WebSphere® software

IBM

High Availability and Scalability for WebSphere Presence and XML Document Management Servers

Go from base to highly available and scalable environments

Architecture, topology and configuration

Sample implementation

Edward Oguejiofor
Vijay Bhadriraju
Rajasekhar Durvasula
Andrew Jones
Ravikumar R Kanamatareddy
Yutaka Obuchi
Sebastian Pietrzak
Galina Rubinshtein

Redbooks

ibm.com/redbooks

International Technical Support Organization

High Availability and Scalability for WebSphere Presence and XML Document Management Servers

June 2008

SG24-7370-00

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

First Edition (June 2008)

This edition applies to Version 6.2, of the IBM WebSphere Presence Server and IBM XML Document Management Servers (XDMS).

Contents

Notices

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

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

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

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

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

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

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

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

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

COPYRIGHT LICENSE:

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

Trademarks

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

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

AIX 5L™	IBM®	Redbooks (logo) ®
AIX®	IMS™	Tivoli®
DB2 Universal Database™	Rational®	WebSphere®
DB2®	Redbooks®	xSeries®

The following terms are trademarks of other companies:

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

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

EJB, Enterprise JavaBeans, J2EE, Java, JavaBeans, JDBC, JMX, JVM, and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Intel, Pentium, Pentium 4, Intel logo, Intel Inside logo, and Intel Centrino logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries 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.

Preface

In this IBM® Redbooks publication, we describe how to create scalable and highly-available solutions that incorporate only two of the WebSphere products for Telcom: the IBM WebSphere® Presence and the WebSphere XML Document Management Servers (XDMS). IBM WebSphere products for Telcom is a set of service enablers that extend the IBM WebSphere Application Server to deliver full IP Multimedia Subsystem (IMS) standards-compliant platform for creation and delivery of next-generation telecommunications services.

This book is structured into five parts:

- Part 1, "Introduction" - in this introductory part, we provide an overview of the IBM WebSphere Presence Server and XDMS Document Management Servers and describe the sample scenario that we use throughout this book to demonstrate availability and scalability.
- Part 2, "The base scenario" - in part 2 of the book, we provide an overview of the base environment for our sample scenario, which includes the architecture and topology. We also describe how to set up and configure the different components in the base environment. We conclude with the execution of test cases that validate the base environment.
- Part 3, "The high-availability (HA) scenario" - in part 3, we describe the enhancements to the base environment to make it highly available. We demonstrate fail-over processing following failure of servers in the environment.
- Part 4, "The scalable scenario" - in this part, we describe how to configure additional servers to make the highly available environment scalable. We demonstrate the distribution of messages to additional servers that are introduced to the environment.
- Part 5, "Appendixes" - we provide additional information and pointers to related publications and other useful information that includes how to obtain the scripts for the sample scenarios.

The team that wrote this book

A team of specialists from around the world working at the International Technical Support Organization, Raleigh Center produced this book.

Figure 0-1 This IBM Redbooks® publication team, (front row - left to right) Rajasekhar Durvasula, Ravikumar R Kanamatareddy, and Vijay Bhadriraju; (back row - left to right) Andy Jones, Edward Oguejiofor, Sebastian Pietrzak, and Yutaka Obuchi. Participating remotely was Galina Rubinshtein.

Edward Oguejiofor is a Project Leader at the International Technical Support Organization, Raleigh Center. He has over 20 years of experience in distributed and enterprise systems architecture and design. His areas of expertise are Web Services and service-oriented architecture. He also provides technical and thought leadership in emerging technologies and collaborative services. He has a Bachelor of Science degree in Software Engineering from the University of London's Imperial College of Science and Technology.

Vijay Bhadriraju is a Certified Senior IT Specialist working for the IBM Software Group in IBM Research Triangle Park Lab, USA. He has 13 years of IT industry experience with six years spent in developing Rational® and WebSphere Studio IDEs and six years spent in Smalltalk consulting services. He currently works on enabling IBM Business Partners on the IBM IP Multimedia Subsystem (IMS) Product set and WebSphere software. He has a Bachelor of Engineering degree in Computer Science from the University of Pune, India.

Rajasekhar Durvasula is an Advisory IT Architect with the IBM India Software Labs IMS™ BPTEWW Team that focuses on Customer and Business Partners enablement in the Asia Pacific geographic area. Raj has over 10 years of industry experience in software development, systems design, and systems deployment of large-scale Enterprise Application Integration/OSS/BSS solutions in telecom space. His areas of interest are EAI, SOA, and Project Management Methodologies.

Andrew Jones is a Senior Solution Architect in the WebSphere Business Partner Technical Enablement group specializing in the telecom area. Andrew is an IBM Certified IT Architect, IBM Certified SOA Solution Designer, and Open Group Master Certified IT Architect. Based in his home town of Austin, Texas, Andrew has worked for IBM for 18 years, with the majority of those years focused on customer and partner enablement of leading-edge technologies.

Ravikumar R Kanamatareddy is a Senior Solution Manager in IBM India Global Business Solutions. He has 17 years of experience in the signaling and digital switching field of telecom. Ravi also has experience in working on OSS/BSS systems for global telecom providers. He has a degree in Engineering from JNTU, India and Post Graduation from CEPT, India. His areas of expertise are building real-time subsystems on signaling gateways and data management solutions for integration. He has written on subjects varying from migration issues between 2G and 3G, subscriber provisioning and management, and charging event management for future services.

Yutaka Obuchi is a Software Engineer for Hitachi Software Engineering in Japan, which is an IBM Business Partner. Currently he is working at the IBM WebSphere Technology Institute in Raleigh as a trainee. He has three years of experience in J2EE™ technologies and two years of experience in several areas of telecom system solutions, such as functional analysis, customization, implementation, and system integration. He has a Master of Science degree in Physics from Kyoto University in Japan.

Sebastian Pietrzak is an IT Architect in the IBM Telecommunications Sector, Poland. He has over five years of experience as an IT Architect. His background is in the architect of complex solutions and the technical leadership of its delivery. Sebastian has worked with several telecommunication companies across Central and Eastern Europe and the Middle East. He has been involved with Customer enablement in such areas as enterprise systems architecture, application development, and end-to-end integration. His areas of specialty are security, high availability, and performance. He is currently focused on IMS and SDP enablement. He has a Master's degree in Information Technology from Wroclaw University of Technology, Poland.

Galina Rubinshtein is a Senior Software Engineer with IBM Israel Software Labs. She has over ten years of experience in the design and development of server-side applications, with three of those years spent developing the IBM WebSphere Presence Server. Galina is currently focused on providing technical services to Customers and Business Partners for IBM WebSphere products for Telcom. She has a Master's degree in Applied Mathematics from the National University of Railway Transport, Ukraine.

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

IBM Software Group

- Benson Chen
- Girish Dhanakshirur
- Chris Germano
- Mark Gilmore
- Ivan Heninger
- Curtis E Hrischuk
- Yun Huang
- Ronnie Jones
- Jeffrey Martin
- Brian Pulito
- Tim Smith
- Cristi Nesbitt Ullmann
- Douglas Williams
- Anthony Wrobel

IBM Software Group, Israel

- Shai Beem
- Avshalom Houri

International Technical Support Organization, Raleigh Center

- Rufus Credle
- Carolyn Briscoe
- Tamikia Barrow
- Margaret Ticknor

International Technical Support Organization, Poughkeepsie Center

- Michael B. Schwartz

International Technical Support Organization, Business Controls

- Erica B. Wazewski

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

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 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

Part 1

Introduction

In this introductory part, we provide an overview of the IBM WebSphere Presence Server and XML Document Management Servers (XDMS). We describe the sample scenario that we use throughout this book to demonstrate availability and scalability using the IBM WebSphere Presence Server and XML Document Management Servers (XDMS).

1

IBM WebSphere products for Telcom

In this chapter, we provide an overview of the IBM WebSphere products for Telcom and the functional description of the IBM WebSphere Presence Server and the IBM XML Document Management Servers (XDMS).

This chapter contains:

1.1 Introduction to IBM WebSphere products for Telcom

The Telecommunications industry is poised for growth with the introduction of IP Multimedia Subsystem (IMS). IMS defines the architecture and framework for delivering revenue-generating IP services over different types of wired and wireless access networks.

1.2 A bit of history

IMS was initially defined by the 3rd Generation Partnership Project (3GPP), which is a collaboration among a number of telecommunications standards bodies as part of the standardization work for supporting Global System for Mobile communications (GSM) networks and radio technology evolution. IMS was first introduced in 3GPP Release 5, where Session Initiation Protocol (SIP), which the Internet Engineering Task Force (IETF) defined, was chosen as the main protocol for IMS. It was further enhanced in Releases 6 and 7 of 3GPP to include additional features, such as Presence and Group Management, interworking with Wireless Local Area Networks (WLAN) and Circuit Switched (CS) based systems, and Fixed Broadband access.

The 3rd Generation Partnership Project 2 (3GPP2), which also standardized on IMS, was born to evolve North American and Asian Cellular Radio-telecommunication Intersystem Operations into a third-generation system. The initial release of 3GPP2 specifications on IMS was largely based on 3GPP Release 5. The two IMS specifications that were defined by two organizations are fairly similar but not exactly the same. 3GPP2 added appropriate adjustments for their specific issues. Nevertheless, the purpose of both organizations is to ensure that the IMS applications work consistently across different network infrastructures.

In addition to the 3GPP and 3GPP2, Open Mobile Alliance (OMA) also plays an important role with respect to IMS service standardization. OMA defines standardized services that are built on top of IMS infrastructure, and examples of such services include Instant Messaging (IM), Presence service, and Group Management Service.

1.2.1 IMS architecture overview

The IMS architecture is split into three main planes/layers:

- Service or Application layer

This layer defines the infrastructure for the provisioning and management of services. It contains various Application Servers that provide additional services beyond the IMS core. The services that adhere to the control and restrictions of IMS can be provided by the service provider or by third parties. An example of such services are Presence Server (although some think that it is more of an enabler than a service) and Push-to-talk service.

- Control or Signaling layer

  This layer is responsible for routing call signaling and controlling sessions traffic through the system. It contains various SIP proxies and other control elements. The key elements are:

  – Call Session Control Function (CSCF)

    CSCF refers to SIP servers or proxies. It handles SIP registration of end points and processes SIP signal messaging of the appropriate Application Server in the service layer.

  – Home Subscriber Server (HSS)

    HSS database stores the unique service profile for each user. The service profile might include a user's IP address, telephone records, buddy lists, voice mail greetings, and so on.

- User or Transport layer

  This layer includes the various access network servers, such as IPv6 network and WiFi and broadband networks. It also contains the radio components. This layer initiates and terminates Session Initiation Protocol (SIP) sessions and provides the conversion of data that is transmitted between different packets' format.

Figure 1-1 on page 6 illustrates the IMS architecture.

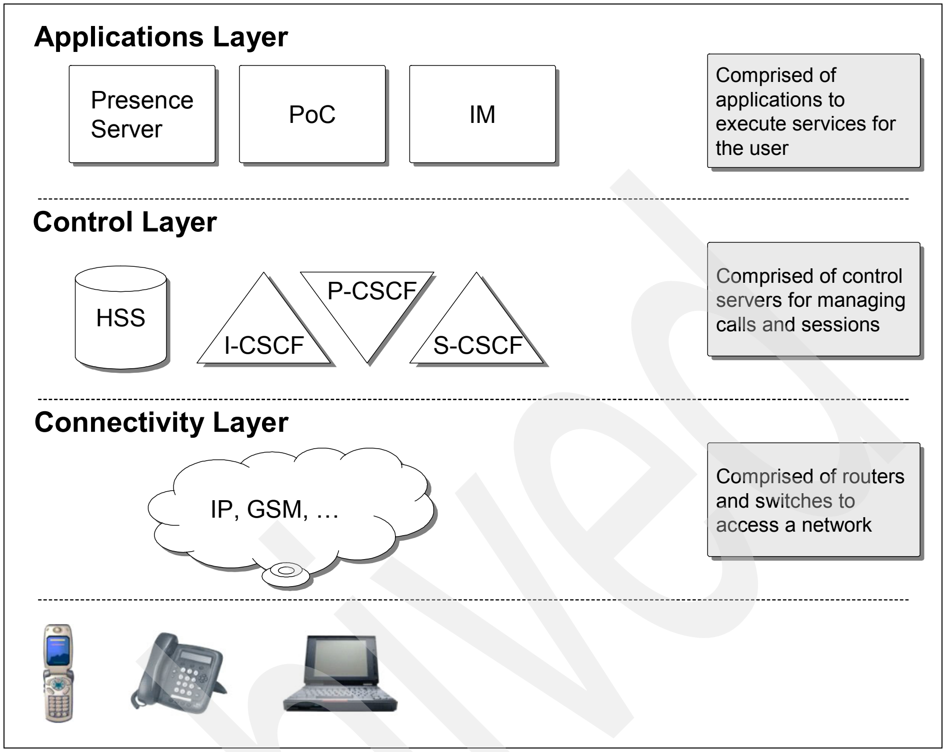


Figure 1-1 IMS architecture overview

1.2.2 IMS protocols

OMA specifications describe the usage of two primary network protocols in IMS: XML Configuration Access Protocol (XCAP) and SIP.

- XML Configuration Access Protocol (XCAP)

  XCAP is an IETF specification for how XML documents are transported over HTTP. XCAP enables clients to read, write, and modify application configuration data that is stored in XML format on a server. XCAP maps XML document subtrees and element attributes to HTTP URIs so that HTTP can directly access these components.

- Session Initiation Protocol

  SIP is an application-layer signaling protocol for initiating, modifying, or terminating communication and collaborative sessions over Internet Protocol (IP). A session can be an IP telephony call, a multi-user conference that incorporates voice, video and data, instant messaging chat, or multi player online game. You can use SIP to invite participants to a scheduled or already existing session. Participants can be a person, an automated service, or a physical device, such as a handset. You can also use SIP to add or remove media to a session.

Note: IMS was designed for the Telecommunications environment; however, you can also use it for enterprises environments. IMS for enterprises enables easier integration of services that are provided by or for service providers. The architecture does not include some components, such as the P-CSCF and other components that are not needed in the secure environment of an enterprise; instead, it includes many features that are needed in the enterprise, which includes Location, Policy, and Application management.

1.2.3 IBM solution for IMS

IBM solution for IMS consists of full life-cycle service delivery environments that focus on the service and application plane. The solution includes two service Enablers that are the subject of this Redbooks publication: the IBM Presence Server and the IBM XML Document Management Servers (XDMS), which are developed on top of WebSphere Application Server. The IBM Presence Server and XDMS together provide a standard-compliant presence solution that you can easily integrated into Next Generation networks and IMS infrastructure, enabling real-time communication across IMS-based applications.

The WebSphere platform

IBM WebSphere products for Telcom run on the IBM WebSphere Application Server. The IBM WebSphere Application Server provides a converged HTTP/SIP execution environment that enables the development of applications that leverage converged HTTP/SIP functionality.

The IBM WebSphere Application Server provides a truly converged HTTP/SIP execution environment, which includes both a converged HTTP/SIP container and converged HTTP/SIP proxy. It implements the JSR 116 SIP Servlet API specification and supports SIP PUBLISH, SUBSCRIBE, NOTIFY.

Utilizing WebSphere Application Server also simplifies deployment, management, and administration and leverages security, reliability, redundancy and high availability.

1.3 IBM WebSphere Presence Server

IBM WebSphere Presence Server is an IMS-compliant service enabler that collects, manages, and distributes real-time presence information to enhance collaboration services across devices and applications in the Next Generation networks. The main purpose of the WebSphere Presence Server is to provide awareness on an as-needed basis, allowing both devices and applications to publish presence information and subscribe to it in order to receive notifications of information changes.

WebSphere Presence Server component is developed using industry-standard protocols, such as those that are defined by the IETF, SIP, and the extension defined by the SIP Instant Messaging and Presence Leveraging Extensions (SIMPLE) working group.

IBM Presence Server is implemented on top of the WebSphere JSR-116-compliant SIP Container and fully integrates with Java™ Platform, Enterprise Edition (Java EE) architecture. Developed as an application on IBM WebSphere Application Server, the IBM Presence Server is a highly scalable, reusable service enabler that supports the delivery of rich, next generation services.

1.3.1 The benefits

IBM WebSphere Presence Server enhances end user value by providing common intelligence that is used across applications. It provides the following benefits by enabling the extension of presence information to enhanced composite services and applications.

- Simplifies the management of and reduces the cost of aggregating presence information across disparate devices, applications, and network elements. You no longer need to learn and manage numerous, siloed presence environments from multiple vendors.
- Speeds time-to-market for new integrated services by providing a single presence platform that spans all applications and services, which makes Presence a centralized enabler that applications simply plug into.
- Improves the intelligence and capabilities of applications, which increases the value to customers and revenue opportunity for service providers. Service providers can add presence information to applications that do not have their own presence infrastructure.

IBM WebSphere Presence Server has numerous distinguishing advantages that include:

- Application and service independence

Supports the integration of presence information from both clients and applications to any standards-based application, such as instant-messaging and presence-enabled phone books

- Runs on WebSphere Application Server

  Enables you to build integrated applications that take advantage of the converged HTTP/SIP container while providing proven manageability, scalability, and security

- Compliant with IETF, OMA ETSI, and 3GPP standards

  Promotes the platform for widespread integration among multi-vendor applications, devices, and network infrastructure

- IMS compliant

  Integrates with Service Provider IMS networks through IMS Session Control (ISC) interfaces that are tested with leading Network Equipment Providers for interoperability.

- Supports both standard and custom presence objects

  Extends presence beyond people to objects, such as delivery trucks and conference rooms

1.3.2 Presence in IMS

Presence or awareness on the status of people, documents, and objects is a very powerful platform for building new and interesting applications. The ability to get near real time awareness on many types of resources can enable more automation in processes that now require you to actively check the status of some resource in order to know when to do something.

Consider the processing of a customer's order, and that currently there are many ways to automate parts of that process. Using presence enables you to subscribe to the status of the process and enables smooth execution of the order processing process.

Another area where presence can make a significant difference is industries where the communication inside is done partly by old technology and partly by new technology. Some examples of this bipolar existence is healthcare and aviation industries where presence information can be a valuable component for creating systems that make communication much more efficient. In the healthcare industry, you can create systems that use presence and source of information on entities, which makes it easier to create integrated applications where the sources can vary according to the need of the observer, for example, a nurse can subscribe to the presence of vital information, such as patience heartbeat.

IBM Presence Server implements the following reference points from the “3GPP TS 123 141” technical specification:

- Pen – Presence Network Agent<->Presence Server

  Enables presentity's presence information to be supplied to the Presence Server. IBM Presence Server implements this reference point through SIP protocol.

- Pep – Presence UA<->Presentity Presence Proxy

  This reference point allows presentity's presence information to be supplied to the Presence Server. It uses the SIP protocol (PUBLISH).

- Pex – Presence External Agent<->Presence Server

  This reference point enables external presence agents (elements outside the provider's network) to supply presence information to the Presence Server. IBM Presence Server implements this reference point through SIP protocol.

- Pi – S-CSCF<->Presence Network Agent

  This reference point enables networks to get IMS registration status for a user from the S-CSCF. The interface is defined for the ISC and the implementation by IBM Presence Server is through SIP SUBSCRIBE/NOTIFY.

- Pw – Watcher Apps<->Presentity Presence Proxy.

  Provides the ability to request and get presence information either through fetch or subscribe. IBM Presence Server implements this reference point using SIP protocol.

- Pwp – Presentity Presence Proxy<->Presence Server.

  The Pwp interface is an intra-operator interface. This reference point allows all of the functionalities that the Pw and Pep reference points provide.

Figure 1-2 on page 11 shows the overview of the IBM Presence Server reference points in the IMS environment. It coallocates the Presence Server, the presentity presence proxy, and the watcher presence proxy, and also acts as a network agent for retrieving information from the environment.

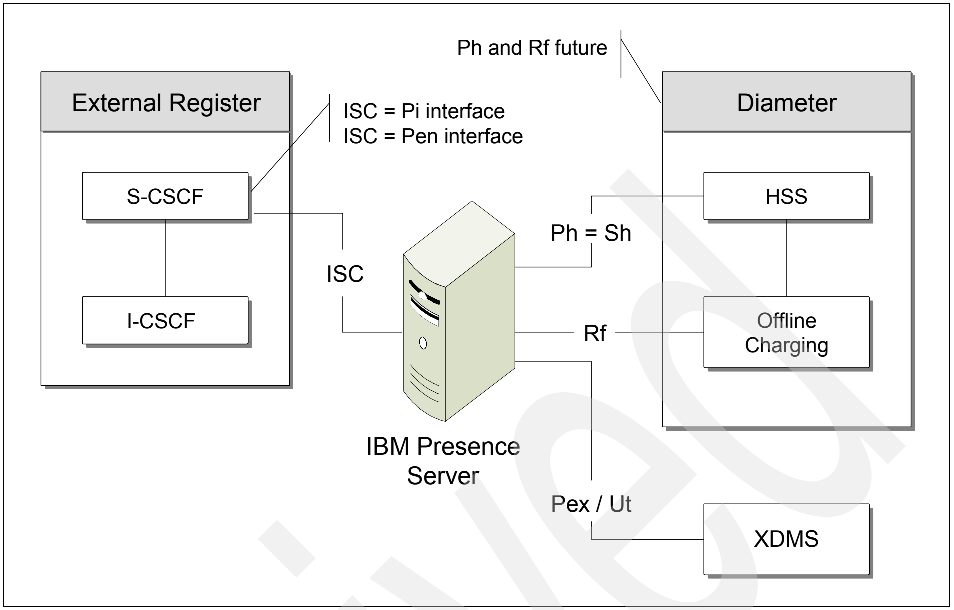


Figure 1-2 Overview of IMS Presence architecture

1.3.3 The basic functionality

The IBM Presence Server implements a number of basic functionalities:

- Presence documents
- Publish-Subscribe-Notify
- Presence data storage
- Events service
- Aggregation from multiple sources
- Group lists
- Presence authorization rules
- Partial publication/notification
- Watcher information
- Usage records

Presence documents

A presentity is generally associated with a person, although it can relate to other things, for example, a conference room. The WebSphere Presence Server identifies the presentity with unique URI, either a SIP URI or a TEL URI. The URI is in turn associated to a presence document.

WebSphere Presence Server supports the standard format of presence documents:

- Presence Information Data Format (PIDF) - the base type of presence document
- Rich Presence Information Data Format (RPID) - rich extensions to PIDF

Publish-Subscribe-Notify

The Publish-Subscribe-Notify (PSN) services are at the core of the IBM WebSphere Presence Server. The set of services implement the major functionalities of the Presence Server according to standards that the IETF defined in RFC3265, RFC3856, RFC3857, RFC3858, RFC3903, and many other draft documents.

Specifically, the PSN server provides the following services:

- Receives and stores all published presence documents
- Receives and handles all subscriptions from watchers
- Aggregates, filters, and composes the presence data according to the publisher/watcher requests
- Notifies watchers when a presence state of the presence in which they are interested changes

To provide this service the following SIP messages are used:

- PUBLISH

  A client-generated request to create, modify, and remove an event state associated with the Address of Record. The event state is carried in the body of the PUBLISH request and maintained by the Presence Server as a presence document. WebSphere Presence Server stores this document until the document expires or the presentity removes it. The combination of the PUBLISH Event package and Content type headers define the format of the published presence document.

- SUBSCRIBE

  A client generated request for a current state and the state updates from a remote node.

- NOTIFY

  A request that the Presence Server generates to notify subscribers of the changes in the event state.

Example: When Sue adds John to her IM list, Sue’s IM client goes to the Presence Server and asks for John’s presence document, which occurs through SUBSCRIBE. The Presence Server then returns John’s presence document to Sue through NOTIFY. Additionally, the Presence Server keeps the SUBSCRIBE open and sends a NOTIFY to Sue for any changes to that document prior to the expiration of the SUBSCRIBE. Upon expiry, Sue’s IM must resubscribe.

Presence data storage

WebSphere Presence Server allow presentities to publish presence documents. It stores the data until the documents expire or until the presentities desire to remove them. To do this, the IBM WebSphere Presence Server handles all PUBLISH messages that the presentities send. Messages are initiated by a registrar server or from external services. The WebSphere Presence Server combines the PUBLISH information with other presence data into well-formed presence documents that are then stored in the database, which Figure 1-3 illustrates. The combination of the PUBLISH Event package and the Content type headers define the format of the published presence document.

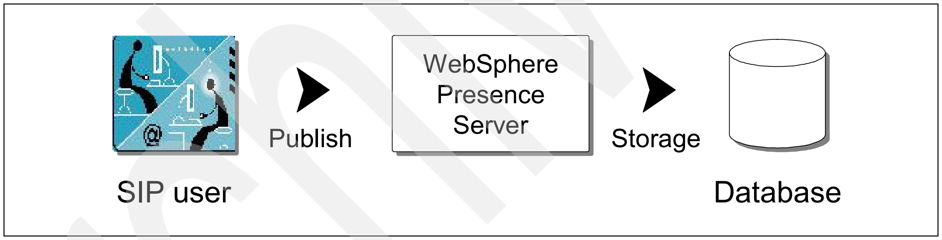


Figure 1-3 Publication storage

Note: WebSphere Presence Server uses JDBC™ to present a uniform interface to databases and therefore can connect to any JDBC-compliant database.

Events Server

WebSphere Presence Server supports subscriptions for notifications to changes in presence information. It receives the SIP SUBSCRIBE requests, verifies the compliance of those requests with RFC 3261 and 3265 (presence of all MUST headers and reasonable syntax of the request), and creates a subscription on the presence. The WebSphere Presence Server collects the presence document and sends NOTIFY messages to the subscribers. The type of each presence is determined by the Event package name in the SUBSCRIBE message.

Figure 1-4 illustrates the subscription notification process.

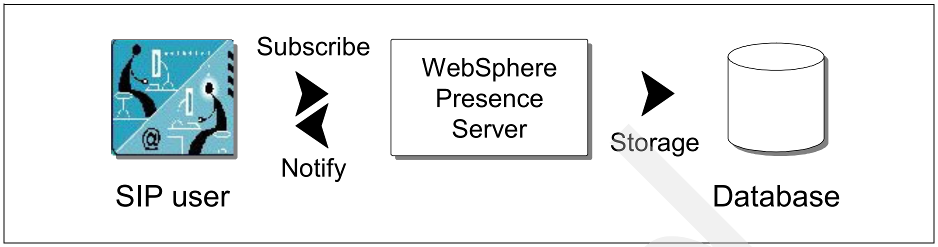


Figure 1-4 Subscription / Notification

Aggregation from multiple sources

In Presence publication, each PUBLISH carries PIDF with exactly one segment of a specific presentity. The presentity's presence document can contain information from multiple applications or devices as long as they are associated with the same public ID (SIP URI or Tel URI). Therefore, the information in a PUBLISH might only contain a segment of the presentity's overall presence information. WebSphere Presence Server uses the initial PUBLISH to set up the presentity's presence document. When an ensuing PUBLISH includes additional elements, WebSphere Presence Server appends these elements as additional tuples within the presence document.

Figure 1-5 on page 15 provides examples of document aggregation.

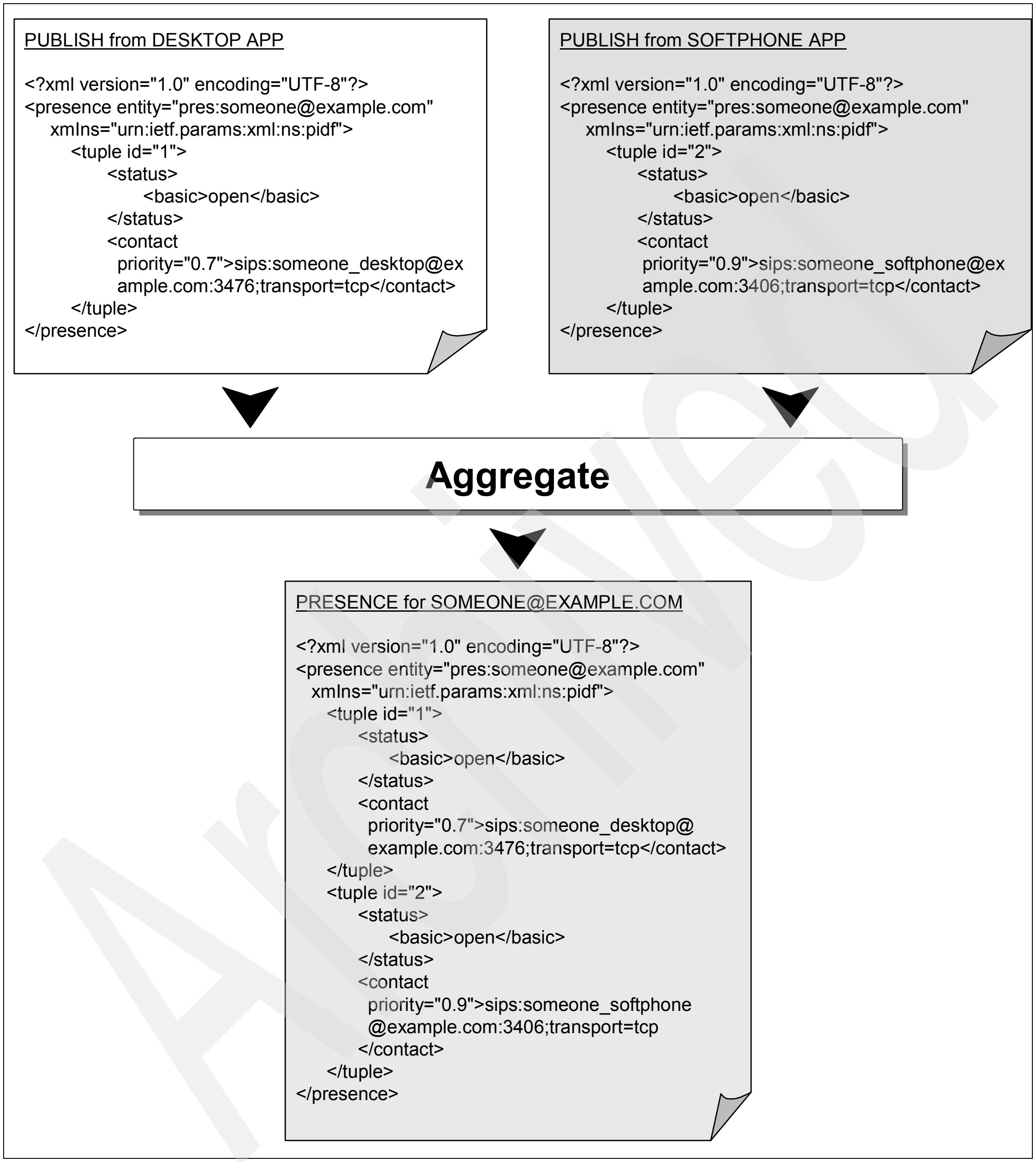


Figure 1-5 Example of document aggregation

Group lists

WebSphere Presence Server supports subscriptions to list of presentities, which includes hierarchical lists. WebSphere Presence Server acts as a Resource List Server (RLS) that accepts subscriptions to resource lists and notifies the subscribers of the state of all presentities in the list. WebSphere Presence Server retrieves group lists from an external XML Document Management Servers (XDMS), which makes an XML Document Management Enabler a requirement for group list capabilities.

