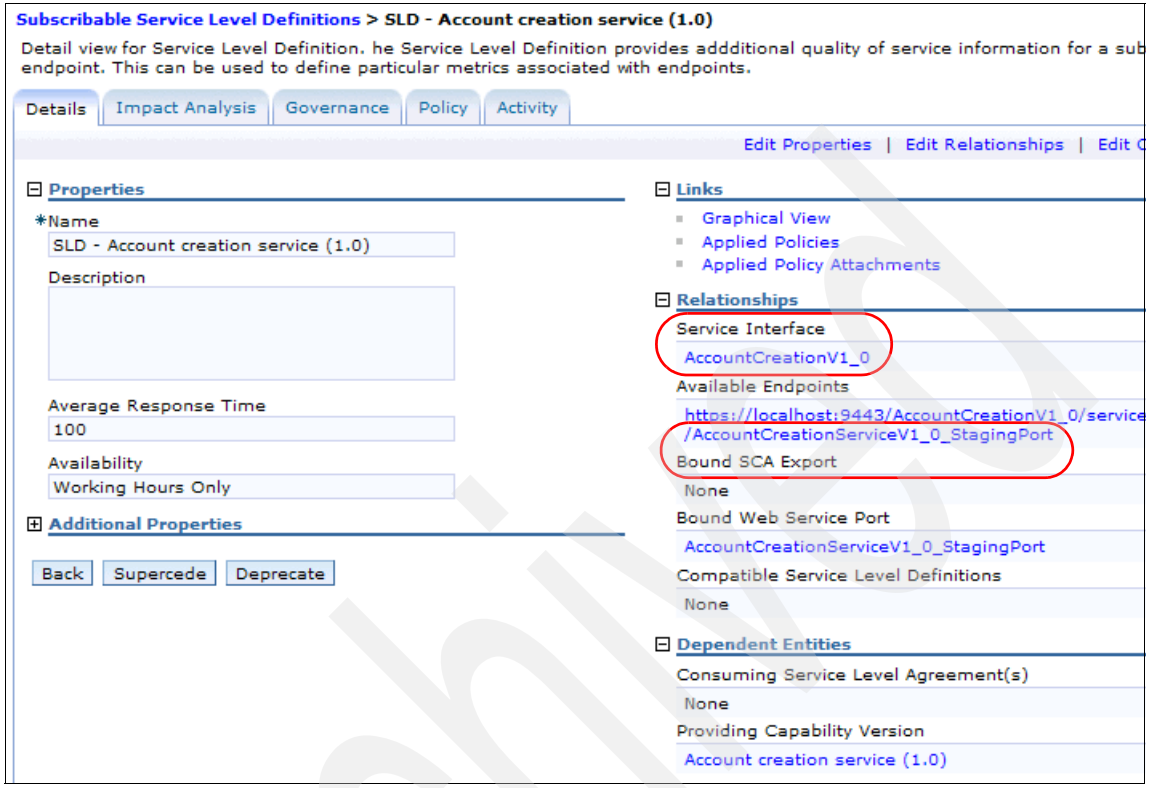


Figure 14-78 Service level definition relationships

Archived

14.9.4 Classifying the endpoint

The Operations team approves the staging endpoint, which verifies that the service is running but not necessarily functioning correctly.

To classify the endpoint, log on to WSRR, and select the **Operations** perspective. Then, follow these steps:

1. Click **Tasks** → **Endpoints** → **Endpoints For Activation** as shown in Figure 14-79.

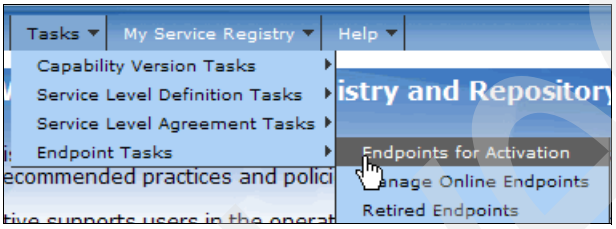


Figure 14-79 Navigate to endpoint for activation

2. Click the offline endpoint **https://localhost:9443/AccountCreationV1_0/services/AccountCreationServiceV1_0_StagingPort** (Figure 14-80).

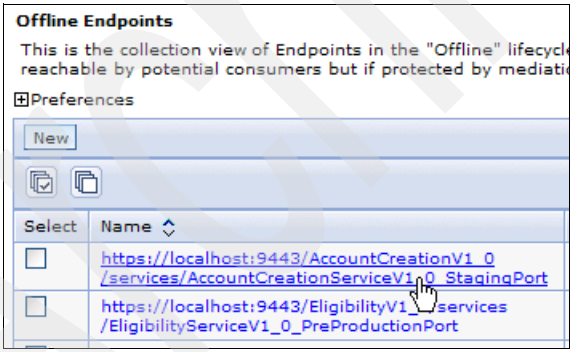


Figure 14-80 Selecting the offline endpoint

3. Click **Edit Classifications**.

Archived

4. Expand **Governance Profile Taxonomy** → **Environment**. Select **Staging** (Figure 14-81), and click **Add**. The Staging classification is added to the Classification list.

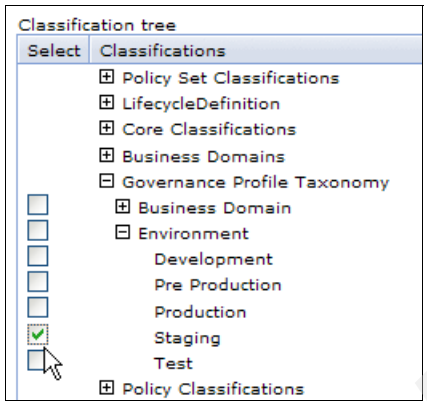


Figure 14-81

5. Click **OK** to assign the classification.

14.9.5 Approving staging deployment

At this point, the service is released officially for use in the staging environment as follows:

1. Click **Tasks** → **Capability Version Tasks** → **Define SLD & prepare for staging** as shown in Figure 14-82.

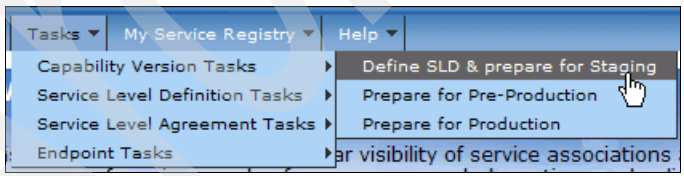


Figure 14-82 Navigate to versions for staging

2. Click **Account Creation Service (1.0)** to display the service version details.

Archived

3. Click **Approve Staging Deployment**. Note that the new governance state is Staged as shown in Figure 14-83.

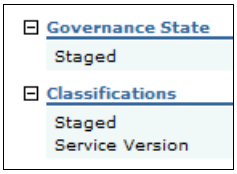


Figure 14-83 Service version staged governance state

14.9.6 Activating the endpoint

Although the service is promoted to the staging environment and approved for use, the endpoint needs to be activated, which updates its status to Online. To activate the endpoint, follow these steps:

1. Click **Tasks** → **Endpoints** → **Endpoints For Activation**.
2. Click the offline endpoint **https://localhost:9443/AccountCreationV1_0/services/AccountCreationServiceV1_0_StagingPort** (Figure 14-80).
3. Click **Approve For Use**. Note that the new governance state is Online as shown in Figure 14-84.

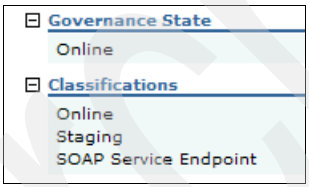


Figure 14-84 Service endpoint online governance state

The account creation service is now promoted to the staging environment. In the staging environment, initial testing of the account creation service is carried out before deployment to pre-production for final acceptance testing.

Archived

Figure 14-85 shows the status of the WSRR entities after the staging environment endpoint is available.

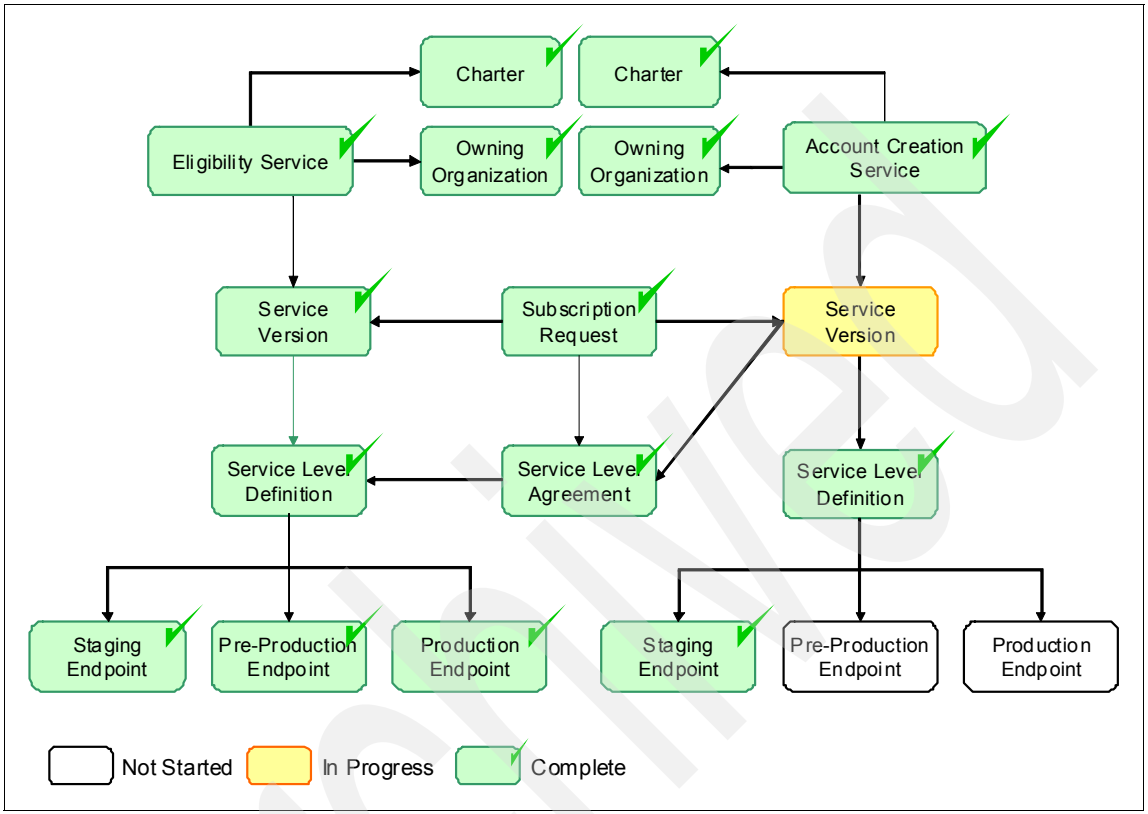


Figure 14-85 Status after deployment to the staging environment

Archived

14.10 Deploying a service version to pre-production

After functional testing completes successfully, the service is deployed to the pre-production environment in a similar manner, and its state becomes Certified.

Figure 14-86 illustrates the process for deploying a service version to pre-production. This section describes this process.

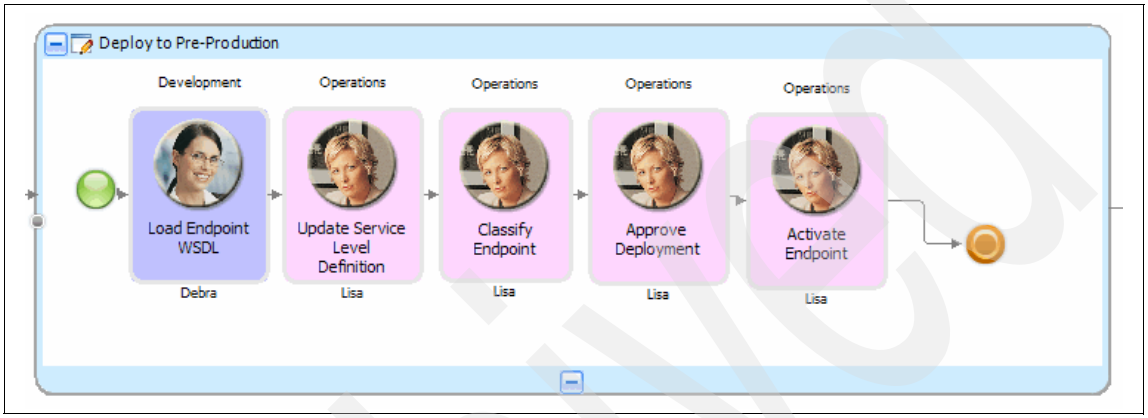


Figure 14-86 Deploying a service version to pre-production process

Archived

Figure 14-87 shows the deployment topology including the pre-production environment.

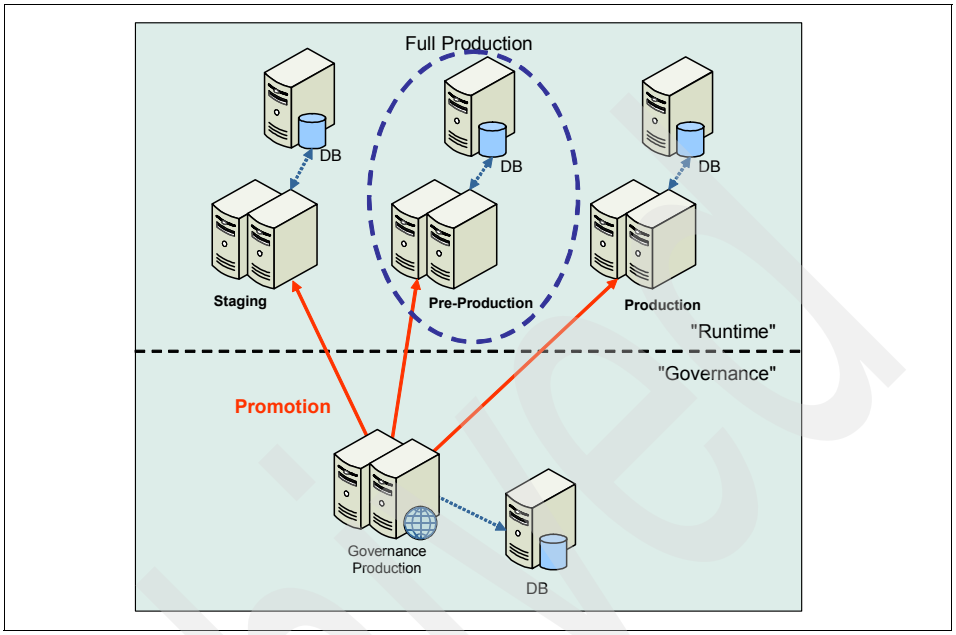


Figure 14-87 Deploying to the pre-production environment

14.10.1 Loading the pre-production endpoint WSDL

The service implementation WSDL is loaded into the WSRR. Debra, the Development Manager for the commercial line of business, is responsible for this phase of the scenario.

To load the pre-production endpoint WSDL, log on to WSRR, and select the **Development** perspective. Then, follow these steps:

1. Click **Actions** → **Load Documents**.
2. Ensure that the Document type is set to **WSDL**.
3. Click **Browse**, and navigate to the directory where the pre-production WSDL is located.
4. Select **AccountCreationV1_0_PreProductionPort.wsdl**, and click **Open**. Enter a document version of `1.0`, and then click **OK**.

5. Click **Finish** to load the WSDL document.

The pre-production endpoint WSDL is loaded as shown in Figure 14-88.

Figure 14-88 Loading the pre-production endpoint WSDL

14.10.2 Updating the service level definition

Having loaded the pre-production endpoint into the WSRR, the final step in making the service consumable in the pre-production environment is to associate the pre-production endpoint with the service level definition.

The Operations team has the responsibility of updating the service level definition. Lisa, the Operations Release Manager for the commercial line of business, is responsible for this phase of the scenario.

To update the service level definition, log on to WSRR, and select the **Operations** perspective. Then, follow these steps:

1. Click **Tasks** → **Service Level Definition Tasks** → **Manage Subscribable Service Level Definitions**.
2. Click **SLD - Account creation service (1.0)**.
3. Click **Edit Relationships**.
4. Click **Add Service Endpoint** to the right of the **Available Endpoints** relationship.
5. Enter `*Acc` in the Name field, select **https://localhost:9443/AccountCreationV1_0/services/AccountCreationServiceV1_0_PreProductionPort** from the auto suggest list, and click **Add**. The service endpoint is added as a target of the Available Endpoints relationship.
6. Click **Add Service Port** to the right of the **Bound Web Service Port** relationship.
7. Enter `A` in the Name field, select **AccountCreationServiceV1_0_PreProductionPort** from the auto suggest list, and click **Add**. The service port is added as a target of the Bound Web Service Port relationship.

Archived

8. Click **Finish** to save your changes.

The SLD relationships are updated to include the Pre Production endpoint information as shown in Figure 14-89.

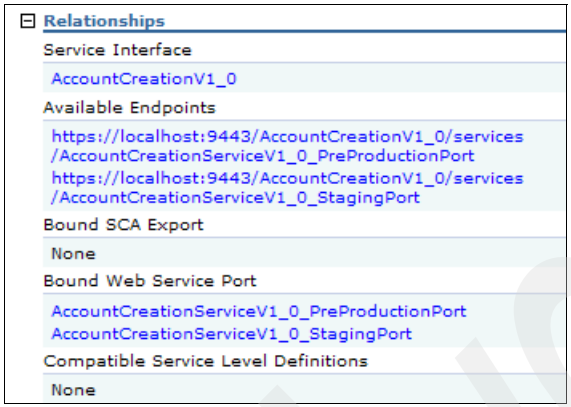


Figure 14-89 Service level definitions with pre-production endpoints

14.10.3 Classifying the endpoint

The Operations team approves the pre-production endpoint. To classify the endpoint, log on to WSRR, and select the **Operations** perspective. Then, follow these steps:

1. Click **Tasks** → **Endpoint Tasks** → **Endpoints For Activation**.
2. Click the development offline endpoint **https://localhost:9443/AccountCreationV1_0/services/AccountCreationServiceV1_0_PreProductionPort**.
3. Click **Edit Classifications**.
4. Expand **Governance Profile Taxonomy** → **Environment**. Then, select **Pre-Production**, and click **Add**. The pre-production classification is added to the Classification list.
5. Click **OK** to assign the classification.

Archived

14.10.4 Approving pre-production deployment

At this point, the service is released officially for use in the pre-production environment as follows:

1. Click **Tasks** → **Capability Version Tasks** → **Prepare for Pre-Production** as shown in Figure 14-90.

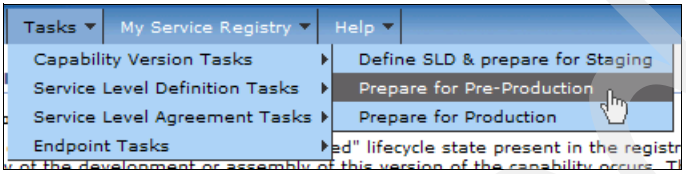


Figure 14-90 Navigate to versions for pre-production

2. Click **Account creation service (1.0)** to display the service version details.
3. Click **Approve Certification**. Note that the new governance state is Certified as shown in Figure 14-91.

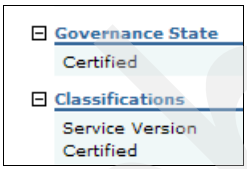


Figure 14-91 Certified service version governance state

14.10.5 Activating the endpoint

Although the service is promoted to the pre-production environment and approved for use, the endpoint needs to be activated, which update its status to Online. To activate the endpoint:

1. Click **Tasks** → **Endpoints** → **Endpoints For Activation**.
2. Click the offline endpoint **https://localhost:9443/AccountCreationV1_0/services/AccountCreationServiceV1_0_PreProductionPort**.
3. Click **Approve For Use**. Note that the new governance state is Online.

The account creation service is now promoted to the pre-production environment.

Archived

In the pre-production environment, final acceptance testing of the account creation service is carried out to ensure that, if it is made available in production, it will meet all of its proposed service level definitions and service level agreements.

Figure 14-92 shows the status of the WSRR entities after the pre-production environment endpoint is available.

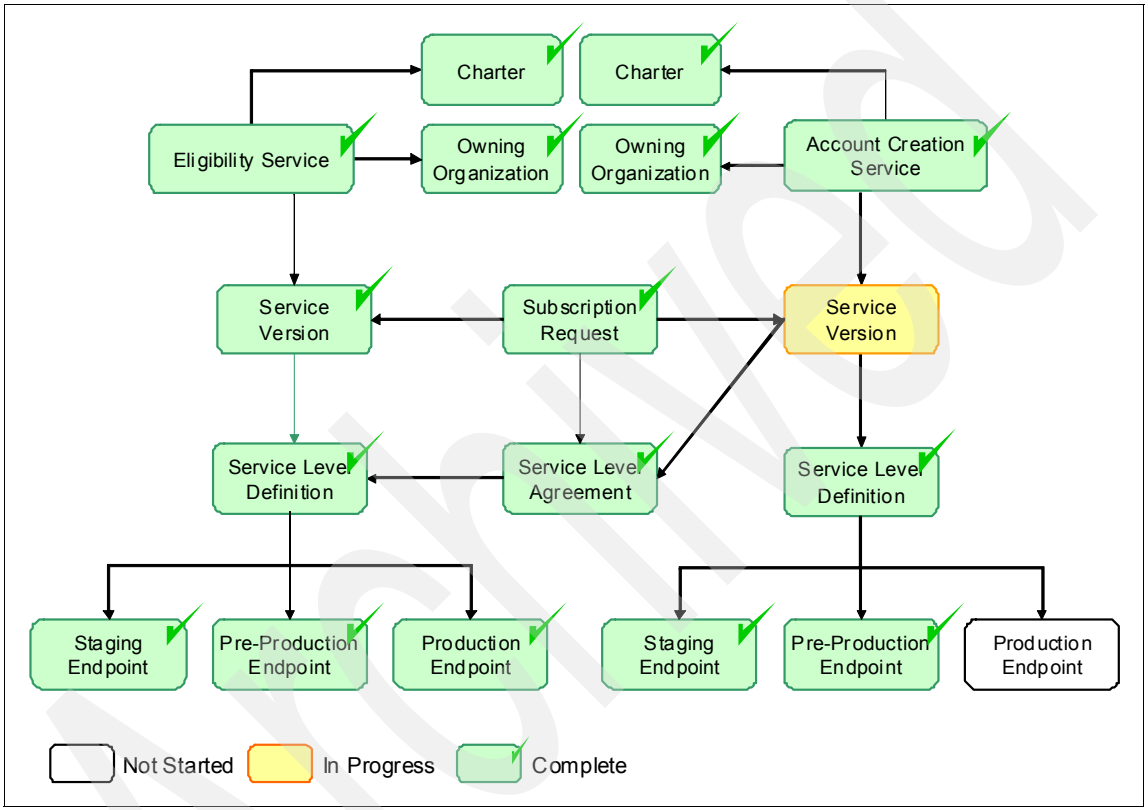


Figure 14-92 Status after deployment to the pre-production environment

Archived

14.11 Deploying a service version to production

After final testing and verification concludes successfully, the service is deployed to the production environment, and its state will become Operational.