WebSphere Presence Server interoperates with XML Document Management Servers (XDMS) using XCAP and SIP. In addition to XCAP GET, WebSphere Presence Server can subscribe to list documents and retrieve group documents through XCAP. WebSphere Presence Server uses standardized interfaces and reference points to interoperate with an XDMS; therefore, it can work with the WebSphere XDMS or another XDMS.

> **Note:** Without an XDMS, WebSphere Presence Server cannot provide group list capabilities because it requires an XDMS to store the group list documents. We recommend that WebSphere Presence Server utilize the WebSphere XDMS.
>
> WebSphere Presence Server does not act as an intermediary for the management or retrieval of group list documents by clients or applications. Clients and applications manage group lists directly through the XDMS.

Presence authorization rules

Presence authorization rules allow presentities to specify which users are allowed to watch their presence information. The rules also allow filtering of the presence documents by specifying which parts of the presence data is to be sent to specific watchers.

WebSphere Presence Server supports presence authorization rules. The rules are stored as XML documents in the XDMS (XML Document Management Servers (XDMS)). WebSphere Presence Server interacts with the XDMS through XCAP.

> **Note:** An external XDMS is required to support presence authorization rules. It is required to store Presence Rules documents.
>
> WebSphere Presence Server does not act as an intermediary for the management or retrieval of authorization rules documents by clients or applications. Clients and applications manage authorization rules documents directly through the XDMS.

Partial publication/notification

Publish requests might contain either the full presence document or a set of changes from the previously published presence document. If the document contains changes, WebSphere Presence Server updates the previously published presence document. It then continues the processing of the partial publish request in the same manner as a publish requests with a full presence document.

The partial notification mechanism allows a watcher to request notifications for only changed presence information. WebSphere Presence Server sends partial notifications only if the subscriber that is indicated in the SUBSCRIBE request supports partial notifications.

Watcher information

WebSphere Presence Server allows for subscriptions to watcher information. Watcher information shows the watchers that are subscribed to a presentity.

The combination of watcher information and presence authorization rules enable the use of reactive authorization, where authorization occurs through direct user intervention. You can subscribe to the watcher information for your presentity, which enables you to obtain when a new watcher is added that is not covered by the existing authorization rules. You can then add a new authorization rule for the new watcher.

A watcher information document is full or partial. A document with full state information contains the complete list of watchers for the resource. A document with partial state information contains new watchers or watchers whose state changed since the last notification was sent to the subscriber. The initial response to a watcher information SUBSCRIBE request contains the full state information. Subsequently, the server sends partial state information whenever there are changes in the watcher information.

Usage record

WebSphere Presence Server generates a usage record that describes how the service was used for accounting and billing purposes. Usage records are stored in relational table format such that you can search for them using SQL queries. Each usage record contains common event data that you can use to uniquely identify each record and references a properties table that contains application specific attributes. This provides a uniform infrastructure for creating and storing service usage records.

1.3.4 The architecture

IBM WebSphere Presence Server provides a flexible and extensible architecture that enables the composition and publication of documents for different types of objects. It allows for the extension of presence information by additional information received from external sources. The architecture also allows you to add awareness solutions to support customer needs. Figure 1-6 provides an overview of the WebSphere Presence Server architecture.

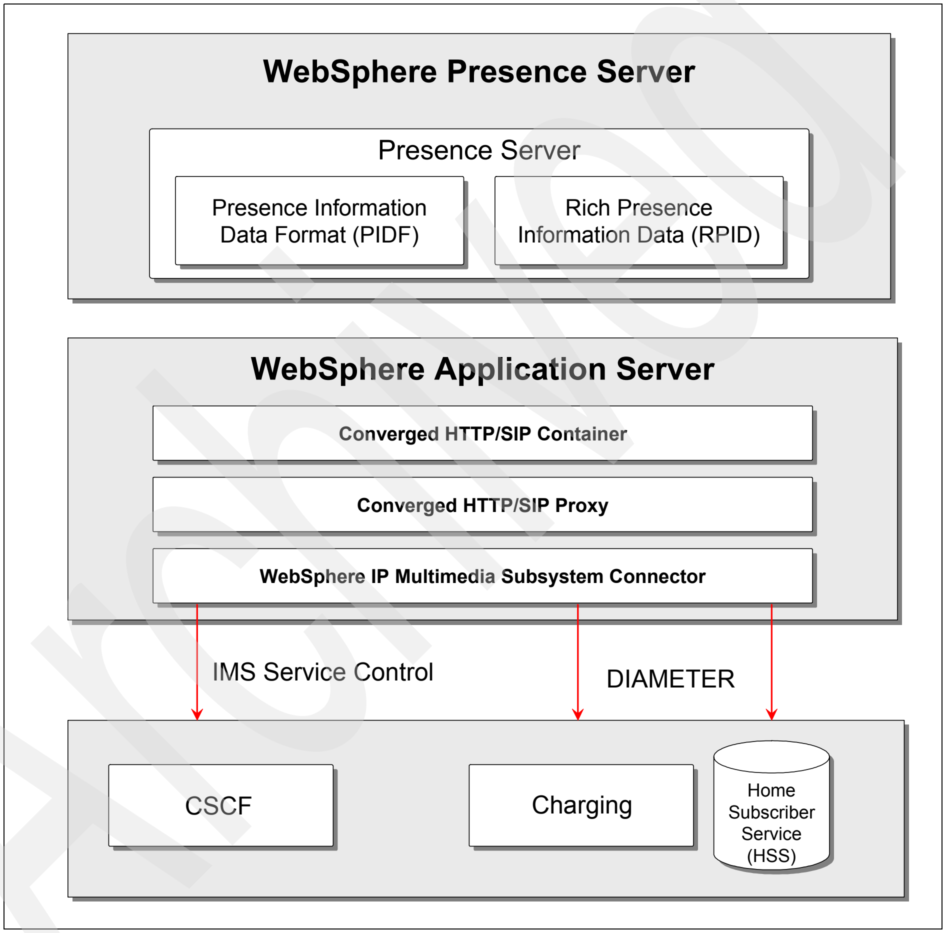


Figure 1-6 Overview of the WebSphere Presence Server architecture

WebSphere Presence Server is designed in layers. Figure 1-7 on page 19 illustrates the major components and layers.

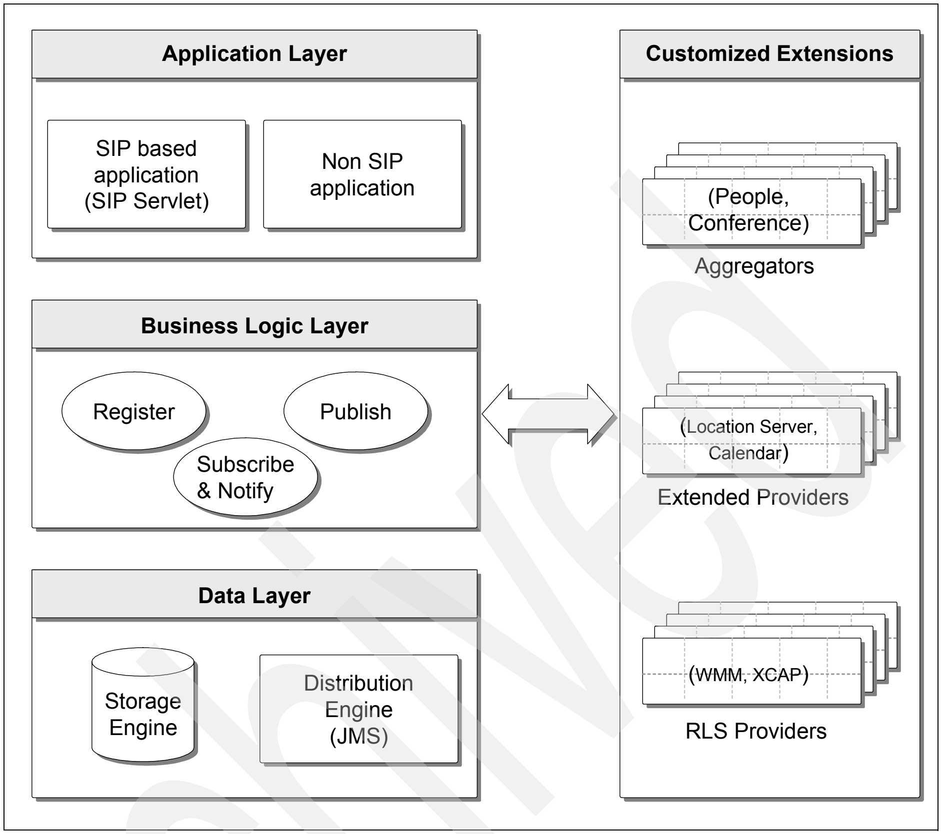


Figure 1-7 Layers of the WebSphere Presence Server architecture

The three layers are:

- Data layer

  The Data layer encapsulates a number of functions, including:

  – Providing access to presence data storage

  – Notifying presence data consumers (Presence Server components) of changes in the data

  – Distributing events on data change to all servers in the cluster

  Figure 1-8 on page 20 illustrates the functions of the data management layer.

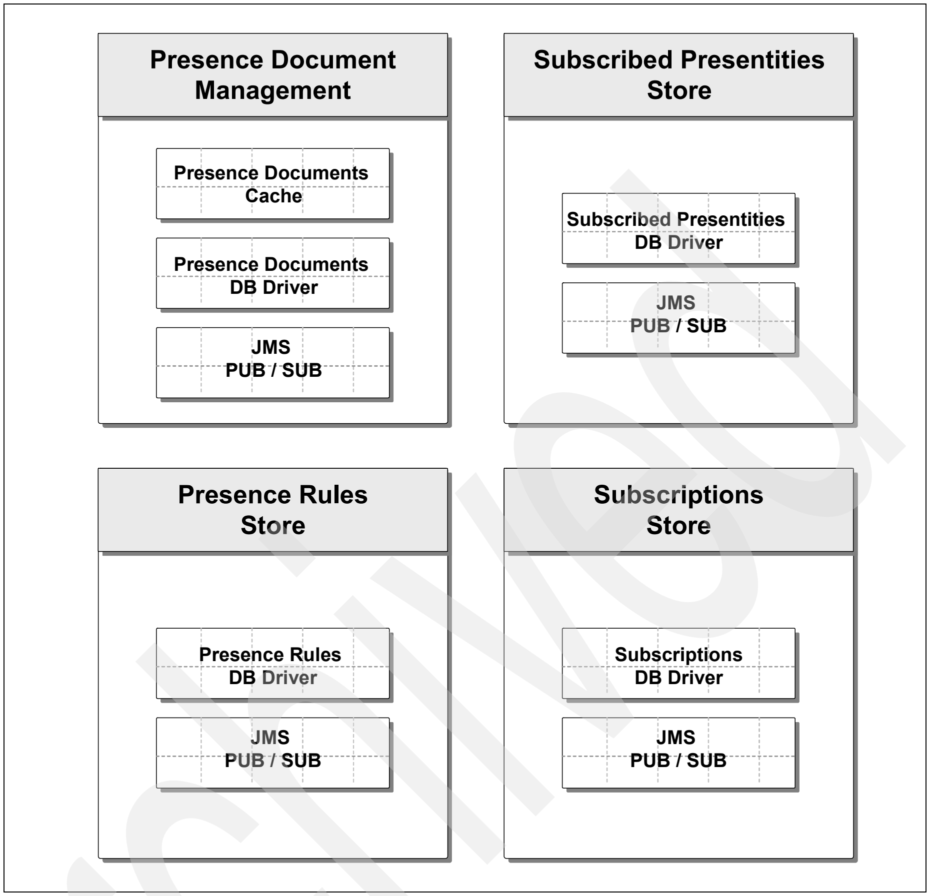


Figure 1-8 Functions in the Data layer

- Business logic layer

  The Business logic layer encapsulates the functions for handling Publish, Subscribe, and Notify operations. The functions include operations, such as Privacy, matching subscriber identities against known allowed/disallowed users, Filtering, applying rules to include or exclude elements from an XML document, and applying partial updates to XML documents.

- Application layer

  The Application Layer enables the creation of presence-based applications that use the generic presence services and specific utilities that extend the Presence Server.

Note: SIP-based applications run in the SIP container and utilize additional functionalities that the SIP container provides. These applications are the only components in the architecture that are familiar with the SIP messaging format. They are therefore responsible for validating the SIP client requests and invoking the appropriate methods in the Presence Server.

1.3.5 Key components of WebSphere Presence Server

WebSphere Presence Server is developed as a Java Platform, Enterprise Edition (Java EE) application. It consists of SIP Servlets, Enterprise JavaBeans™ (EJB™), Java Database Connectivity (JDBC), and Java Message Service (JMS).

SIP Servlets

WebSphere Presence Server implements a number of SIP Servlets to handle SIP messages. The set of Servlets include: Servlet that validates the syntax of the incoming SIP PUBLISH and SUBSCRIBE requests that are processed in the Business Logic layer, the Servlet that generates outgoing SIP SUBSCRIBE request on *reg* events, and the SIP Servlet that handles SIP NOTIFY messages from the external registrar.

Presence Service façade

WebSphere Presence Service facade is a Session Bean component that provides a framework public API. You can view this service as an entry-point for all requests. The WebSphere Presence Service implements the common functionality for all specific services, such as management calls and database calls. It invokes the specific awareness service to process the request for a specific presence.

Storage Engine (database usage)

The Storage Engine is a Session Bean component that is responsible for data storage. It uses JDBC to interact with the WebSphere Presence Server database. The Storage Engine component performs basic SQL operations SELECT, INSERT, UPDATE, and DELETE on data in the Presence Server database tables. Table 1-1 shows the tables that the WebSphere Presence Server database contains.

Table 1-1 Tables in the WebSphere Presence Server database

Table name	Description
PUBLISH	This table contains basic details regarding each published presence document that is not yet expired.
FULLDOC	This table contains the full presence information document for each presentity.
CONFIG	This table stores all Presence Server configuration attributes, such as *publishExpiration*, *documentChangeJMS*, and more.
MANAGING_SERVERS	This table contains the information of the managing server for each subscription.
WATCHERS	This table contains pending requests for subscribers on user watcher information.
PRESENCE_AUTH_RULES	Database cache for Presence Rules documents—only for users that are being interested in (Users that someone subscribed on)

Notifier (JMS usage)

Figure 1-9 on page 23 illustrates the relationship between two WebSphere Presence Servers in a cluster. When presentity publishes its document, WebSphere Presence Server invokes the Storage Engine to store this data in the database and sends a JMS message to notify other WebSphere Presence Servers in the cluster about the change. The message is sent to the appropriate topic, in the JMS server, that is responsible for publishing this kind of event. Each Presence Server has its own Message-Driven Bean (MDB) component that acts as a JMS events listener and keeps the Presence Server aware of updates.

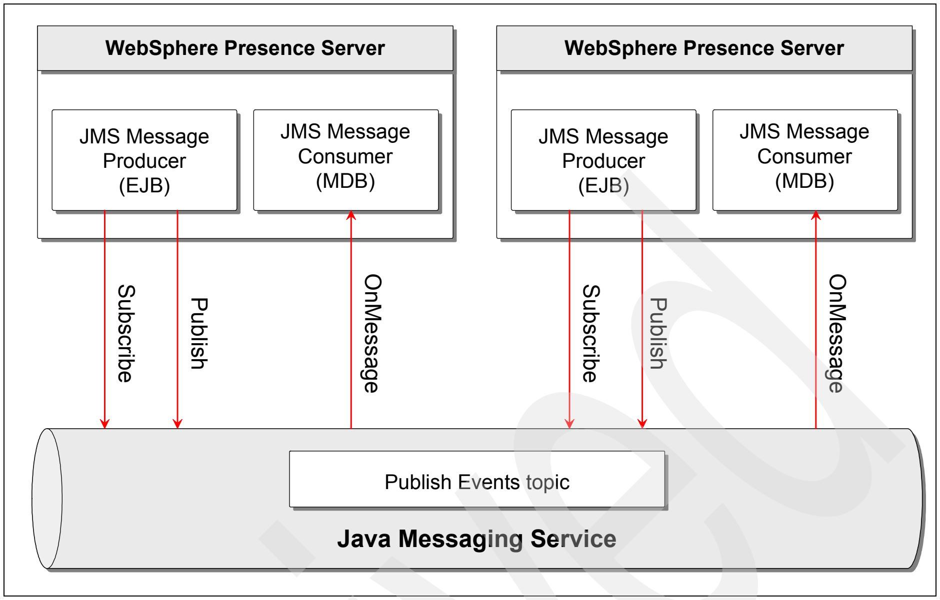


Figure 1-9 Presence information distribution across servers

Aggregators

Aggregators are utilities that assist to compose the presence document. The WebSphere Presence Server implements a set of aggregators for Presence PIDF documents with Rich Presence Information Data Format (RPID) extension).

SIP XProvider

WebSphere Presence Server implements a general external provider to interact with servers that support the SIP protocol. WebSphere Presence Server uses the external provider to subscribe to an external registrar. When the Presence Server invokes the subscribe method, the SIP XProvider generates an outgoing SIP SUBSCRIBE on *reg* event to the external registrar that is defined in the system configuration file. The external provider is also responsible for receiving notifications from the external registrar and for providing the registration data to the Presence Server. It generates a SIP PUBLISH request with the received registration information and sends the request to the Presence Server. The WebSphere Presence Server, in a regular situation when it receives published data, composes the information to an existing presence document for a presentity and stores this information in the Presence Server repository.

1.4 IBM WebSphere XML Document Management Servers (XDMS)

IBM WebSphere XML Document Management Servers (XDMS) is an IMS-compliant application that enables users, applications, and administrators to create and manage network-based group definitions and associated lists of members of those groups in XML documents.The main purpose of the WebSphere XDMS is to provide network accessible storage, retrieval, and management of XML documents that are owned by entities within the IMS solution.

The IBM WebSphere XDMS component is developed using two industry-standard network protocols:

- XML Configuration Access Protocol (XCAP)

  XCAP is an IETF specification for how XML documents are transported over HTTP.

- Session Initiation Protocol (SIP)

  SIP is an IETF specification for application-layer signaling protocol for initiating, modifying, or terminating communication and collaborative sessions over Internet Protocol (IP).

IBM WebSphere XDMS provides three standard enablers for:

- Shared List XDMS: For managing URI lists (resource-lists) and resource lists (rls-services) that are common to other IMS Components
- Resource List XDMS: For managing presence-specific groups
- Presence Rules XDMS: For handling authorization rules documents to support the IBM WebSphere Presence Server Component

1.4.1 The benefits

The core XDMS provides generalized foundation to store, retrieve, and manipulate XML documents. Enablers add specific application usage that extends the core XDMS. They add additional processing for checking constraints, data semantics, naming conventions, resource interdependencies, authorization, and other policy-based validation. Some examples of standards-based OMA Enabler XDMS include Push-To-Talk over Cellular (PoC), XDMS Enabler, and Presence Rules XDMS Enabler.

> **Note:** It does not preclude the ability to define custom XDMS Enablers that are outside of the scope of the standards.

1.4.2 XDMS in IMS

XDMS is at the heart of IMS communication services. You can use it across any number of IMS applications for users communication. IBM WebSphere XDMS can work with IMS devices that are using the IPv4 or IPv6 protocols.

Figure 1-10 illustrates the various reference points of the XDMS.

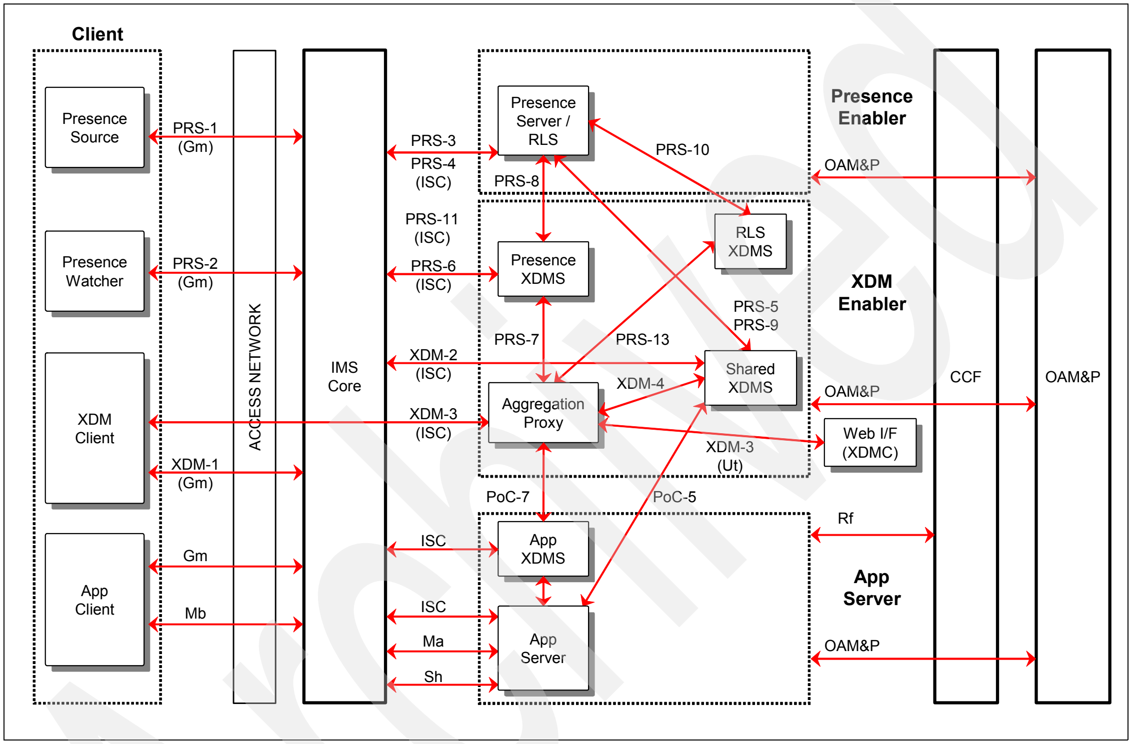


Figure 1-10 OMA XDMS architecture

IBM WebSphere XDMS supports the following reference points:

- SIP protocol reference points:
  - XDM-1 Reference Point to support the communication between the XDM Client and the SIP/IP Core network
  - XDM-2 Reference Point to support the communication between the Shared XDMS and the SIP/IP Core
- HTTP/XCAP protocol:
  - XDM-3 Reference Point to support the communication between the XDM Client and the Aggregation Proxy

– XDM-4 Reference Point to support the communication between the Aggregation Proxy and the Shared XDMS
– PRS-5 Reference Point to support the communication between the Shared XDMS and the Presence Server
– PRS-7 Reference Point to support the communication between the Aggregation Proxy and Presence Rules XDMS
– PRS-9 Reference Point to support the communication between the Shared XDMS and Resource List Server (RLS)

- ISC Interface

  WebSphere XDMS can interact with IMS networks through the WebSphere IMS Connector, which provides:

  – IMS Service Control (ISC):
    - ISC Interfaces enable communication with the Call Session Control Function (CSCF) in the IMS control plane.
    - Interoperability tested with all of the major Network Equipment Providers.

  – DIAMETER:
    - Rf - The accounting information (usage records) from the PS DB can be passed through the Rf interface to a CCF.
    - Sh – The connector enables the PS to use the Sh interface to connect to an HSS.
    - Ro – Online charging.

> **Note:** WebSphere XDMS does not use the DIAMETER interfaces. Services can utilize the DIAMETER interfaces to create IMS assets, for example, XDMS writes usage records to the database. A services asset could inject those records through the Rf interface.

Figure 1-11 on page 27 illustrates the WebSphere IMS Connector.

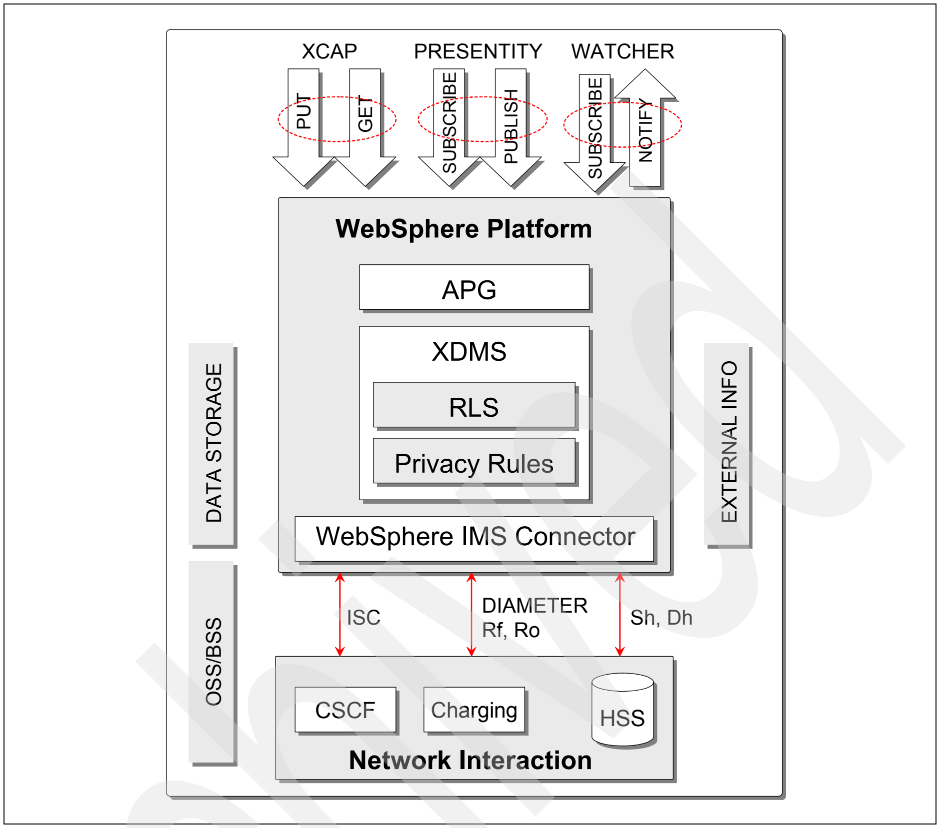


Figure 1-11 WebSphere IMS Connector

Through these interfaces, the WebSphere XDMS can:

- Send and receive SIP traffic through a CSCF
- Manage HTTP traffic through the Aggregation Proxy
- Generate usage records into an external database
- Generate JMX™ events for alerts and monitoring

1.4.3 The basic functionality

The IBM WebSphere XML Document Management Servers (XDMS) implements a number of basic functionalities for manipulating XML documents. The functions include XML repository, Presence XDMS, and Shared XDMS.

XML repository

The IBM WebSphere XDMS component enables the storage, access, and manipulation of XML documents that are stored in a set of centralized DB2® databases. The databases contain tables that are identified and named using an Application Unique Identifier (AUID). The stored documents can then be accessed and manipulated using one of two protocols:

- XML Configuration Access Protocol (XCAP)

  The IETF specification for how XML documents are transported over HTTP. XCAP maps XML document sub-trees and element attributes to HTTP URIs so that HTTP can directly access these components.

- Session Initiation Protocol (SIP)

  The application-layer that signals the protocol for initiating, modifying, or terminating communication and collaborative sessions over Internet Protocol (IP).

IBM XDMS reduces overhead and bandwidth by enabling the selection and manipulation of specific elements within an XML document instead of the entire document.

Figure 1-12 on page 29 illustrates the flow of XCAP and SIP requests through the system.

XCAP Flow

In this section, we provide the process for the XCAP message flow:

1. The XCAP Servlet receives an XCAP request.
2. The Servlet creates a chain of XCAP Filters based on the AUID.
3. The XCAP Servlet invokes the first XCAP Filter in the chain.
4. As each XCAP Filter completes, it invokes the next XCAP Filter in the chain.
5. When the request finally reaches the XCAP DataStore Filter, the DataStore Filter performs the respective XCAP operation (GET, PUT, DELETE, and so on.)
6. Control is returned back through each Filter.
7. The XCAP Servlet now regains control and passes the response back to the client.

SIP Flow

In this section, we provide the process for SIP:

1. The SIP Servlet receives a SIP Subscribe request for a certain document that IBM XDMS manages.

2. The Servlet creates a chain of SIP Filters.
3. The SIP Servlet invokes the first SIP Filter in the chain.
4. As each SIP Filter completes, it invokes the next SIP Filter in the chain.
5. When the request finally reaches the SIP Authorization Filter, the SIP Authorization Filter determines if the requestor has at least READ authority on the document.
6. The DataStore Filter determines if the document is in the database.
7. Control is then passed back through each Filter.
8. The SIP Servlet now regains control and passes the response back to the client with the response.

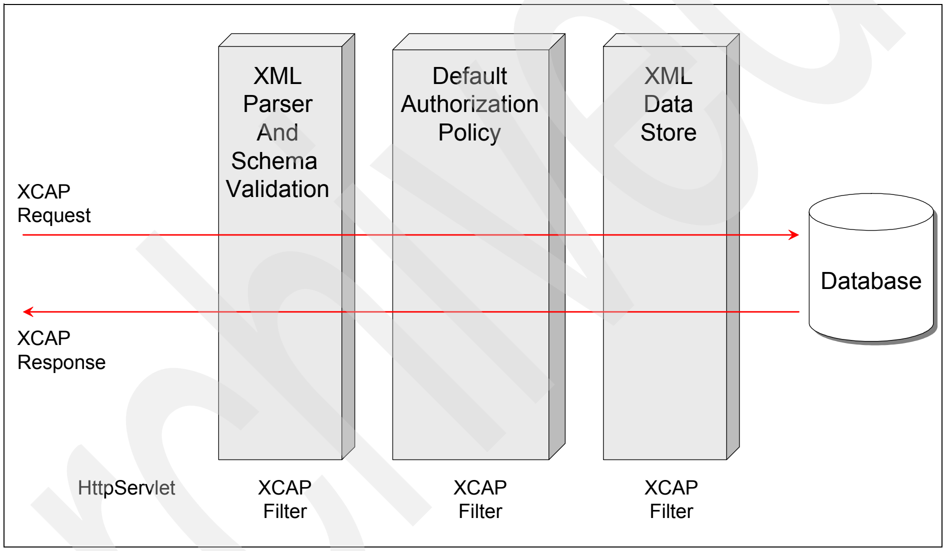


Figure 1-12 XCAP message flow

Presence XDMS

IBM WebSphere XDMS implements the Presence XDMS that acts as an XCAP server and SIP Notifier. The Presence XDMS supports the following functions:

- Manages Presence authorization policy XML documents that are specific to the Presence service enabler.
- Enables single subscriptions to changes to multiple documents that are stored in the Presence XDMS.
- Notifies subscribers of changes to Presence-specific documents that are stored in the network.

RLS XDMS

IBM WebSphere XDMS implements the RLS XDMS that acts as an XCAP server and SIP Notifier. The RLS XDMS supports the following functions:

- Manages Presence group's XML documents, which are specific to the use of an RLS.
- Enables single subscription changes to multiple documents that are stored in the RLS XDMS.
- Notifies subscribers of changes to such documents that are stored in the network.

Shared XDMS

The Shared XDMS is responsible for storing and managing the shared lists and the group lists in the network. The IBM SharedList XDMS supports both the resource-lists (URI Lists) and rls-services document storage, as defined by the OMA-TS-XDM_Shared_List-V2_0 specification.

The resource-list's AUID allows you to create shared documents that store lists of members. Standard support for XCAP and SIP are provided for resource-lists documents. Validation or constraint checks are enforced as mandated by the previously stated specifications. In addition, the IBM resource-lists enforce the policy for maximum external list depth and recursion checks for <external> references that are local to the XDMS. The <external> references can be expensive because they require multiple XCAP requests to traverse the list hierarchy. The IBM SharedList XDMS limits how deeply <external> references can be nested. OMA provides an extension to the resource-lists to specify the <appusages> element. The element contains an XCAP URI of another application element that references a list. The SharedList XDMS uses the <appusages> element to denote which <external> references are made against a particular list. This provides an easy way for the SharedList XDMS to traverse the parent hierarchy to determine recursion.

> **Note:** The `<external>` references to another server besides the local SharedList XDMS are not checked for their depth or for recursion.

The rls-services AUID allows you to create RLS documents that can either store embedded lists or reference a local resource-lists document that is stored within the SharedList XDMS. Standard support for XCAP and SIP are provided for rls-services documents. Any validation or constraint checks are enforced as mandated by the above specifications. Even though the SharedList XDMS checks for <external> depth and recursion for resource-lists AUID, the same checks are not applied to the rls-services lists because they are essentially hidden. No <external> elements should reference an rls-services list.

IBM SharedList XDMS has two primary scenarios for handling SIP subscriptions to rls-services <service> elements:

- If the <service> element contains a <resource-list> element that references a specific resource list within a resource-lists document (XCAP with node selector), the subscription is not against the specific resource list but against the entire resource-lists document in which it is contained; therefore, changes to another list within the same document causes the subscriber to be notified of the change.
- If the <service> element contains a <list> element, then only changes to the embedded list element causes the subscriber to be notified of the change. Changes to other <service> elements within the same rls-services document do not cause the subscriber to be notified of a change.

1.4.4 The architecture

WebSphere XDMS provides three standard Enablers:

- SharedList XDMS
- Presence Rules XDMS
- Resource List XDMS

These Enablers describe specific usage of XDMS along with specialized schemas and XML constraints that are associated with each enabler. The XDMS platform provides the foundation on which Enablers build specific application logic to handle things, such as schema validation, data constraints, custom authorization, and custom policy enforcement.

Figure 1-13 on page 32 shows the WebSphere XDMS architecture.

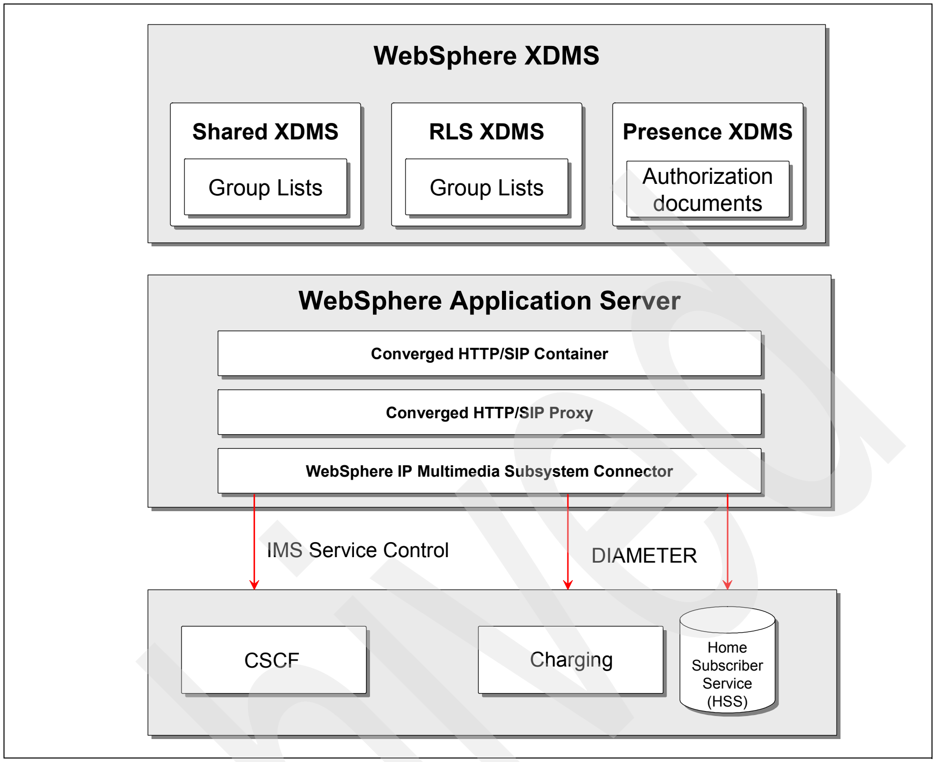


Figure 1-13 WebSphere XDMS architecture

The IBM WebSphere XDMS implementation provides an extensibility framework for use in defining XCAP and SIP processing for custom Enablers. You can define custom Enablers to be flexible enough to handle any scenario while optimizing performance.

1.4.5 The key components of WebSphere XDMS

The IBM WebSphere XDMS implementation consists of two key components: the Aggregation Proxy and the Filter Chains.

Aggregation Proxy

An Aggregation Proxy is a component included with the WebSphere XDMS that routes incoming messages to both the WebSphere XDMS or third-party XDMS elements. Aggregation Proxy is the contact point for the XDM Client that is implemented in a User equipment (UE) to access XML documents that are stored in any XDMS. Figure 1-14 on page 33 shows the architectural overview of the Aggregation Proxy.

As illustrated Figure 1-14, the proxy is built as an HTTP Servlet in the Web container based on the WebSphere Application Server product. The Aggregation Proxy receives the requests from XDM clients and forwards them to the appropriate Application Servers. The Trust Association Interceptor (TAI) is the authentication component that is used to guarantee the legitimacy of XDM clients.

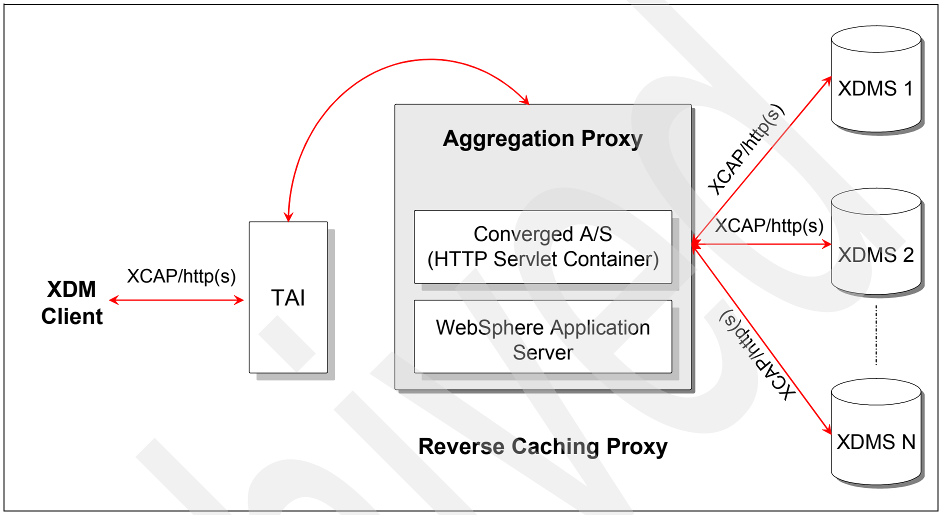


Figure 1-14 Architecture overview of Aggregation Proxy

Filter chains

The XCAP Filter Chain describes the sequence of XCAP Filter classes which process XCAP requests. Similarly for SIP, the SIP Filter Chain is defined, which describes the sequence of SIP Filter classes that process SIP requests.

The XCAP extensibility model follows closely with the HTTP Servlet Filter model where multiple filters are chained together to handle either XCAP or SIP request/response for specific tasks. Each application usage (AUID) has its own descriptor file that describes the application usage and the XCAP and SIP Filter Chains, which contain a sequence of corresponding XCAP or SIP Filter classes.

2

About the scenario

In this chapter, we describe the presence service scenario for realizing the highly available and scalable environments in this Redbooks publication. We present the scenario and discuss the implementation approach.

This chapter contains:

2.1 The scenario overview

A common approach for deploying new services in the enterprise or Telecom environment is to start small and grow the service offering. This approach can be realized using different strategies that include starting with a small set of functions and growing the capabilities or starting with a small number of users and growing the user base. Often, the functionality and level of service grow incrementally with deployment.

In this IBM Redbooks publication we consider a presence-enabled service scenario configured using the IBM WebSphere Presence and XML Document Management Servers (XDMS), starting with a small number of users and growing the user base. Starting with a base setup, the configuration is such that it can support the increase in the user base and growth in the number of messages in the environment.

Presence technology was initially conceived for use in Instant Messaging services to enable users to see if a friend or colleague is online and available to exchange messages. The use of presence expanded to include the online status and availability of devices and services, in addition to users. The presence services interaction model includes three core entities, as illustrated in Figure 2-1:

- Presentity (presence entity) - provides presence information to Presence Services
- Watcher - requests presence information about a presentity
- Presence service - accepts, stores, and distributes presence information

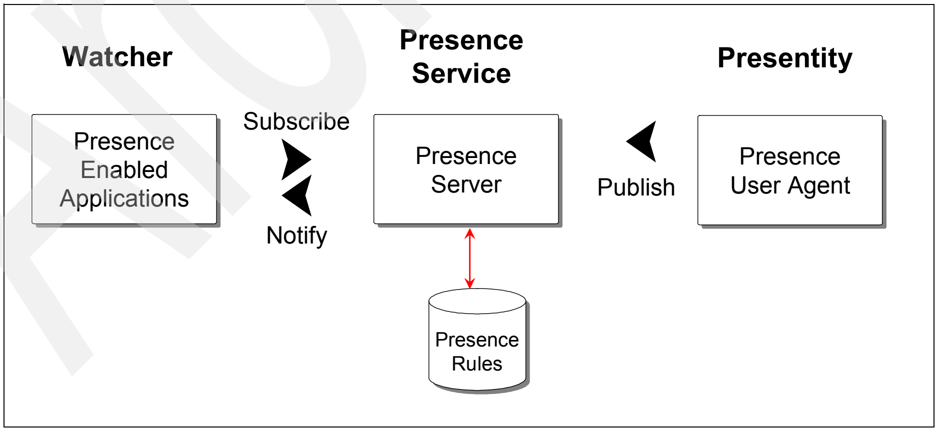


Figure 2-1 Presence technology interaction model

Mobile Instant Messaging is a typical presence-enabled service, where users connect to lists of contacts who publish their presence. In this Redbooks publication, we use Mobile Instant Messaging for our scenario to demonstrate the base, highly-available and scalable environments based on WebSphere Presence Server and WebSphere XML Document Management Servers (XDMS). The presence interaction that Figure 2-2 depicts is described as follows:

- Alice, Bill, Bob, John, Jane, and Joe are all subscribers to a Mobile Instant Messaging service.
- Alice has 5 buddies namely Bill, Bob, John, Jane, and Joe.
- Alice is interested in knowing the status and availability of her buddies to initiate Instant Messaging sessions.
- To get the status, Alice sends a SUBCRIPTION message to the network to get the presence status of her buddies.
- Alice gets a NOTIFY message for each PUBLISH by each presentity (her buddies).

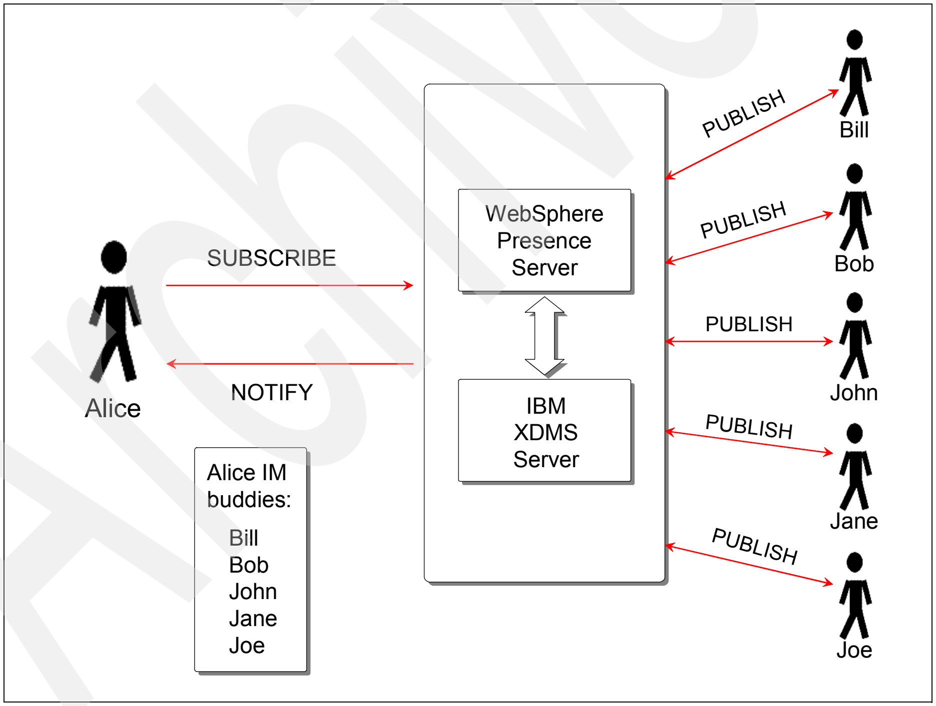


Figure 2-2 Presence notification scenario

Consider that as a service provider, having completed a Mobile Instant Messaging pilot project, you are now ready to proceed with the full service rollout. You typically establish Service Level Agreements (SLAs), and you will most likely want the service to be highly available. Likewise, as the user base grows, the infrastructure should easily scale to support the increase in the messaging traffic. The design and architecture presented in this IBM Redbook's publication demonstrates how to create highly-available and scalable solutions for this type of scenario using the IBM WebSphere Presence and the IBM XML Document Management Servers (XDMS).

Figure 2-3 shows the scenario support for increased user base.

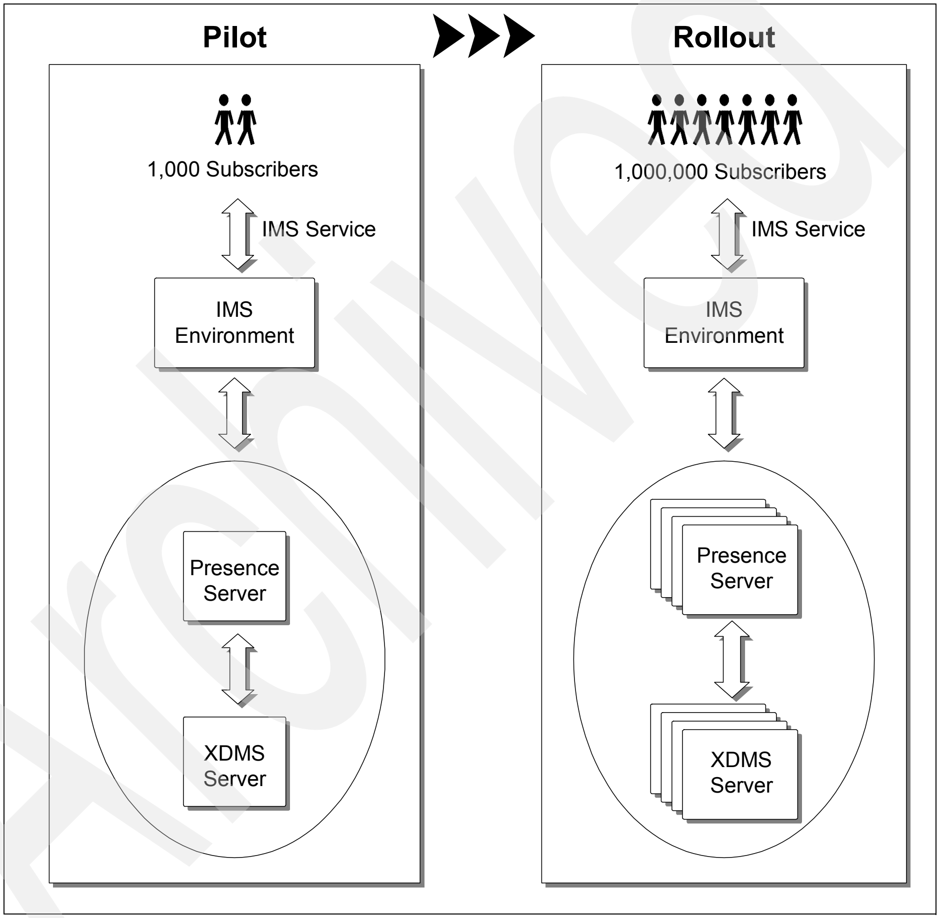


Figure 2-3 Scenario support for increased user base

2.2 The Presence and XDMS services

The WebSphere Presence and IBM WebSphere XML Document Management (XDMS) Servers provide a number of services that enable a user to subscribe to presence services in a telecom services provider network. Figure 2-4 illustrates the services.

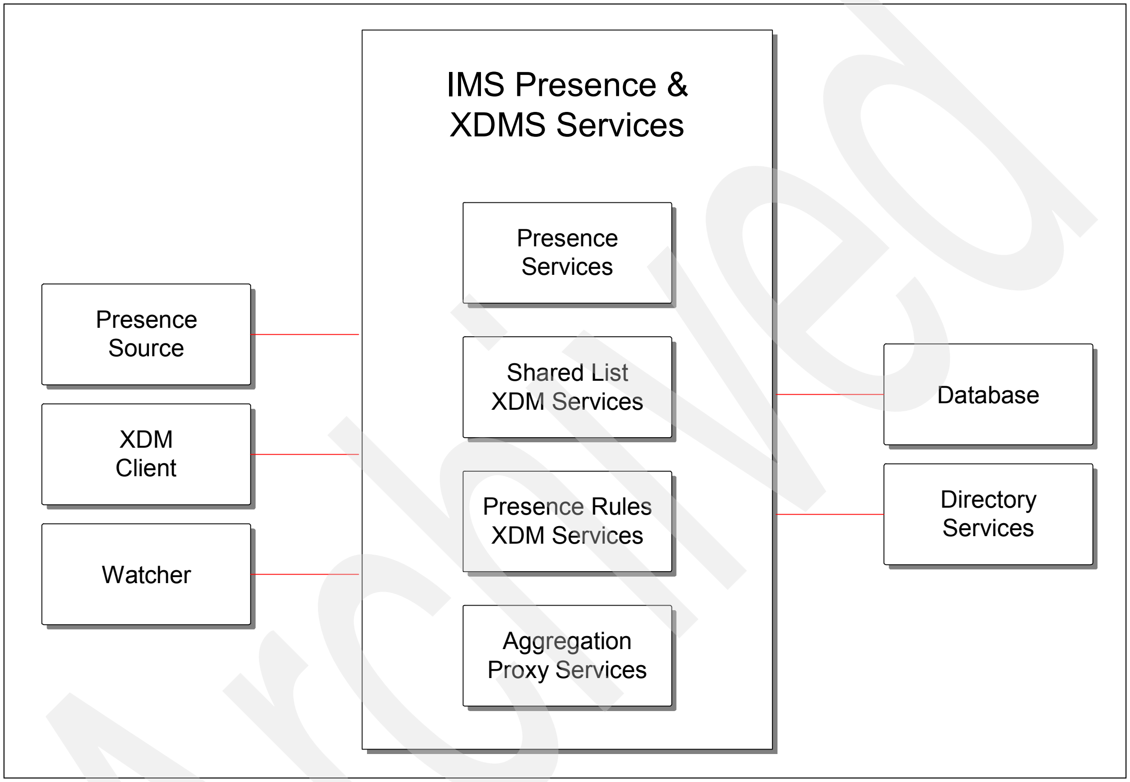


Figure 2-4 IMS Presence and XDMS services

- Presence services

  These are a collection of services that are provided to users who subscribe to presence services in a Telecom services provider network. The services include:

  – Presence publish service

    Lets users publish their presence or change their presence information in the network. The published presence information is available to other users in the network who are authorized to see the user's presence information. The authorization is controlled by a set of Presence Rules that the user sets.

– Subscription to presence service

  Lets a user subscribe to changes in the presence information of other users in their buddy list. The list of users in the buddy list are normally other users in the network that authorized the subscriber to view their presence information changes through a set of predefined rules of white and black lists.

– Notification to Presence changes service

  This service is provided to all presence subscribers to get notification of changes in the presence information of the presentitys they are subscribed to.

– Presence Watcher Info service

  This service, defined by IETF RFC 3857 and 3858, is provided by the Presence Server. Watcher information consists of the current state of all subscriptions to a particular resource. Monitoring watcher information enables you to control access to a resource because it provides the raw data that you need to create and deploy presence authorization rules.

- Shared List services

  The IBM XDMS Server provides services for managing (storage and retrieval) user-defined XML documents, such as buddy lists that are stored in a datastore. The documents are used along with the Presence Server to provide presence services. The XML document management services that the XDMS Server provides are:

  – Resource List service

    Resource List is an XML document that can have one of more lists with each list that contains a set of entries, with each entry representing a resource (SIP) URI. Example 2-1 shows a resource list document from the scenario implementation in this Redbooks publication.

Example 2-1 Sample Resource List document from the scenario implementation

```
<message>
    <!-- header -->
    <![CDATA[PUT
/services/resource-lists/users/alice@itso.ral.ibm.com/alicefriends.xml
HTTP/1.1
      Content-Type: application/resource-lists+xml
      X-XCAP-Asserted-Identity: "sip:alice@itso.ral.ibm.com"
      Host: xdms.itso.ral.ibm.com:9082
      Connection: keep-alive
```

```
User-Agent: seagull]]>
<!-- body -->
<![CDATA[<?xml version="1.0" encoding="UTF-8"?>
  <resource-lists xmlns="urn:ietf:params:xml:ns:resource-lists">
  <list name="alicefriends">
    <display-name>Alice Buddies</display-name>
    <entry uri="sip:bill@itso.ral.ibm.com">
      <display-name>Bill</display-name>
    </entry>
    <entry uri="sip:bob@itso.ral.ibm.com">
      <display-name>Bob</display-name>
    </entry>
    <entry uri="sip:jane@itso.ral.ibm.com">
      <display-name>Jane</display-name>
    </entry>
    <entry uri="sip:joe@itso.ral.ibm.com">
      <display-name>Joe</display-name>
    </entry>
    <entry uri="sip:john@itso.ral.ibm.com">
      <display-name>John</display-name>
    </entry>
  </list>
  </resource-lists>]]>
</message>
```

– RLS-Service

  The RLS service document is an XML document that defines URI that represent the services and the services that are provided for that URI by the Resource List Server (RLS), which is the XDMS Server in our case.

  The RLS service document refers to the resource lists documents or can contain local resource lists. When local resource lists are used, the lists are available only to that RLS-Service. Example 2-2 shows an example of the message section of an RLS-Service document.

Example 2-2 Sample RLS-Service document from the scenario implementation

```
<message>
       <!-- header -->
    <![CDATA[PUT
/services/rls-services/users/alice@itso.ral.ibm.com/alicefriends.xml
HTTP/1.1
      Content-Type: application/rls-services+xml
      X-XCAP-Asserted-Identity: "sip:alice@itso.ral.ibm.com"
      Host: xdms.itso.ral.ibm.com:9082
```

```
      Connection: keep-alive
      User-Agent: seagull]]>
    <!-- body -->
    <![CDATA[<rls-services xmlns="urn:ietf:params:xml:ns:rls-services"
xmlns:rl="urn:ietf:params:xml:ns:resource-lists">
      <service uri="sip:alicefriends@itso.ral.ibm.com">

<resource-list>http://xdms.itso.ral.ibm.com:9082/services/resource-list
s/users/alice@itso.ral.ibm.com/alicefriends.xml/~~/resource-lists/list%
5b@name=%22alicefriends%22%5d</resource-list>
        <packages>
          <package>presence</package>
          <package>test</package>
        </packages>
      </service>
    </rls-services>]]>

    </message>
```

- Presence Rules service

  The Presence Rules service creates and manipulates presence authorization rules documents. Presence Rules documents are defined for subscribers to validate if a subscriber is allowed to view the presence state changes of a presentity. Example 2-3 shows a fragment of the Presence Rules document for “Alice” showing the presence rule for subscriber “Bill”.

Example 2-3 Sample Presence Rules document for Alice

```
<?xml version="1.0" encoding="UTF-8" ?>
    <cr:ruleset xmlns="urn:oma:xml:prs:pres-rules"
xmlns:pr="urn:ietf:params:xml:ns:pres-rules"
xmlns:cr="urn:ietf:params:xml:ns:common-policy">
      <cr:rule id="bill">
        <cr:conditions>
          <cr:identity>
            <cr:one id="sip:bill@itso.ral.ibm.com"/>
          </cr:identity>
        </cr:conditions>
        <cr:actions>
          <pr:sub-handling>allow</pr:sub-handling>
        </cr:actions>
        <cr:transformations>
          <pr:provide-services>
            <pr:all-services/>
```

```
        </pr:provide-services>
        <pr:provide-persons>
          <pr:all-persons/>
        </pr:provide-persons>
        <pr:provide-devices>
          <pr:all-devices/>
        </pr:provide-devices>
        <pr:provide-all-attributes/>
      </cr:transformations>
    </cr:rule>
  </cr:ruleset>
```

- Aggregation Proxy Service

  Aggregation Proxy service routes requests based on the application (Shared List or Presence Rules) usage to the corresponding IBM XDMS enablers or applications.