Figure 14-93 illustrates the process for deploying a service version to production. We describe this process in this section.

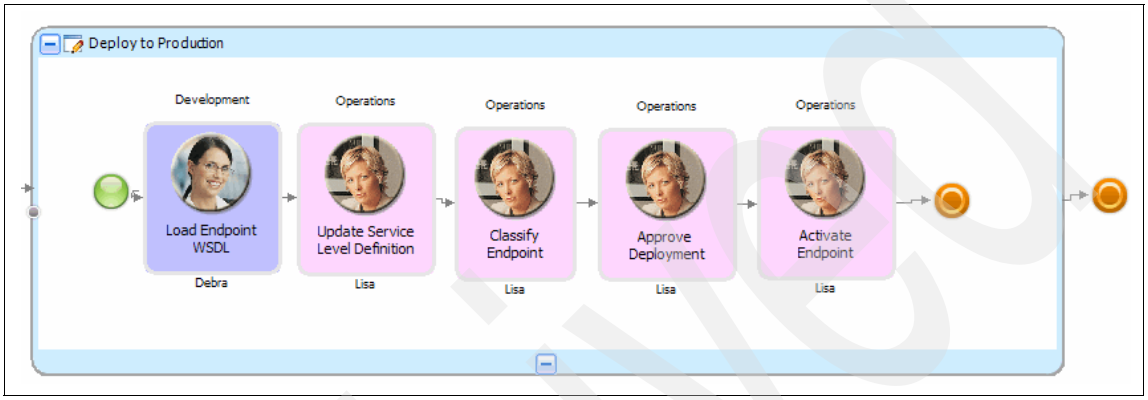


Figure 14-93 Deploying a service version to production process

Figure 14-94 shows the deployment topology including the production environment.

Archived

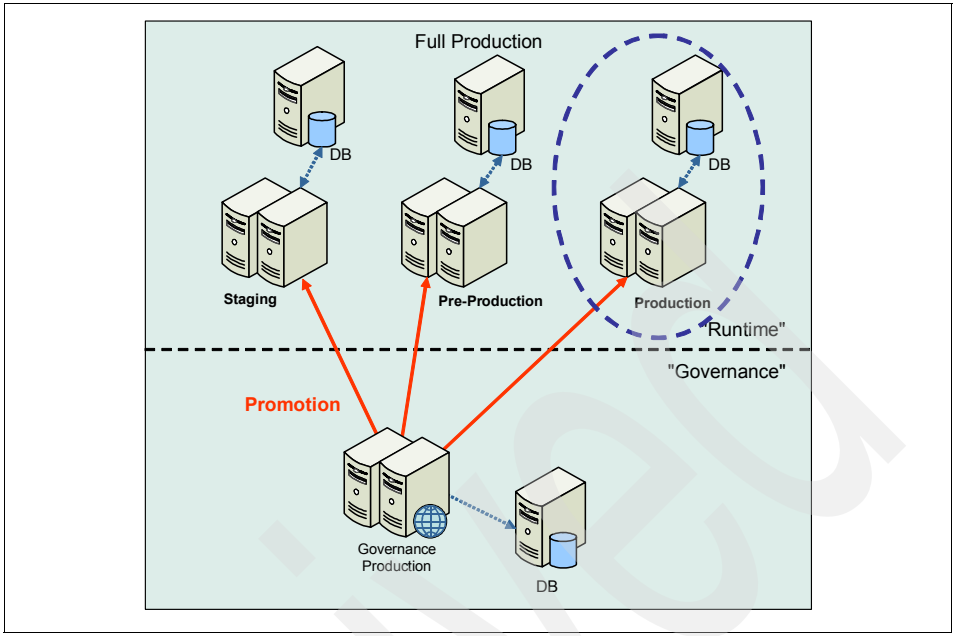


Figure 14-94 Deploying to the production environment

14.11.1 Loading the production endpoint WSDL

The service implementation WSDL is loaded into the WSRR. Debra, the Development Manager for the commercial line of business, is responsible for this phase of the scenario.

To load the production endpoint WSDL, log on to WSRR, and select the **Development** perspective. Then, follow these steps:

1. Click **Actions** → **Load Documents**.
2. Ensure that the Document type is set to **WSDL**.
3. Click **Browse**, and navigate to the directory where the pre-production WSDL is located.
4. Select **AccountCreationV1_0_ProductionPort.wsdl**, and click **Open**. Enter a document version of `1.0`, and then click **OK**.
5. Click **Finish** to load the WSDL document.

The production endpoint WSDL is loaded (Figure 14-95).

Figure 14-95 Loading the production endpoint WSDL

14.11.2 Updating the service level definition

Having loaded the endpoint into the WSRR, the final step in making the service consumable in the production environment is to associate the production endpoint with the service level definition.

The Operations team has the responsibility of updating the service level definition. Lisa, the Operations Release Manager for the commercial line of business, approves the service level definition.

To update the service level definition, log on to WSRR, and select the **Operations** perspective. Then, follow these steps:

1. Click **Tasks** → **Service Level Definition Tasks** → **Manage Subscribable Service Level Definitions**.
2. Click **SLD - Account creation service (1.0)**.
3. Click **Edit Relationships**.
4. Click **Add Service Endpoint** to the right of the **Available Endpoints** relationship.
5. Enter `*Acc` in the Name field, select **https://localhost:9443/AccountCreationV1_0/services/AccountCreationServiceV1_0_ProductionPort** from the auto suggest list, and click **Add**. The service endpoint is added as a target of the Available Endpoints relationship.
6. Click **Add Service Port** to the right of the **Bound Web Service Port** relationship.
7. Enter `A` in the Name field, select **AccountCreationServiceV1_0_ProductionPort** from the auto suggest list, and click **Add**. The service port is added as a target of the Bound Web Service Port relationship.
8. Click **Finish** to save your changes.

Archived

The SLD relationships are updated to include the Production endpoint information as shown in Figure 14-89.

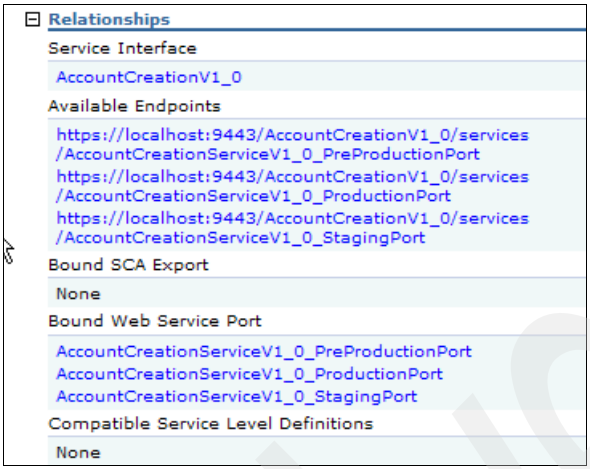


Figure 14-96 Service level definitions with production endpoints

14.11.3 Classifying the endpoint

The Operations team approves the production endpoint.

To classify the endpoint, log on to WSRR, and select the **Operations** perspective. Then, follow these steps:

1. Click **Tasks** → **Endpoint Tasks** → **Endpoints For Activation**.
2. Click the development offline endpoint **https://localhost:9443/AccountCreationV1_0/services/AccountCreationServiceV1_0_ProductionPort**.
3. Click **Edit Classifications**.
4. Expand **Governance Profile Taxonomy** → **Environment**.
5. Select **Production**, and click **Add**. The Production classification is added to the Classification list.
6. Click **OK** to assign the classification.

Archived

14.11.4 Approving production deployment

At this point, the service is ready to be released officially as follows:

1. Click **Tasks** → **Capability Version Tasks** → **Prepare for Production** as shown in Figure 14-97.

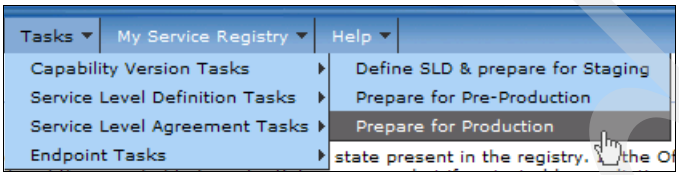


Figure 14-97 Navigating to deploy versions to production

2. Click **Account creation service (1.0)** to display the service version details.
3. Click **Approve Production Deployment**. Note that the new governance state is Operational as shown in Figure 14-98.

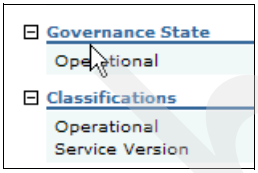


Figure 14-98 Operational service version governance state

14.11.5 Activating the endpoint

Although the service is promoted to the production environment and approved for use, the endpoint needs to be activated, which update its status to Online. To activate the endpoint:

1. Click **Tasks** → **Endpoint Tasks** → **Endpoints For Activation**.
2. Click the offline endpoint **https://localhost:9443/AccountCreationV1_0/services/AccountCreationServiceV1_0_ProductionPort**.
3. Click **Approve For Use**. Note that the new governance state is Online.

The account creation service is now promoted to the production environment.

Archived

Figure 14-92 shows the status of the WSRR entities after the production environment endpoint is available.

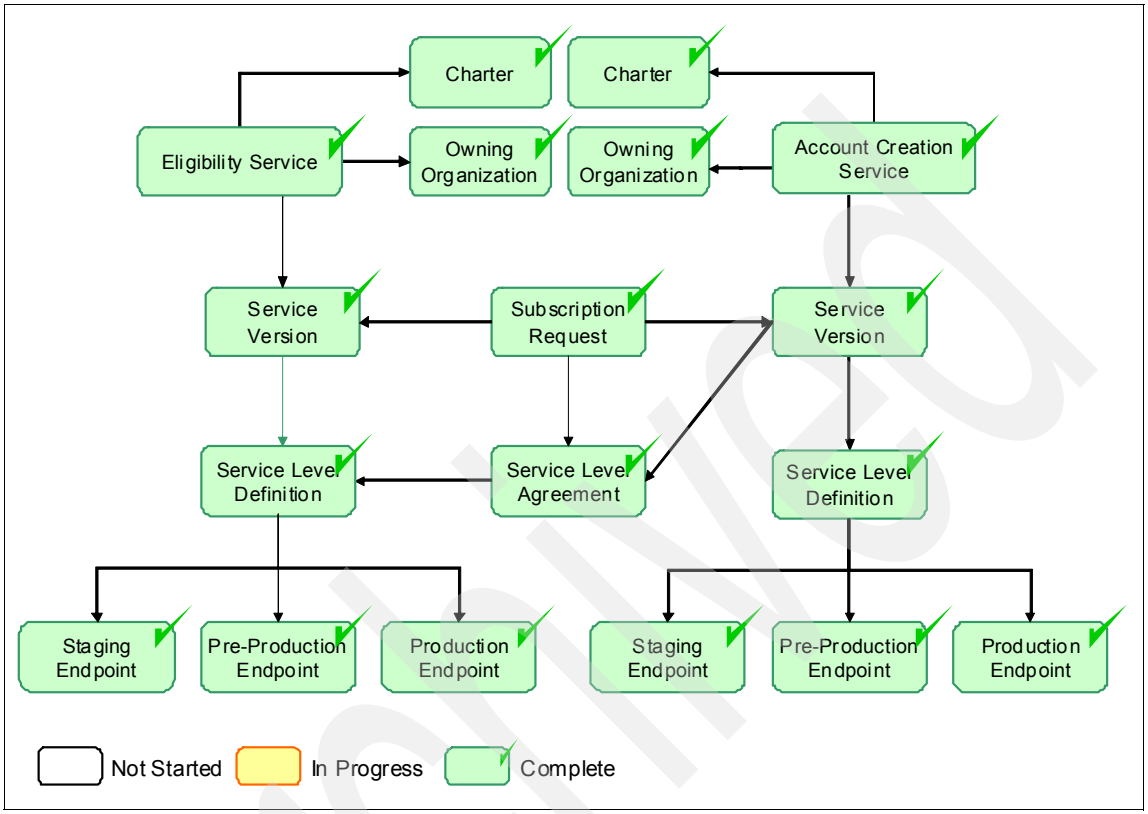


Figure 14-99 Status after deployment to the production environment

Archived

15

Governing a minor upgrade

This chapter provides step-by-step instructions for governing a minor upgrade to an existing service.

Note: In the examples in this book, we use a case study about a fictional company named *JKHL Enterprises* (JKHLE). For information about this case study, see Chapter 4, “JKHL Enterprises case study” on page 153. For information about the roles used throughout this scenario refer to 3.3, “Roles in the governance enablement profile” on page 103.

This chapter includes the following topics:

- Governing a minor upgrade to a service
- Scoping a new service version
- Creating and approving a subscription request
- Planning a service version
- Creating a service level definition
- Creating a service level agreement
- Deploying the upgrade version to staging
- Deploying the upgrade version to pre-production
- Deploying the upgrade version to production

© Copyright IBM Corp. 2009. All rights reserved.

15.1 Governing a minor upgrade to a service

JKHLE currently has an Account Creation service running in the production environment. This service was upgraded recently to expose an additional `verifyCreation` operation as shown in Figure 15-1. This minor upgrade of the service is backwardly compatible with the first version.

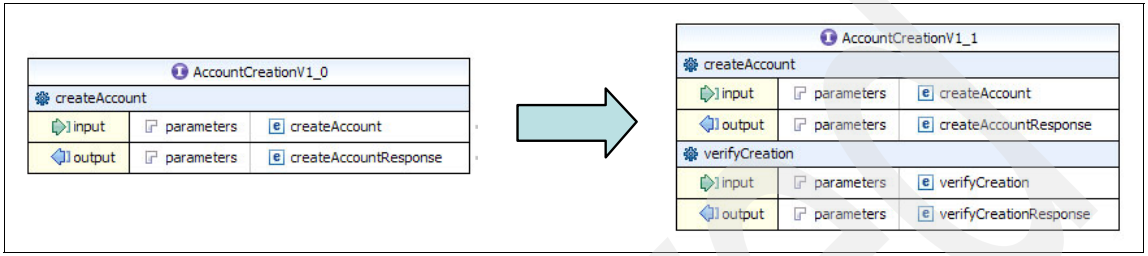


Figure 15-1 Upgrade to service interface

The steps shown in Figure 15-2 describe adding details of the minor upgrade service version to WSRR and making the new endpoint active in all runtime environments.

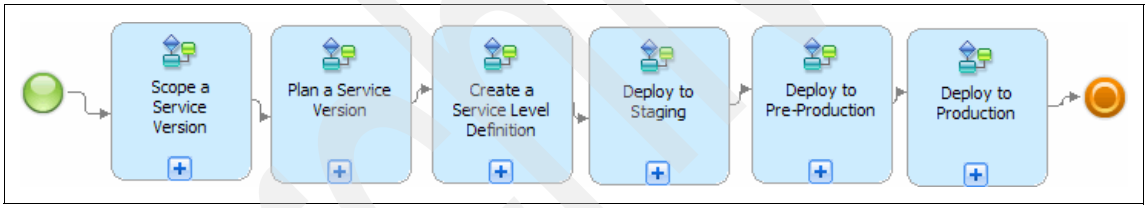


Figure 15-2 Governing a minor upgrade process

Note: A number of the steps in this scenario are described in more detail in the govern a new service scenario. Refer to Chapter 14, “Governing a service that reuses an existing service” on page 475.

Figure 15-3 highlights the major entities in WSRR involved this process.

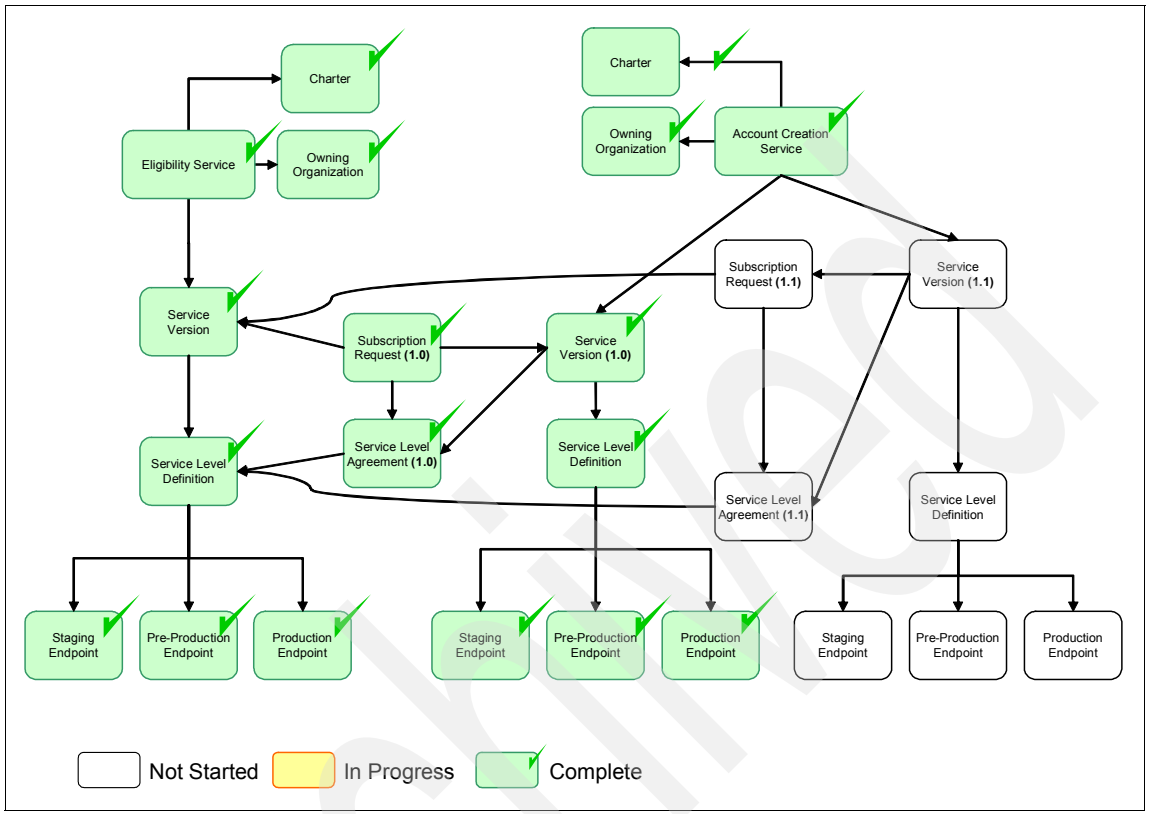


Figure 15-3 WSRR entities for service governance

Archived

15.2 Scoping a new service version

The process for governing a new minor version of the account creation service begins with scopng the service version as illustrated in Figure 15-4. We describe this process in this section.

Figure 15-4 Scoping the service version process

15.2.1 Creating a new service version

After the new business functionality is defined, reviewed, and approved, a new service version is defined.

Debra, the Development Manager for the commercial line of business, is responsible for enhancing the account creation service.

To create the new service version, log on to WSRR, and select the **Development** perspective. Then, follow these steps:

1. Click **View** → **Business Governance** → **Business Capabilities** → **Business Services**.
2. Click **Account creation business service** to display the business service details.
3. Assign a new service version to the business service by clicking N**ew Capability Version**.

4. Under the relationship called **Versions**, click **Account creation business service** to display the details of the new service version.
5. Click **Edit Properties**.
6. Change the following property values:
   - Name: `Account creation service (1.1)`
   - Version: `1.1`
   - Description: `Minor upgrade to add verifyCreation operation.`
   - Consumer Identifier: `ACSV000`
   - Version Requirements Link: `http://requirements.jkhle.com/requirements.jsp?id=8978`
7. Click **OK** to save the changes.
8. Click **Propose Scope** to transition the new service version to a governance state of Scope Review.

15.2.2 Approving the service version scope

In the Scope Review state, David from the SOA governance team reviews the service version requirements.

When the service version scope review is complete, the scope can be approved.

To transition the service version to the Scoped state, log on to WSRR, and select the **SOA Governance** perspective. Then, follow these steps:

1. Click **Tasks** → **Capability Version Tasks** → **Versions for Scope Review**.
2. Click **Account creation service (1.1)** to display the service version details.
3. Click **Approve Scope**. Note that the new governance state is Scoped

Archived

Figure 15-5 shows the business service and two service versions.