  > **Note:** XDMS enablers in accordance to Open Mobile Alliance (OMA) XML Document Management 1.0 architecture are defined as applications that the XDMS Server provides that service shared list and Presence Rules documents.

Now we examine how the services can be used for presence processing in our Mobile Instant Messaging scenario.

User Alice adds her presence buddy list

Alice's presence buddy list:

- Alice, which is a SIP/XCAP client, sends an XCAP PUT request from her device to store her buddy list. The buddy list document is a resource list document that contains the list of Alice's buddies.
- The Share List Server processes the XCAP PUT request from Alice by storing the resource list XML document in the DB2 datastore.
- After processing the request, the Shared List Server notifies Alice of the successful storage of the document.

> **Note:** Alice could also query her buddy list to update her existing buddy list. In case of a query, Alice sends an XCAP GET request that results in Alice getting back her resource list document from XDMS.

User Alice subscribes to her presence buddy list

User Alice subscribes to her presence buddy list:

- Alice is interested in knowing the presence status of her buddies in her buddy list. To get the presence status, she sends in a SIP SUBSCRIBE request. Alice is Notified when her buddies publish changes in their presence status to the network.
- The Presence service processes the SUBSCRIBE request by sending it to the XDMS Server to verify that Alice is authorized to view the presence status of her buddies. Each buddy in Alice's buddy list can add a presence rule document listing so that people can view the change in his or her presence status. The Presence Rules documents are handled and stored in the datastore by the Presence Rules service.
- After the authorization check is performed on Alice, the Presence Server sends a notification to Alice when any of her presence-authorized buddies change their presence status by sending in a publish to the presence service.

The Figure 2-5 on page 45 illustrates the message flow for just three users, Alice, Bill, and Jane, who subscribe to each others group list. In addition to the group list that each user creates, an authorization rules document is created and managed to control who can see a particular users presence information. This document is stored in the Presence Rules XDM Server.

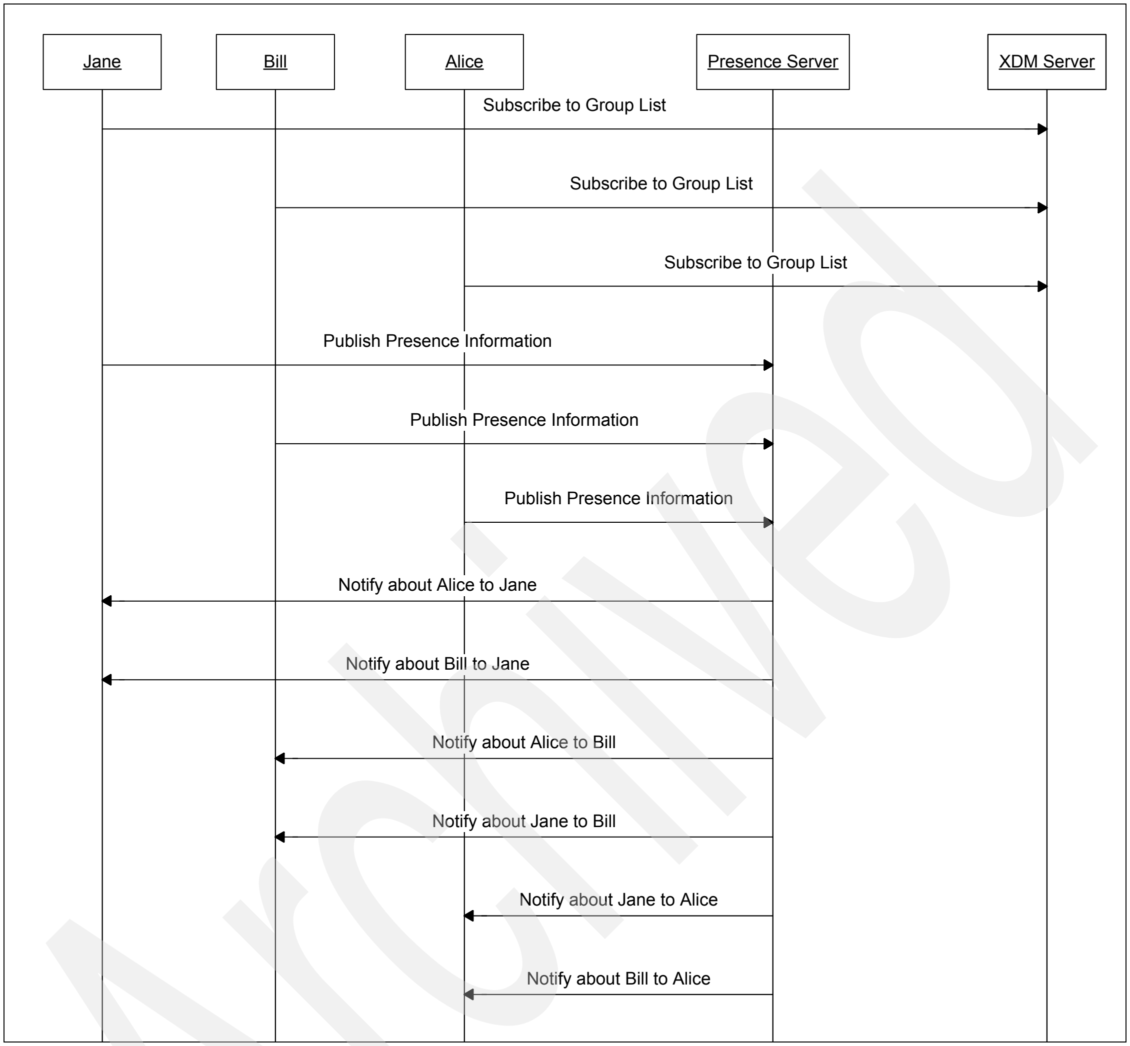


Figure 2-5 Simulated sequence diagram

2.3 Implementation approach

For the purpose of this Redbooks publication, we are not going to focus on the capabilities of the IBM WebSphere Presence Server and the IBM WebSphere XML Document Management Servers (XDMS). Rather, our focus is on how to deploy highly-available and scalable Presence and XDMS services. In essence, we treat the services as a black box, as illustrated in Figure 2-6 on page 46.

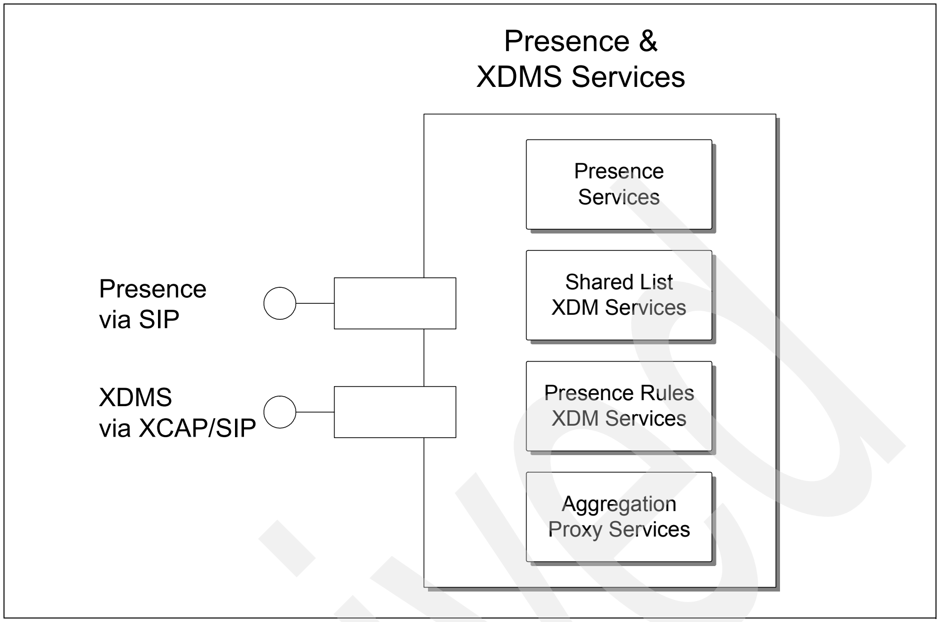


Figure 2-6 Black box view of Presence and XDMS service

> **Note:** To deploy highly-available and scalable solution, you must consider availability and scalability for other components, such as hardware (multiple power supplies in a system, use of RAID on disk subsystems, and so on), networking resources (redundant network adapters, network capacity, and so on), load balancers, authentication servers (LDAP), and the database servers. While these are important topics that must be addressed in a production environment, they are outside of the scope of this book. There are many books and articles that are available on these topics, and many IBM products provide or assist in providing these capabilities.

Three-phase implementation

We identified three phases that we follow to realize the scenario that will create and deploy a highly-available and scalable Presence and XDMS services. Rather than increase the functionality or capabilities of the services, we choose to grow the scenario by increasing the level of support (availability) and increasing the subscriber base (scalability). Figure 2-7 on page 47 illustrates these phases.

For the rest of this Redbooks publication, we describe each phase, define the requirements, discuss the approaches to meet the requirements, and implement the phase in the context of the scenario.

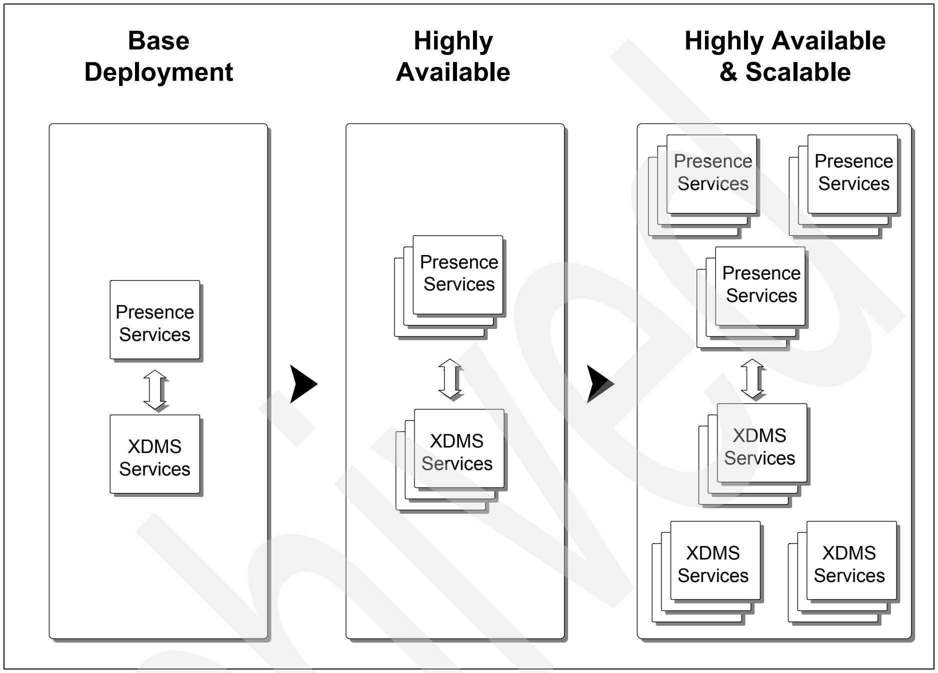


Figure 2-7 Scenario phases

Implementation objectives

The objectives of the scenario can best be described in terms of the three following phases:

- Base Deployment:
  - Build a base deployment of the Presence and XDMS black box services.
  - Leverage common components and technologies to build an environment that can be made highly available and scaled.
  - Validate that the base deployment provides the functions defined by the black box service interfaces.
- Highly Available Deployment:
  - Leveraging the base deployment, add the required components and services that are required to provide fail over capability of all services.

- Validate that the highly available deployment not only provides the functions of the black box service but does so even as failures occur in the deployment (for example, machine crash or network link failure).

- Highly Available, Scaled Deployment:
  - Leveraging the highly available deployment, scales the solution to meet new growth requirements.
  - Validates that the new deployment utilizes all available resources such that it provides adequate service for the larger subscriber base.

Note: Sizing and capacity planning are critical aspects of defining growth of any solution and are used to determine the right amount of availability and scalability that is required for a given solution. Part of the process includes some level of performance monitoring and analysis, growth projection, cost analysis, and justification. All of these variables are important when determining when more capacity should be added (scalability) and when more redundancy should be added based upon service level agreements (availability). However, these variables are very dependant on actual deployment. Because this information is very dependant on the actual deployment and characteristics of the usage patterns, we have not included it in our scenario objectives.

2.3.1 Scenario test cases

Using the simulated Mobile Instant Messaging application, we execute a series test cases to validate the proper installation and configuration of the three deployment environments:

- Base deployment environment scenario

  The base deployment environment provides a basic presence and XDMS environment that is appropriate for a proof-of-concept or pilot program. It is configured to provide functional Presence and XDM services, which you can enhance to provide both high availability and scalability. Using the simulated application, multiple instances of the test cases are run to validate that the Presence and XDM capabilities are operating properly.

- Highly-available (HA) deployment environment scenario

  The HA deployment is an extension to the base deployment environment and shows how horizontal clustering of the Presence Server, Shared List Server, Presence Rules Server, and associated Proxy Servers is achieved. Adding these capabilities allows a service provider to enhance the level of service and reliability of the services. Two categories of tests are conducted in the HA deployment environment:

– Planned shutdown of Presence Server

  In this case, we consider controlled shutdown situations, such as maintenance shutdown of a Presence Server in the cluster during the execution of multiple instances of the simulated Mobile Instant Messaging application. Another variant that we consider is the shutdown of a single instance of more than one of the Presence, XDM, and Proxy Servers.

  These tests establish the following functions of the HA environment:

  - Continued Presence and XDM services to new and existing sessions when a single instance of any server is taken out of active service
  - Continued Presence and XDM services to new and existing sessions when a single instance of multiple server types are taken out of active service

– Unexpected shutdown of Presence Server

  This test case illustrates the fail over capabilities of an unexpected shutdown of a server as a result of the termination of a Java Virtual Machine process while the server is processing incoming requests. For the purpose of this scenario test case, we consider unexpected shutdown of one of the cluster members in Presence Cluster. In highly available environments, this test leverages the following infrastructure services in the event of an unexpected shutdown of one or more servers:

  - The high availability manager component of the WebSphere Application Server infrastructure handles the requests targeting them to available servers. In the Presence Server environment that we configured for this scenario, all of the server instances are configured as members of DefaultCoreGroup. This setting enables high availability through a peer-to-peer exchange of work load management information.
  - The replication domain components of the WebSphere Application Server infrastructure allows for replication of SIP sessions in case a SIP dialog is established. In the Presence Server environment, separate replication domains are created for each cluster. This setting ensures servicing of SIP sessions that are already established in the event of an unexpected shutdown of a server in the cluster.

- Scalability deployment environment scenario

  As part of the growth case, the next deployment scenario demonstrates a scalable environment. In this scenario, we describe the transition from a high availability environment to achieve scalability. This exercise involves both:

  – Vertical scaling, which is the addition of Application Server instances to existing machines to better utilize existing capacity, memory, processor, and so on.

  – Horizontal scaling, which is the addition of additional machines that host Application Server instances).

  Using the simulated Instant Messaging (IM) application, we validate that the new server and Application Server instances are being utilized and providing additional capacity to the Presence and XDM services.

2.4 Implementation overview

We use the following artifacts to design and architect the scenario and to describe the different aspects of the infrastructure:

- System context diagram

  The system context diagram illustrates how the Presence and XDMS Server components that are in the infrastructure inter-operate with other systems. It also establishes boundaries on the scope of the scenario and highlights interfaces that communicate with the environment.

- Architecture overview diagram

  The architecture overview diagram captures the building blocks of the scenario solution. It provides the conceptual overview and relationships between components, nodes, connections, data stores, users, and external systems.

- Operational model

  The operational model defines and documents the distribution of the Presence and XDMS Server components onto geographically distributed nodes, together with the connections that are necessary to support the required component interactions. In our scenario, this model is progressively enhanced from the base to highly available and scalable aspects of the infrastructure.

2.4.1 System context diagram

The system context diagram, which we illustrate in Figure 2-8 shows an overview of the Presence and XDMS Server infrastructure. Figure 2-8 illustrates the communications between users (UA-n) and external systems. The external systems are:

- User repository
- Database
- Load Balancer

Figure 2-8 shows the solution architecture that is designed for the purpose of this book. In a real production environment, the Presence and XDMS Server infrastructure are part of a much bigger infrastructure and connected to Core IP/IMS Network as well as to other telecommunications network elements.

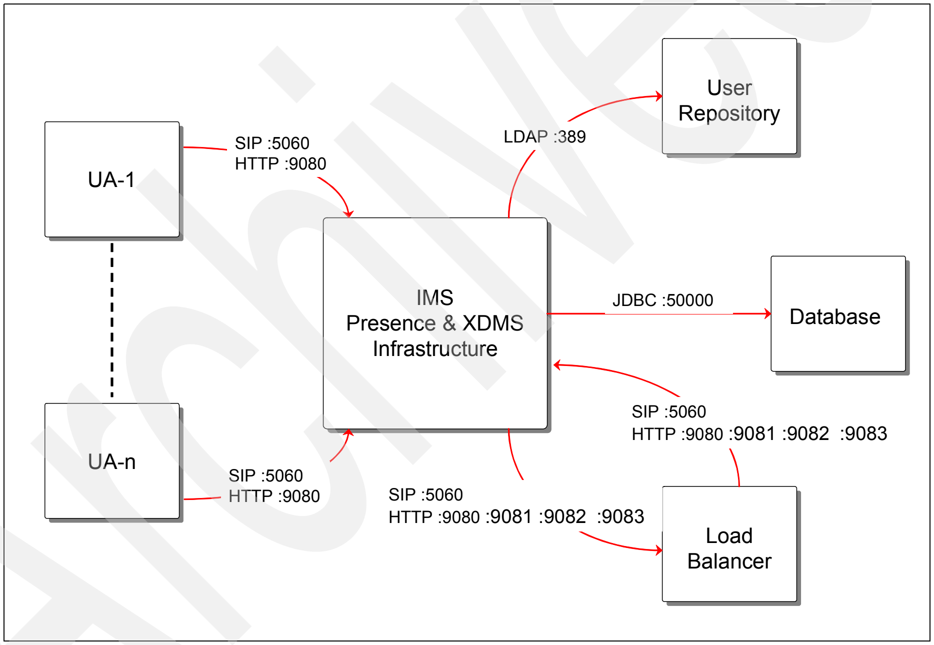


Figure 2-8 System context diagram

2.4.2 Architecture overview diagram

The architecture overview includes the services view that focuses on the Presence and XDMS services as well as the systems view, which focuses on the components and the relationships between the components.

Figure 2-4 on page 39 illustrates the architecture services view. It shows the services that enable a user to subscribe to presence services in a Telecom services provider network.

The systems view, as illustrated in Figure 2-9, provides additional levels of detail and shows the Presence and XDMS components of the architecture. The components support two types of functionalities:

- Functional requirements - these functionalities support the Presence and XDMS services.
- Non-functional requirements - these support functionalities, such as availability provided by Proxy Nodes.

The components have clearly defined APIs and protocols for communicating with them and should be thought of as logical elements and not necessarily as servers or physical machines.

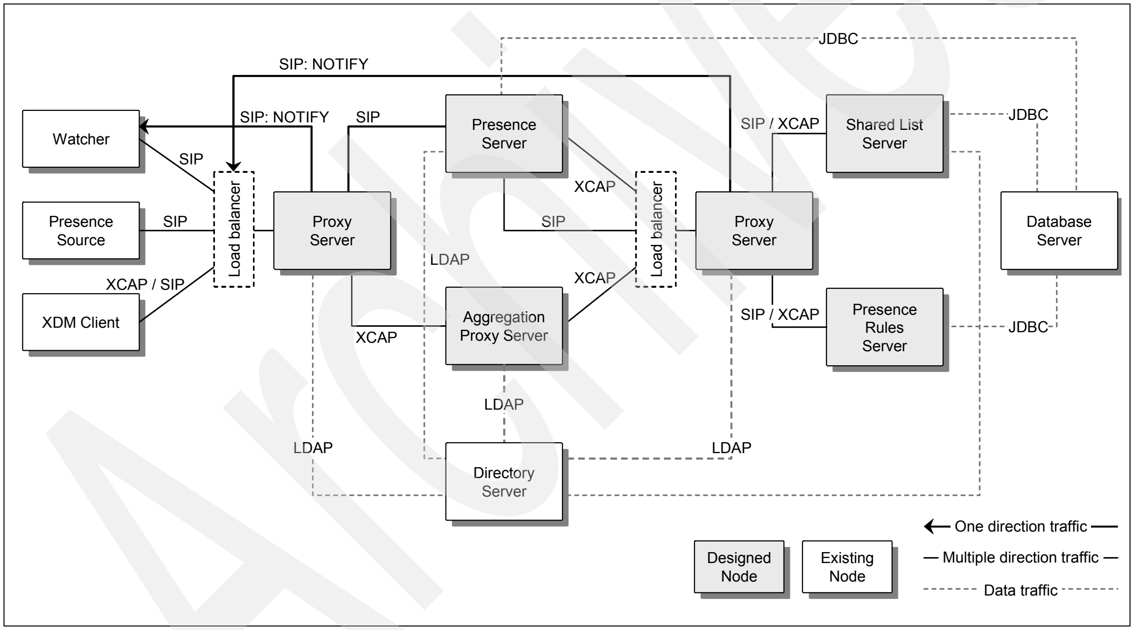


Figure 2-9 Architecture overview - systems view

Client components

Client components represent client systems that request presence information from Presence and XDMS services, receive notifications about requested presence, and provide Presence and XDMS services information about their presence:

- Watcher
- Presence Source
- XDM Client

Service components

The role of the service components is to provide Presence and XDMS services functionality. The high availability and scalability of these components make up the most significant part of our consideration for this book:

- Presence Server
- Aggregation Proxy Server
- Shared List Server
- Presence Rules Server

Infrastructure components

The following components are dedicated to improving the availability of the IMS Presence and XDMS Infrastructure:

- Proxy Node
- Load Balancer

Other components

The other components are:

- Directory Node

  This component represents the User Repository and provides Directory Services functions. Directory Services are invoked to register, authenticate, and manage users and their roles. They include an LDAP repository that the other components use for authentication and authorization purposes.

- Database Node

  Provides a common platform that is used to store the application data.

Product mapping

Table 2-1 shows the mapping of products to the components in the architecture systems view.

Table 2-1 Product mapping

Component name	Product name	Version
Presence Server	IBM WebSphere Presence Server	6.2.0
Aggregation Proxy Server	IBM WebSphere XML Document Management Servers (XDMS)	6.2.0
Shared List Server	IBM WebSphere XML Document Management Servers (XDMS)	6.2.0
Presence Rules Server	IBM WebSphere XML Document Management Servers (XDMS)	6.2.0
Proxy Server	IBM WebSphere Application Server	6.1.0.11
Load Balancer	IBM Edge Server	6.1.0
Watcher	SIPp	2.0.1
Presence Source	SIPp	2.0.1
XDMS Client	XCAPut	6.2.0
Directory Node	IBM Tivoli® Directory Server	6.0.4
Database Node	IBM DB2	9.1.0.3

2.4.3 Operational model

Figure 2-10 on page 55 illustrates the operational model, which is based on the Architecture Overview Model and provides the foundation for implementation decisions in subsequent chapters.

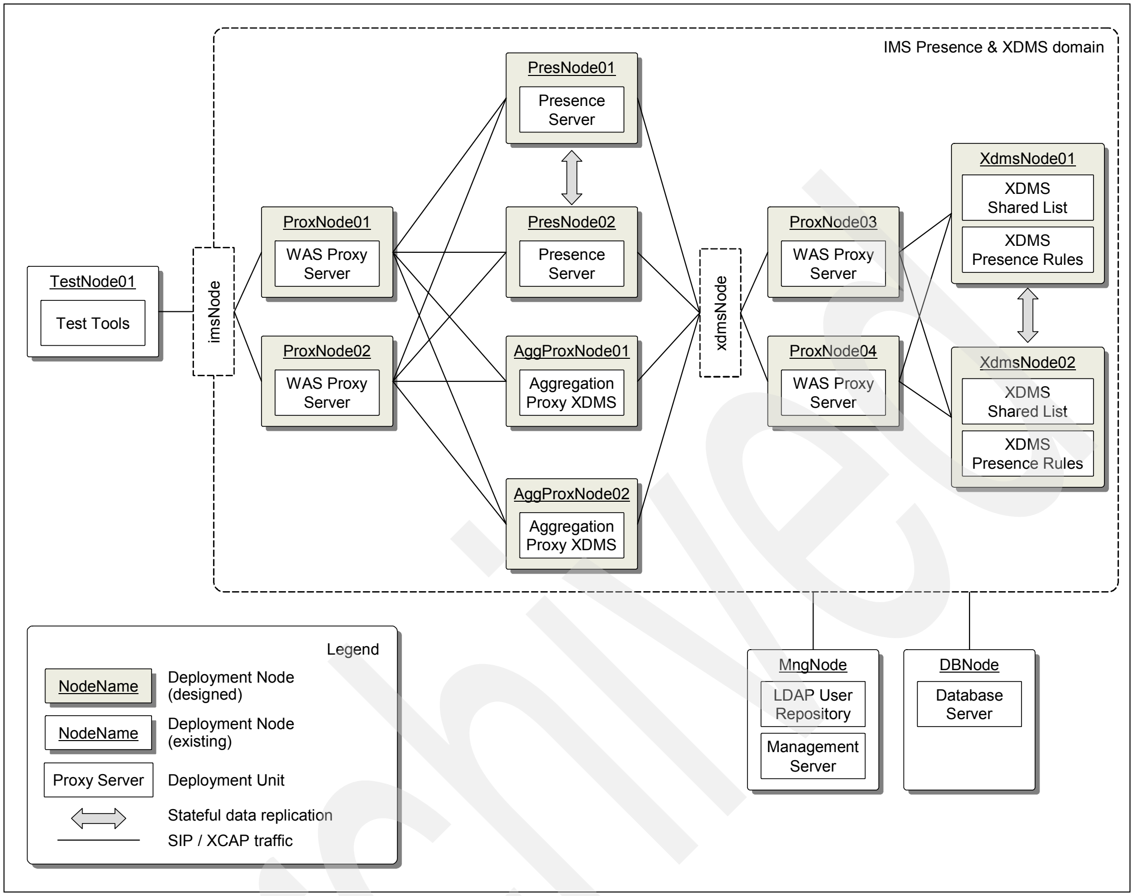


Figure 2-10 Operational model - topology diagram

Part 2

The base scenario

In this part of the book, we provide an overview of the base environment and describe the architecture, the topology, and how to set up the different components that we use in the base environment. Finally, we describe the test cases that we executed to validate the base environment.

3

Overview of the base environment

In this chapter, we provide an overview of the base scenario environment and describe how to prepare the environment before you install the Presence Server and XDMS components.

This chapter contains:

3.1 Overview of the base scenario

The scenario environment is designed with the idea of starting with a base environment, extending that environment for the high availability, and further expanding it for the scalability phases of realizing the scenario. Figure 3-1 shows a topological view of the base deployment environment. The environment shows components that support Presence and XDMS services, which we consider to be within Presence and XDMS services domain. The components in grey are considered to be outside of the domain.

Note: The components within the Presence and XDMS services domain will be extended and expanded in the high availability and scalability phases of the scenario.

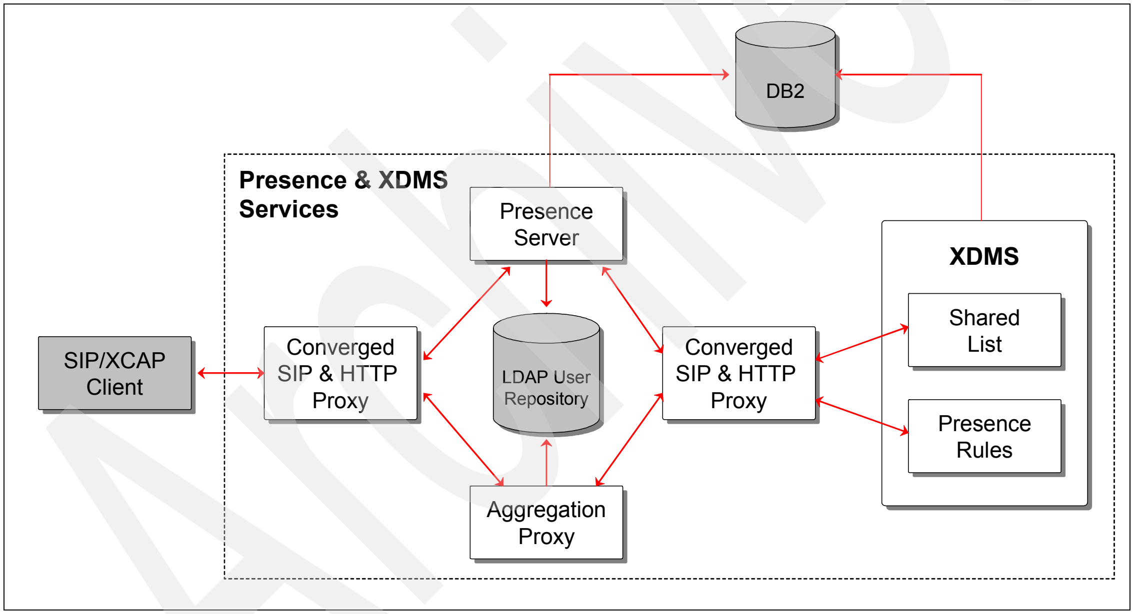


Figure 3-1 Topology view of base deployment environment

The following information further explains Figure 3-1:

- Components within the Presence and XDMS services domain

  The following components are core components that support Presence and XDMS services:

– WebSphere Application Server V6.1

The WebSphere Application Server is the core platform on which the WebSphere Presence and XDMS Servers are built. Both servers run on the WebSphere Application Server Network Deployment V6.1.0.11 and use various components of WebSphere Application Server ND V6.1 to provide Presence and XDMS services.

WebSphere Application Server is installed and configured first before you install the components for the Presence and XDMS services.

– Converged (SIP and HTTP) Proxy Server

The converged Proxy Server is configured in WebSphere Application Server Network Deployment V6.1. The proxy routes SIP traffic to the Presence Server and the XCAP over HTTP traffic to the Aggregation Proxy. The Proxy Server uses default SIP (5060) and HTTP (9080) ports for routing.

– IBM WebSphere Presence Server

The base installation of the WebSphere Presence Server requires that you install the server in a clustered configuration with one node in the cluster. Clustering is needed for high availability, but in this case the Presence Server needs to be fronted by the SIP Proxy Server that forwards SIP requests to all Presence Servers.

> **Note:** A SIP Proxy Server in WebSphere Application Server 6.1 can only be configured for clusters; therefore, the base installation of Presence Server requires a clustered configuration.

– IBM WebSphere XML Document Management Server (XDMS)

The XDMS Server provides two primary services Shared List and Presence Rules services and an optional service, the Aggregation proxy. Shared List and Presence Rules services are provided as enterprise applications:

- Shared List enterprise application
- Presence Rules enterprise application

The base deployment requires that you deploy the Shared List application in a separate WebSphere Application Server cluster with a single node or more (for high availability). The requirement for a separate cluster is to support the multiple domain situation, where you can install more than one XDMS Server in the environment, and the Aggregation Proxy, which fronts the XDMS Server, can route the XML Configuration Access Protocol (XCAP) requests to the right server in the cluster that is based on AUID (Application Unique ID) partitioning.

> **Note:** Refer OMA Resource List Server (RLS) XDM Specification Approved Version 1.0.1 for details on application routing using AUIDs.

- Components outside of the Presence and XDMS services domain

  The following components are core components that support Presence and XDMS services:

  – SIP/XCAP Client

    The client represents user devices that send SIP and XCAP requests and accept corresponding responses.

  – DB2 version 9.1.0.2

    Both Presence Server and XDMS use several database tables to store and retrieve presentity-related data and various XDMS documents that include Resource Lists, RLS Services, and Presence Rules. The Presence and XDMS Servers use JDBC to communicate with DB2 databases.

    > **Note:** For the 6.2 release of the products, Presence Server supports both DB2 and Oracle® databases. XDMS Server supports only DB2 for storing XML documents.

  – Tivoli Directory Server V6.0 (TDS) for LDAP

    The Presence Server and the Aggregation proxy component in XDMS need to authenticate all users and groups of users that use the Presence and XDMS services. The IBM Tivoli Directory Server is used as the LDAP Server for user authentication information.

3.2 Hardware and software prerequisites

To install the WebSphere Presence Server and WebSphere XDMS Server with Aggregation proxy, which is part of it, your server must meet at least the minimum system requirements. The minimum hardware and operating system requirements for WebSphere Presence and XDMS Server installation are:

Minimum hardware requirements:

- Processor:
  – Pentium® 4, a minimum of two processors are required
  – 2.8 GHz (32- or 64-bit)

- L2 cache:
  - L2 cache for 2.8 GHz processor must be 512 KB
  - L2 cache for 3.4 GHz processor must be 1 M
- Physical memory:
  - 4 GB minimum, 2 GB per JVM™ recommended
- Disk space:
  - 2 GB of free space (minimal)
  - 4 GB of free space recommended

Software requirements:

- Supported operating systems:
  - Red Hat Enterprise Linux® AS 4.0 Update 5
  - SUSE® Linux Enterprise Server 9 SP 3
  - AIX® 5L™ 5.3 TL 05 SP 6
- Supported Application Server:
  - WebSphere Application Server 6.1.0.11
- Supported databases:
  - IBM DB2 Universal Database™ Version 9.1 FixPak 2
  - Oracle Database Version 10.2.0.2

Additionally, you need CD-ROM drive or access to the shared network area with installation images. In our case, we use a shared disk that is mounted under the /mnt directory on each server in the infrastructure. Under this directory, IMS 6.2 installation images are available in an unpacked form.

Example 3-1 shows the network shared area with installation images.

Example 3-1 Network shared area with installation images

```
[root@ldap IMS_62]# pwd
/mnt/IMS_62_Builds
```

Note: In this sample scenario, we used System x servers running RedHat Enterprise Linux Server v 4 update 5, and WebSphere Application Server Network Deployment V6.1.0.11 installed on each server.

3.3 Preparing the base environment

The out-of-box deployment of the base environment for IBM Presence Server and XDMS Servers requires installation and configuration of the following components:

- SIP/HTTP Proxy Server on WebSphere Application Server V6.1.0.11
- Tivoli Directory Server V6.0 for LDAP
- DB2 9.1.0.2

Table 3-1 shows the set of steps for preparing the base environment, which includes the installation and configuration of the components in the previous bulleted list.

Table 3-1 Environment preparation checklist

Step	Activity	Remarks
1.0	Infrastructure	
1.1	Install OSes	Linux RHEL 4u5
1.2	Configure network	DNS
1.3	Install WebSphere Application Server Deployment Manager	Dmgr01
1.4	Create Deployment Manager Profile	Dmgr01
1.5	Install WebSphere Application Server Application Servers	without profiles
1.6	Install directory server	Tivoli Directory Server V6.0.4
1.7	Install databases	IBM DB2 V9.1.3
1.8	Create database instance	db2Inst1
2.0	WebSphere	
2.1	Set up WebSphere Application Server variables	DB2UNIVERSAL_JDBC_DRIVER_PATH, DB2UNIVERSAL_JDBC_DRIVER_NATIVEPATH
2.2	Configure WebSphere Application Server security	wasadmin
2.3	Install DHAIMSConnectorTAI modules	
2.4	Install AggProxy TAI modules	

Step	Activity	Remarks
2.5	Configure the TAI modules	

To be better prepared for installation process, we recommend that you collect, for your environment, the information in Table 3-2 to use during the installations and configurations to prepare the base environment.

Table 3-2 Base environment parameters for our scenario

Parameter	ITSO Value	Description
DB_HOST	db2	Host name of database server
DB_IP_ADDR	<xxx.xxx.xxx.xxx>	IP address of db host
DB_PORT	50000	Database listening port number
DB_INST	db2inst1	Database instance name
DB_ADMIN	db2inst1	Database administrator name
DB_PASSWORD	<password>	Database administrator password
LDAP_HOST	ldap	Host name of LDAP server
LDAP_IP_ADDR	<xxx.xxx.xxx.xxx>	IP address of LDAP host
LDAP_PORT	389	LDAP server listening port number
LDAP_ADMIN	cn=root	LDAP administrator account name
LDAP_PASSWD	<password>	LDAP administrator password
LDAP_BASED_DN	cn=itso	Based Distinguished Name od ldap directory used for this installation
WAS_DM_HOST	ldap	Host name of Deployment Manager Server for WebSphere Application Server Cell. In our case, we use the same server for LDAP and Deployment Manager Servers.
WAS_DM_IP	<xxx.xxx.xxx.xxx>	IP address of Deployment Manager Server
WAS_CELL_NAME	imsCell	Cell name of the WebSphere Application Server infrastructure that you create.

Figure 3-2 on page 67 shows the hardware components and the installation of the following components:

- WebSphere Application Server Network Deployment V6.1.0.11 with Deployment Manager Server (dmgr)
- Tivoli Directory Server v. 6.0.4 running on MngNode

> **Note:** It is not necessary to run the Deployment Manager Server and the LDAP Server on the same machine. In your environment, two different machines might be used.

- IBM DB2 database v.9.1.3

We assume that the components are installed and ready for use to complete the preparation of the environment.

Figure 3-2 on page 67 shows the hardware configuration for the base environment.

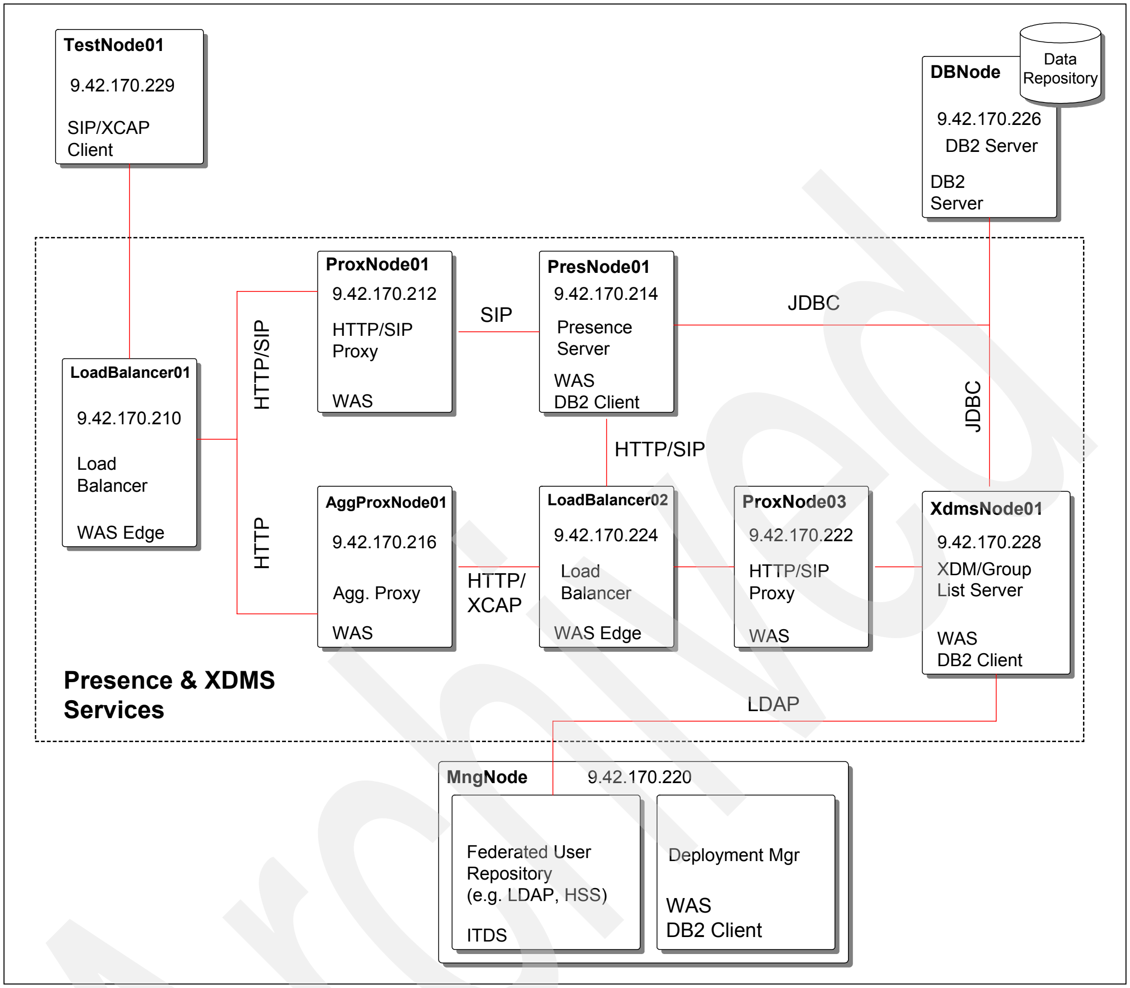


Figure 3-2 The hardware configuration for the base environment

3.3.1 Setting WebSphere Application Server variables

WebSphere variables define parameters for the system and enable communication between the Application Server and the database. These variables are created at the WebSphere Application Server Cell level so that they are available to all WebSphere Application Server nodes in the base environment. For the Cell level DB2 variables to be available to all WebSphere Application Server nodes, you need to make sure that the DB2 V9.2 client is installed in the same path on all WebSphere Application Server nodes, for example: /opt/ibm/db2/V9.1.

To create the WebSphere Application Server DB2 variables:

1. Log into the Integrated Solutions console, and click **Environment** → **WebSphere Variables**.
2. Click the **New** button to create a new variable, and enter the following data, as shown in Figure 3-3:
   – Name: **DB2UNIVERSAL_JDBC_DRIVER_PATH**
   – Value: **/opt/ibm/db2/V9.1/java**

   Click **OK**.

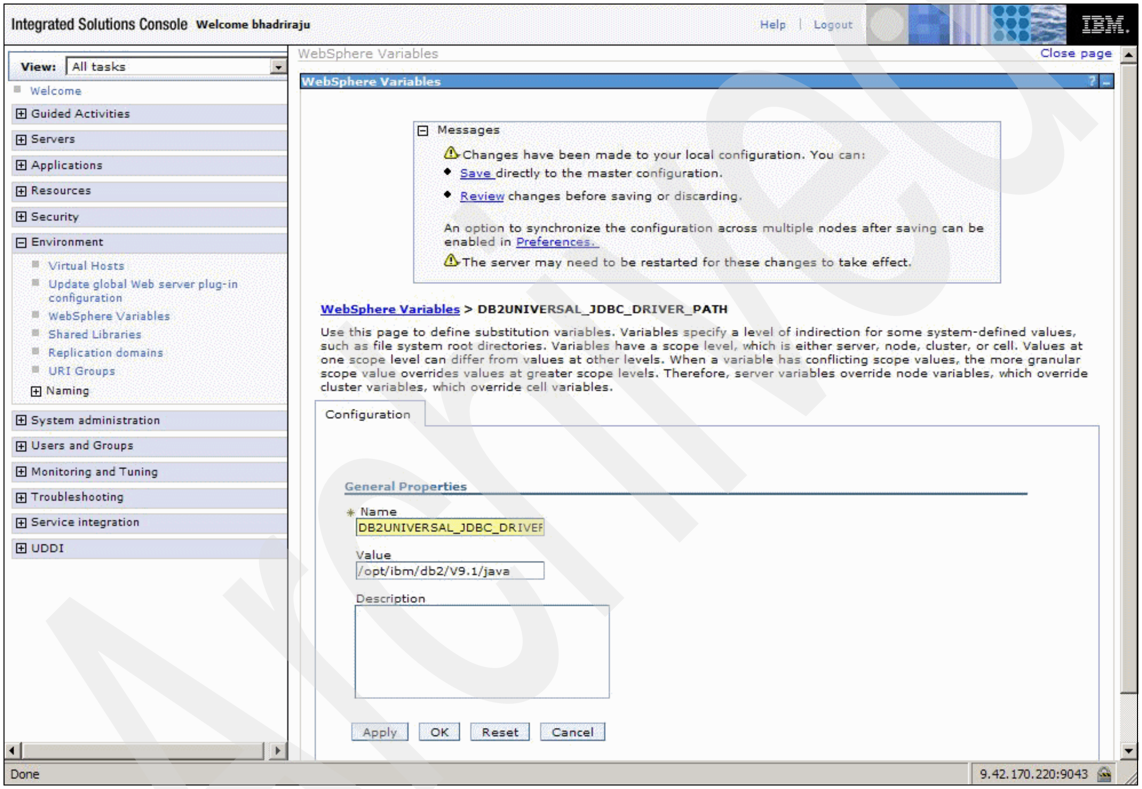


Figure 3-3 Creating WebSphere Variables: DB2UNIVERSAL_JDBC_DRIVER_PATH variable

3. Repeat step 2 with the following data:
   – Name: **DB2UNIVERSAL_JDBC_DRIVER_NATIVEPATH**
   – Value: **/opt/ibm/db2/V9.1/lib32**
4. Repeat step 3 with the following data:
   – Name: **UNIVERSAL_JDBC_DRIVER_NATIVEPATH**
   – Value: **/opt/ibm/db2/V9.1/lib32**

5. Repeat step 3 with the following data:

   – Name: **UNIVERSAL_JDBC_DRIVER_PATH**
   – Value: **/opt/ibm/db2/V9.1/java**

6. Click **Save** to store the WebSphere Application Server master configuration. After you compete the previous four steps, all variables should show, as shown in Figure 3-4.

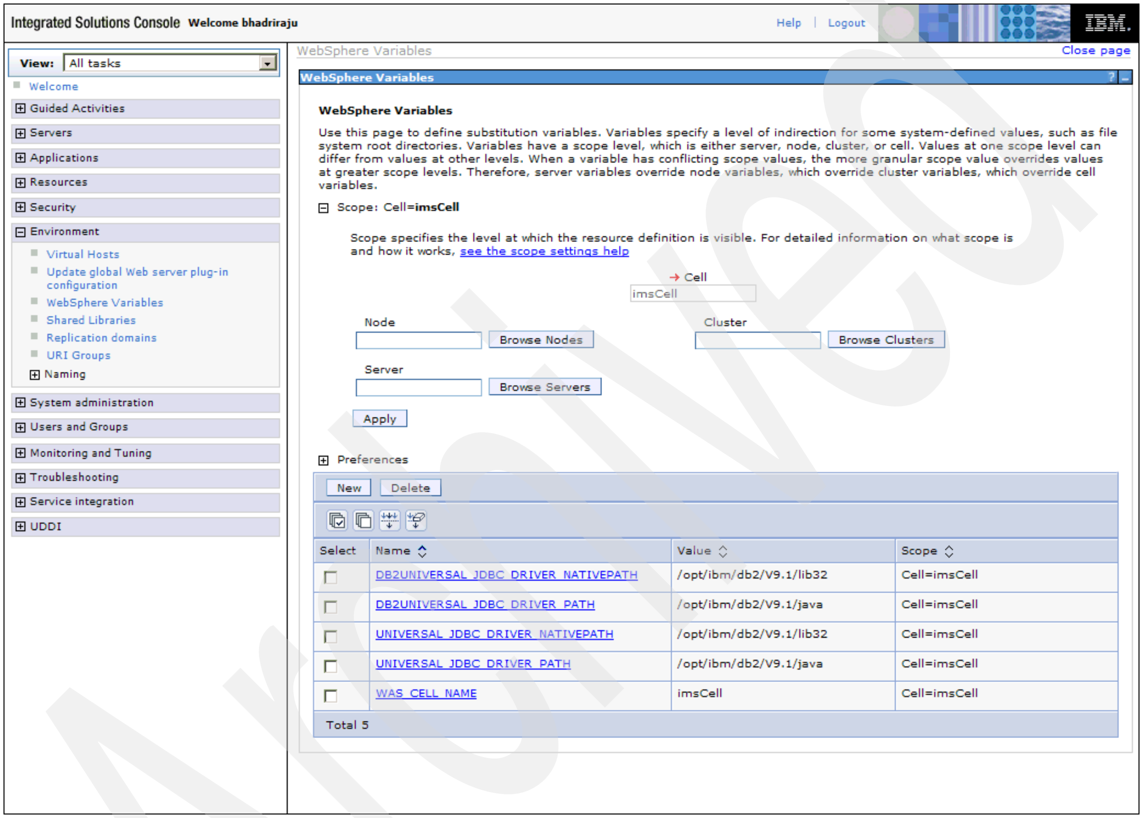


Figure 3-4 WebSphere variables creation: ISC showing that all variables are created

3.3.2 WebSphere Application Server security configuration

Enabling security in the base environment allows configuration of identity check in Presence Server. As a consequence of enabling identity check, Presence User Agents or Presence-enabled SIP User Equipments are required to send identity information about the user in the request header to Presence Server environment. Configure the TAI component on the Proxy Server to enforce the asserted-identity mechanism.

> **Note:** By enabling TAI settings, XCAP requests should have X-XCAP-Asserted-Identity, and SIP requests should have P-Asserted-Identity header value pairs.

Presentity information is comprised of users' personal information and preferences. This information has to be protected from unauthorized access. IBM Presence Server provides the following configuration options to enable privacy functions:

- Black Lists and White lists

  In addition to Identity check, Black and white lists provide additional protection from unauthorized access to presentity services. Black and white lists are stored in the Shared List server repository. Presence Server provides mechanisms in the configuration settings.

  > **Note:** Black and White list functionality is not used in this deployment scenario.

- Authorization rules

  Authorization rules allow you to perform CRUD operations on Presence Rules, as defined in RFC 4745 and draft-ietf-simple-presence-rules-09 specifications. These authorization rules are persisted in the Presence Rules Server repository on a per user basis. In the base environment, all of the scenario users are configured with authorization rules.

LDAP security configuration

In the first step, we create an LDAP object for the WebSphere Application Server and XDMS Servers, and after that, we configure WebSphere application Security.

Creating the LDAP object

To start WebSphere Application Server configuration, we need administration accounts and groups in LDAP Directory Server:

1. Using your preferred LDAP tool, create LDAP accounts for the WebSphere Application Server administrator and the XDMS administrator. They should be similar Example 3-2.

Example 3-2 LDAP administrators accounts

```
[root@ldap ~]# ldapsearch -h ldap -p 389 -D cn=root \
>-w <password> -b cn=itso "(|(cn=wasadmin)(cn=superadmin))"
cn=wasadmin,cn=itso
```

```
userpassword=<password>
objectclass=inetOrgPerson
objectclass=organizationalPerson
objectclass=person
objectclass=top
sn=WASAdmin
cn=wasadmin
uid=wasadmin

cn=superadmin,cn=Users,cn=itso
uid=superadmin
userPassword=<password>
objectclass=top
objectclass=inetOrgPerson
objectclass=person
objectclass=organizationalPerson
sn=superadmin
cn=superadmin
```

> **Note:** Attributes objectClass=Person, cn=<name>, and uid=<name> are crucial to WebSphere Application Server LDAP configuration. Be sure they exist and are set like Example 3-2 on page 70.

2. Create LDAP groups administrator and configurator, as shown in Example 3-3.

Example 3-3 LDAP Groups for WebSphere Application Server

```
[root@ldap ~]# ldapsearch -h ldap -p 389 -D cn=root \
> -w <password> -b cn=itso "(|(cn=administrator)(cn=configurator))"
cn=administrator,cn=Users,cn=itso
objectclass=groupOfNames
objectclass=top
cn=administrator
member=cn=superadmin,cn=users,cn=itso

cn=configurator,cn=Users,cn=itso
objectclass=groupOfNames
objectclass=top
member=cn=superadmin,cn=users,cn=itso
cn=configurator
```

> **Note:** Be sure that account cn=superadmin, cn=Users, cn=itso were added to both groups.

Configuring WebSphere Application Server Security to use LDAP

In configuring WebSphere Application Server security to use LDAP, we used two attributes for authentication: one is **cn** and the second is **uid**. This is because we authenticate users either based on the full user name with domain name (for example user1@itso.ral.ibm.com), which is saved in uid or based on a short name (for example user1) that is saved in cn.

To configure WebSphere Application Server Security to use LDAP:

1. Logon to the WebSphere Application Server Console as wasadmin.
2. In the navigation pane, click **Security** → **Secure Administration, Applications, and infrastructure**.
3. From the Available realm definitions, select **Standalone LDAP registry**, and click **Set as current**.
4. Click **Configure**, as shown in Figure 3-5.

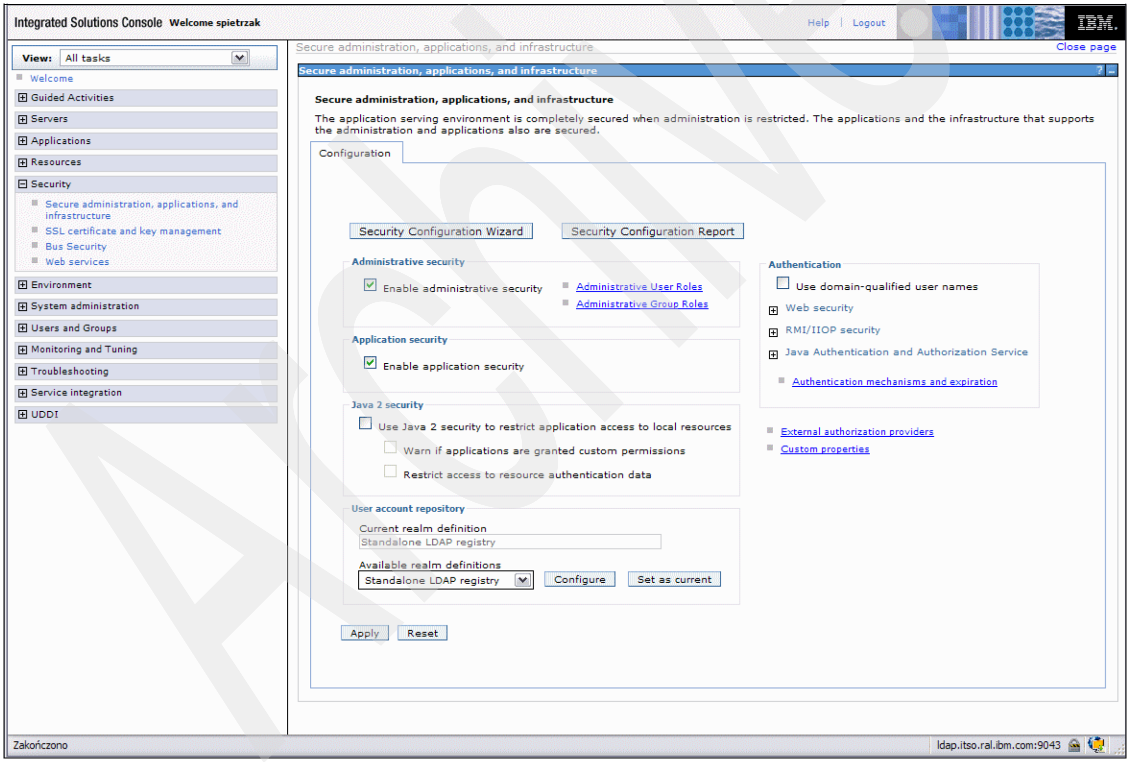


Figure 3-5 WebSphere Application Server Security - LDAP registry - Security configuration wizard

5. Click **Advanced Lightweight Directory Access Protocol (LDAP) user registry settings**, and change the user filter field as follows, as shown in Figure 3-6:
(&(|(cn=%v)(uid=%v))(objectclass=person))
6. Click **OK**, and return to the **Secure administration, applications, and infrastructure** → **Standalone LDAP registry** configuration page.