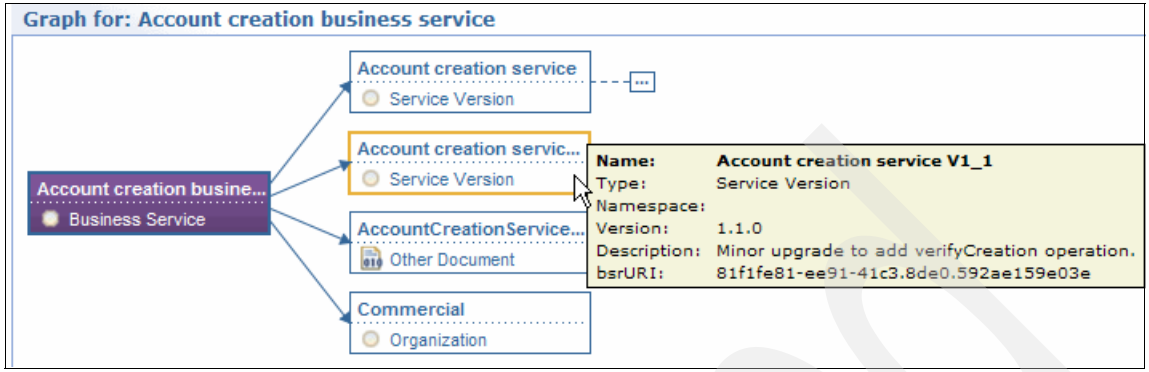


Figure 15-5 Business and service version entities

Figure 15-6 shows the status of the WSRR entities after a new service version is created.

Archived

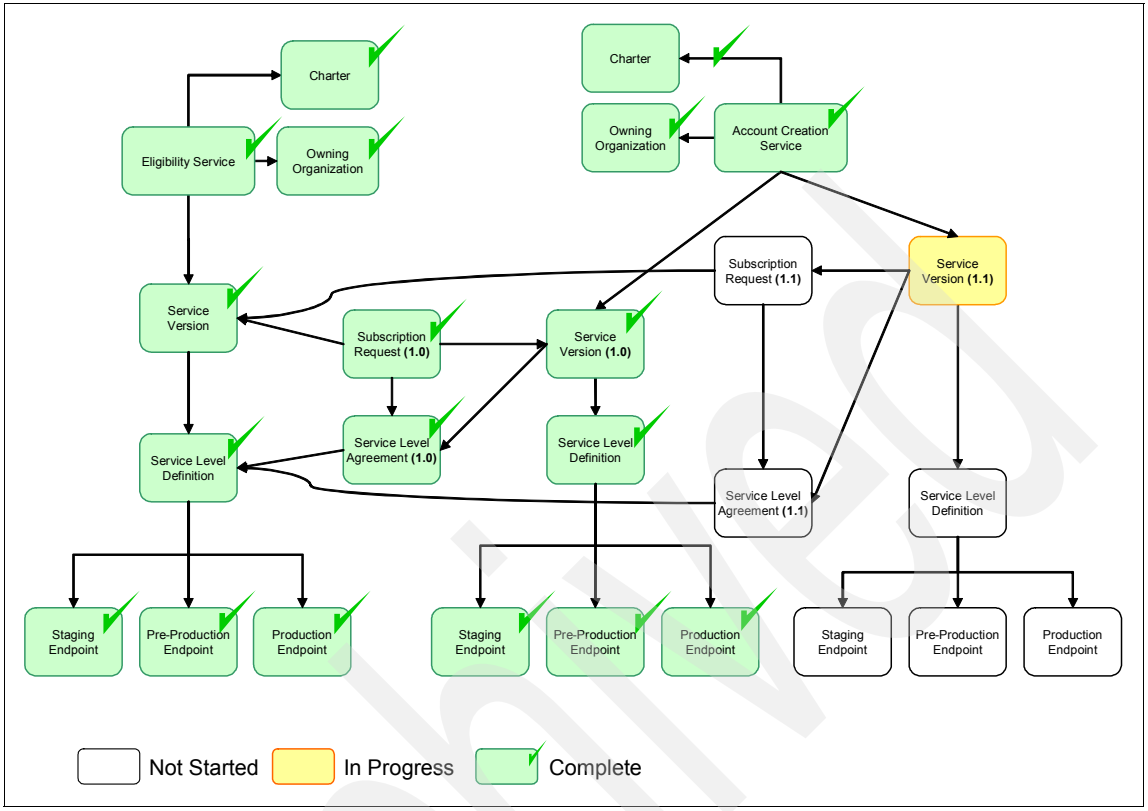


Figure 15-6 Status after service version created

15.3 Creating and approving a subscription request

The new version of the account creation service (v1.1) includes the same requirement to verify the eligibility of a customer as version 1.0 of the account creation service. As the Development Manager for the commercial line of business, Debra creates a subscription request to formalize the reuse of the eligibility service.

The next phase in the governing a new service scenario at JKHLE is creating the subscription request as illustrated in Figure 15-7. We describe this process in this section.

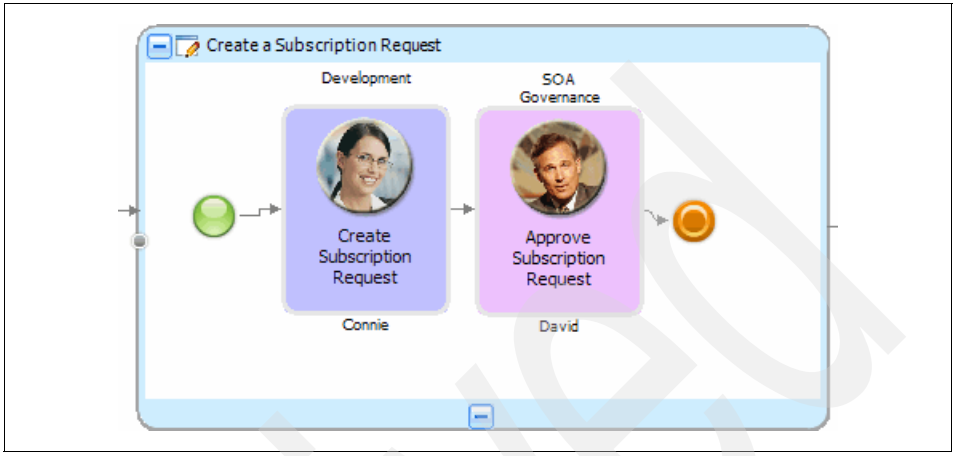


Figure 15-7 Creating the subscription request process

15.3.1 Creating a subscription request

Debra, the Development Manager for the commercial line of business, is responsible for creating the subscription request to use the eligibility service.

To create the new subscription request, log on to WSRR, and select the **Development** perspective. Then, follow these steps:

1. Click **Actions** → **Create** → **Subscription Request**.
2. Enter the following information:
   - Name: `Account creation (1.1) consumption of eligibility service`
   - Requirements Link: `http://requirements.jkhle.com/requirements.jsp?id=7786`

     This is a link, fictitious in this case, to the relevant Subscription Request item in JKHLE's requirements tracking tool. Note that if the requirements are specified in a document rather than in a requirements tracking tool, you add the document as a target of the Artifacts relationship.
3. Click **Add Capability Version** to the right of the **Consumer** relationship.

4. Enter `A` in the Name field, select **Account creation service (1.1)** from the auto suggest list, and click **Add**. The service version is added as a target of the Consumer relationship.
5. Click **Add Capability Version** to the right of the **Provider** relationship.
6. Enter `E` in the Name field, select **Eligibility service (1.0)** from the auto suggest list, and click **Add**. The service version is added as a target of the Provider relationship.
7. Click **Finish** to save your changes.
8. Click **Propose**. Note that the new governance state is Asset Scope Review.

15.3.2 Approving the subscription request

The SOA governance team reviews and approves the subscription request. They are responsible for ensuring that the following commitments are made:

- The service provider commits to providing the necessary resources to meet the service level that the account creation service requires according to the project schedule.
- The consumer agrees that the capabilities detailed in the eligibility service version specifications will meet the requirements of the account creation service. Additionally, this confirmation of acceptance of the subscription request indicates that they have all of the detailed interface, binding, and policy information required in order for them to continue the development of the account creation service.

After the subscription request is approved by all of the consumer and provider stakeholders, David, the chairman of the SOA governance team, approves the subscription request. This approval is the seal of approval from the governance team that the relationship between the consumer and provider can be entered into.

To transition the subscription request to the Asset Approved state, log on to WSRR, and select the **SOA Governance** perspective. Then, follow these steps:

1. Click **Tasks** → **Subscription Request Tasks** → **Subscription Requests for Scope Review**.
2. Click **Account creation (1.1) consumption of eligibility service** to display the subscription request details.
3. Click **Approve**. Note that the new governance state is Asset Approved.

Figure 15-8 shows the status of the WSRR entities after the new subscription request is approved.

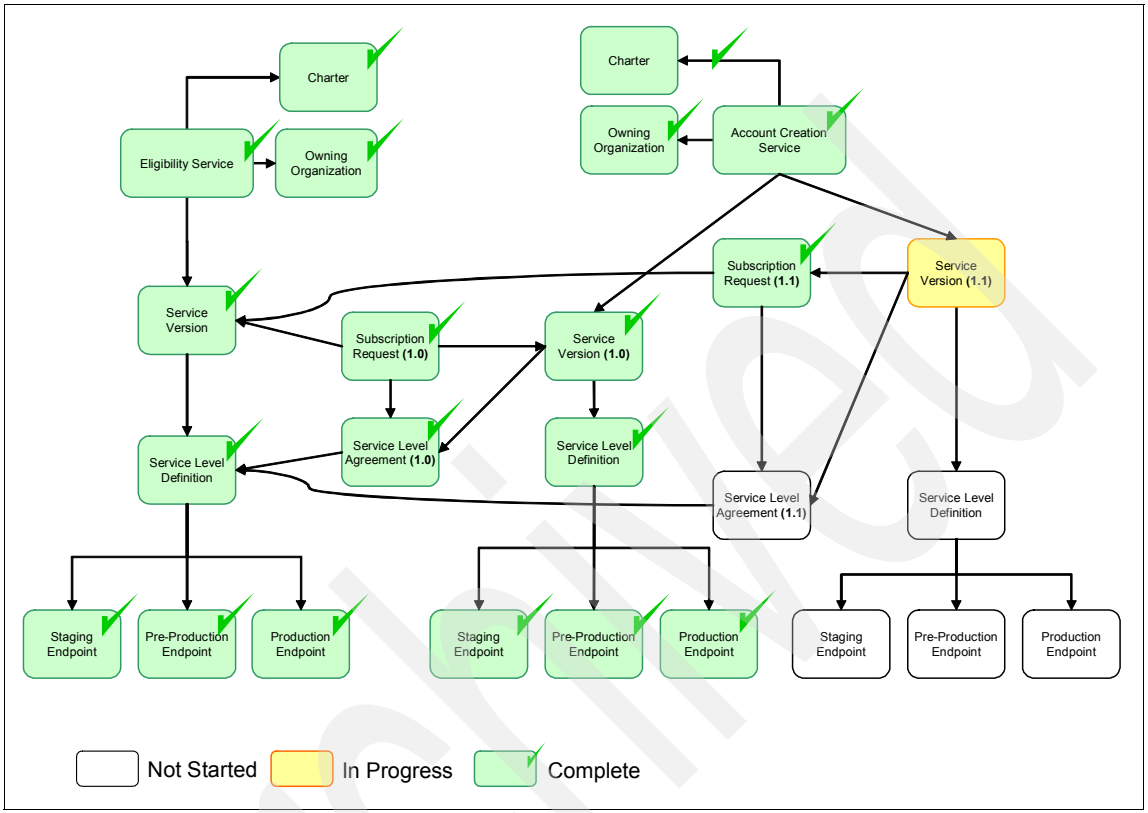


Figure 15-8 Status after subscription request is approved

Archived

15.4 Planning a service version

The scoped version is now in the planning phase where the development and owning organizations must work together to define the funding and timeframes for the upgrade. Figure 15-9 on page 555 illustrates this planning process. We describe this process in this section.

Figure 15-9 Planning a service version process

15.4.1 Completing service version details

The service version must be updated to include proposed availability and termination dates. The plan is then proposed, which makes the plan available for review.

Updating the service version details is the responsibility of the development organization. Debra, the Development Manager for the commercial line of business, is responsible for this phase of the scenario.

To create the new service version, log on to WSRR, and select the **Development** perspective. Then, follow these steps:

1. Click **Tasks** → **Capability Version Tasks** → **Version Planning**.
2. Click **Account creation service (1.1)** to display the service version details.
3. Then click **Edit Properties**.

Archived

4. Enter values of your choice in the Version Availability Date and Version Termination Date fields.
5. Click **OK** to save your changes.

15.4.2 Loading the WSDL

You can now load the abstract WSDL that defines the updated service interface as follows:

1. Click **Edit Relationships**.
2. Click **Add Document** to the right of the **Artifacts** relationship. You might have to scroll down in the relationships pane.
3. Ensure that the Document Type in the Load Document pane is set to **WSDLDocument.** Click **Load Document**.
4. Click **Browse**, and navigate to the directory where the WSDL document is located.
5. Select **AccountCreationInterfaceV1_1.wsdl**, and click **Open**. Then, click **OK**.
6. Cick **Finish** to load the WSDL document.
7. Click **Finish** to save your changes.

15.4.3 Adding the interface specification relationship

Now, you need to create a relationship between the new service version and the interface specification as follows:

1. Click **Edit Relationships**.
2. Click **Add Service Interface** to the right of the **Interface Specifications** relationship.
3. Enter `A` in the Name field, select **AccountCreationV1_1** from the auto suggest list, and click **Add**. The AccountCreation Service Interface is added as a target of the Interface Specification.
4. Click **Finish** to save your changes.

Archived

15.4.4 Approving the service version

After all parties have agreed to the details in the plan, David from the SOA governance team can approve the service version plan.

To transition the service version to the Planned state, log on to WSRR, and select the **SOA Governance** perspective. Then, follow these steps:

1. Click **Tasks** → **Capability Version Tasks** → **Version Planning**.
2. Click **Account creation service (1.1)** to display the service version details
3. Click **Approve Specification**. Note that the new governance state is Specified.

Figure 15-10 shows the status of the WSRR entities after the service version is approved.

Archived

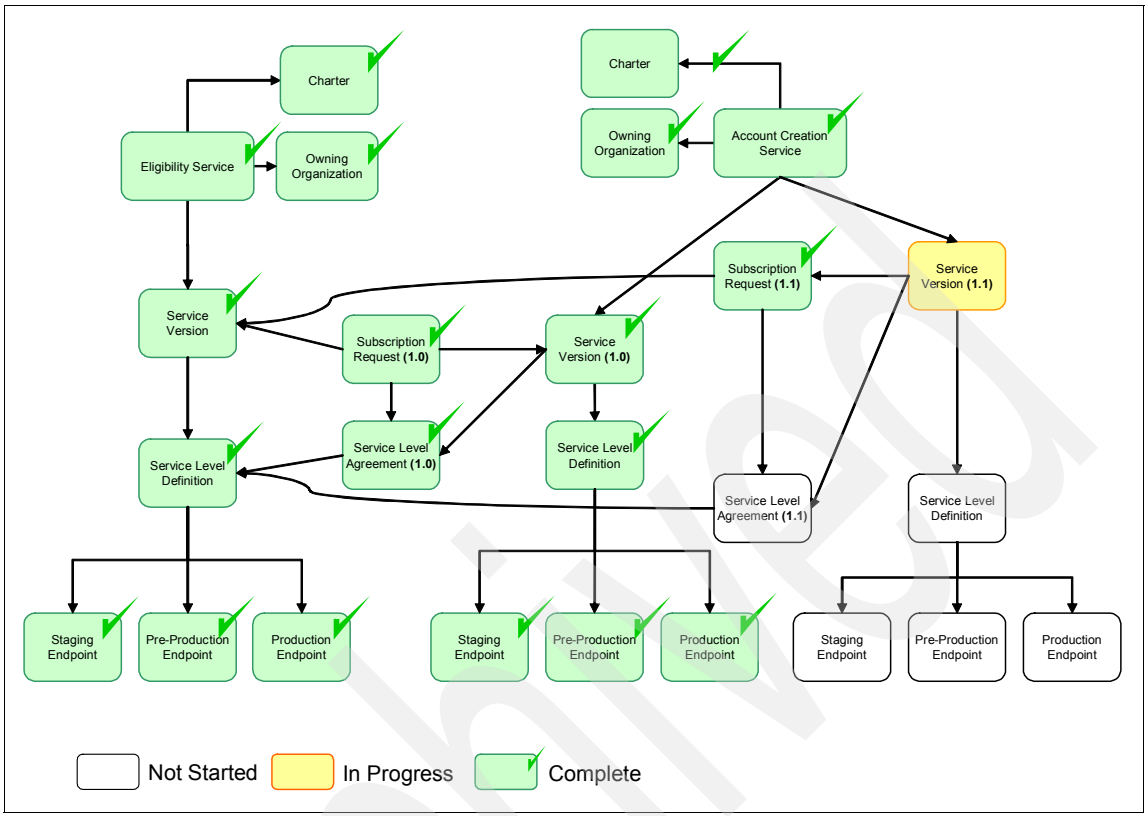


Figure 15-10 Status after service version is approved

Archived

15.5 Creating a service level definition

Now that the upgraded interface objects are defined, the development team must define a new service level definition to which the service version will adhere. The service level definition specifies non-functional requirements for interacting with the provided service. Figure 15-11 illustrates the process for creating a service level definition. We describe this process in this section.

Figure 15-11 Creating a service level definition process

15.5.1 Creating the service level definition

Creating the service level definition (SLD) is the responsibility of the development organization. Debra, the Development Manager for the commercial line of business, is responsible for this phase of the scenario.

To create the new service level definition, log on to WSRR, and select the **Development** perspective. Then, follow these steps:

1. Click **Tasks** → **Capability Version Tasks** → **Define SLD and prepare for staging**.
2. Click **Account creation service (1.1)** to display the service version details.
3. Click **New SLD**.
4. Click **SLD - Account creation service (1.1)** under the **Provides** relationship to display the SLD details.

Archived

5. Click **Edit Properties** and enter the following information:
   - Average Response Time: `100`
   - Availability: `Working Hours Only`
6. Click **OK** to save your changes.

15.5.2 Adding the service interface

Now, you need to create a relationship between the service level definition and the service interface as follows:

1. Click **Edit Relationships**.
2. Click **Add Service Interface** to the right of the **Service Interface** relationship.
3. Enter `A` in the Name field, select **AccountCreationV1_1** from the auto suggest list, and click **Add**. The service interface is added as a target of the Service Interface relationship.
4. Click **Finish** to save the relationship change.
5. Click **Propose Scope**. Note that the new governance state is SLD Scope Review.

15.5.3 Approving the service level definition

The SOA governance team confirm that the service level definition meets the non-functional requirements. David from the SOA governance team can then approve the service level definition scope.

To transition the service version to the Subscribable state, log on to WSRR and select the **SOA Governance** perspective. Then, follow these steps:

1. Click **Tasks** → **Service Level Definition Tasks** → **Service Level Definition for Scope Review**.
2. Click **SLD - Account creation service (1.1)** to display the SLD details.
3. Click **Approve Scope**. Note that the new governance state is SLD Subscribable.

Archived

15.5.4 Creating a compatible service level definition relationship

You need to create a compatible service level definitions relationship from the service level definition for V1_0 to the new service level definition for V1_1 to identify that this new version is backwardly compatible with V1_0.

The Operations team has the responsibility of updating the service level definition. Jon, the Operations release manager for Common services, approves the service level definition.

To update the service level definition, log on to WSRR, and select the **Operations** perspective. Then, follow these steps:

1. Click **View** → **SOA Governance** → **Service Level Definitions** as shown in Figure 15-12.

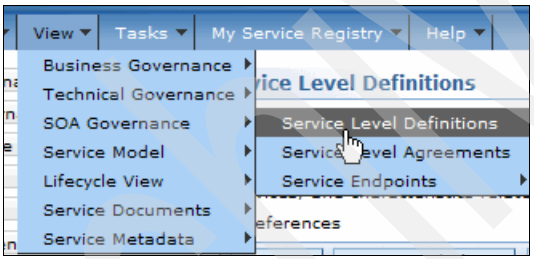


Figure 15-12 Navigating to service level definitions

2. Click **SLD - Account creation service V1_0** to display the original service version details.
3. Click the **Edit Relationships**.
4. Click **Add Service Level Definition** to the right of the compatible service level definitions relationship.
5. Enter `*1.1` in the Name field, select **SLD - AccountCreation (1.1)** from the auto suggest list, and click **Add**.
6. Click **Finish**.

Archived

The compatible service level definition relationship is created as shown in Figure 15-13.

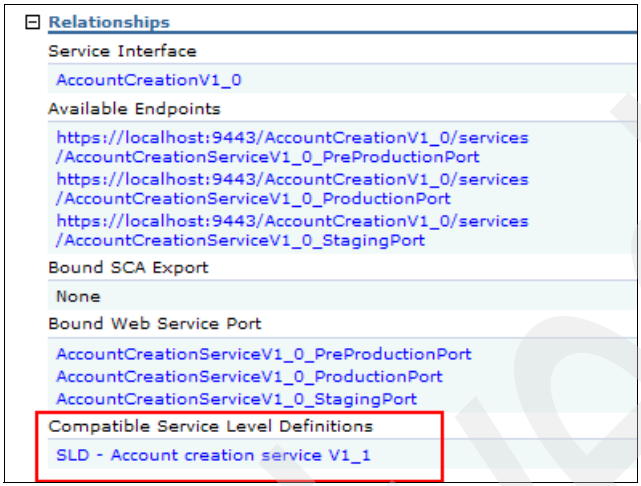


Figure 15-13 Version 1_0 SLD compatible service level definition relationship

Archived

Figure 15-14 shows the status of the WSRR entities after the service level definition is created.