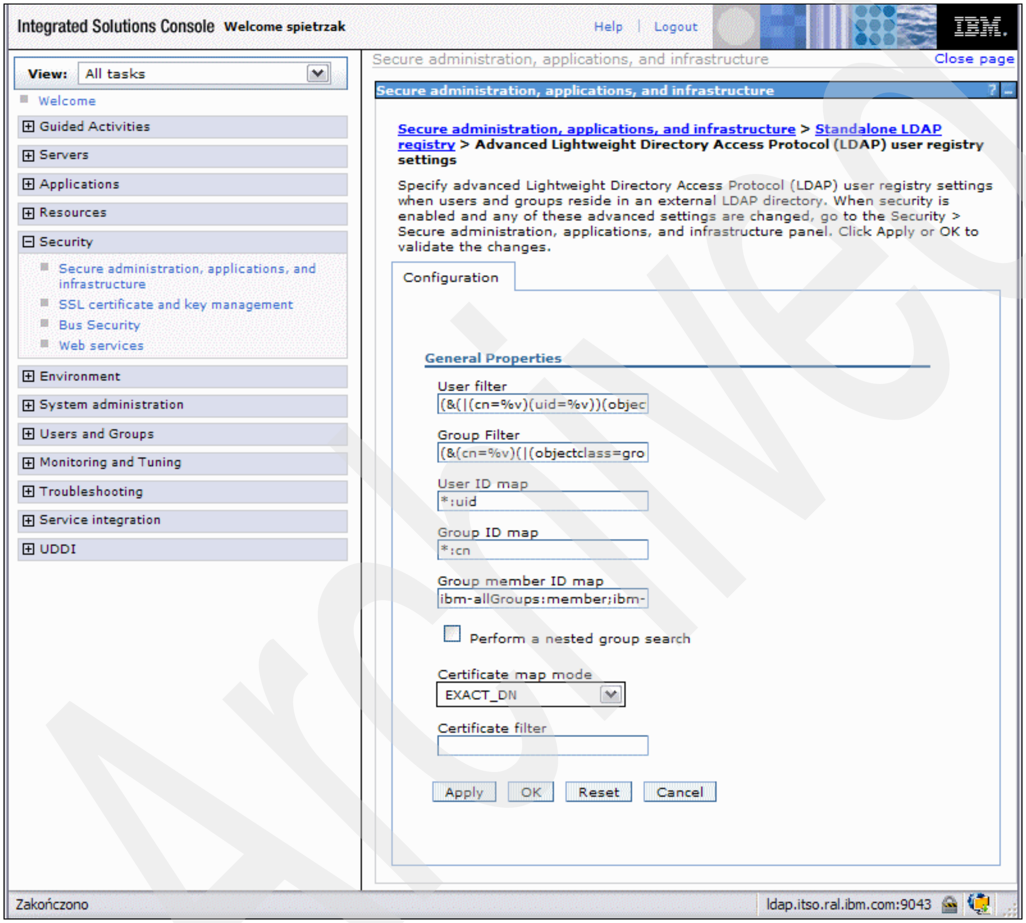


Figure 3-6 WebSphere Application Server Security - LDAP registry - User registry settings

7. Type the following Standalone LDAP registry parameters (see Figure 3-7 on page 74):
   - Primary administrative user name: **cn=wasadmin,cn=itso**
   - Type of LDAP server: **Custom**
   - Host: **ldap.itso.ral.ibm.com**
   - Port: **389**

– Based DN: **cn=itso**
– Bind DN: **cn=wasadmin,cn=itso**
– Bind Password: **<password>**

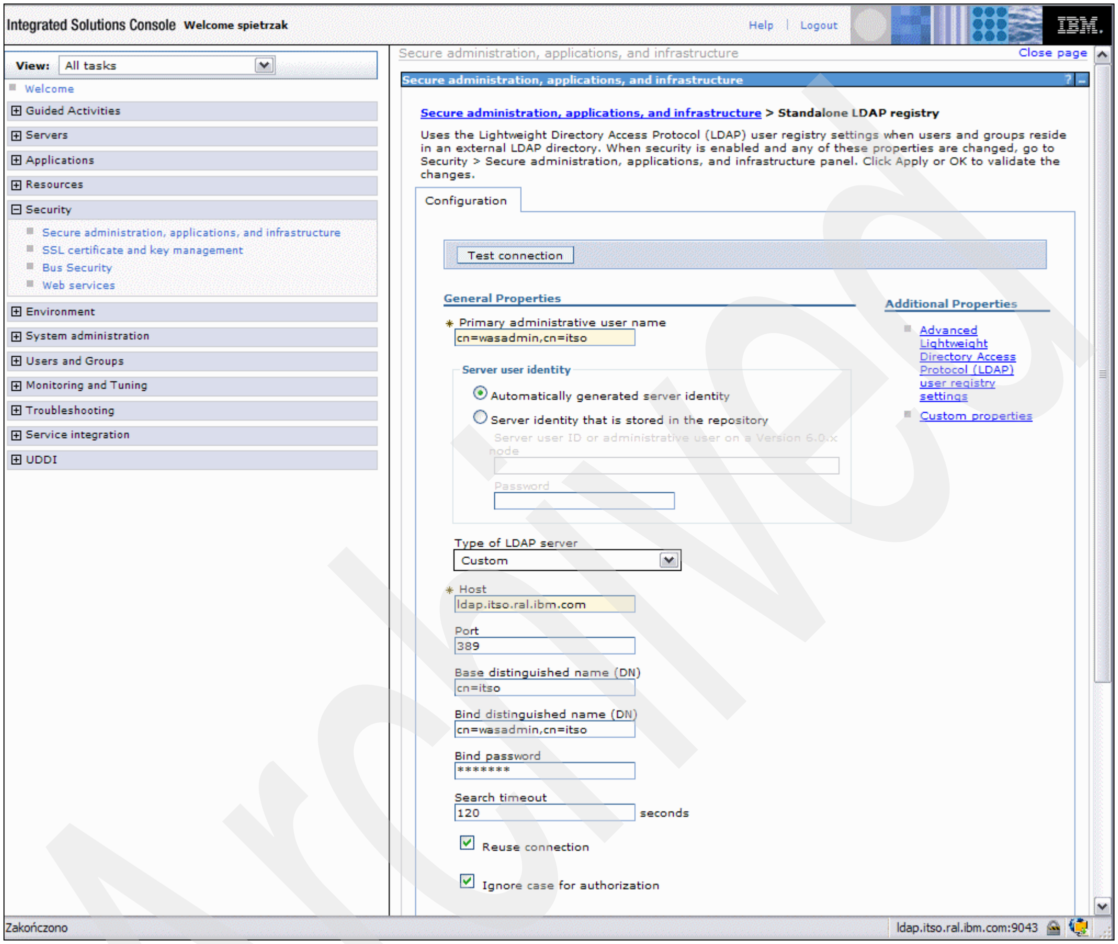


Figure 3-7 WebSphere Application Server Security - LDAP registry - Standalone LDAP registry parameters

8. Click **Apply**.
9. Click **Test connection**, as shown in Figure 3-8 on page 75.
10. After successful test, click **OK**.

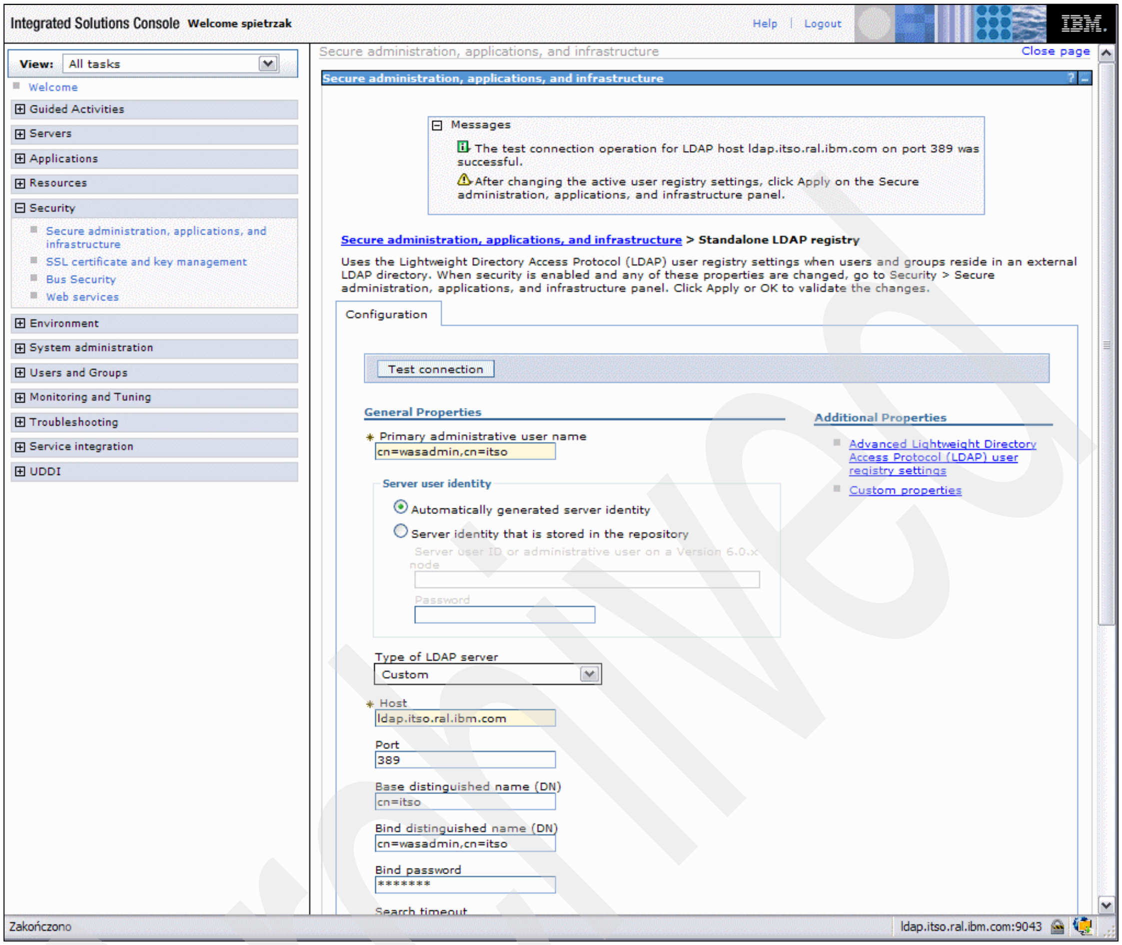


Figure 3-8 WebSphere Application Server Security - LDAP registry - Test connection

11. Click **Apply** on the Secure administration, applications, and infrastructure page, as shown in Figure 3-9 on page 76.
12. **Save** changes to the master configuration.

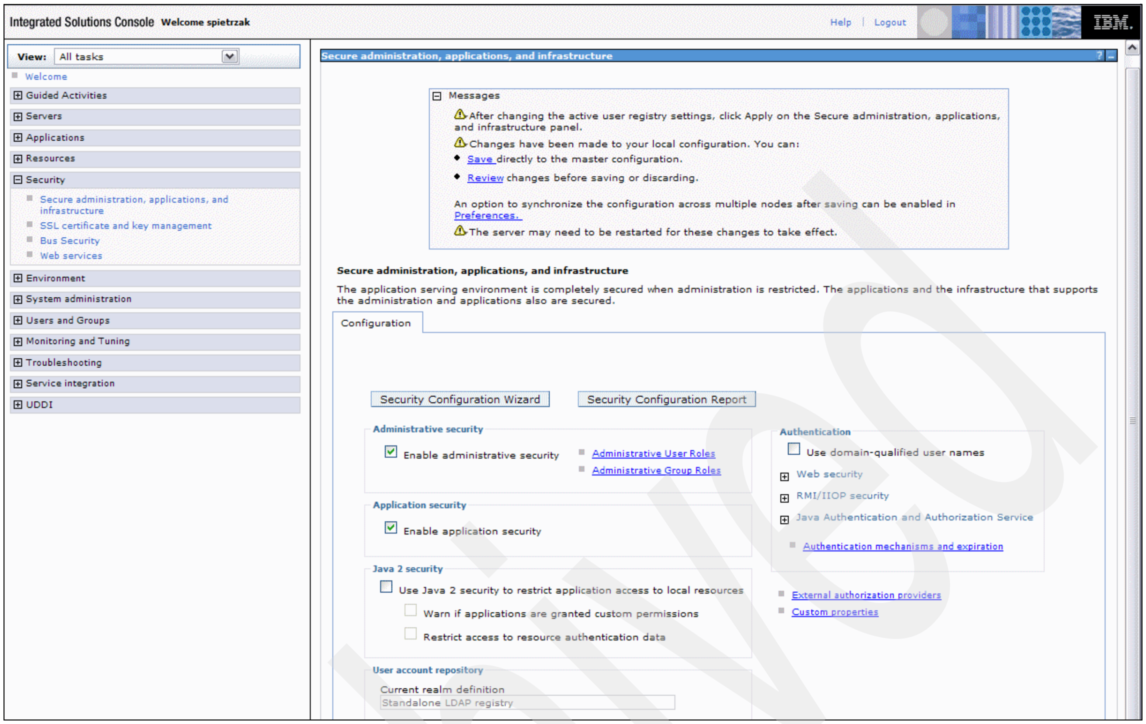


Figure 3-9 WebSphere Application Server Security - LDAP registry

Restarting all servers and nodes

To restart all servers and nodes:

1. From the navigation pane, click **System administration** → **Nodes**.
2. Select all nodes in the list, and click **Full Resynchronize**.
3. Restart all servers, nodes, and the Deployment Manager Node.

3.3.3 Web security configuration

In the next sections, we show you how to configure Web security for Presence and XDMS infrastructure. We describe the following steps:

- Install TAIs modules.
- Configure the Trust Association Interceptor.

Installing Trust Association Interceptor

Trust Association Interceptors are used for authentication purposes to request incoming components of Presence Server and XDMS Infrastructure.

To install Trust Association Interceptor:

1. Unpack the IMSConnectorInstallPackage_6.2.tar to the temporary directory.
2. Distribute file DHAIMSConnectorTAI.jar to location **<was_root>/lib/ext** on all servers in the infrastructure, at least they are:
   - MngNode (ldap.itso.ral.ibm.com)
   - PresNode01 (pres1.itso.ral.ibm.com)
   - XdmsNode01(xdms1.itso.ral.ibm.com)
   - AggProxNode01 (agg1.itso.ral.ibm.com)

   Example 3-4 shows how to copy the IMSConnector to the TAI library.

Example 3-4 Copy IMSConnector TAI library

```
[root@ldap ext]# pwd
/opt/ibm/WebSphere/AppServer/lib/ext
[root@ldap ext]# cp /mnt/IMS_62/DHAIMSConnectorTAI.jar ./
```

Trust Association Interceptor configuration

After security is enabled and the TAI are installed, next we configure the components for usage in the infrastructure.

Configuring WebSphere Security for the IMS Presence infrastructure

To configure WebSphere Security for the IMS Presence infrastructure:

1. Logon to the WebSphere Application Server Console as user with Administration role.
2. From the Integration Solutions Console, click **Security** → **Secure Administration, Applications, and infrastructure**.
3. Under Authentication, click **Web Security** → **General settings**. See Figure 3-10 on page 78.

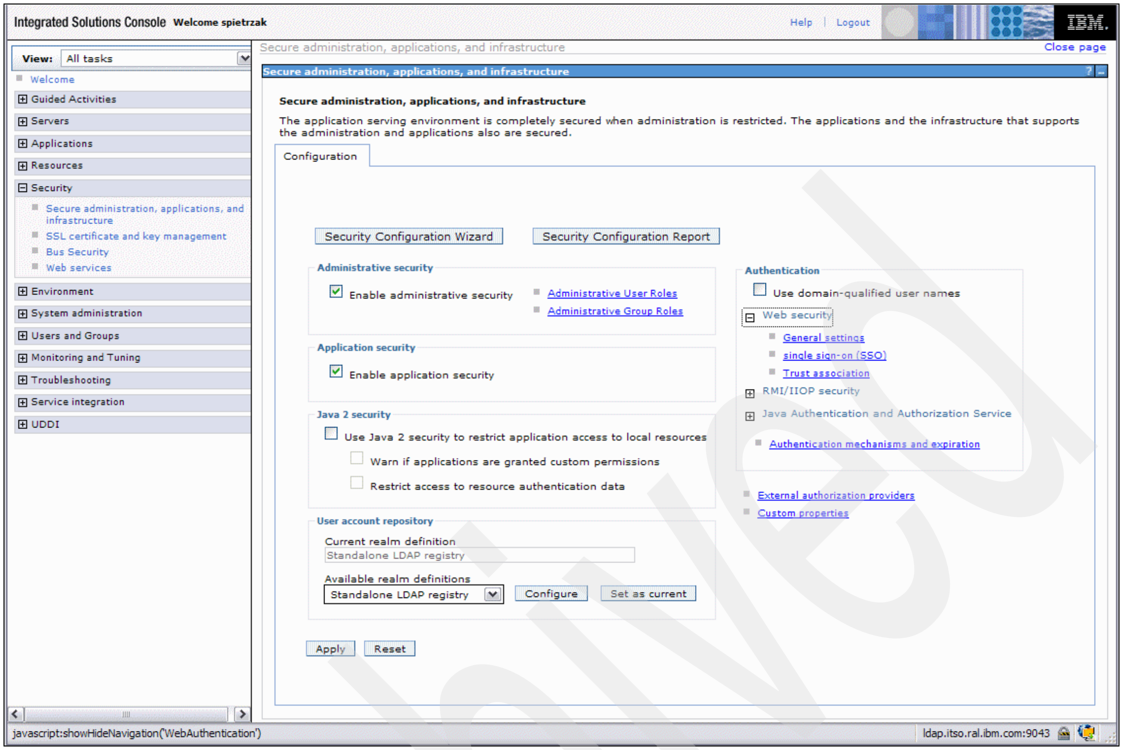


Figure 3-10 WebSphere Application Server Security - General settings

4. Select **Authenticate only when the URI is protected**.
5. Select the **Use available authentication data when an unprotected URI is accessed** option, as shown in Figure 3-11 on page 79.
6. Click **OK**.

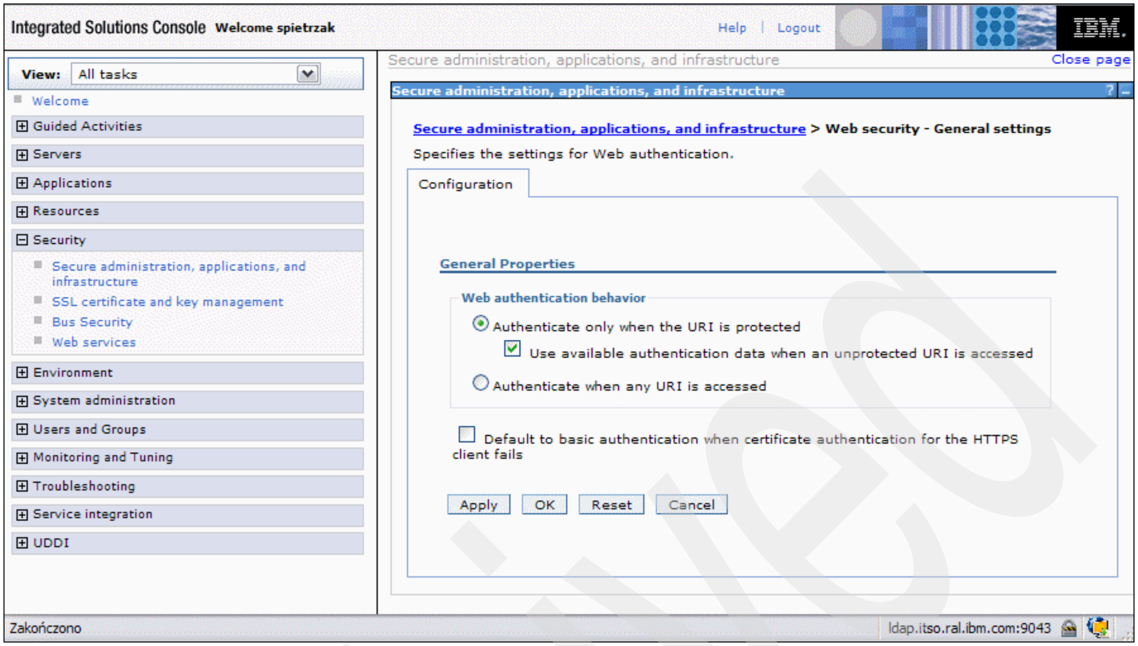


Figure 3-11 WebSphere Application Server Security - General settings

7. From the navigation pane, click **Security** → **Secure administration, applications, and infrastructure**.

8. Under **Authentication**, click **Web Security** → **Trust Association**, as shown in Figure 3-12 on page 80.

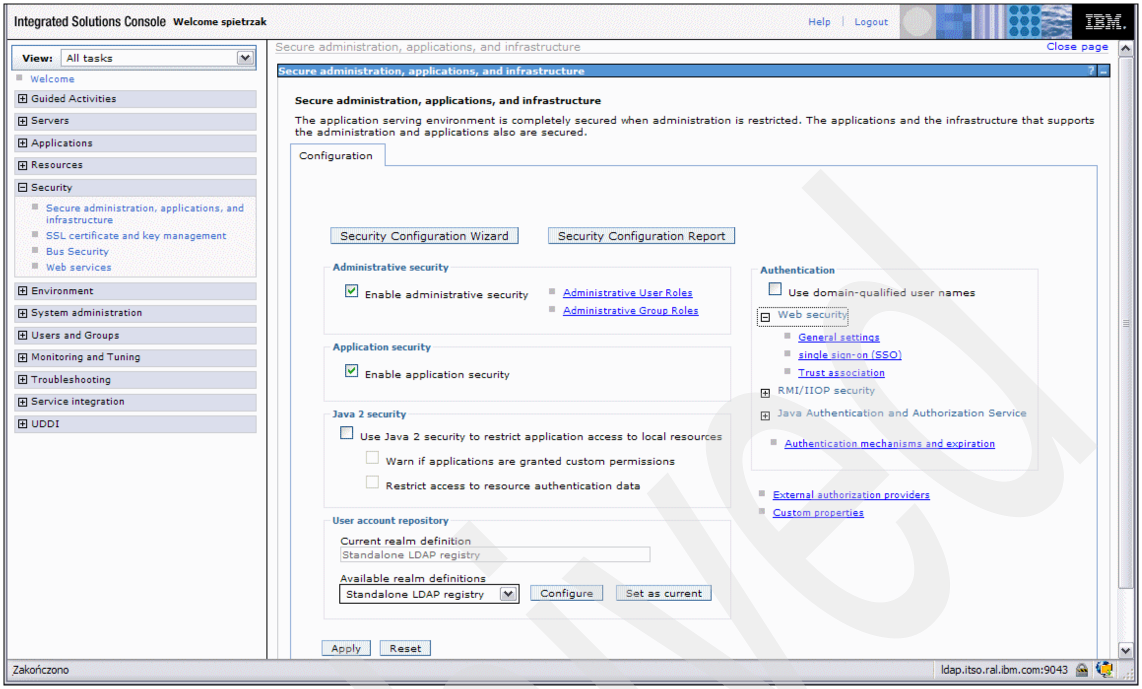


Figure 3-12 WebSphere Application Server Security - Trust Association

9. Under **Additional Properties**, click **Interceptors**, as shown in Figure 3-13.
10. Select all interceptors, and click **Delete**.

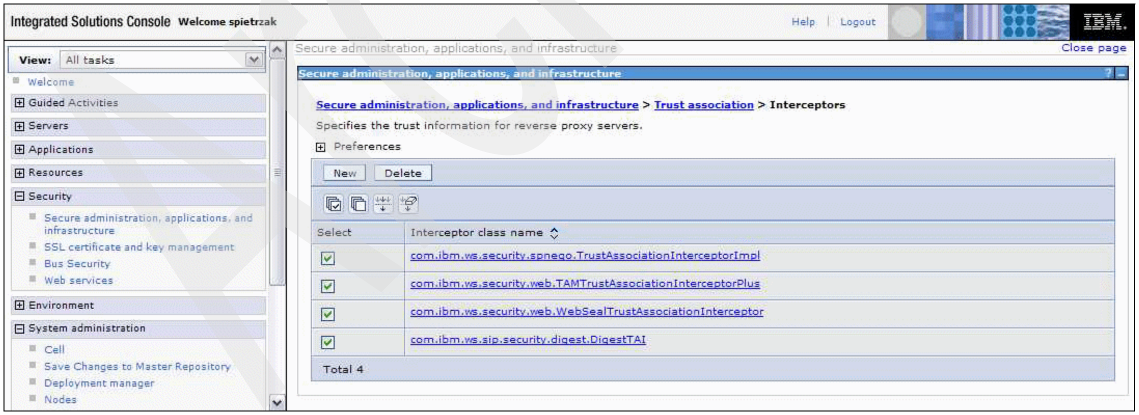


Figure 3-13 WebSphere Application Server Security - Trust Association Interceptors

> **Note:** All TAIs for WebSEAL are removed because of usage of network level authentication instead of Tivoli Access manager WebSEAL as the Authorization Proxy.

11. Select **Security → Secure administration, applications, and infrastructure → Trust association**.
12. Select the **Enable trust association** option, as shown in Figure 3-14.
13. Click **OK**.

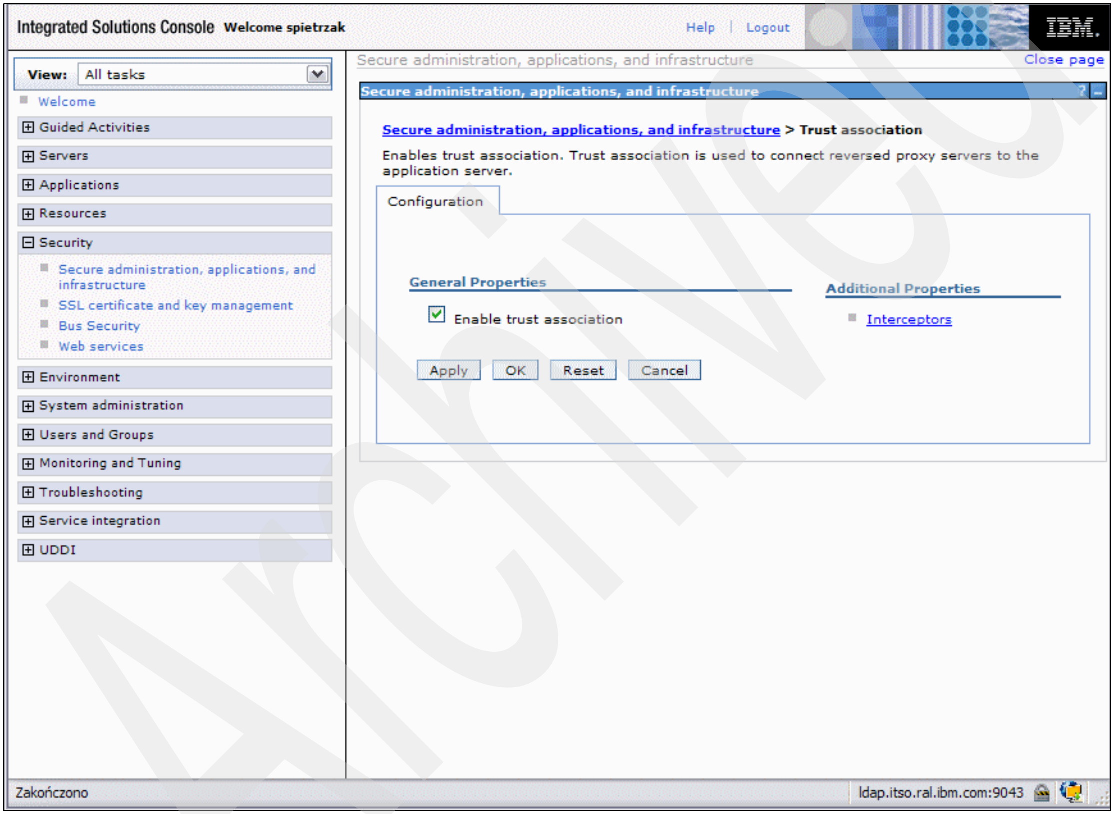


Figure 3-14 WebSphere Application Server Security - Enable trust associations

14. Select **Security → Secure administration, applications, and infrastructure**.
15. Select **Enable application security**, as shown in Figure 3-15 on page 82.
16. Click **Apply**.
17. Save changes to the master repository.

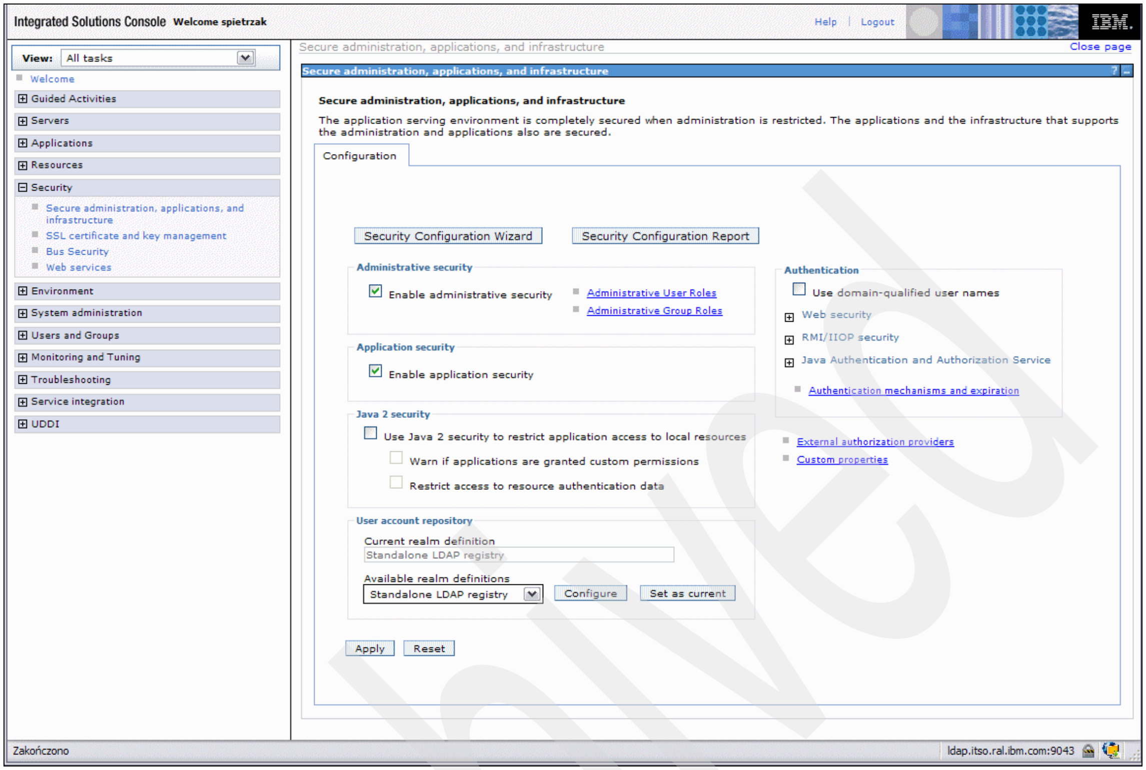


Figure 3-15 WebSphere Application Server Security - Enable application security

Creating IMS Connector Interceptors

To create IMS Connector Interceptors:

1. From the navigation pane, click **Security** → **Secure administration, applications, and infrastructure**, as shown in Figure 3-16 on page 83.