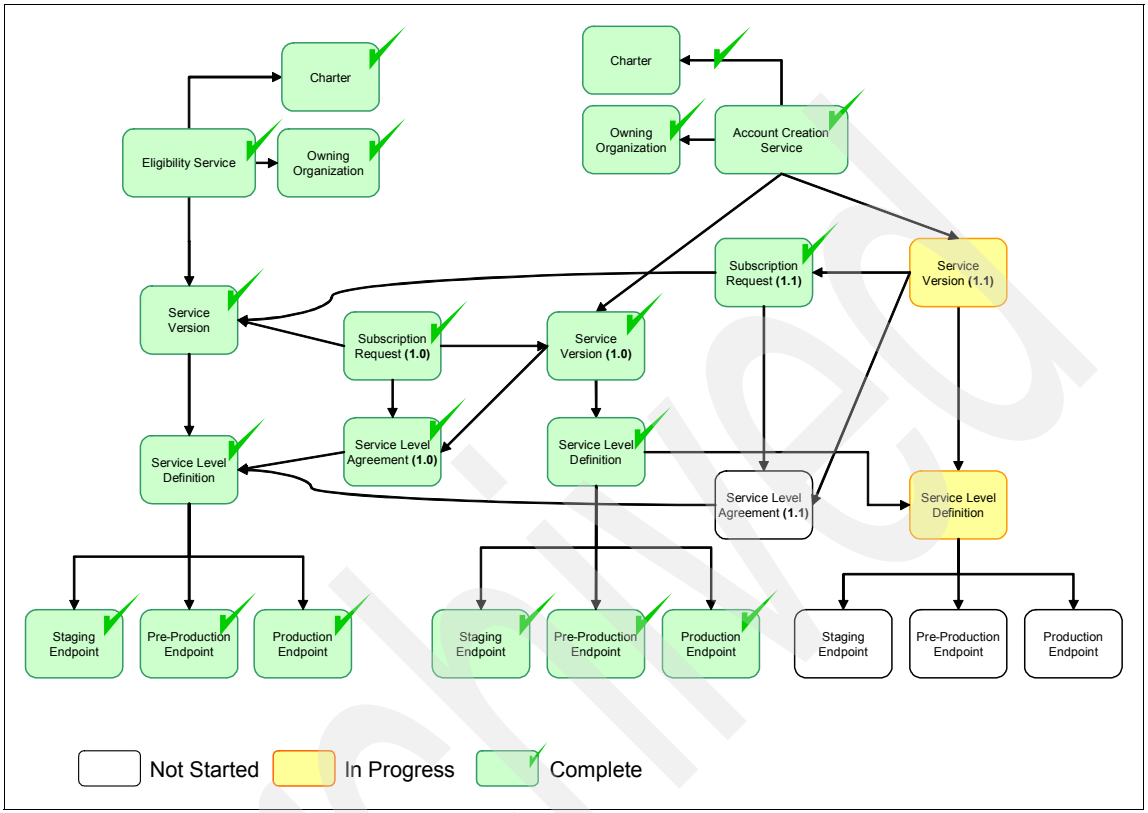


Figure 15-14 Status after SLD is created

15.6 Creating a service level agreement

The service level agreement (SLA) is the technical implementation of the subscription request. It allows the context and actual interaction patterns to be selected and refined for:

- Service level definition
- Interface
- Bindings
- Ports
- Endpoints
- Policies

Archived

The service version consumer account creation, in this case, selects from the subscribable service level definitions that the provider has made available.

The consumer then defines the specific quality of service and resource requirements for the subscription service. At this time, the provider can negotiate, and thus approve, reject, or refine the request.

Figure 15-15 illustrates the process for creating an SLA. We describe this process in this section.

Figure 15-15 Creating the service level agreement process

15.6.1 Creating the SLA

Creating the SLA is the responsibility of the development organization. Debra, the Development Manager for the commercial line of business, is responsible for creating the service level agreement.

To create the new SLA, log on to WSRR, and select the Development perspective. Then, follow these steps:

1. Click **Tasks** → **Subscription Request Tasks** → **Manage Approved Subscription Requests**.
2. Click **Account creation (1.1) consumption of eligibility service.**
3. Click **New SLA**.
4. Under the relationship called **Subscribed Service Level Agreements**, click **SLA - Account creation (1.1) consumption of eligibility service** to display the details of the new service version.

5. Enter the following values:
   - Name: `SLA - Account creation (1.1) consumption of eligibility service`
   - Description: `This is the SLA between the Eligibility Service (provider) and the Account Creation Service 1.1 (consumer).`
   - Service Level Agreement Availability Date: A date of your choosing
   - Service Level Agreement Termination Date: A date of your choosing
6. Click **OK** to save your changes.

15.6.2 Adding the agreed endpoints relationship

Now, you need to create a relationship between the service level agreement and the service level definition of the service that is being reused (the agreed endpoints). Follow these steps:

1. Click **Edit Relationships**.
2. Click **Add Service Level Definition** to the right of the **Agreed Endpoints** relationship.
3. Enter an asterisk (*) in the Name field, select **SLD - Eligibility service (1.0)** from the auto suggest list, and click **Add**. The service level definition is added as a target of the Agreed Endpoints relationship.
4. Click **Finish** to save the relationship change.
5. Click **Request SLA**. Note that the new governance state is SLA Requested.

15.6.3 Approving the service level agreement

The provider of the service has the option to approve or reject the request or to ask for it to be revised. Here, the request is approved, which moves it to the Inactive state. Thus, the development team that wants to consume the service can continue development based on the consumption of this specific service level definition, but they do not yet have authorization to access any endpoints.

The Operations team has the responsibility of approving the service level agreement. Jon, the Operations release manager for Common services, approves the service level agreement.

Archived

To transition the service level agreement to the SLA Inactive state, log on to WSRR, and select the **Operations** perspective. Then, follow these steps:

1. Click **Tasks** → **Service Level Agreement Tasks** → **SLA Requests for Approval**.
2. Click **SLA - Account creation (1.1) consumption of eligibility service** to display the SLA details.
3. Click **Approve SLA Request**. Note that the new governance state is SLA Inactive.

15.6.4 Activating the service level agreement

When there is a suitable endpoint available for the consumer to invoke, the operations manager for the provided eligibility service activates the SLA. Because the endpoints for the eligibility service are already in the service registry, the SLA can be activated as follows:

1. Click **Tasks** → **Service Level Agreement Tasks** → **SLAs For Activation**.
2. Click **SLA - Account creation (1.1) consumption of eligibility service**.
3. Click **Activate SLA**. Note that the new governance state is SLA Active.

Archived

Figure 15-16 shows the status of the WSRR entities after the new service level agreement is created.

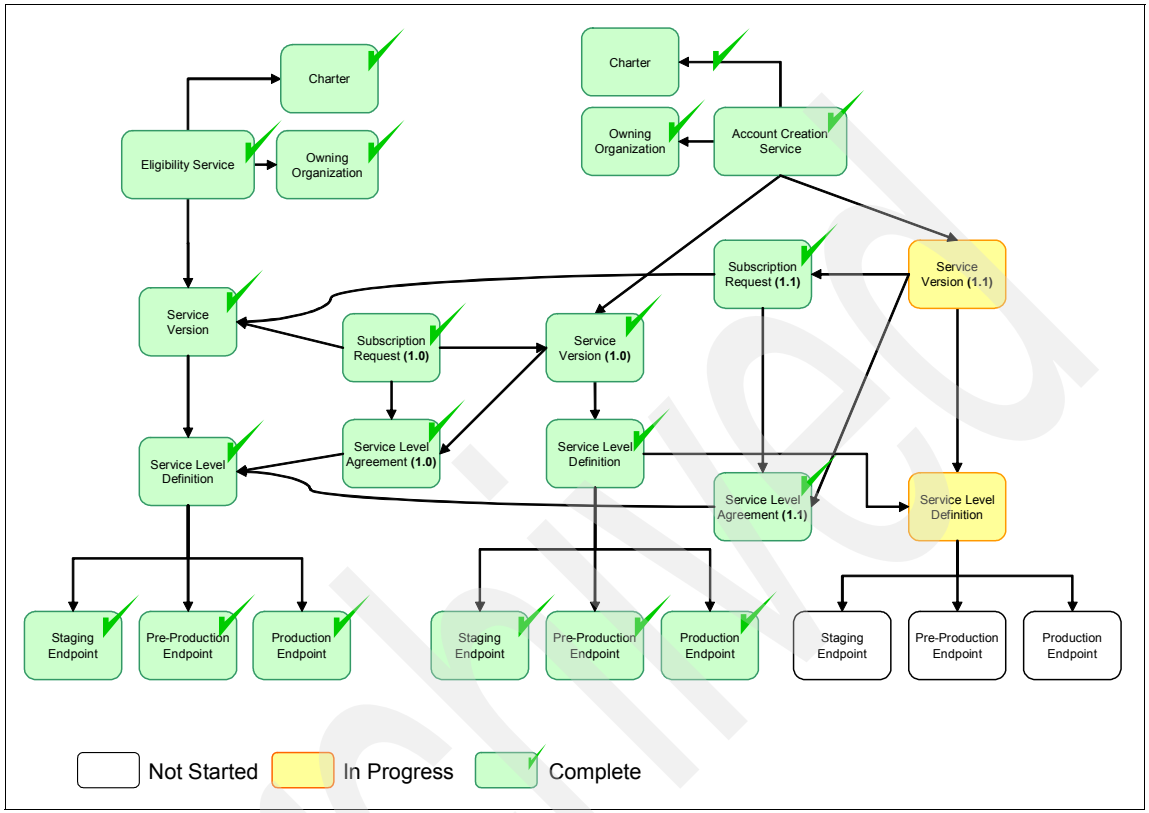


Figure 15-16 Status after creation of a service level agreement

Archived

15.7 Deploying the upgrade version to staging

Having created and unit tested the upgraded implementation of the service, Development is ready to pass the service to the Operations team for deployment to a staging environment for testing. Figure 15-17 illustrates the process for deploying a service version. We describe this process in this section.

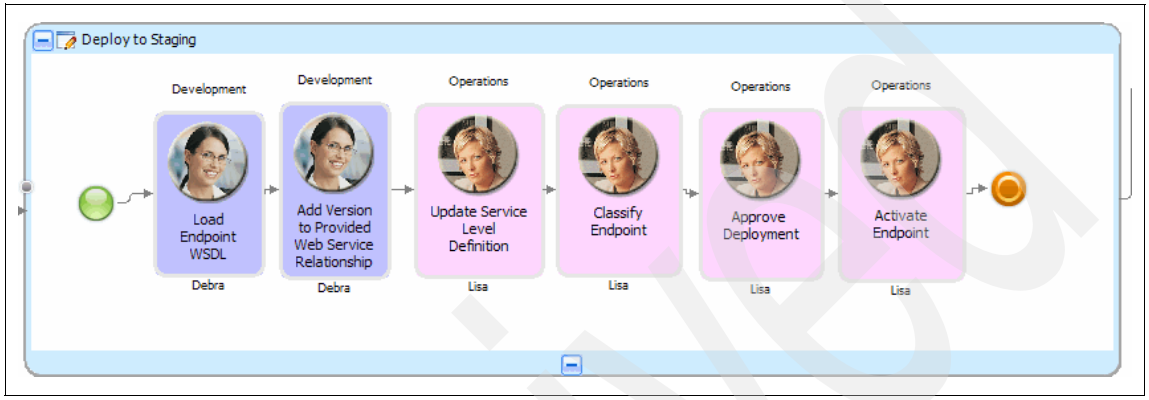


Figure 15-17 Deploying a service version process

15.7.1 Loading the endpoint WSDL

The service implementation WSDL for the minor upgrade is loaded into the WSRR. Debra, the Development Manager for the commercial line of business area, is responsible for this phase of the scenario.

To load the V1_1 endpoint WSDL, log on to WSRR, and select the **Development** perspective. Then, follow these steps:

1. Click **Actions** → **Load Documents**.
2. Ensure that the Document type is set to **WSDL**.
3. Click **Browse**, and navigate to the directory where the V1_1 endpoint WSDL is located.
4. Select **AccountCreationV1_1_StagingPort.wsdl**, click **Open**, and then click **OK**.
5. Click **Finish** to load the WSDL document.

Archived

15.7.2 Adding a relationship to the provided Web service

Next, create a relationship between the service version and the provided Web service as follows:

1. Click **Tasks** → **Capability Version Tasks** → **Version Realization**.
2. Click A**ccount creation service V1_1** to display the service version details.
3. Click **Edit Relationships**.
4. Click **Add Service** to the right of the **Provided Web Services** relationship.
5. Enter `A` in the Name field, select **AccountCreationV1_1** from the auto suggest list, and click **Add**. The endpoint is added as a target of the Provided Web Services relationship.
6. Click **Finish** to save your changes.

15.7.3 Updating the service level definition

Having loaded the endpoint into the WSRR, the final step in making the upgraded service implementation consumable in the staging environment is to associate the staging endpoint with the service level definition.

The Operations team has the responsibility of updating the service level definition. Lisa, the Operations release manager for commercial line of business area, approves the service level definition.

To update the service level definition, log on to WSRR, and select the **Operations** perspective. Then, follow these steps:

1. Click **Tasks** → **Service Level Definition Tasks** → **Manage Subscribable Service Level Definitions**.
2. Click **SLD - Account creation service V1_1**.
3. Click **Edit Relationships**.
4. Click **Add Service Endpoint** to the right of the **Available Endpoints** relationship.
5. Enter an asterisk (*) in the Name field, select **https://localhost:9443/AccountCreationV1_1/services/AccountCreationServiceV1_1_StagingPort** from the auto suggest list, and click **Add**. The service endpoint is added as a target of the Available Endpoints relationship.

6. Click **Add Service Port** to the right of the **Bound Web Service Port** relationship.
7. Enter `A` in the Name field, select **AccountCreationServiceV1_1_StagingPort** from the auto suggest list, and click **Add**. The service port is added as a target of the Bound Web Service Port relationship.
8. Click **Finish** to save your changes.

The service level definition relationships is updated as shown in Figure 15-18.

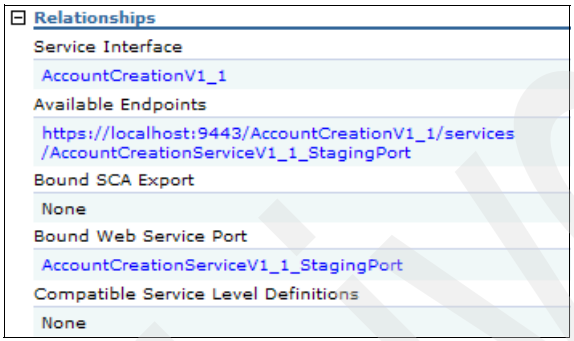


Figure 15-18 Version 1_1 SLD relationships

15.7.4 Classifying the endpoint

The Operations team approves the staging endpoint, which verifies that the service is running but not necessarily functioning correctly.

To classify the endpoint, log on to WSRR, and select the Operations perspective. Then, follow these steps:

1. Click **Tasks** → **Endpoints** → **Endpoints For Activation**.
2. Click the offline endpoint **https://localhost:9443/AccountCreationV1_1/services/AccountCreationServiceV1_1_StagingPort**.
3. Click **Edit Classifications**.
4. Expand **Governance Profile Taxonomy** → **Environment**.
5. Select **Staging**, and click **Add**. The Staging classification is added to the Classification list.
6. Click **OK** to assign the classification.

Archived

15.7.5 Approving staging deployment

At this point the service is released officially for use in the staging environment as follows:

1. Click **Tasks** → **Capability Version Tasks** → **Define SLD & prepare for staging**.
2. Click Account creation service (1.1) to display the service version details.
3. Click **Approve Staging Deployment**. Note that the new governance state is Staged.

15.7.6 Activating the staging endpoint

The Operations team approves the new staging endpoint, which verifies that the service is running, but not necessarily functioning correctly. To activate the endpoint:

1. Click **Tasks** → **Endpoints** → **Endpoints For Activation**.
2. Click the offline endpoint **https://localhost:9443/AccountCreationV1_1/services/AccountCreationServiceV1_1_StagingPort**.
3. Click **Approve For Use**. Note that the new governance state is Online.

The new account creation service is verified and promoted from the staging environment to the pre-production environment before final deployment to production. See 15.8, “Deploying the upgrade version to pre-production” on page 573 for information about deploying the service to these other environments.

Archived

Figure 15-19 shows the status of the WSRR entities after the staging environment endpoint is available.

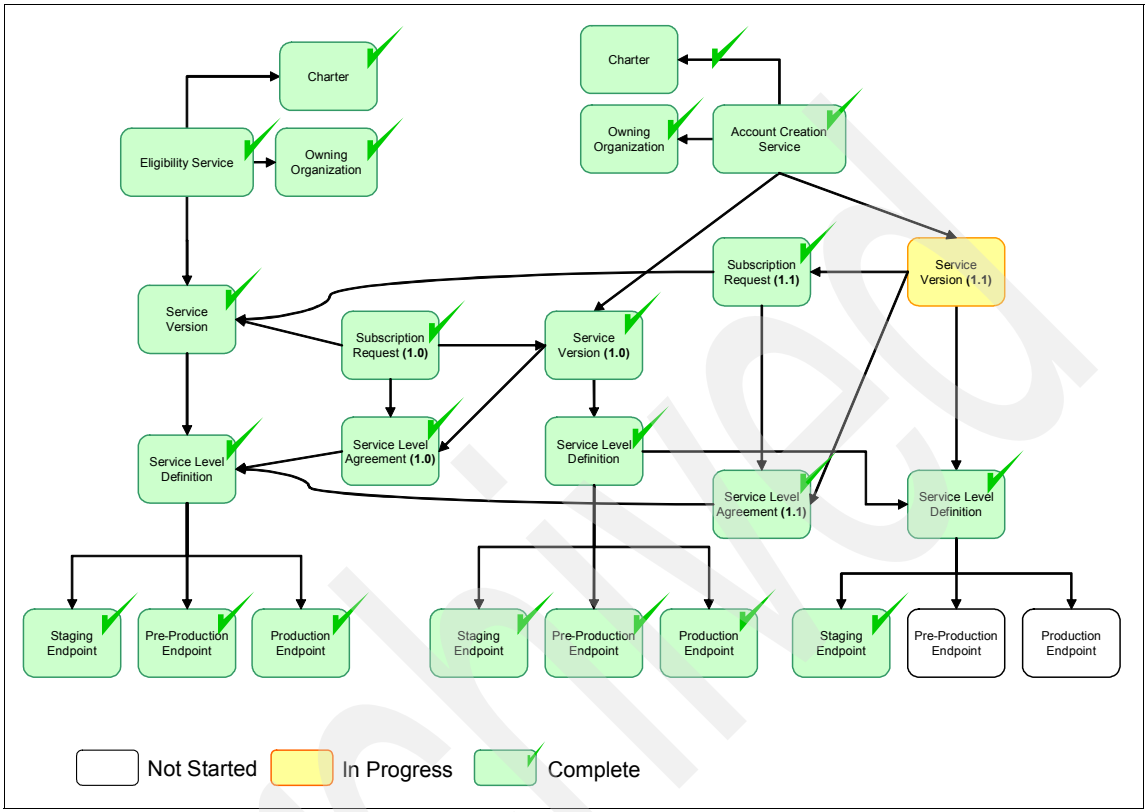


Figure 15-19 Status after deployment to the staging environment

Archived

15.8 Deploying the upgrade version to pre-production

After the system testing completes successfully, the service is deployed to the pre-production environment in a similar manner, and its state becomes Certified. Figure 15-20 illustrates the process for deploying a service version to pre-production. We describe this process in this section.

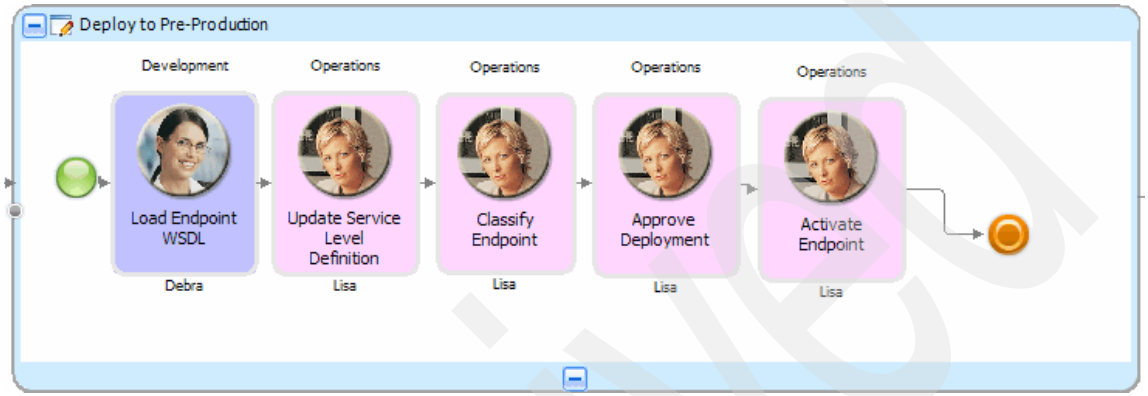


Figure 15-20 Deploying a service version to pre-production process

15.8.1 Loading the pre-production endpoint WSDL

The service implementation WSDL is loaded into the WSRR. Debra, the Development Manager for the commercial line of business area, is responsible for this phase of the scenario.

To load the pre-production endpoint WSDL, log on to WSRR, and select the **Development** perspective. Then, follow these steps:

1. Click **Actions** → **Load Documents**.
2. Ensure that the Document type is set to **WSDL**.
3. Click **Browse**, and navigate to the directory where the pre-production WSDL is located
4. Select **AccountCreationV1_0_PreProductionPort.wsdl**, click **Open**, and then click **OK**.
5. Click **Finish** to load the WSDL document.

Archived

15.8.2 Updating the service level definition

Having loaded the endpoint into the WSRR, the final step in making the service consumable in the pre-production environment is to associate the pre-production endpoint with the service level definition.

The Operations team has the responsibility of updating the service level definition. Lisa, the Operations Release Manager for commercial line of business area, approves the service level definition.

To update the service level definition, log on to WSRR, and select the **Operations** perspective. Then, follow these steps:

1. Click **Tasks** → **Service Level Definition Tasks** → **Manage Subscribable Service Level Definitions**.
2. Click **SLD - Account creation service V1_1**.
3. Click **Edit Relationships**.
4. Click **Add Service Endpoint** to the right of the **Available Endpoints** relationship.
5. Enter an asterisk (*) in the Name field, select **https://localhost:9443/AccountCreationV1_1/services/AccountCreationServiceV1_1_PreProductionPort** from the auto suggest list, and click **Add**. The service endpoint is added as a target of the Available Endpoints relationship.
6. Click **Add Service Port** to the right of the **Bound Web Service Port** relationship.
7. Enter `A` in the Name field, select **AccountCreationServiceV1_1_PreProductionPort** from the auto suggest list, and click **Add**. The service port is added as a target of the Bound Web Service Port relationship.
8. Click **Finish** to save your changes.

Archived

The SLD relationships are updated to include the Pre Production endpoint information as shown in Figure 15-21.

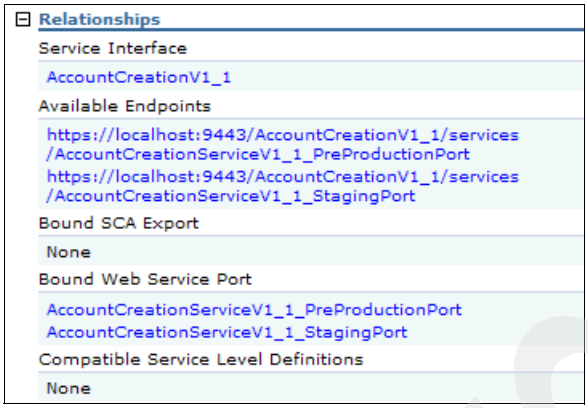


Figure 15-21 Service level definition with pre-production endpoints

15.8.3 Classifying the endpoint

The Operations team approves the pre-production endpoint. To classify the endpoint, log on to WSRR, and select the **Operations** perspective. Then, follow these steps:

1. Click **Tasks** → **Endpoint Tasks** → **Endpoints For Activation.**
2. Click the development offline endpoint **https://localhost:9443/AccountCreationV1_1/services/AccountCreationServiceV1_1_PreProductionPort**.
3. Click **Edit Classifications**.
4. Expand **Governance Profile Taxonomy** → **Environment**.
5. Select **Pre-Production**, and click **Add**. The Pre-Production classification is added to the Classification list.
6. Click **OK** to assign the classification.

15.8.4 Approving pre-production deployment

At this point, the service is released officially for use in the pre-production environment as follows:

1. Click **Tasks** → **Capability Version Tasks** → **Prepare for Pre-Production**.
2. Click **Account creation service (1.1)** to display the service version details.
3. Click **Approve Certification**. Note that the new governance state is Certified.

Archived

15.8.5 Activating the endpoint

Although the service is promoted to the pre-production environment and approved for use, the endpoint needs to be activated, which updates its status to Online. To activate the endpoint:

1. Click **Tasks** → **Endpoints** → **Endpoints For Activation**.
2. Click the offline endpoint **https://localhost:9443/AccountCreationV1_1/services/AccountCreationServiceV1_1_PreProductionPort**.
3. Click **Approve For Use**. Note that the new governance state is Online.

The account creation service is promoted to the pre-production environment.

In the pre-production environment, final acceptance testing of the account creation service is carried out to ensure that if it is made available in production it meets the proposed service level definitions and service level agreements.

Archived

Figure 15-22 shows the status of the WSRR entities after the pre-production environment endpoint is available.

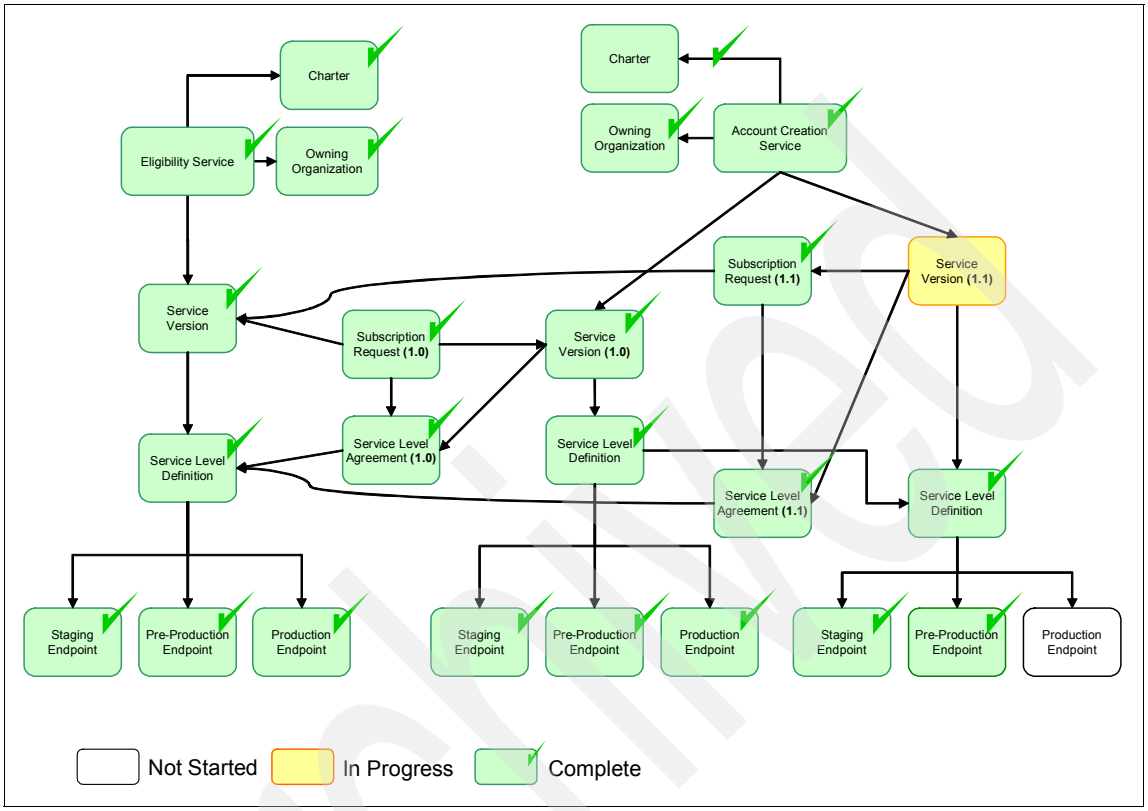


Figure 15-22 Status after deployment to the pre-production environment

Archived

15.9 Deploying the upgrade version to production

After final testing and verification completes successfully, the service is deployed to the production environment, and its state becomes Operational. Figure 15-23 illustrates the process for deploying a service version to production. We describe this process in this section.

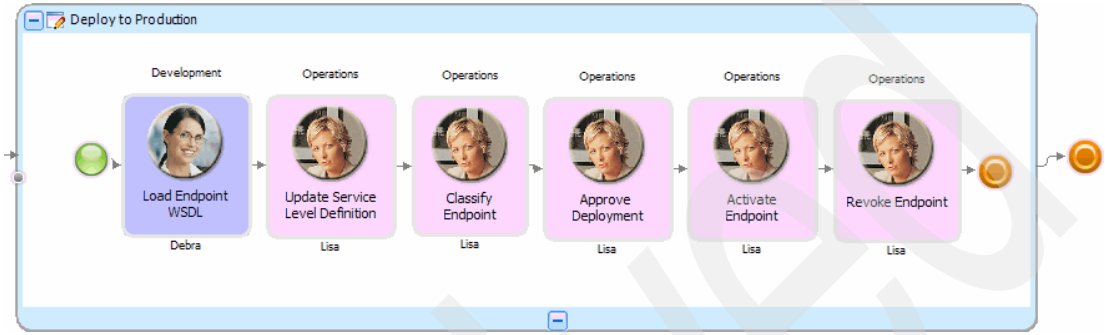


Figure 15-23 Deploying a service version to production process

15.9.1 Loading the production endpoint WSDL

The service implementation WSDL is loaded into the WSRR. Debra, the Development Manager for the commercial line of business area, is responsible for this phase of the scenario.

To load the production endpoint WSDL, log on to WSRR, and select the **Development** perspective. Then, follow these steps:

1. Click **Actions** → **Load Documents**.
2. Ensure that the Document type is set to **WSDL**.
3. Click **Browse**, and navigate to the directory where the production WSDL is located.
4. Select **AccountCreationV1_1_ProductionPort.wsdl**, click **Open**, and then click **OK**.
5. Click **Finish** to load the WSDL document.

The production endpoint WSDL is loaded.

Archived

15.9.2 Updating the service level definition

Having loaded the endpoint into the WSRR, the final step in making the service consumable in the production environment is to associate the production endpoint with the service level definition.

The Operations team has the responsibility of updating the service level definition. Lisa, the Operations Release Manager for the commercial line of business area, approves the service level definition.

To update the service level definition, log on to WSRR, and select the **Operations** perspective. Then, follow these steps:

1. Click **Tasks** → **Service Level Definition Tasks** → **Manage Subscribable Service Level Definitions**.
2. Click **SLD - Account creation service**.
3. Click **Edit Relationships.**
4. Click **Add Service Endpoint** to the right of the **Available Endpoints** relationship.
5. Enter an asterisk (*) in the Name field, select **https://localhost:9443/AccountCreationV1_1/services/AccountCreationServiceV1_1_ProductionPort** from the auto suggest list, and click **Add**. The service endpoint is added as a target of the Available Endpoints relationship.
6. Click **Add Service Port** to the right of the **Bound Web Service Port** relationship.
7. Enter `A` in the Name field, select **AccountCreationServiceV1_1_ProductionPort** from the auto suggest list, and click **Add**. The service port is added as a target of the Bound Web Service Port relationship.
8. Click **Finish** to save your changes.

Archived

The SLD relationships are updated to include the Production endpoint information as shown in Figure 15-24.

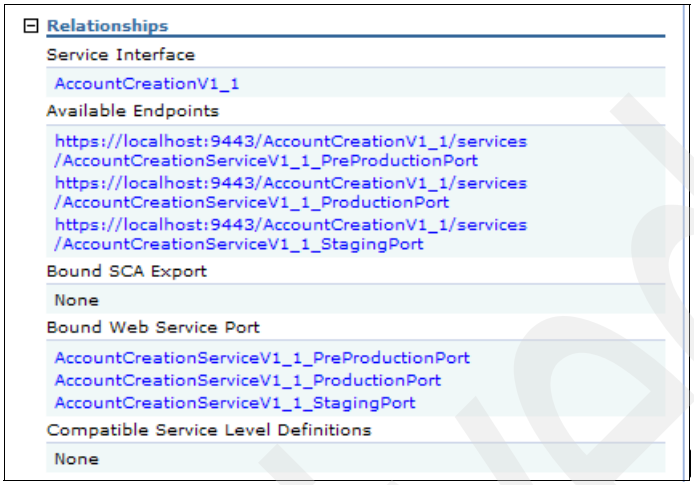


Figure 15-24 Service level definition with production endpoints

15.9.3 Classifying the endpoint

The Operations team approves the production endpoint. To classify the endpoint, log on to WSRR, and select the **Operations** perspective. Then, follow these steps:

1. Click **Tasks** → **Endpoint Tasks** → **Endpoints For Activation.**
2. Click the development offline endpoint **https://localhost:9443/AccountCreationV1_1/services/AccountCreationServiceV1_1_ProductionPort**.
3. Click **Edit Classifications.**
4. Expand **Governance Profile Taxonomy** → **Environment**.
5. Select **Production**, and click **Add**. The Production classification is added to the Classification list.
6. Click **OK** to assign the classification.

15.9.4 Approving production deployment

At this point, the service is ready to be officially released as follows:

1. Click **Tasks** → **Capability Version Tasks** → **Prepare for Production**.
2. Click **Account creation service (1.1)** to display the service version details.

3. Click **Approve Production Deployment**. Note that the new governance state is Operational.

15.9.5 Activating the endpoint

Although the service is promoted to the production environment and approved for use, the endpoint needs to be activated, which updates its status to Online. To activate the endpoint:

1. Click **Tasks** → **Endpoint Tasks** → **Endpoints For Activation.**
2. Click the offline endpoint **https://localhost:9443/AccountCreationV1_1/services/AccountCreationServiceV1_1_ProductionPort.**
3. Click **Approve For Use**. Note that the new governance state is Online.

The account creation service is now promoted to the production environment.

Archived

Figure 15-25 shows the status of the WSRR entities after the production environment endpoint is available.

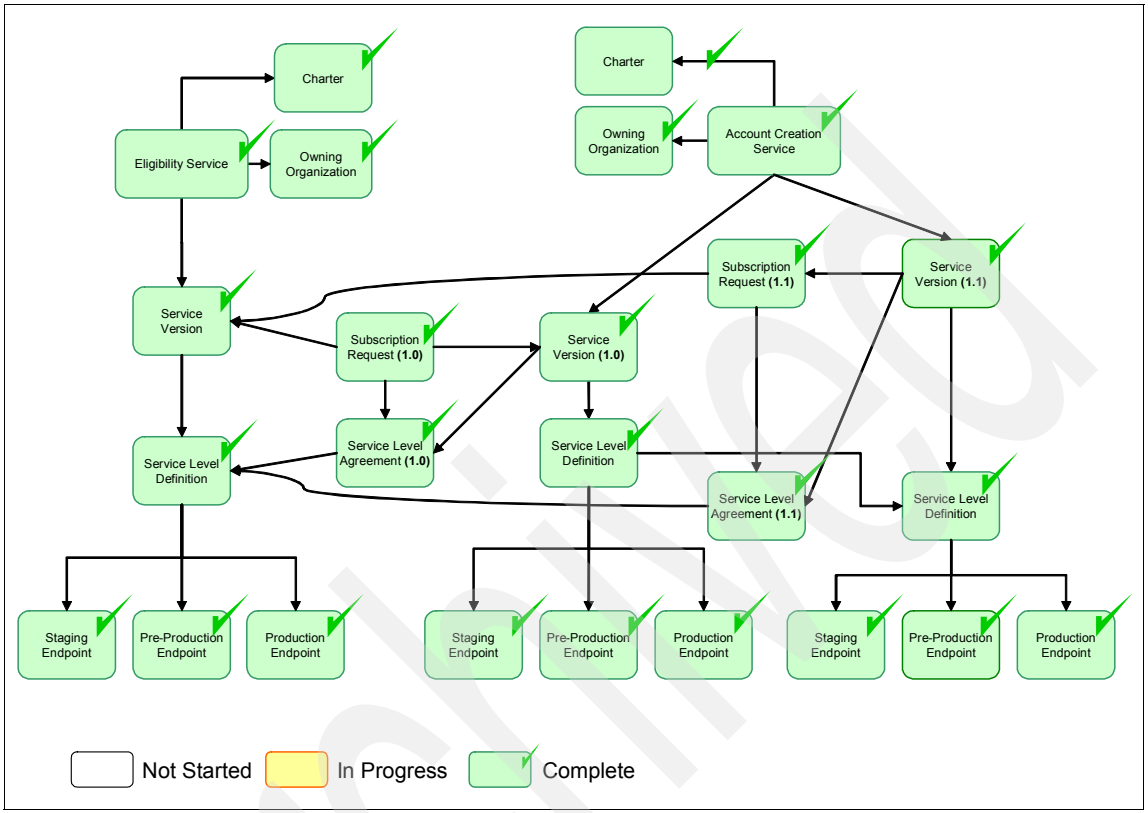


Figure 15-25 Status after deploying to the production environment

15.9.6 Superceding the v1.0 service version

Because the 1.1 service version of the Account Creation Service is now in production, the v1.0 version can be transitioned to the superceeded state, which is the responsibility of the Operations team. Lisa, the Operations Release Manager for the commercial line of business area is responsible for this process.

To supercede the service version, log on to WSRR, and select the **Operations** perspective. Then, follow these steps:

1. Click **View** → **Technical Governance** → **Capability Versions** → **Service Versions**.
2. Click **Account creation service (1.0)**.

3. Click **Supercede** to transition the new service version to a governance state of Superceded.

15.9.7 Revoking the V1_0 staging endpoint

The Operations team has the responsibility of revoking the v1.0 endpoints. They first run a report to check that there are no consumers of the v1.0 version of the service before bringing the endpoints offline. Lisa, the Operations Release Manager for the commercial line of business area, revokes the v1.0 staging endpoint of the Account Creation Service.

For more information about reporting, refer to Chapter 10, “Reports” on page 349.

> **Note:** It is recommended that you do not bring the test endpoints for the v1.0 version of the service offline until the v1.1 version is fully in production. This method provides the capability to cope with the scenario whereby a bug fix needs to be tested on the v1.0 version of the service before the v1.1 version is tested and is available in production.

To revoke the V1_0 staging endpoint, log on to WSRR, and select the **Operations** perspective. Then, follow these steps:

1. Click **Tasks** → **Endpoints** → **Manage Online Endpoints** as shown in Figure 15-26.

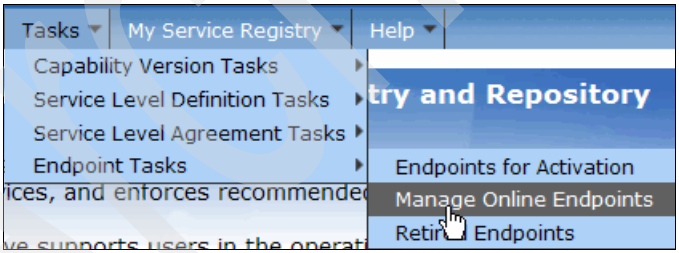


Figure 15-26 Navigating to online endpoints

2. Click the V1_0 staging endpoint **https://localhost:9443/AccountCreationV1_0/services/AccountCreationServiceV1_0_StagingPort**.

3. Click **Revoke From Use**. Note that the new governance state is Offline as shown in Figure 15-27.

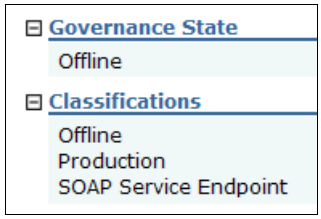


Figure 15-27 SOAP service endpoint offline governance state

Figure 15-28 shows the life cycle for an endpoint.

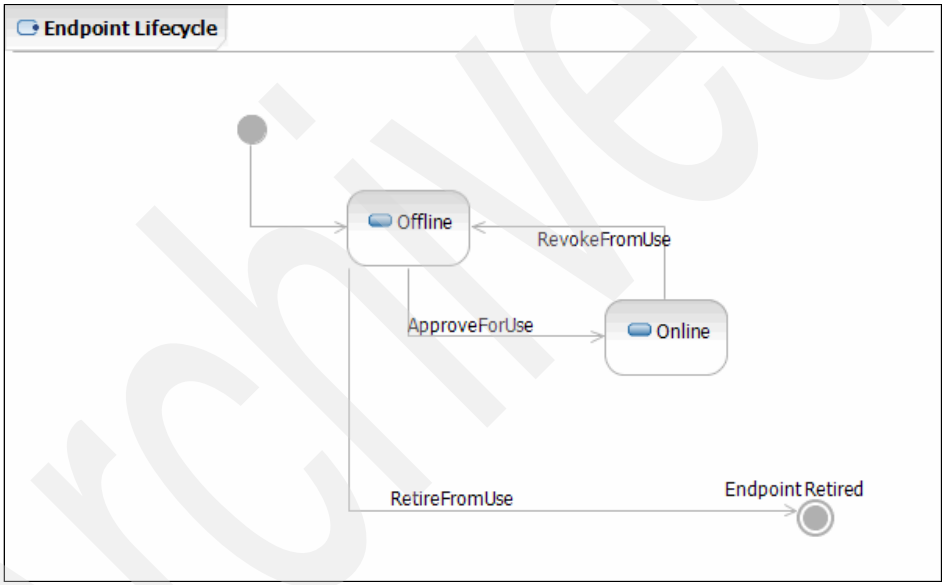


Figure 15-28 Endpoint life cycle

Archived

Figure 15-29 shows the status of the WSRR entities after the staging endpoint is brought offline.

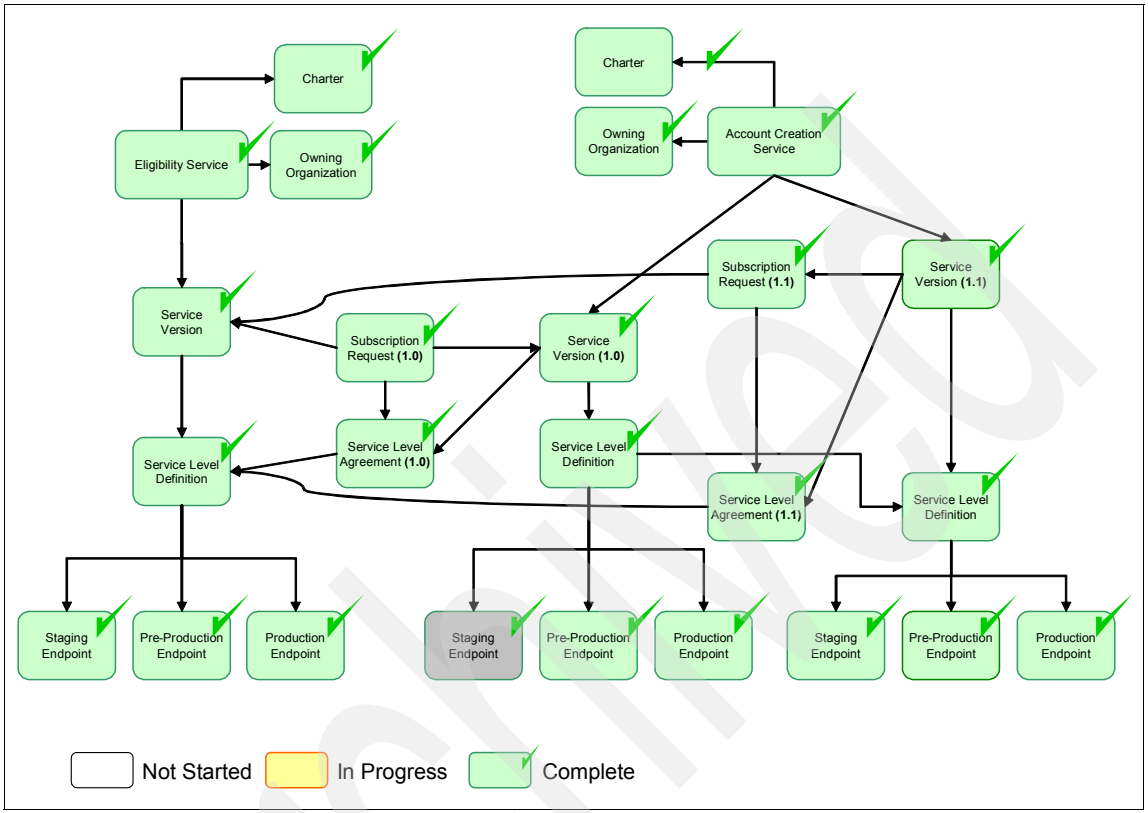


Figure 15-29 Status after V1.0 staging endpoint has been brought offline

15.9.8 Revoking the V1_0 pre-production endpoint

The Operations team can now take V1_0 of the account creation service offline in the pre-production environment.