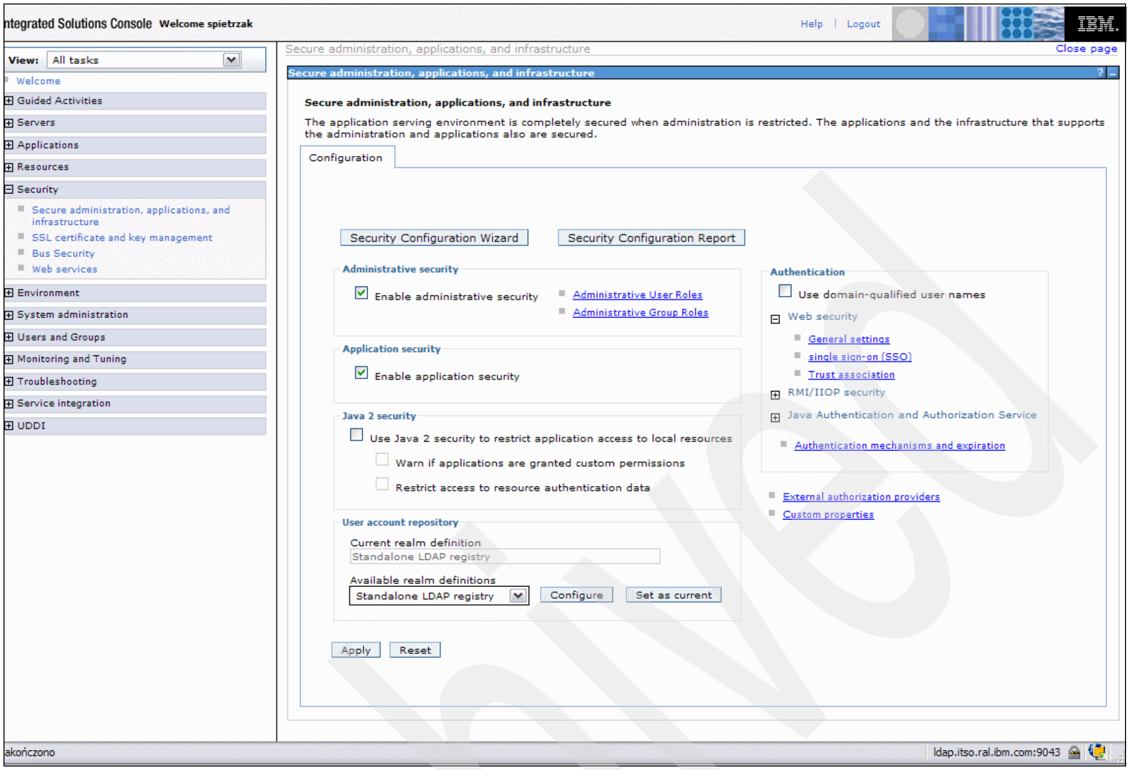


Figure 3-16 WebSphere Application Server Security - IMS Connector TAI

2. Under Authentication, click **Web Security** → **Trust Association**, as shown in Figure 3-17 on page 84.

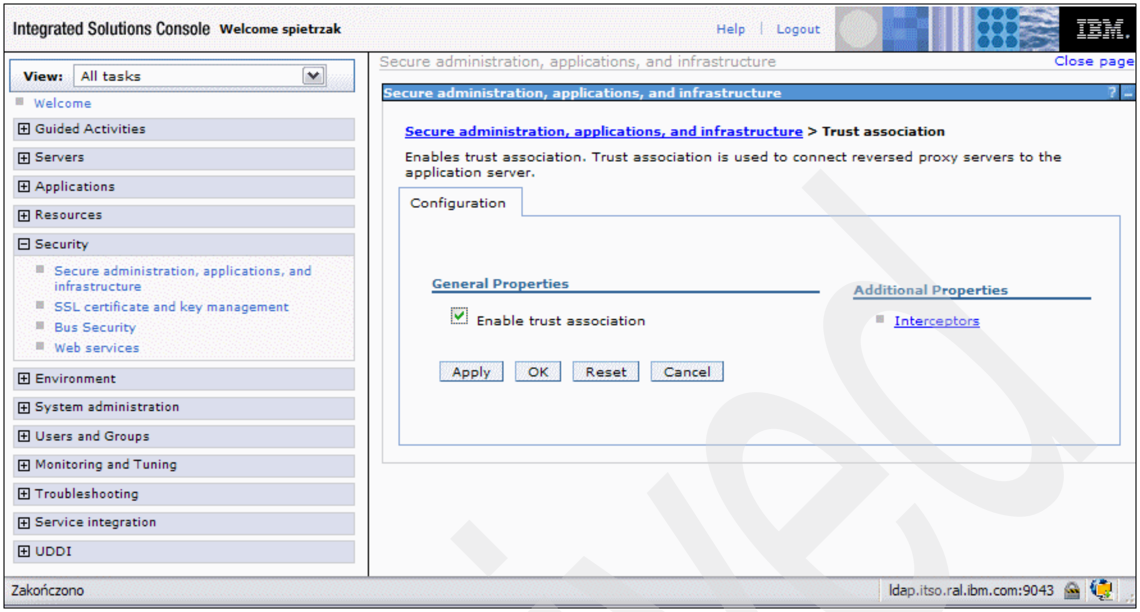


Figure 3-17   WebSphere Application Server Security - IMS Connector TAI - Trust Association

3. Under Additional Properties, click **Interceptors** → **New**.
4. Enter the class name for the new interceptor: **com.ibm.imsconnector.tai.HttpInterceptor**, as shown in Figure 3-18 on page 85 and Figure 3-19 on page 85.
5. Click **OK**.

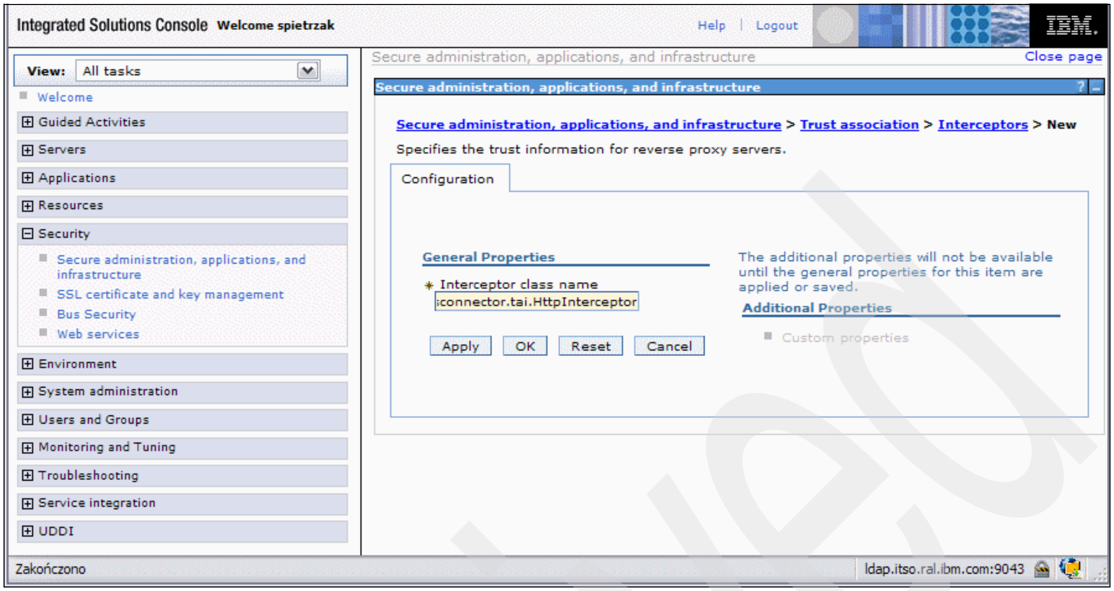


Figure 3-18 WebSphere Application Server Security - IMS Connector TAI - New interceptors

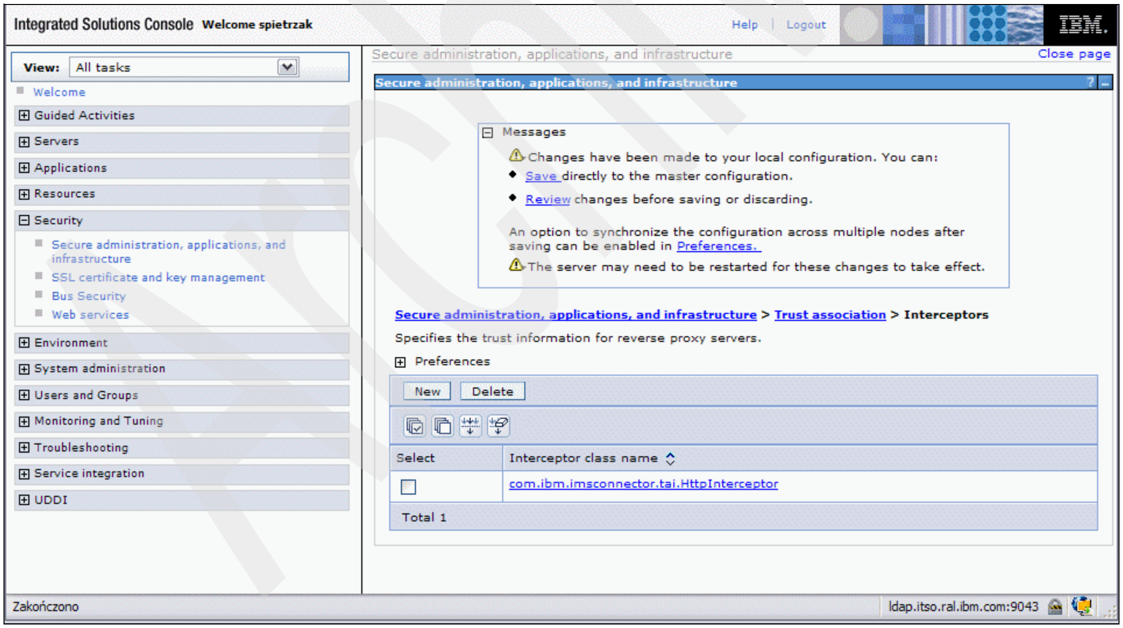


Figure 3-19 WebSphere Application Server Security - IMS Connector TAI - Interceptors

6. Click **New**.

7. Enter the class name for the new interceptor: **com.ibm.imsconnector.tai.SipInterceptor**, as shown in Figure 3-20.
8. Click **OK**.

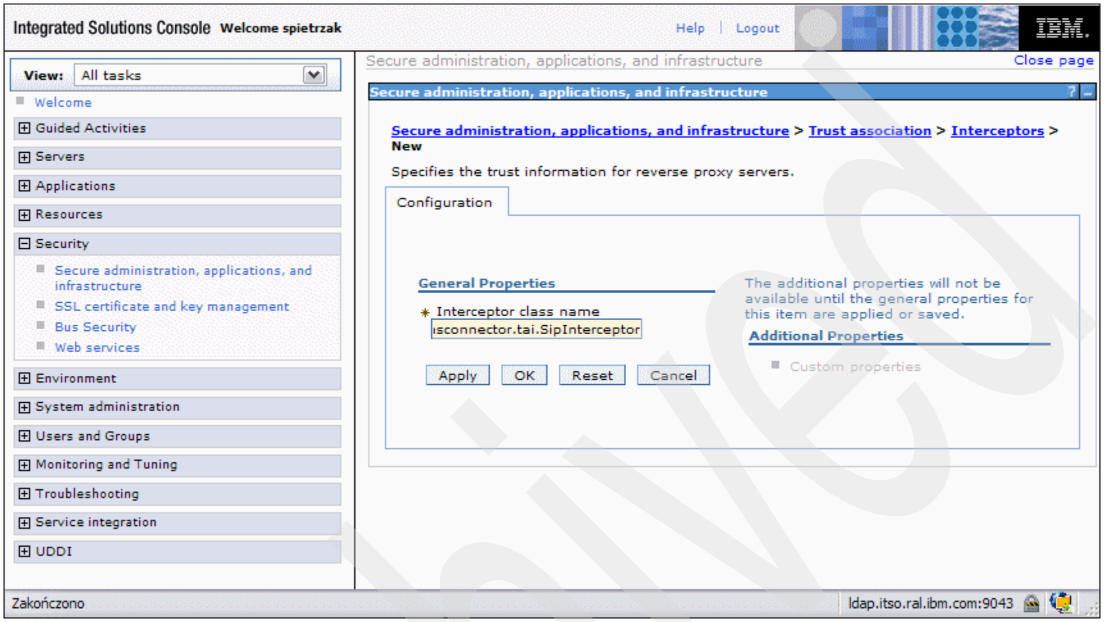


Figure 3-20 WebSphere Application Server Security - IMS Connector TAI - New interceptor SipInterceptor

9. Save changes to the master repository, as shown in Figure 3-21 on page 87.

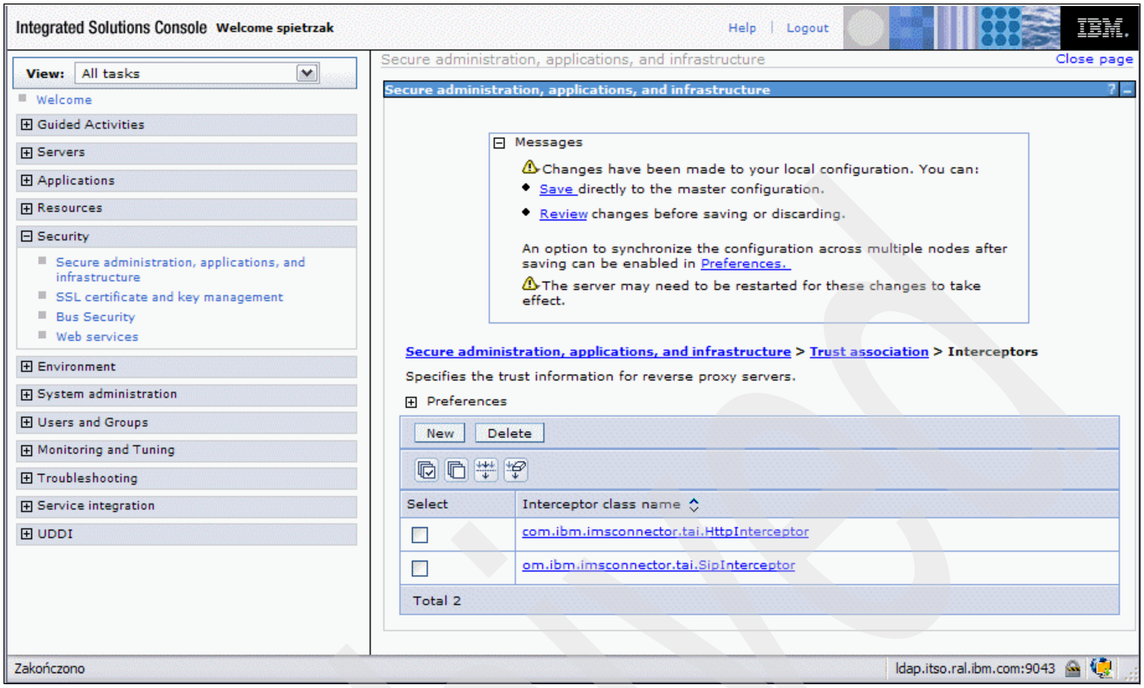


Figure 3-21 WebSphere Application Server Security - IMS Connector TAI - Interceptors

Configuring WebSphere IMS SIPInterceptor

To configure WebSphere IMS SIPInterceptor:

1. Click on the recently create interceptor **com.ibm.imsconnector.tai.SipInterceptor**.
2. Under Additional Properties, click **Custom properties**, as shown in Figure 3-22 on page 88.

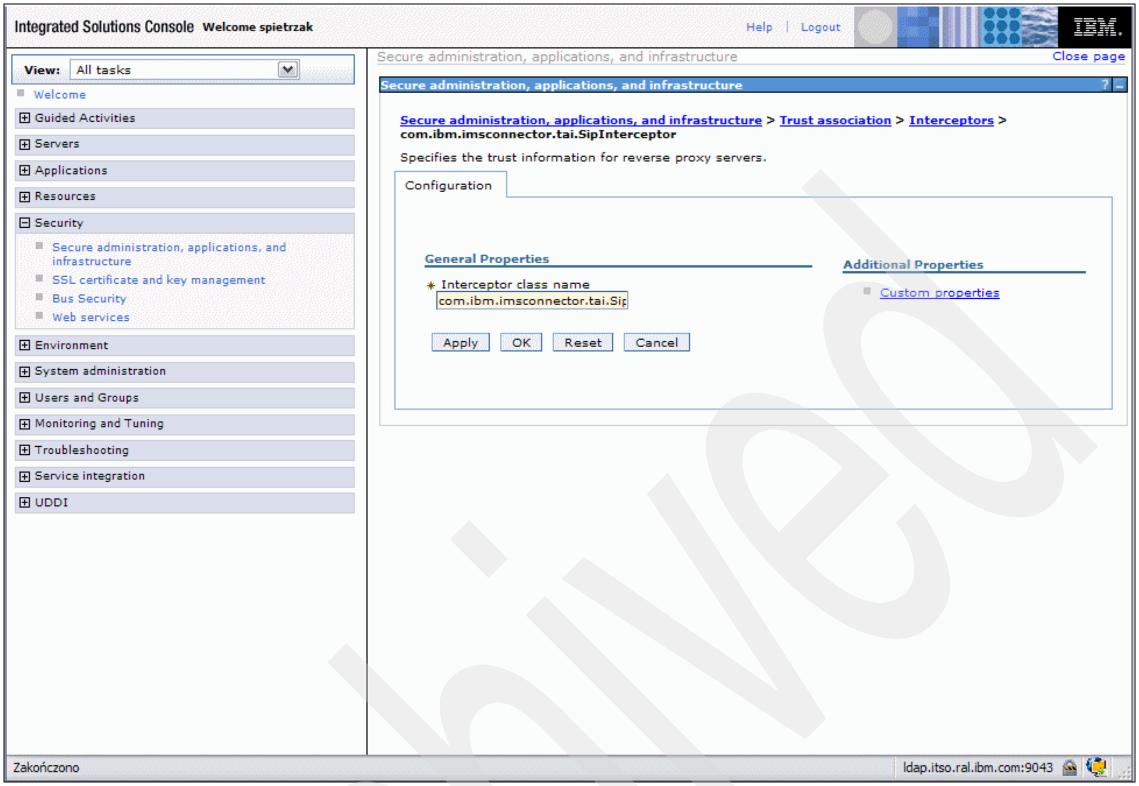


Figure 3-22 Configure SipInterceptor - custom properties

3. Click **New** to create a new property, as shown in Figure 3-23 on page 89, and enter the following values:
   - Name: **allowedSenderList**
   - Value: ***.itso.ral.ibm.com**
4. Click **OK**.

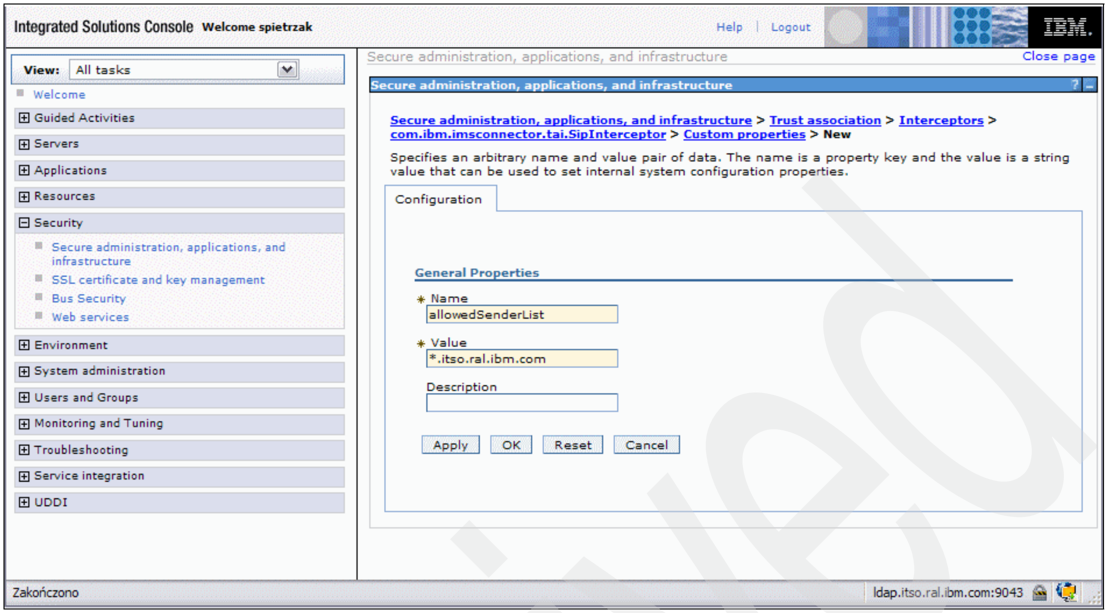


Figure 3-23 Configure SipInterceptor - new property

5. Click **New** to create a second property, as shown in Figure 3-24 on page 90, and enter the following data:
   - Name: **enableSenderVerification**
   - Value: **false**
6. Click **OK**.

> **Note:** For test purposes, we accept requests from all hosts, and the parameter **enableSenderVerification** is set as false. We recommend that you set it to true in the production environment.

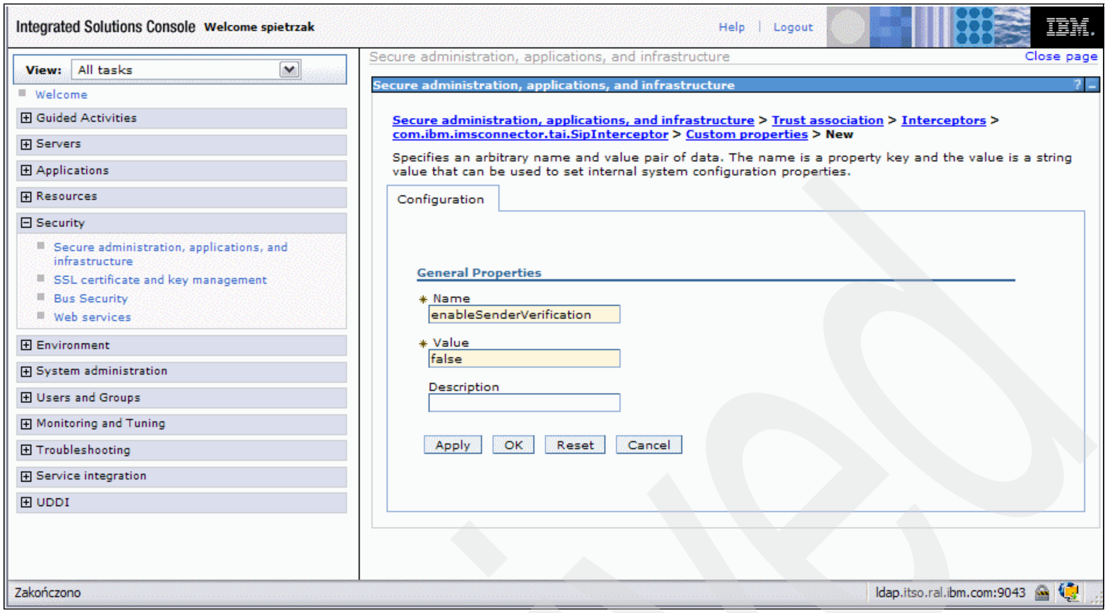


Figure 3-24 Configure SipInterceptor - new property

7. Save the changes to the master repository.

Configuring the HTTP Interceptor

Configuring the HTTP Interceptor involves creating the custom properties for com.ibm.imsconnector.tai.HttpInterceptor. Perform the following steps to create the custom properties:

1. From the Integrated Solutions Console of the WebSphere Application Server Deployment Manager for the environment, select **Secure administration, applications, and infrastructure** → **Trust association** → **Interceptors**.
2. Click the **com.ibm.glm.http.security.tai.HttpDigestTAI** interceptor.
3. Under Additional Properties, click **Custom properties**.

   See Figure 3-25 on page 91.

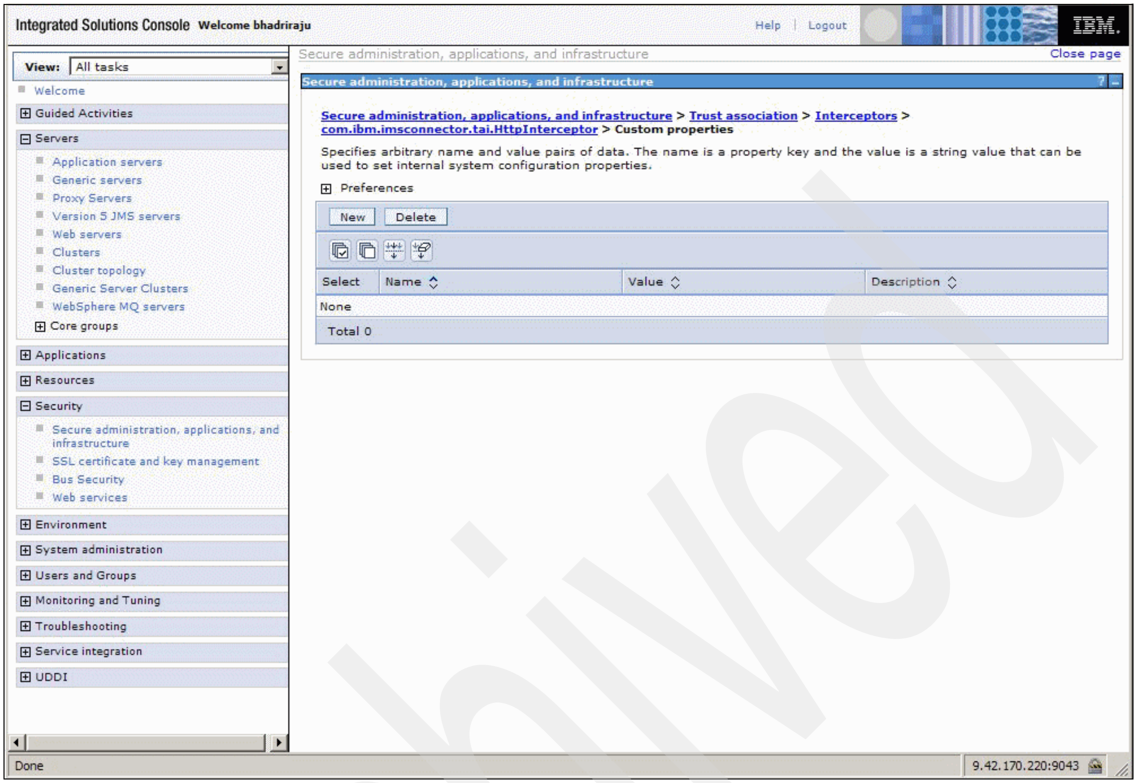


Figure 3-25 Configure HTTPInterceptor - Create custom properties

4. To create a new Custom Property, click **New**.
5. Enter the following data, and click **OK**, as shown in Figure 3-26 on page 92.
   – Name: **allowedSenderList**
   – Value: ***.itso.ral.ibm.com**

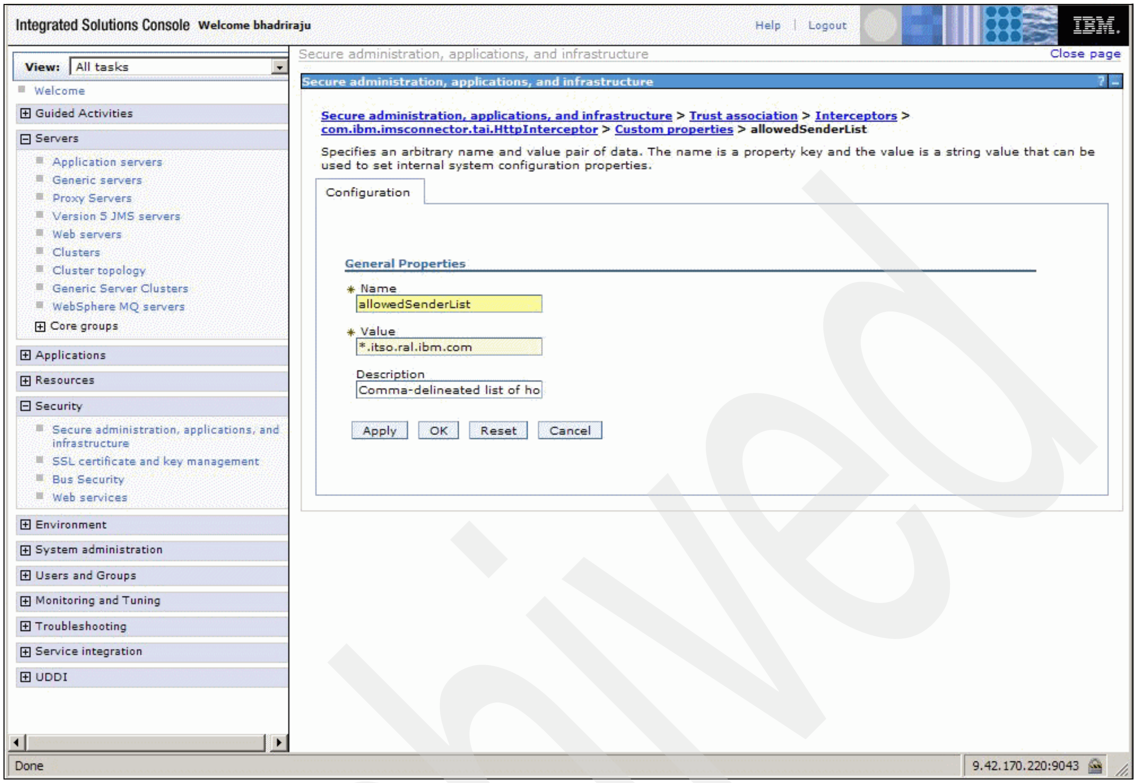


Figure 3-26 Configure HTTPInterceptor - New property

6. Repeat steps 4 and 5 with the following data:
   – Name: **assertedIdentityHeaderType**
   – Value: **X-XCAP-Asserted-Identity**
7. Repeat steps 4 and 5 with the following data:
   – Name: **enableDefaultRoleMapping**
   – Value: **true**
8. Repeat steps 4 and 5 with the following data:
   – Name: **enableMultipleIDMapping**
   – Value: **true**
9. Click **Save** to save the WebSphere Application Server master configuration in the Secure administration, applications, and infrastructure panel.