To revoke the V1_0 pre-production endpoint, log on to WSRR, and select the **Operations** perspective. Then, follow these steps:

1. Click **Tasks** → **Endpoints** → **Manage Online Endpoints**.
2. Click the V1_0 pre-production endpoint **https://localhost:9443/AccountCreationV1_0/services/AccountCreationServiceV1_0_PreProductionPort**.
3. Click **Revoke From Use**. Note that the new governance state is Offline.

Figure 15-30 shows the status of the WSRR entities after the pre-production endpoint is brought offline.

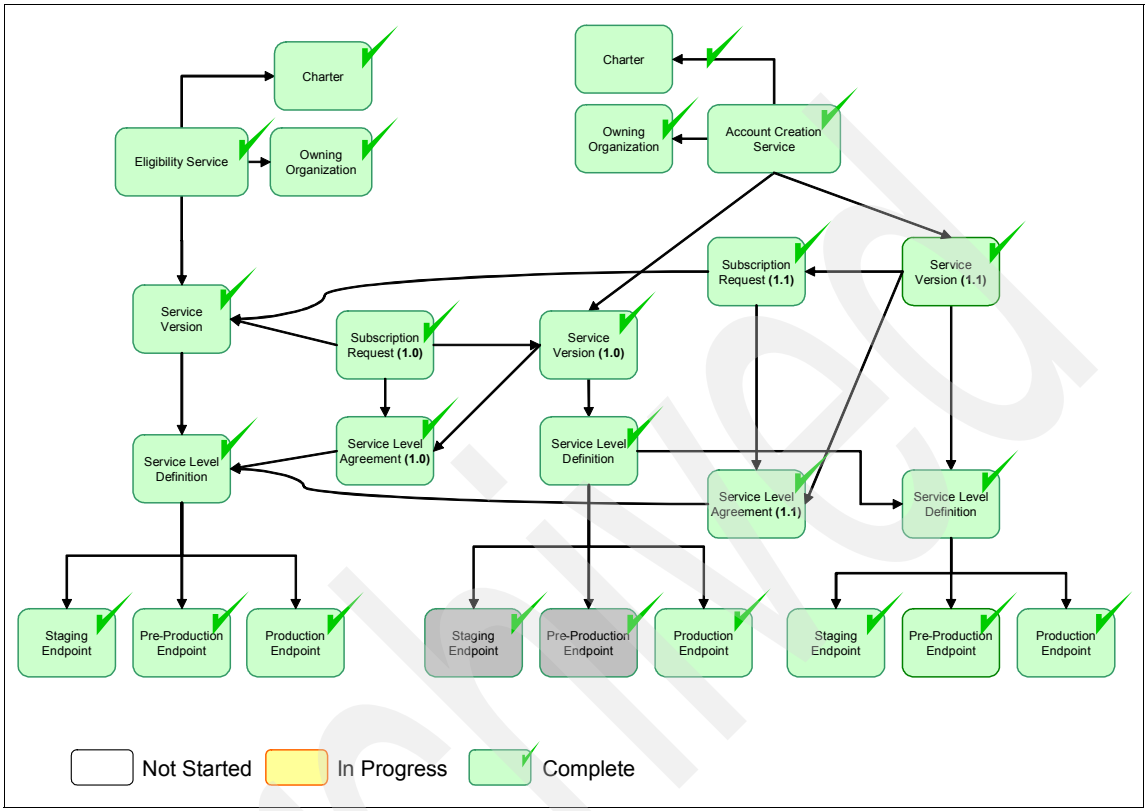


Figure 15-30 Status after the pre-production endpoint has been brought offline

15.9.9 Revoking the V1_0 production endpoint

The Operations team can now take V1_0 of the account creation service offline in the production environment. To revoke the V1_0 production endpoint, log on to WSRR, and select the **Operations** perspective. Then, follow these steps:

1. Click **Tasks** → **Endpoints** → **Manage Online Endpoints**.
2. Click the V1_0 pre-production endpoint **https://localhost:9443/AccountCreationV1_0/services/AccountCreationServiceV1_0_ProductionPort**.
3. Click **Revoke From Use**. Note that the new governance state is Offline.

Figure 15-31 shows the status of the WSRR entities after the production endpoint is brought offline.

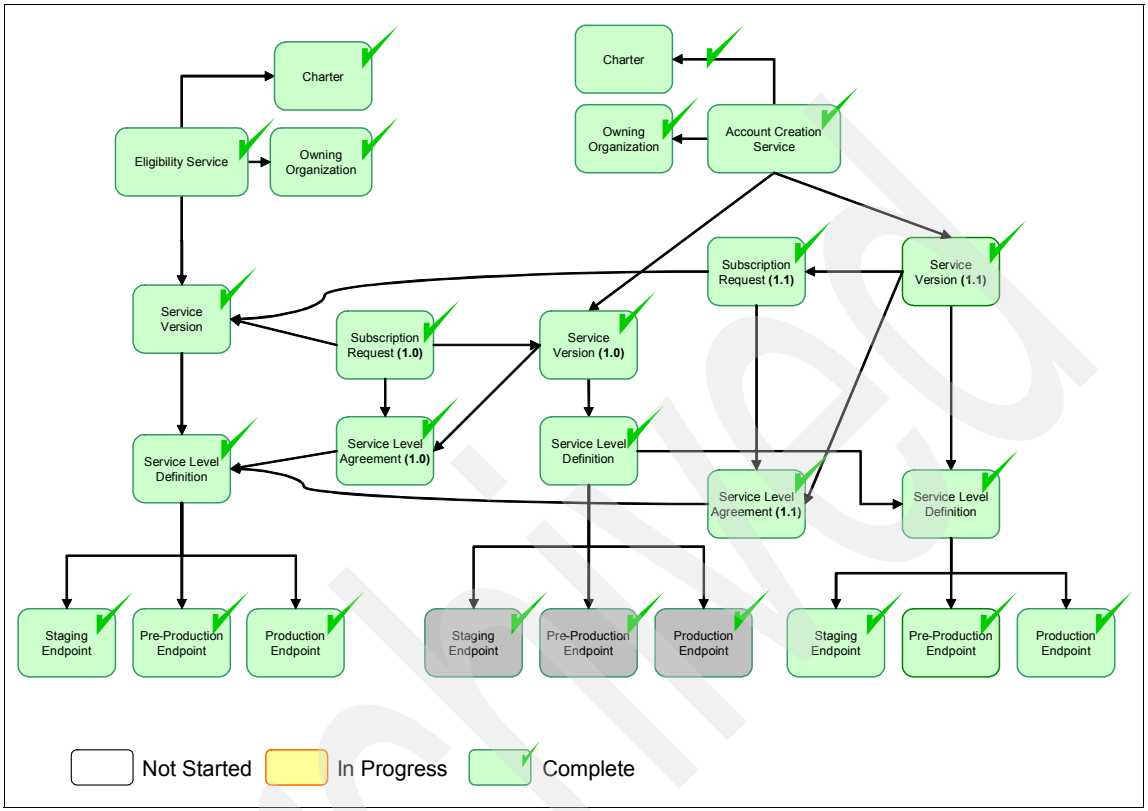


Figure 15-31 Status after the staging endpoint has been brought offline

Archived

Archived

16

Governing a business process

This chapter provides step-by-step instructions for governing a new business process.

Note: In the examples in this book, we use a case study about a fictional company named *JKHL Enterprises* (JKHLE). For information about this case study, see Chapter 4, “JKHL Enterprises case study” on page 153. For information about the roles used throughout this scenario refer to 3.3, “Roles in the governance enablement profile” on page 103.

This chapter includes the following topics:

- Governing a new business process
- Creating a business capability
- Reviewing and approving business process
- Scoping a process version
- Creating and approving a subscription request
- Planning a process version

© Copyright IBM Corp. 2009. All rights reserved.

16.1 Governing a new business process

In this scenario, we discuss the process for creating and governing a new business process at JKHLE.

Figure 16-1 illustrates the main steps that are required to complete this process. We describe only these steps in this chapter.

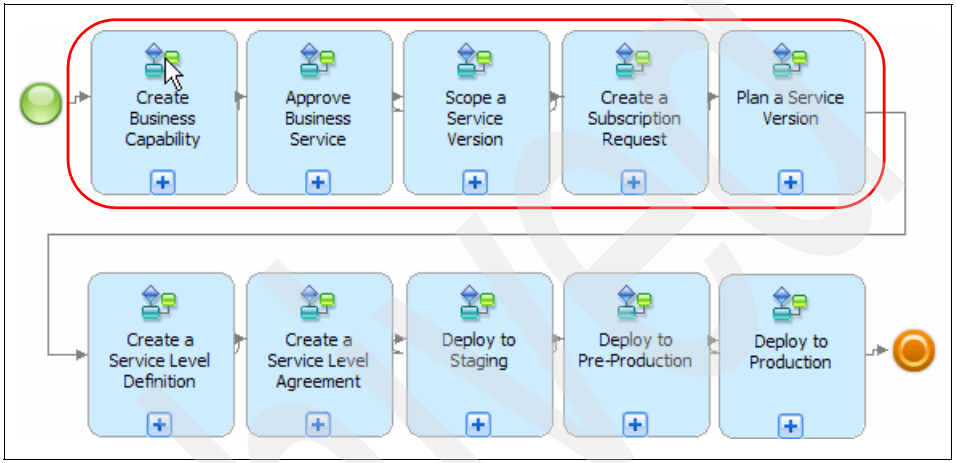


Figure 16-1 Governing a new business process steps

The business process is implemented as an SOA service using WS-BPEL, which exposes a WSDL interface. The new business process uses the account creation service to create customer account records automatically.

In WebSphere Service Registry and Repository (WSRR), the business process and version are defined in a similar manner to the service scenario that we describe in Chapter 14, “Governing a service that reuses an existing service” on page 475.

The subsequent steps are identical to those for governing a new service.

Figure 16-2 highlights the major entities that we create in WSRR throughout this process.

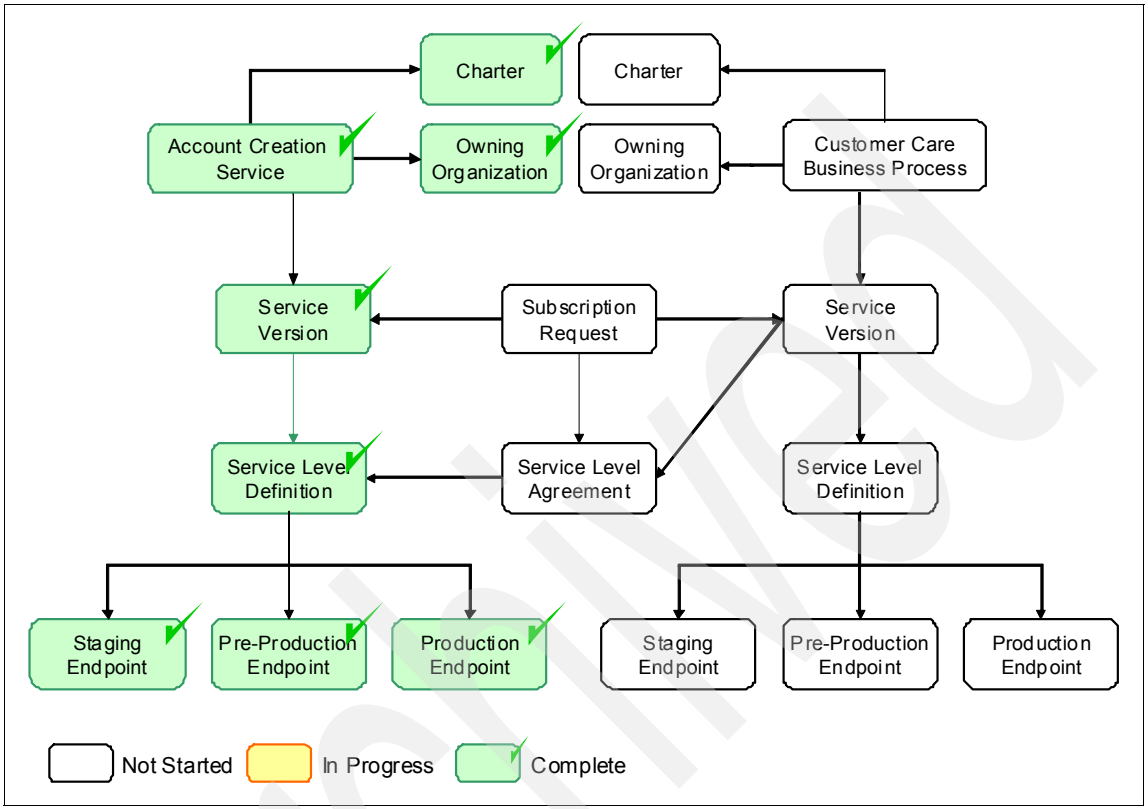


Figure 16-2 WSRR entities for business process governance

Archived

16.2 Creating a business capability

The first phase in governing a new business process is creating the business capability as illustrated in Figure 16-3. We describe this process in this section.

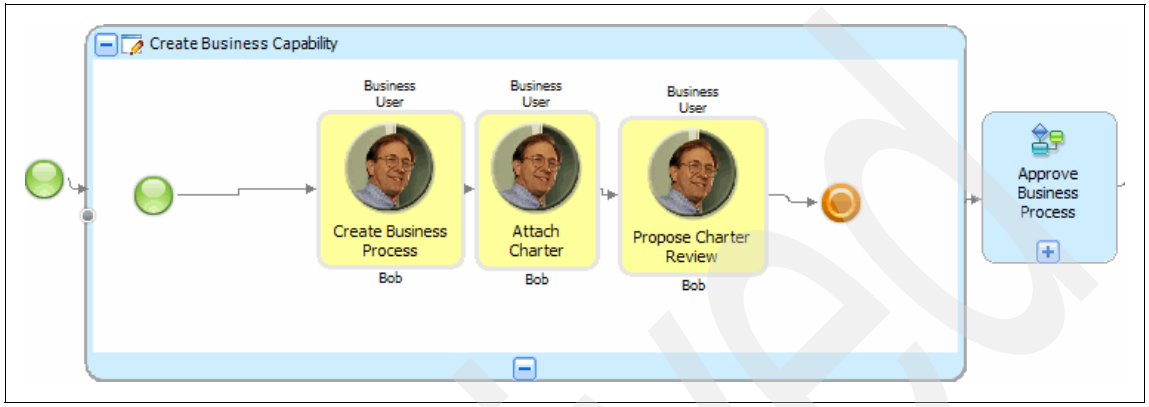


Figure 16-3 Creating a business capability

16.2.1 Creating a new business capability

Bob, the Business Analyst for the commercial line of business at JKHLE, is tasked with defining a new process for establishing a customer. This process is invoked from the Customer Care Web front end. It creates account records for existing customers automatically by invoking the account creation service.

To create the new business capability, log on to WSRR, and select the **Business** perspective. Then, follow these steps:

1. Click **Actions** → **Create** → **Business Process** as shown in Figure 16-4.

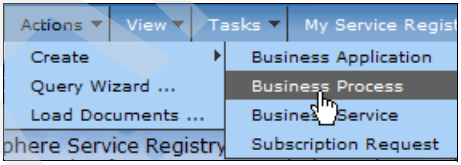


Figure 16-4 Creating a business process

Archived

2. Enter the following information:
   - Name: `Customer Care business process`
   - Description: `Automated process for identifying a customer and creating their new account.`
3. Click **Finish**. The details page for the new business process displays.

The governance state of the new business process is Business Capability Identified as shown in Figure 16-5.

Figure 16-5 Business capability identified governance state

16.2.2 Attaching a charter

The *charter* is a separate document that describes the required capability in detail, including all functional and non-functional requirements. The charter is authored externally and then loaded into WSRR and attached to the business process.

You need to load the document that contains the charter and attach it to the business process using the Charter relationship as follows:

1. Click **Edit Relationships**.
2. Click **Add Other Document** alongside the **Charter** relationship.
3. Click **Load Document**.
4. Click **Browse**, and navigate to the directory where the charter document is located.
5. Select **CustomerCareBusinessProcessCharter.doc**, click **Open**, and then click **OK**.
6. Click **Finish** to load the charter document. The charter document is added as a target of the Charter relationship.
7. Click **Finish** to save your changes.

The charter document is loaded for the Customer Care business process.

Archived

16.2.3 Proposing the business process

The business user has created the initial definition of the process, and it is now ready for review by the SOA governance team. To indicate the its readiness for review and to make it available to the reviewers, the charter must be proposed.

To transition the business process to the Charter Review state, click **Propose for Charter Review**. Note that the new governance state is Charter Review.

The business process and charter entities shown in Figure 16-6 are now ready for review.

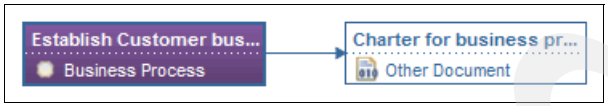


Figure 16-6 Business process and charter in graphical view

Archived

Figure 16-7 shows the status of the WSRR entities after the business capability is created.

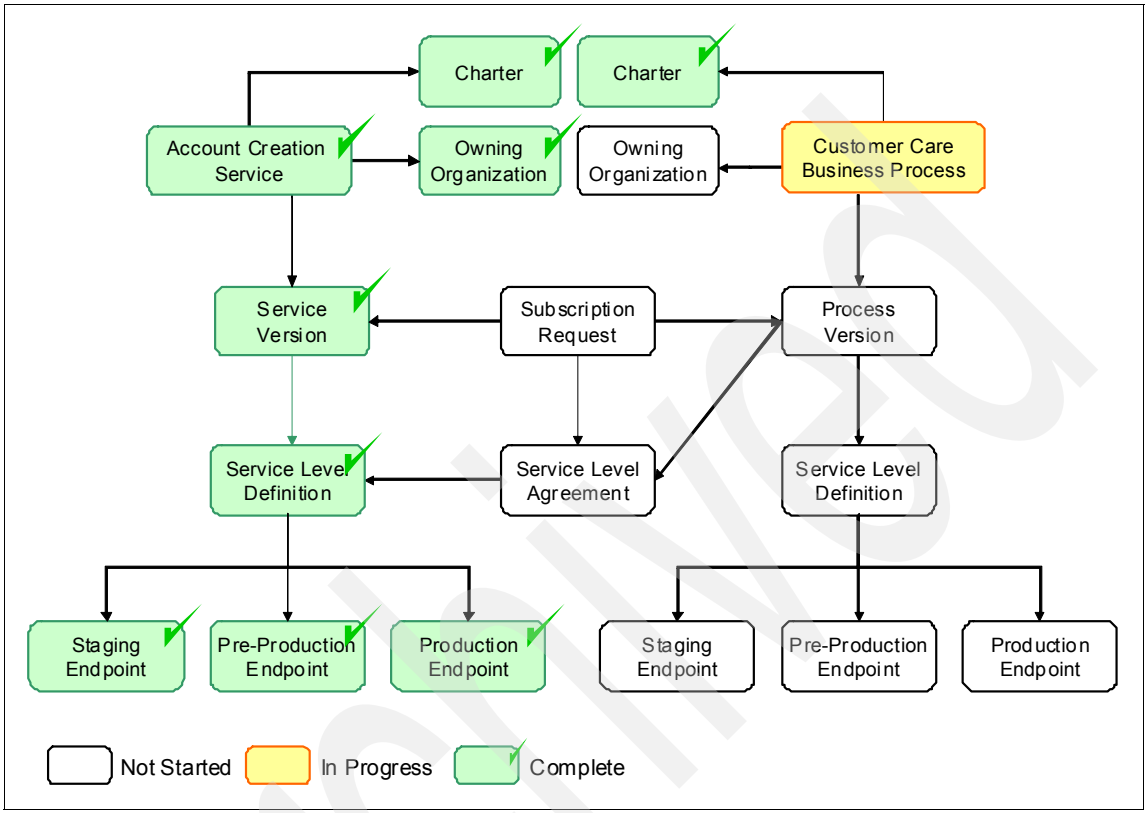


Figure 16-7 Status after business capability created

16.3 Reviewing and approving business process

The SOA governance team is responsible for ensuring that governance processes are enforced. Before the business process can be transitioned to the next stage in the life cycle, the team must ensure that the proposed capability does not duplicate other processes in the registry and that an owning organization is assigned that is responsible for all versions of this capability and for managing requirements for this process.

Archived

The next phase in the governing a new business process scenario at JKHLE is approving the business process capability as illustrated in Figure 16-8. We describe this process in this section.

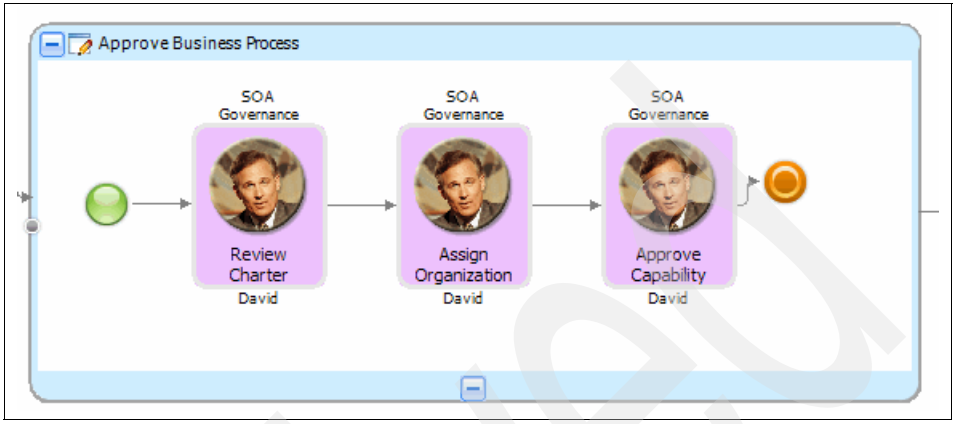


Figure 16-8 Approving the business process

16.3.1 Reviewing the new business process

David, chairman of the SOA governance team, reviews the business process definition and charter.

To review the business process definition, log on to WSRR, and select the **SOA Governance** perspective. Then, follow these steps:

1. Click **Tasks** → **Business Capability Tasks** → **Business Capabilities for Approval** to display all business capabilities in the Charter Review state.
2. Click **Customer Care business process** to display the business process details, and review the basic information in the business process definition.
3. Click **Charter for business process.doc** under the **Charter** relationship.
4. Click **Content**, then click **Download Document**.
5. Click **OK** to open the charter document and review the charter.
6. Close the charter document and return to the WSRR Web UI.

Archived

16.3.2 Assigning an owning organization to the business process

Next, you need to assign the organization that is responsible for this process as follows:

1. Click **Customer Care business process** in the breadcrumb trail to display the business process details.
2. Click **Edit Relationships**.
3. Click **Add Organization** to the right of the **Owning Organization** relationship.
4. Enter `C` in the Name field, select **Commercial** from the auto suggest list, and click **Add**. The Commercial organization is added as a target of the Owning organization relationship.
5. Click **Finish** to save your changes.

16.3.3 Approving the business process

Having ensured that all of the governance requirements are satisfied and having used the search facilities in the WSRR Web UI to confirm that the proposed capability does not duplicate other services in the registry, the SOA governance team member approves the business process.

To transition the business process to the Business Capability Approved state, Click **Approve Capability**. The new governance state is Business Capability Approved as shown in Figure 16-9.

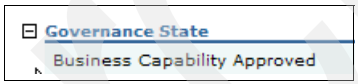


Figure 16-9 Business capability approved business process governance state

Archived

Figure 16-10 shows the status of the WSRR entities after the business process is approved.

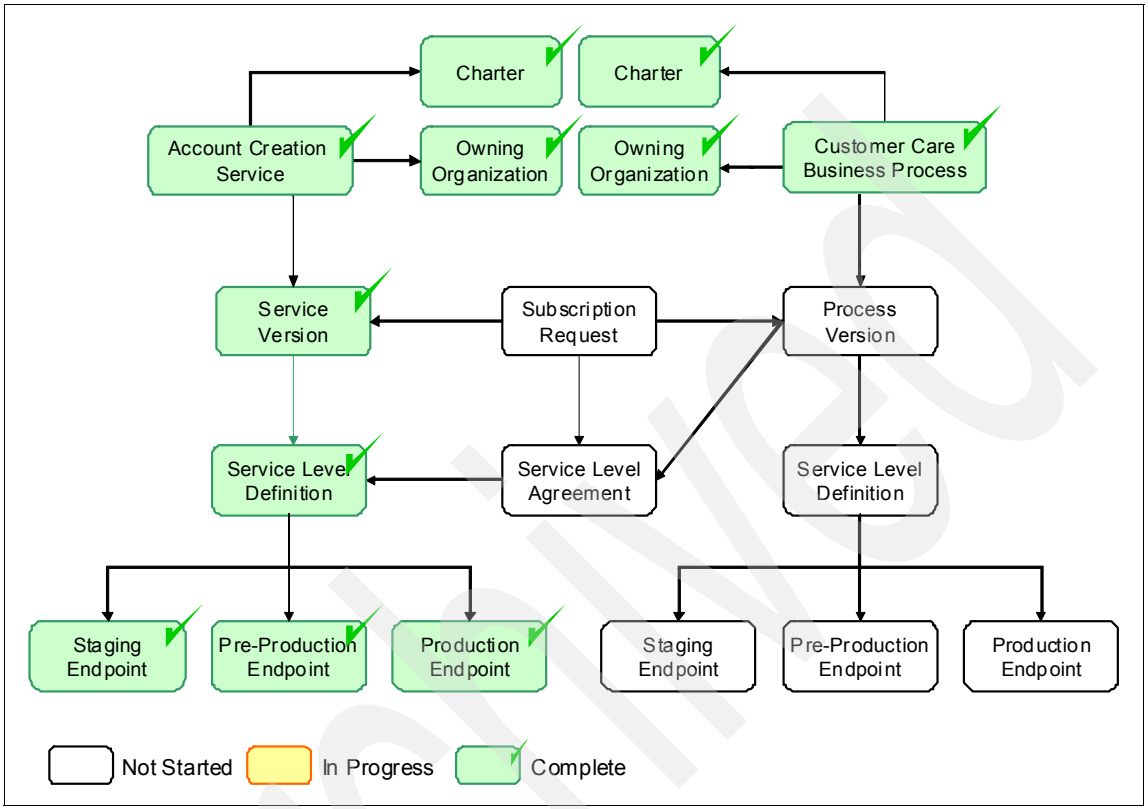


Figure 16-10 Status after business process approved

Archived

16.4 Scoping a process version

A capability version represents a specific version or release of a process and provides a range of functional and non-functional specifications for that version of the process. The next phase in the governing a new process scenario at JKHLE is scoping the process version as illustrated in Figure 16-11. We describe this process in this section.

Figure 16-11 Scoping the new process version

16.4.1 Creating a new capability version

After the business capability is defined, reviewed, and approved, a new process version is defined.

It is now the responsibility primarily of the development organization to create an implementation of the process. Debra, the Development Manager for the commercial line of business, is responsible for developing the new establish customer process for JKHLE.

To create the new capability version, log on to WSRR, and select the **Development** perspective. Then, follow these steps:

1. Click **View** → **Business Governance** → **Business Capabilities** → **Business Processes**.
2. Click **Customer Care business process** to display the business process details.

3. Assign a new capability version to the business process by clicking N**ew Capability Version**.
4. Under the relationship called **Versions**, click **Customer Care business process** to display the details of the new process version. Note that a Chartered Business Capability relationship is provided that you can use to navigate back to the business process.
5. Click **Edit Properties**.
6. Change the following property values:
   - Name: `Customer Care business process (1.0)`
   - Version: `1.0`
   - Description: `Customer Care business process`
   - Consumer Identifier: `ACSV000`
   - Version Requirements Link: `http://requirements.jkhle.com/requirements.jsp?id=8912`

     This is a link, fictitious in this case, to the relevant item in JKHLE's requirements tracking tool.
7. Click **OK** to save the changes.

The governance state is Identified as shown in Figure 16-12. This state is the initial state in the Model phase of the SOA life cycle. A new process version is entered into the SOA life cycle automatically.

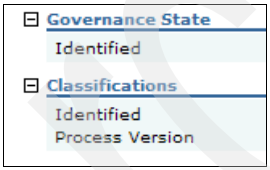


Figure 16-12 Process version identified governance state

16.4.2 Proposing the process version scope

Now that the scope of the process version is defined, you need to circulate it for review so that all potential consumers of the process can verify that the requirements are within the proposed scope by the development team.

Archived

Transition the process version to the Scope Review state by clicking **Propose Scope**. Note that the new governance state is Scope Review as shown in Figure 16-13.

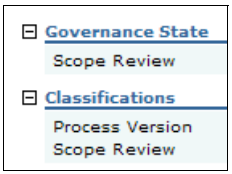


Figure 16-13 Process version scope review governance state

16.4.3 Approving the process version scope

In the Scope Review state, David from the SOA governance team reviews the process version requirements.

David carries out the following verification activities:

- Checks that this process version is warranted across the organization
- Checks that the requirements and stakeholders are agreed
- Checks that the owning organization that is responsible for delivering the requirements has been identified, and assigned to the process version

When the process version scope review is complete, the scope can be approved.

To transition the process version to the Scoped state, log on to WSRR, and select the **SOA Governance** perspective. Then, follow these steps:

1. Click **Tasks** → **Capability Version Tasks** → **Versions for Scope Review**.
2. Click **Customer Care business process (1.0)** to display the process version details.

   As part of the review activity, a member of the SOA governance team would also open the business process, by following the **Customer Care business process** link under the **Chartered Business Capability(s)** relationship, review the charter document to ensure that the requirements for this process version are clear, and then return to the process version by following the Customer **Care business process (1.0)** link under the Versions relationship.

Archived

3. Click **Approve Scope**. Note that the new governance state is Scoped as shown in Figure 16-14.

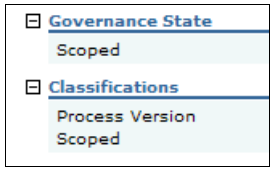


Figure 16-14 Process version scoped governance state

Figure 16-15 shows the status of the WSRR entities after the new process version is scoped.

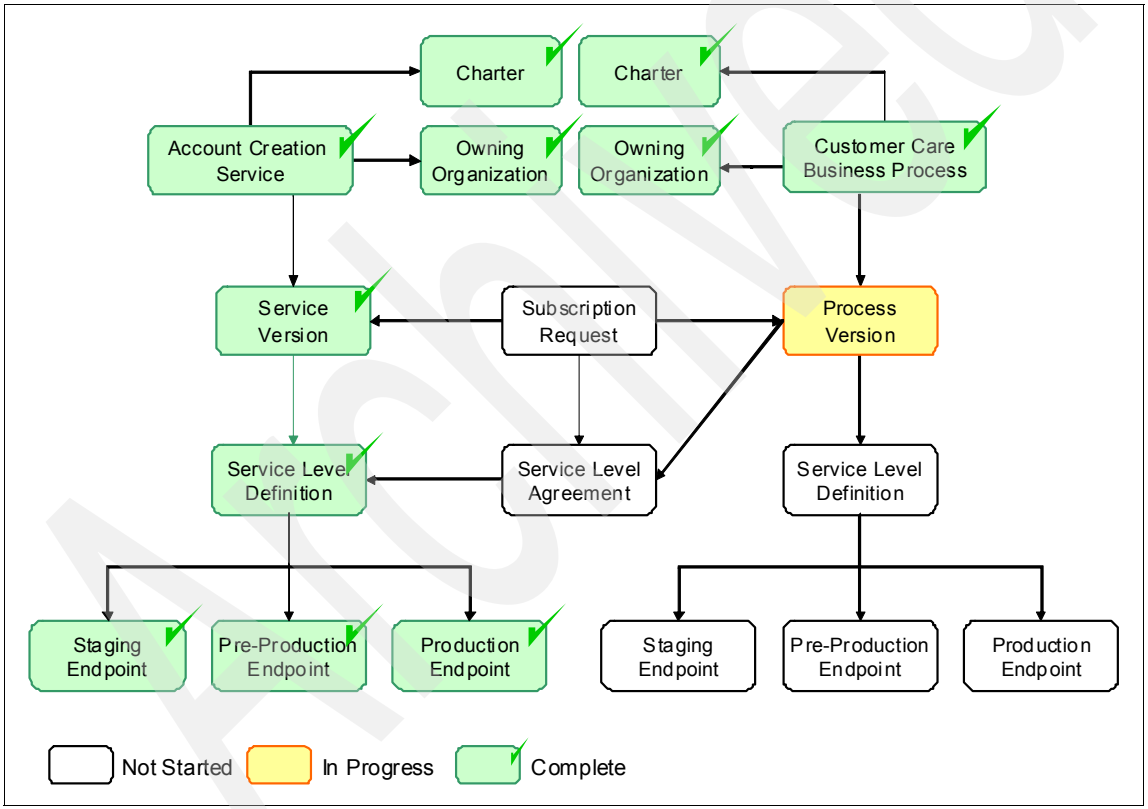


Figure 16-15 Status after process version is scoped

Archived

16.5 Creating and approving a subscription request

The new Customer Care business process makes use of the account creation service. As Development Manager for the Customer Care business process, Debra uses the JKHLE service registry and repository to identify that the existing account creation service will provide the required functionality. A subscription request is created to formalize the reuse of this service.

The next phase in the governing a new business process scenario at JKHLE is creating the subscription request as illustrated in Figure 16-16. We describe this process in this section.

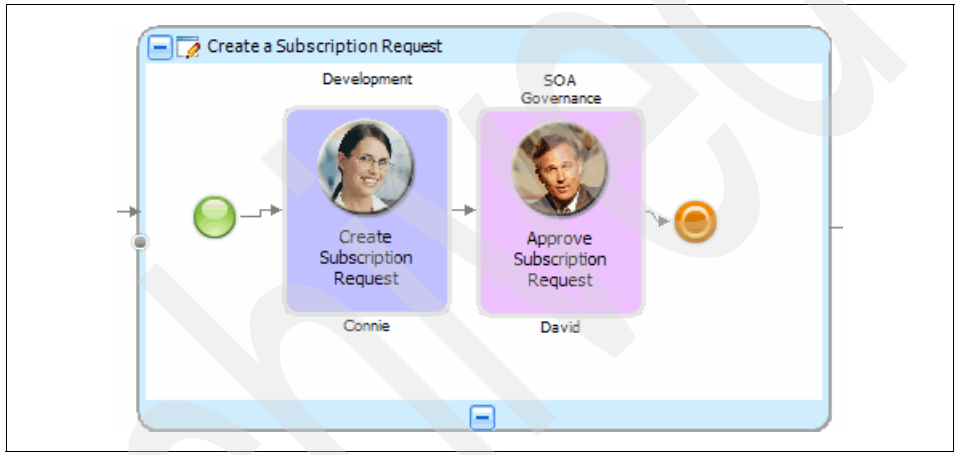


Figure 16-16 Creating the subscription request process

16.5.1 Creating a subscription request

Connie, the Development Manager for Common services, is responsible for creating the subscription request.

Archived

To create the new subscription request, log on to WSRR, and select the **Development** perspective. Then, follow these steps:

1. Click **Actions** → **Create** → **Subscription Request**.
2. Enter the following information:
   - Name: `Customer Care consumption of account creation service`
   - Requirements Link: `http://requirements.jkhl.com/requirements.jsp?id=8997`

     This is a link, fictitious in this case, to the relevant Subscription Request item in JKHLE's requirements tracking tool.
3. Click **Add Capability Version** to the right of the **Provider** relationship.
4. Enter `A` in the Name field, select **Account creation service** from the auto suggest list, and click **Add**. The service version is added as a target of the Consumer relationship.
5. Click **Add Capability Version** to the right of the **Consumer** relationship.
6. Enter `C` in the Name field, select **Customer Care business process (1.0)** from the auto suggest list, and click **Add**. The process version is added as a target of the Provider relationship.
7. Click **Finish** to save your changes.
8. Click **Propose**. Note that the new governance state is Asset Scope Review as shown in Figure 16-17.

Figure 16-17

16.5.2 Approving the subscription request

The SOA governance team reviews and approves the subscription request. They are responsible for ensuring the following commitments:

- The service provider commits to providing the necessary resources to meet the service level that the establish customer process requires according to the project schedule.
- The consumer agrees that the capabilities detailed in the account creation service version specifications meets the requirements of the establish customer process. Additionally, this confirmation of acceptance of the

subscription request indicates that they have all of the detailed interface, binding, and policy information required for them to continue the development of the establish customer process.

After the subscription request is approved by all of the consumer and provider stakeholders, David, chairman of the SOA governance team, approves the subscription request. This approval is the final seal of approval from the governance team that the relationship between the consumer and provider can be entered into.

To transition the subscription request to the Asset Approved state, log on to WSRR, and select the **SOA Governance** perspective. Then, follow these steps:

1. Click **Tasks** → **Subscription Request Tasks** → **Subscription Requests for Scope Review** as shown in Figure 16-18.

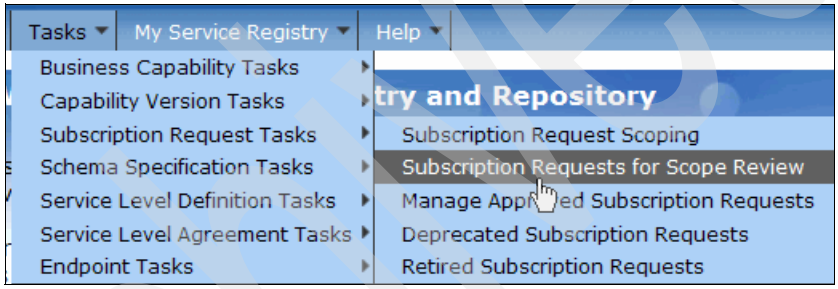


Figure 16-18 Navigating to subscription requests for scope review

2. Click **Customer Care consumption of account creation service** to display the subscription request details.
3. Click **Approve**. Note that the new governance state is Asset Approved as shown in Figure 16-19.

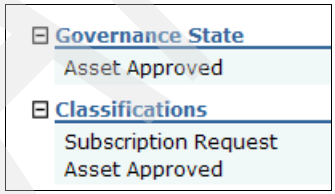


Figure 16-19 Subscription request governance asset approved state

Archived

Figure 16-20 shows the status of the WSRR entities after the new subscription request is approved.

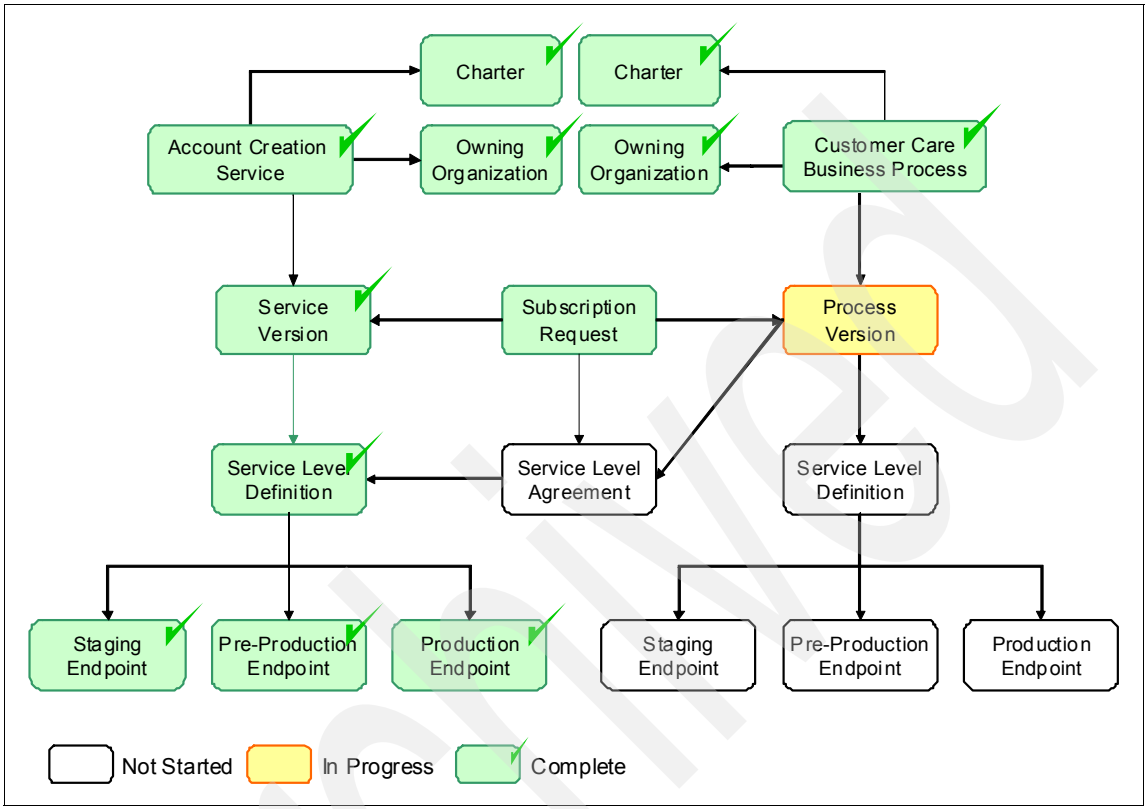


Figure 16-20 Status after subscription request is approved

Archived

16.6 Planning a process version

The scoped version is now in the planning phase where the development and owning organizations must work together to define the funding and timeframes for the project. In this phase, architecture sizing and funding decisions are made, including establishing dependencies on other assets or services. This planning process is illustrated in Figure 16-21. We describe this process in this section.

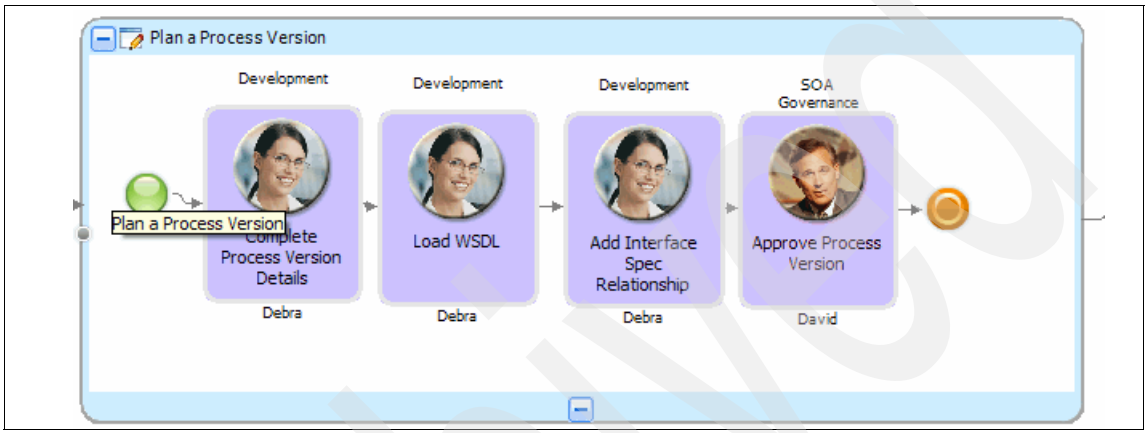


Figure 16-21 Planning a process version

16.6.1 Completing process version details

You need to update the process version to include proposed availability and termination dates. The plan is then proposed, which makes the plan available for review.

Updating the process version details is the responsibility of the development organization. Debra, the Development Manager for the commercial line of business area, is responsible for this phase of the scenario.

Archived

To complete the process version details, log on to WSRR, and select the **Development** perspective. Then, follow these steps:

1. Click **Tasks** → **Capability Version Tasks** → **Version Planning** as shown in Figure 16-22.

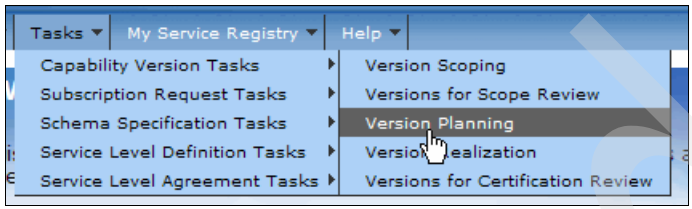


Figure 16-22 Navigating to version planning tasks

2. Click **Customer Care business process (1.0)** to display the process version details.
3. Then click **Edit Properties**. Enter values of your choice in the Version Availability Date and Version Termination Date fields.
4. Click **OK** to save your changes.

16.6.2 Loading the WSDL

Now, you load the abstract WSDL that defines the service interface as follows:

1. Click **Edit Relationships**.
2. Click **Add Document** to the right of the **Artifacts** relationship. You might have to scroll down in the relationships pane.
3. Ensure that the Document Type in the Load Document pane is set to **WSDLDocument**. Click **Load Document**.
4. Click **Browse**, and navigate to the directory where the WSDL document is located. Select **CustomerCare.wsdl**, and click **Open**. Then, click **OK**.
5. Click **Finish** to load the WSDL document. It is added as a target of the Artifacts relationship. Note that the dependent XSDDocument loaded previously is detected and identified as in repository as shown in Figure 16-23.

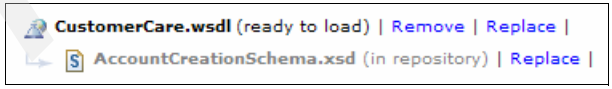


Figure 16-23 Dependent XSD

6. Click **Finish** to save your changes.

16.6.3 Adding the interface specification relationship

Next, create a relationship between the process version and the interface specification as follows:

1. Click **Edit Relationships**.
2. Click **Add Service Interface** to the right of the **Interface Specifications** relationship.
3. Enter `C` in the Name field, select **CustomerCare** from the auto suggest list, and click **Add**. The Customer Care service interface is added as a target of the Interface Specification.
4. Click **Finish** to save your changes.

16.6.4 Approving the process version

After all parties have agreed the details in the plan, David from the SOA governance team can approve the process version plan.

To transition the process version to the Planned state, log on to WSRR, and select the **SOA Governance** perspective. Then, follow these steps:

1. Click **Tasks** → **Capability Version Tasks** → **Version Planning** as shown in Figure 16-24.

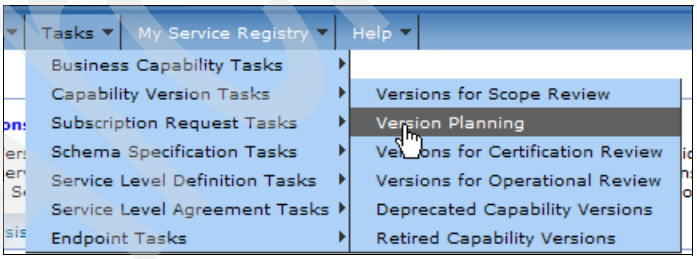


Figure 16-24 Navigating to version planning

2. Click **Customer Care business process (1.0)** to display the process version details.

Archived

3. Click **Approve Specification**. Note that the new governance state is Specified as shown in Figure 16-25.

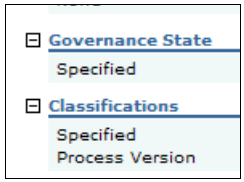


Figure 16-25 Process version specified governance state

Figure 16-26 shows the status of the WSRR entities after the new process version is planned.

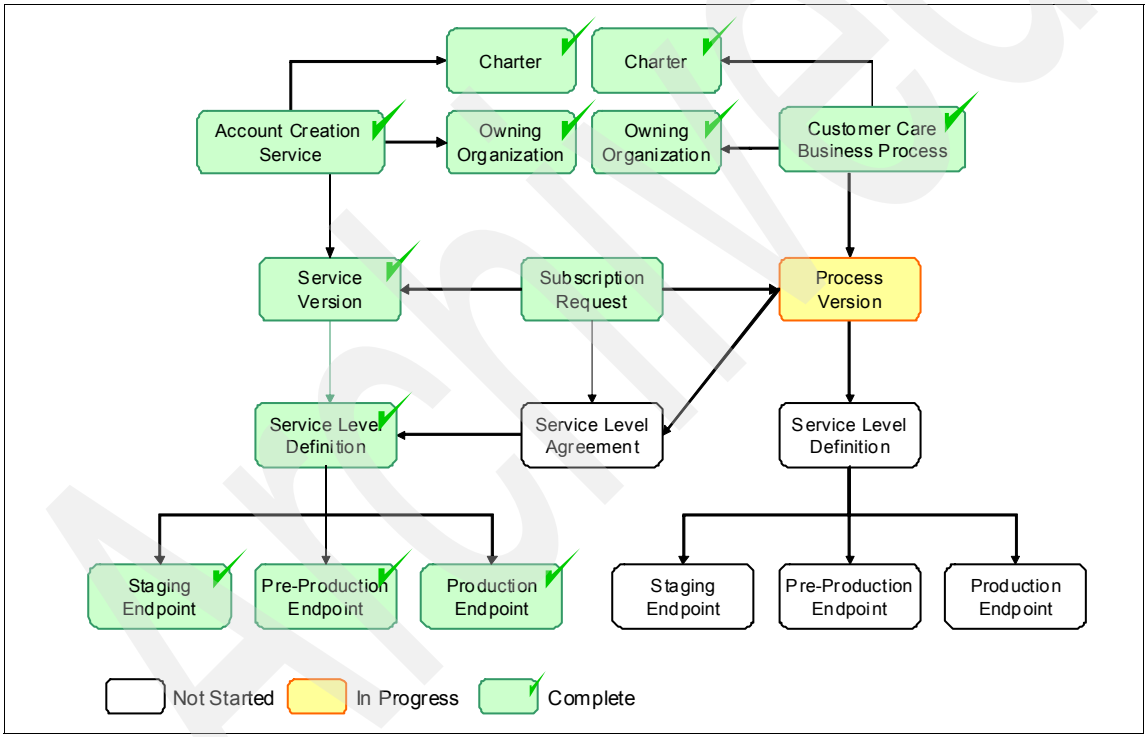


Figure 16-26 Status after process version is planned

The next phase of governing a new process is creating a service level definition. This step and the remainder of the activities that are required to govern a business process are identical to the equivalent steps described in 14.7, “Creating a service level definition” on page 509.

Part 4

Appendixes

Archived

© Copyright IBM Corp. 2009. All rights reserved.

Archived

A

IBM governance enabling tools

This appendix provides a brief overview of IBM software for build time SOA governance automation and runtime SOA governance automation.

© Copyright IBM Corp. 2009. All rights reserved.

IBM software for build time SOA Governance automation

This section addresses the following products:

- IBM Rational Method Composer
- IBM WebSphere Business Glossary
- IBM Rational System Architect
- IBM Rational RequisitePro and RequisitePro Composer
- IBM Rational Software Architect
- IBM Rational Application Developer
- IBM WebSphere Integration Developer
- IBM Rational Asset Manager
- IBM Rational Team Concert
- IBM Rational ClearCase
- IBM Rational Tester for SOA Quality and IBM Rational Performance Tester Extension for SOA Quality
- IBM Rational Clearcase and Buildforge
- IBM WebSphere Service Registry and Repository

IBM Rational Method Composer

Rational Method Composer assists in documenting and communicating governance processes and information. You can use this software to author, configure, view, and publish processes (in this case, SOA Governance processes). Rational Method Composer is a content management system that provides a common management structure and look and feel for all process content. All content managed in Rational Method Composer can be published to HTML and deployed to Web servers for distributed use by all governance stakeholders. The documented governance processes that are created with Rational Method Composer can be published and deployed as Web sites using the corporate intranet.

IBM WebSphere Business Glossary

WebSphere Business Glossary creates and manages a common language between business and IT. A common vocabulary allows the business and IT to share a common view and categorization of data and, therefore, create common data services throughout the enterprise. Data governance assurance programs assign responsibility to business users (data stewards) for the management of data through its life cycle. This assignment of responsibility is important for governance because the data owner provides final approval of requirements for that data and resolves all disputes and grant exceptions.

Archived

Creation of a standard, enterprise-wide business entity and messaging model (for example, schema metadata) is critical to developing enterprise-ready services. Although data transformations are sometimes necessary within the service (bad), or the middleware (good), such transformations can be minimized with a good messaging model and governance that enforces the services to use the messaging model.

IBM Rational System Architect

Transformation planning requires a real enterprise architecture with a fully integrated collection of models and documents across four key architecture domains:

- Business
- Information
- Systems
- Technology

Developing an enterprise architecture produces an organizational blueprint that can help improve business quality, efficiency, and accountability. Rational Systems Architect is the industry leading tool for enterprise architecture.

IBM Rational RequisitePro and RequisitePro Composer

Rational RequisitePro® and RequisitePro Composer trace business requirements back to business goals and to steps in the service development life cycle. This traceability enables organizations to document that all business goals are being addressed. It also allows for impact analysis when requirements need to be changed.

One inhibitor to reuse that we often see is the practice of recording business requirements solely within the documentation of individual IT projects, where it is often difficult or impossible for new projects to locate the requirements. This leads to future unnecessary duplication of analysis efforts and, worse, to developing and having to manage multiple independent implementations of common business functionality that are not fully compatible.

IBM Rational Software Architect

Rational Software Architect creates and documents service solutions so that team members are developing against a common set of project blueprints. Rational Software Architect matches the resulting models to the requirements that are maintained in Rational RequisitePro and RequisitePro Composer.

Archived

IBM Rational Application Developer

Rational Application Developer supports static code analysis. Using predefined and user defined rules, you can use Rational Application Developer to help ensure that no architectural, enterprise, or industry coding standards are violated. Also, during the implementation, the service build team can use the built-in service unit testing capabilities in Rational Application Developer to validate that requirements are implemented correctly.

IBM WebSphere Integration Developer

You can use WebSphere Integration Developer to assemble services into composite applications or to implement processes by re-using services.

IBM Rational Asset Manager

Rational Asset Manager provides the architectural standards and policies that must be followed for asset management and development life cycle management. Using Rational Asset Manager, you can help ensure compliance by centralizing patterns or transformation that can be used when developing the solution through specification, design, and build. In addition, Rational Asset Manager hosts a selection of implementation patterns and existing services that can be used during the build process to allow the build team to analyze the implementation patterns to find which one meets its specifications. When the service build is submitted back to Rational Asset Manager, the policy governor plug-in validates the implementation against architectural and coding standards. Rational Asset Manager uses a policy-based rules checking capability to automate certification before the service can be deployed.

IBM Rational Team Concert

Rational Team Concert™ is the first in a family of products that provides a collaborative portal for automating, integrating, and governing the activities in a team-based software delivery environment. It adds a significant degree of collaborative value to programs using Rational ClearCase®, ClearQuest®, and RSA, as well as many Eclipse-based tools.

IBM Rational ClearCase

Rational ClearCase product centralizes software configuration management. User authentication and audit features help provide security and enforce governance policies on when, what, and by whom services can be changed.

Archived

IBM Rational Tester for SOA Quality and IBM Rational Performance Tester Extension for SOA Quality

You can use Rational Tester for SOA Quality and Rational Performance Tester Extension for SOA Quality to validate that the functional and non-functional requirements are satisfied and that the service will meet any service level agreements (SLAs). It also addresses challenges that surround user acceptance testing for SOA services (both user interface based as well as composed).

IBM Rational Clearcase and Buildforge

Rational Clearcase and Buildforge allow development teams to standardize and automate repetitive build, test, and release tasks.

IBM WebSphere Service Registry and Repository

During the model and assemble phases of the service life cycle, you can use WebSphere Service Registry and Repository (WSRR) to locate copies of record of candidate service interaction metadata or intermediaries, as well as policies that govern the interactions. You can use WSRR to publish and govern service metadata about emerging, to-be-deployed services.

For the deploy phase, WSRR provides the system of record for metadata that describes service interaction endpoints. WSRR is populated with metadata as part of an SOA solution deployment or using discovery of existing endpoints. In addition, it is used by SOA runtimes as well as deployer and administrator roles when detailed information about service endpoints is required to drive operations of the deployed composite applications.

IBM software for runtime SOA Governance automation

This section addresses the following products:

- IBM WebSphere Service Registry and Repository
- IBM Tivoli Change and Configuration Management Database
- IBM Tivoli Composite Application Manager for SOA
- IBM Tivoli Security Policy Manager
- IBM DataPower SOA Appliances
- IBM WebSphere Message Broker and IBM WebSphere Enterprise Service Bus

Archived

IBM WebSphere Service Registry and Repository

SOA runtime management and resilience within the SOA is enhanced by sharing the service metadata that exists in WebSphere Service Registry and Repository (WSRR) with operational data stores, allowing management and monitoring dashboards to present a more comprehensive view of the managed service environment. Summary information about service performance can be fed back into WSRR and used by the execution environment to affect the selection of the best-fit provider. You can also use WSRR to manage policies that are associated with services, so that you can deploy and enforce these policies at service run time.

IBM Tivoli Change and Configuration Management Database

You can use Tivoli Change and Configuration Management Database to acquire and manage detailed information about the environment and topology in which service endpoints execute. Tivoli Change and Configuration Management Database ensures compliance with internal and regulatory requirements by enforcing policies and tracking changes throughout your organization.

IBM Tivoli Composite Application Manager for SOA

IBM Tivoli Composite Application Manager for SOA (ITCAM for SOA) provides SOA infrastructure management software and integrated management tools that speed and simplify identification and resolution of SOA problems. A services topology view displays actual service-to-service relationships, including drill down to service status and metrics, so that you can keep track of your service flow. ITCAM for SOA provides automated SOA management and SOA monitoring software to help meet established service levels with built-in alerts, message mediations, situations and workflows. It also helps manage SOA SLAs and issues alerts when SLAs are not met. This information can be deposited into WSRR for the full service status perspective of the health and availability of the service.

IBM Tivoli Security Policy Manager

Tivoli Security Policy Manager helps strengthen application access, facilitates compliance, and supports operational governance throughout the IT infrastructure. Tivoli Security Policy Manager manages SOA security policies and entitlements throughout the policy life cycle, from authoring and publishing to enforcing and updating. It can then enforce policies at run time. Tivoli Security Policy Manager enables end-to-end application authorization using flexible policy administration and standards-based policy decisions.

IBM DataPower SOA Appliances

This appliance is a firmware-based solution for security enforcement, policy enforcement, EAB, and XML accelerator. The XS40 version delivers a comprehensive set of configurable security and policy enforcement functions. The XM70 delivers predictive, low latency messaging and routing for data distribution. The XB60 extends the Enterprise Service Bus (ESB) with purpose-built business-to-business hardware providing AS2/AS3 messaging and trading partner profile management in a high-performance DMZ-ready appliance. Hardware ESB from IBM, the XI50 is purpose-built for simplified deployment and hardened security, bridging multiple protocols and performing any-to-any conversions at wire speed. The XA35 offloads web and application servers by processing XML, XSD, XPath and XSLT at wire speed.

IBM WebSphere Message Broker and IBM WebSphere Enterprise Service Bus

WebSphere Message Broker and WebSphere Enterprise Service Bus provide connectivity and act as enforcement points for dynamic service selection and mediation. They also distribute information and data generated by business events in real time to people, applications, and devices throughout your extended enterprise and beyond. From a strictly governance perspective, an ESB provides a single point of contact for monitoring some service internal metrics, such as the number and scope of data mediations, and the ability to route to service endpoints depending on operational metadata.

Archived

Archived

B

Additional material

This book refers to additional material that you can download from the Internet as described in this appendix.

Locating the Web material

The Web material 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/SG247793

Alternatively, you can go to the IBM Redbooks Web site at:

ibm.com/redbooks

Select the **Additional materials** and open the directory that corresponds with the IBM Redbooks form number, SG247793.

© Copyright IBM Corp. 2009. All rights reserved.

Archived

Abbreviations and acronyms

API	application programming interface
BAM	Business Activity Monitoring
BIRT	Business Intelligence and Reporting Tool
BPEL	Business Process Execution Language
CB	Component Broker
CRUD	Create, Retrieve, Update, Delete
CVS	Concurrent Versions System
DOU	Document of Understanding
ESB	Enterprise Service Bus
GEP	Governance Enablement Profile
IBM	International Business Machines Corporation
ISC	Integrated Solution Console
IT	Information Technology
ITSO	International Technical Support Organization
JAAS	Java Authentication and Authorization Service
JKHLE	JKHL Enterprises
LTPA	Lightweight Third Party Authentication
LoB	Line of Business
MTOM	Message Transmission Optimization Mechanism
ORB	Object Request Broker
OSIMM	Open Group Service Integration Maturity Model
OWL	Ontology Language
QoS	Quality of Service
RBAC	Role-Based Access Control
REST	Representational State Transfer
SCA	Service Component Architecture
SCDL	Service Component Description Language
SDO	Service Data Object
SGMM	SOA Governance and Management Method
SLA	service level agreement
SLD	service level definition
SOA	service-oriented architecture
SSL	Secure Sockets Layer
SSO	Single Sign-On
TOGAF	The Open Group Architecture Forum
UI	User Interface
UML	Unified Modeling Language
URI	Uniform Resource Identifier
WSDL	Web Services Description Language
XSD	XML Schema Definition

Archived

© Copyright IBM Corp. 2009. All rights reserved.

Archived

Related publications

We consider the publications that we list in this section particularly suitable for a more detailed discussion of the topics that we cover in this book.

IBM Redbooks publications

For information about ordering these publications, see “How to get IBM Redbooks” on page 625. Note that some of the documents referenced here might be available in softcopy only.

- *Integrating WebSphere Service Registry and Repository with WebSphere Process Server and WebSphere ESB, REDP-4557*
- *Integrating WebSphere Service Registry and Repository with WebSphere MQ and WebSphere Message Broker, REDP-4558*
- *Integrating WebSphere Service Registry and Repository with WebSphere DataPower, REDP-4559*
- *Integrating WebSphere Service Registry and Repository with IBM Tivoli Security Policy Manager, REDP-4561*
- *Service Lifecycle Governance with IBM WebSphere Service Registry and Repository Advanced Lifecycle Edition, SG24-7782*
- *WebSphere Application Server V6 Planning and Design WebSphere Handbook Series, SG24-6446*
- *WebSphere Application Server V6 Scalability and Performance Handbook, SG24-6392*
- *WebSphere Application Server V6 Technical Overview, REDP-3918*

How to get IBM Redbooks

You can search for, view, or download IBM Redbooks, Redpapers, Technotes, draft publications and Additional materials, as well as order hardcopy Redbooks publications, at this Web site:

ibm.com/redbooks

© Copyright IBM Corp. 2009. All rights reserved.

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services

Archived

Redbooks

Service Lifecycle Governance with IBM WebSphere Service Registry and Repository

(1.0” spine)
0.875”<->1.498”
460 <-> 788 pages

Archived

Archived

Archived

Service Lifecycle Governance with IBM WebSphere Service Registry and Repository

Discover how to implement service life cycle governance

Examine how to build WSRR solutions

Learn by example with practical scenarios

IBM WebSphere Service Registry and Repository (WSRR) provides service registry and repository functions for service-oriented architecture (SOA) enterprise applications. This IBM Redbooks publication uses business scenarios to illustrate SOA governance using WSRR as the authoritative registry and repository.

We divided this book into the following parts:

- Part one of this book introduces SOA and service governance, provides a technical overview of WSRR, and describes the WSRR governance enablement profile.
- Part two of this book provides step-by-step guidance to building WSRR solutions. This part addresses topics such as modeling in WSRR Studio, creating a WSRR user interface, security, promotion, policies, and reports.
- Part three describes a series of common usage scenarios and describes step-by-step how to implement them in WSRR. These scenarios describe how to govern schemas, existing services, new services, upgrades to services, and business processes.

This book is based on WebSphere Service Registry and Repository V6.3 and is intended for IT architects and IT specialists looking to get a better understanding of how to achieve service life cycle governance.

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

SG24-7793-00 ISBN 0738433829