4

Presence Server installation

In this chapter, we describe the necessary steps to install and configure the IBM WebSphere Presence Server in the base scenario environment. The steps include configuring the WebSphere Application Server infrastructure, setting up the database instances that the Presence Server requires, and the Service Integration Bus. The Presence Server deployment is followed by post installation configurations.

This chapter contains:

4.1 Configuring the WebSphere Application Server infrastructure

We must further configure the WebSphere Application Server before we can install the IBM WebSphere Presence Server. We start by creating the Node for the Presence Server Cluster and then configure the resource for JDBC and JMS. WebSphere Presence Server is deployed on top of the infrastructure as an Enterprise Application after the configuration.

Figure 4-1 shows an overview of the Presence Server installation.

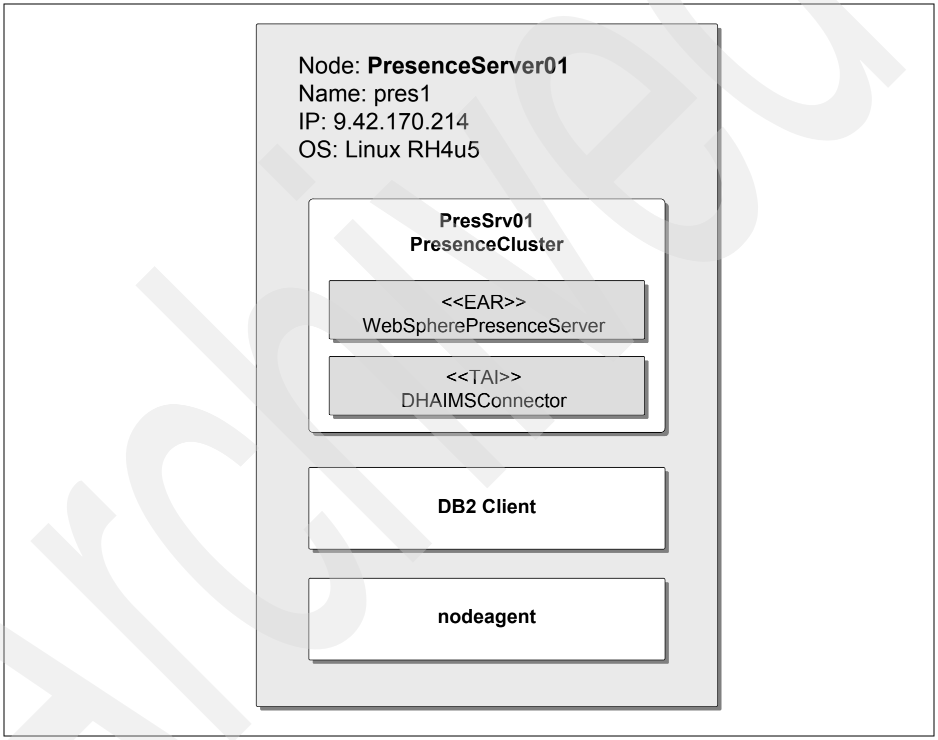


Figure 4-1 Overview of Presence Server installation

4.1.1 Preparing the WebSphere Application Server profile for Presence Server

The first step in preparing for Presence Server installation is to create a WebSphere Application Server profile for Presence Server. The profile is created and federated to the imsCell. In this profile, we create PresenceCluster, which is a management container for all WebSphere Application Servers that are running Presence Server. At this point, there is only one such server. In subsequent phases of this scenario (high availability and scalability phases), we will add more servers.

To prepare the WebSphere Application Server profile for Presence Server:

1. Login to the machine PresNode01, and run the command in Example 4-1.

Example 4-1 Create WebSphere Application Server profile for PresNode01

```
/opt/IBM/WebSphere/AppServer/bin/manageprofiles.sh -create -profileName
PresNode01 -profilePath
/opt/IBM/WebSphere/AppServer/profiles/PresNode01 -templatePath
/opt/IBM/WebSphere/AppServer/profileTemplates/managed -nodeName
PresNode01 -hostName pres1.itso.ral.ibm.com
```

2. Federate the created node to the imsCell using the command in Example 4-2.

Example 4-2 Federate PresNode01 to the imsCell

```
/opt/IBM/WebSphere/AppServer/profiles/PresNode01/bin/addNode.sh
ldap.itso.ral.ibm.com -username wasadmin -password <password>
-profileName PresNode01
```

3. Synchronize the created profile using the **syncNode** command, as shown in Example 4-3.

Example 4-3 Synchronize PresNode01

```
/opt/IBM/WebSphere/AppServer/profiles/PresNode01/bin/syncNode.sh
ldap.itso.ral.ibm.com -username wasadmin -password <password>
```

4. Start the recently created node PresNode01, as shown in Example 4-4.

Example 4-4 Start Presence Node PresNode01

```
/opt/IBM/WebSphere/AppServer/profiles/PresNode01/bin/startNode.sh
```

4.1.2 Creating a WebSphere Application Server Cluster and Server for Presence Server

Presence Server is an Enterprise Application that runs on the Application Server; therefore, before we can install Presence Server, we must create the Application Server for it. The Application Server is managed by the Cluster that is dedicated to Presence Server. In this section, we create the WebSphere Application Server Cluster and Server.

To create the WebSphere Application Server Cluster and Server:

1. Login to the Integrated Solutions Console as a user with the Administrator role, for example wasadmin.
2. Click **Servers** → **Clusters** → **New**.
3. In the Cluster name field, enter `PresenceCluster`.
4. Select the **Configure HTTP session memory-to-memory replication** option, as shown in Figure 4-2.
5. Click **Next**.

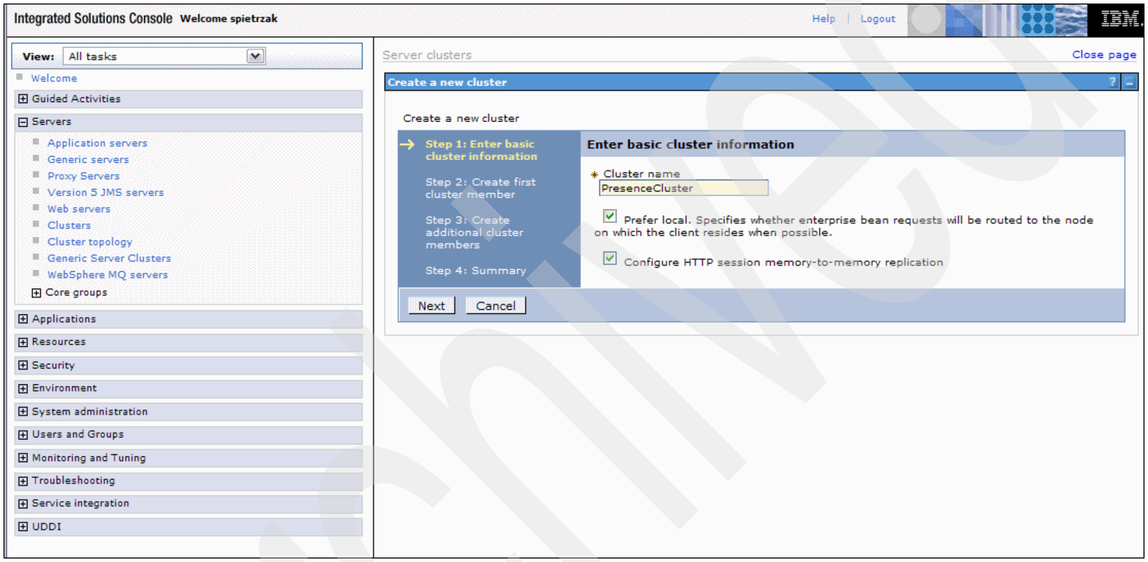


Figure 4-2 Presence Server Cluster - Step 1

6. In the Member N field, type the first server in the cluster, `PresSrv01`, and from the pull-down menu, select **PresNode01**, as shown in Figure 4-3 on page 97.
7. Click **Next** to get to the Summary page.

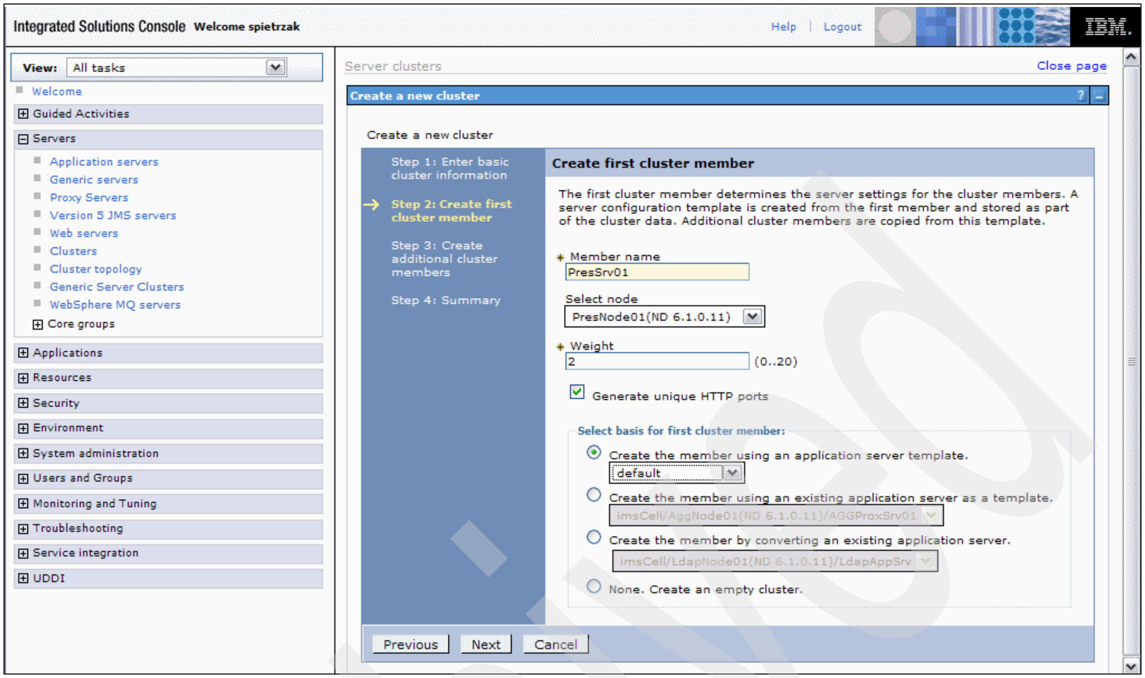


Figure 4-3 Create PresenceCluster - Step 2

8. Check the summary (Figure 4-4 on page 98) to ensure that it is correct, and then click **Finish**.
9. Save changes to the master configuration.

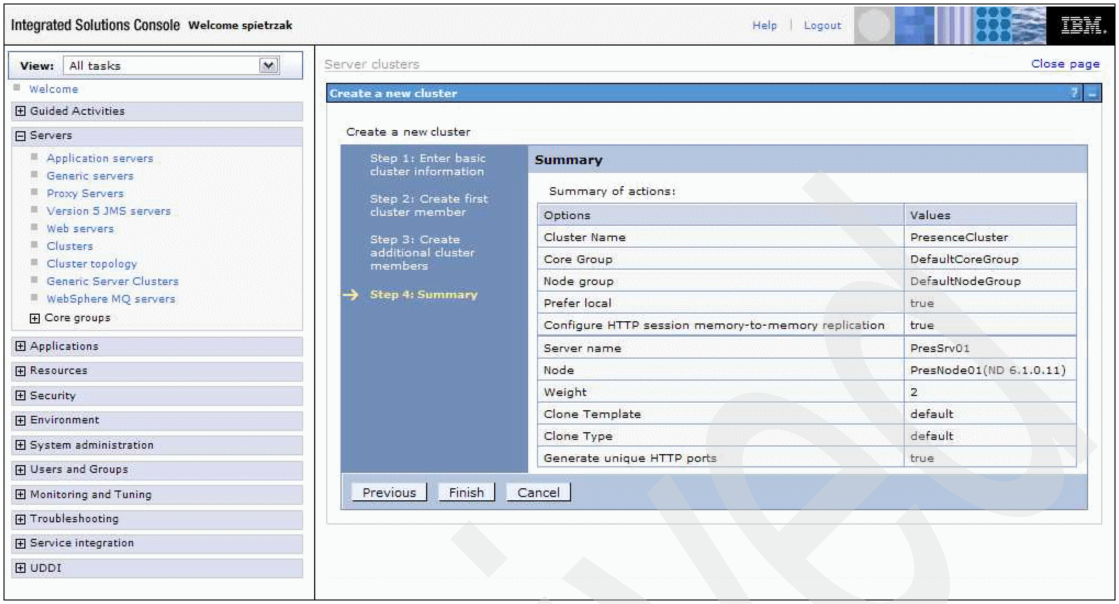


Figure 4-4 Presence Serve - Cluster - Step 3

Starting the Presence Cluster

To start the Presence Cluster:

1. In the Navigation pane, click **Servers** → **Clusters**.
2. Select **PresenceCluster** from the list, as shown in Figure 4-5.
3. Click **Start** to start the Cluster.

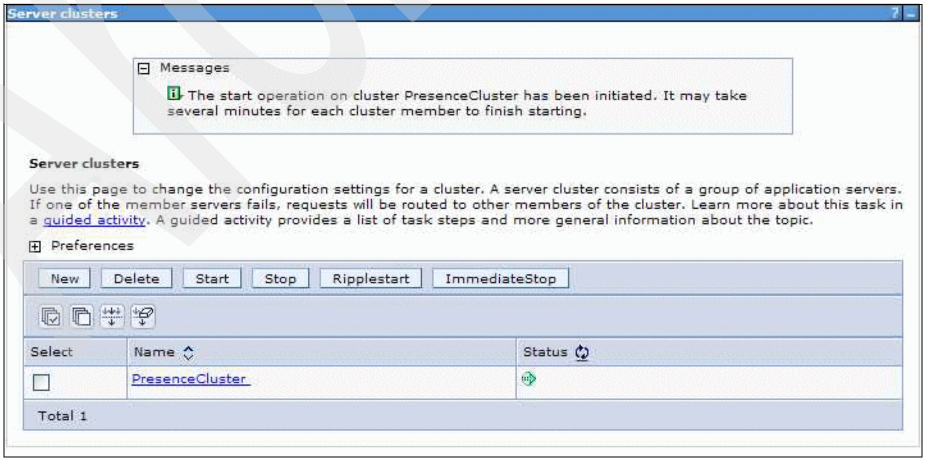


Figure 4-5 Presence Server - Cluster - Step 4

4. Inspect the log files for PresSrv01 to ensure that the configurations are accurate and that there are no errors. Example 4-5 shows a sample log list.

Example 4-5 Sample logs list starting PresenceCluster

```
[root@pres1 PresSrv01]# pwd
/opt/ibm/WebSphere/AppServer/profiles/PresNode01/logs/PresSrv01
[root@pres1 PresSrv01]# ls -la
total 64
drwxr-xr-x  2 root root  4096 Sep 21 14:48 .
drwxr-xr-x  5 root root  4096 Sep 21 14:59 ..
-rw-r--r--  1 root root     0 Sep 21 14:48 native_stderr.log
-rw-r--r--  1 root root     0 Sep 21 14:48 native_stdout.log
-rw-r--r--  1 root root     5 Sep 21 14:49 PresSrv01.pid
-rw-r--r--  1 root root     0 Sep 21 14:48 SystemErr.log
-rw-r--r--  1 root root 23889 Sep 21 14:49 SystemOut.log
```

5. Ensure that SystemOut.log does not have any application errors, and finish with the sentence `Server PresSrv01 open for e-business.`

4.1.3 Creating the WebSphere Application Server profiles for the SIP Proxy Server

In this section, we describe how to create the WebSphere Application Server profile for the Proxy Server, which is used as front-end component for Presence Server.

To create the WebSphere Application Server profiles for the SIP Proxy Server:

1. Login to the ProxNode01 machine, and run the command in Example 4-6.

Example 4-6 Create WebSphere Application Server profile for ProxNode01

```
/opt/IBM/WebSphere/AppServer/bin/manageprofiles.sh -create -profileName
ProxNode01 -profilePath
/opt/IBM/WebSphere/AppServer/profiles/ProxNode01 -templatePath
/opt/IBM/WebSphere/AppServer/profileTemplates/managed -nodeName
ProxNode01 -hostName prox1.itso.ral.ibm.com
```

2. Federate the created node to the imsCell using the command in Example 4-7.

Example 4-7 Federate ProxNode01 to the imsCell

```
/opt/IBM/WebSphere/AppServer/profiles/ProxNode01/bin/addNode.sh
ldap.itso.ral.ibm.com -username wasadmin -password <password>
-profileName ProxNode01
```

3. Synchronize the created profile using the **syncNode** command, as shown in Example 4-8.

Example 4-8 Synchronize ProxNode01

```
/opt/IBM/WebSphere/AppServer/profiles/ProxNode01/bin/syncNode.sh
ldap.itso.ral.ibm.com -username wasadmin -password <password>
```

4. Start the node using the startNode command, as shown in Example 4-9.

Example 4-9 Starting ProxNode01

```
[root@prox1 bin]# pwd
/opt/ibm/WebSphere/AppServer/profiles/ProxNode01/bin
[root@prox1 bin]# ./startNode.sh
ADMU0116I: Tool information is being logged in file

/opt/IBM/WebSphere/AppServer/profiles/ProxNode01/logs/nodeagent/startServer.log
ADMU0128I: Starting tool with the ProxNode01 profile
ADMU3100I: Reading configuration for server: nodeagent
ADMU3200I: Server launched. Waiting for initialization status.
ADMU3000I: Server nodeagent open for e-business; process id is 8798
```

4.1.4 Creating the SIP Proxy Servers

On top of the created Proxy profile, we created the SIP Proxy Server using the following steps:

1. Login to the WebSphere Application Server Console as a user with administration privileges.
2. In the navigation pane, click **Server** → **Proxy Servers**.
3. Click **New**, and perform the following actions, as shown in Figure 4-6 on page 101:
   a. Select the **ProxNode01** node.
   b. In the Server Name field, enter **SIPProxSrv01**.
4. Click **Next**.

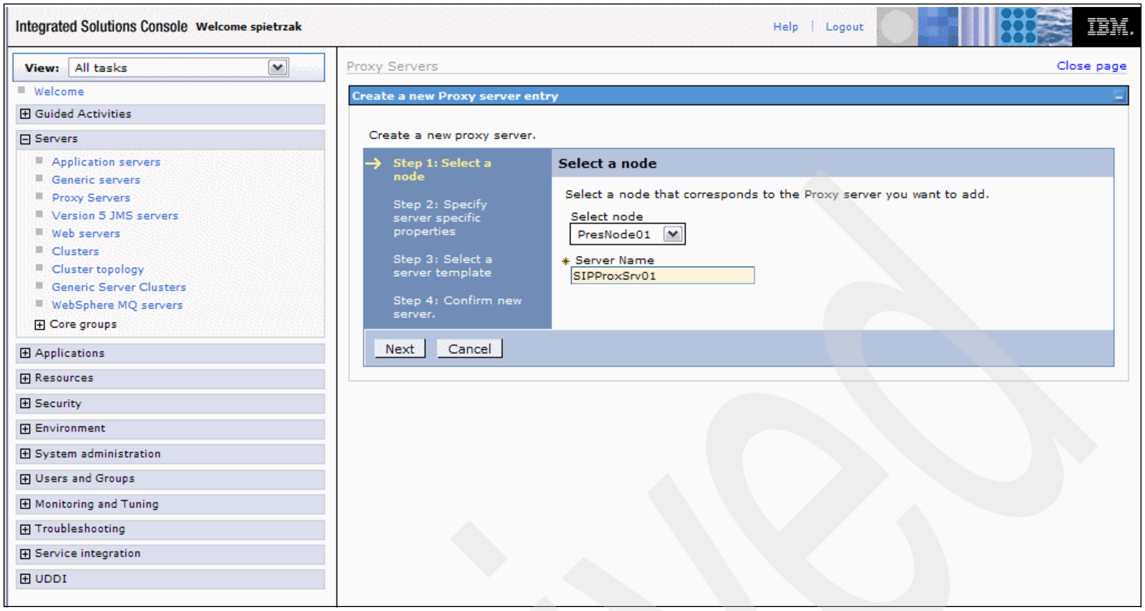


Figure 4-6 WebSphere Application Server Proxy Server - Step 1

5. Select the following options, as shown in Figure 4-7 on page 102:
   - Both HTTP and SIP protocols
   - **Generate Unique ports**
6. Click **Next**.

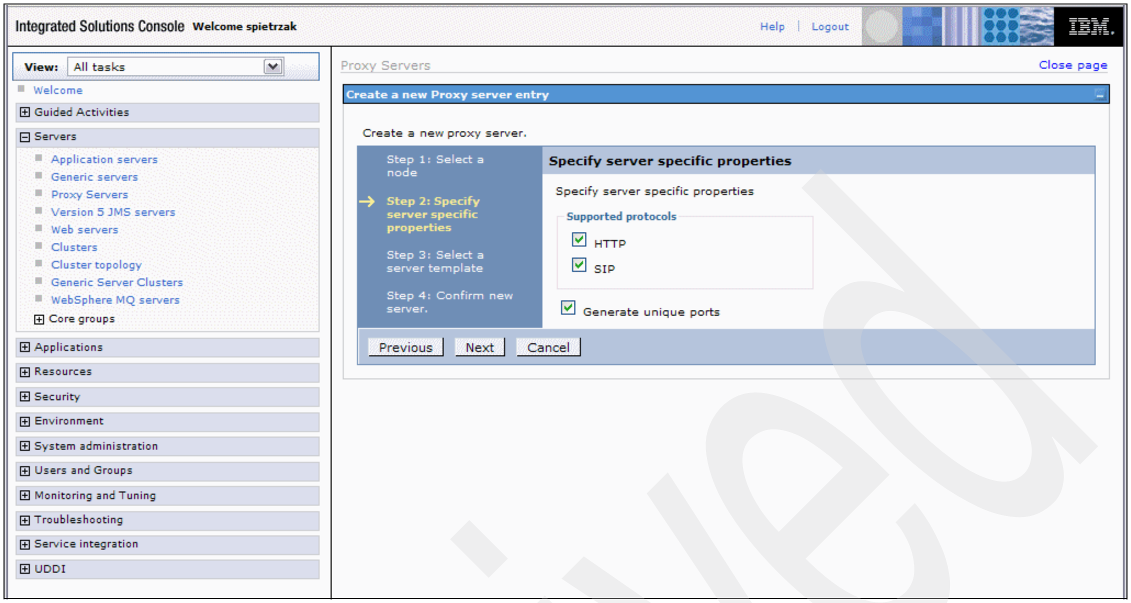


Figure 4-7 WebSphere Application Server Proxy Server - Step 2

7. Verify the summary, as shown in Figure 4-8, and click **Finish**.
8. Save the changes to the master configuration.

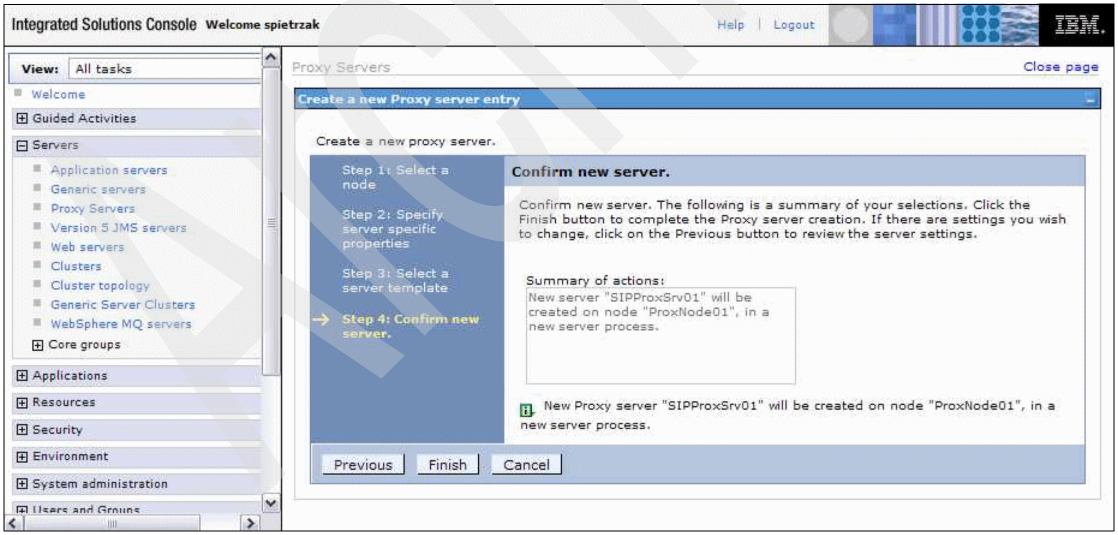


Figure 4-8 WebSphere Application Server Proxy Server - Step 4

4.1.5 Associating SIP Proxies with Application Servers

We created both the Application Server and the Proxy Server. Now it is time to connect them together. Additionally, we configure the Proxy Server to work with the Load Balancer we are using in the environment.

To associate SIP Proxies with Application Servers:

1. From the navigation pane, click **Servers** → **Proxy Servers**.
2. Click the recently created Proxy Server, **SIPProxSrv01**, as shown in Figure 4-9.
3. Under the Proxy Settings section, click **SIP Proxy Server Settings** → **SIP Proxy Settings**.

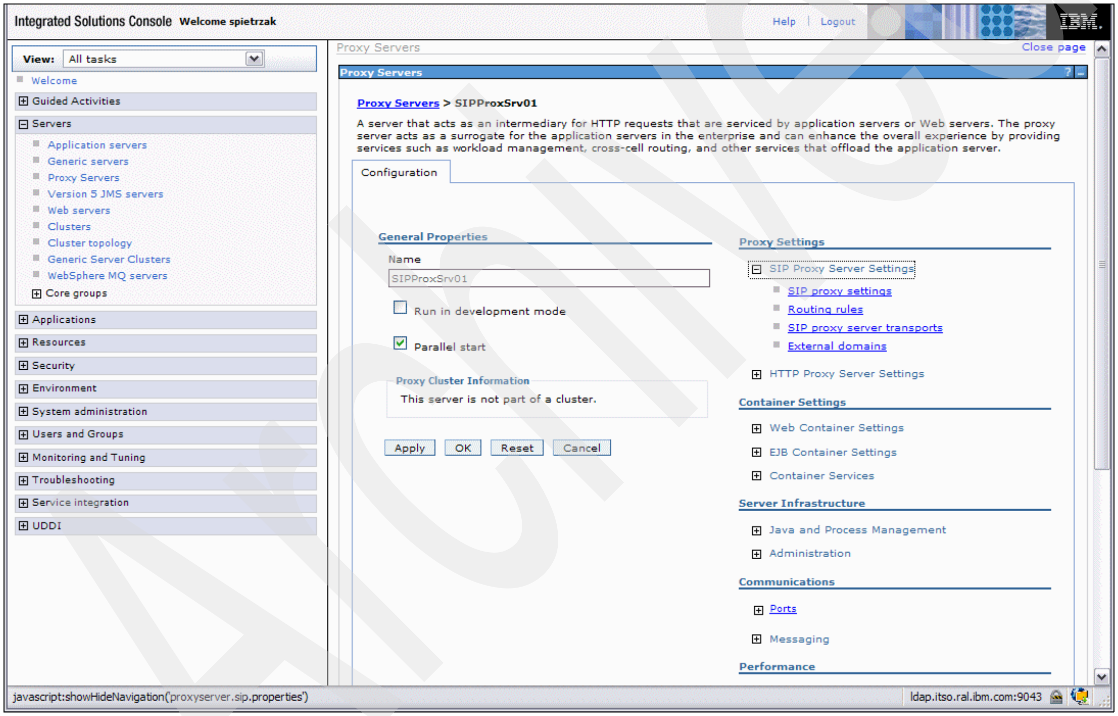


Figure 4-9 WebSphere Application Server Proxy Server - Step 4

4. Enter the following parameters on the configuration page, as shown in Figure 4-10 on page 104:
   - Default cluster: **PresenceCluster**
   - Enable TCP spryer: **checked**
   - TCP host: **9.42.170.211** (IP Address of Load Balancer)

– TCP Port: **5060**
– Enable SSL sprayer: **checked**
– SSL host: **9.42.170.211** (IP Address of Load Balancer)
– SSL Port: **5061**
– Enable UDP sprayer: **checked**
– UDP host: **9.42.170.211** (IP Address of Load Balancer)
– UDP port: **5060**

Additionally you might want to enable logging for SIP traffic, so select **Enable access logging**.

5. Click **OK**, and save changes to the master repository.

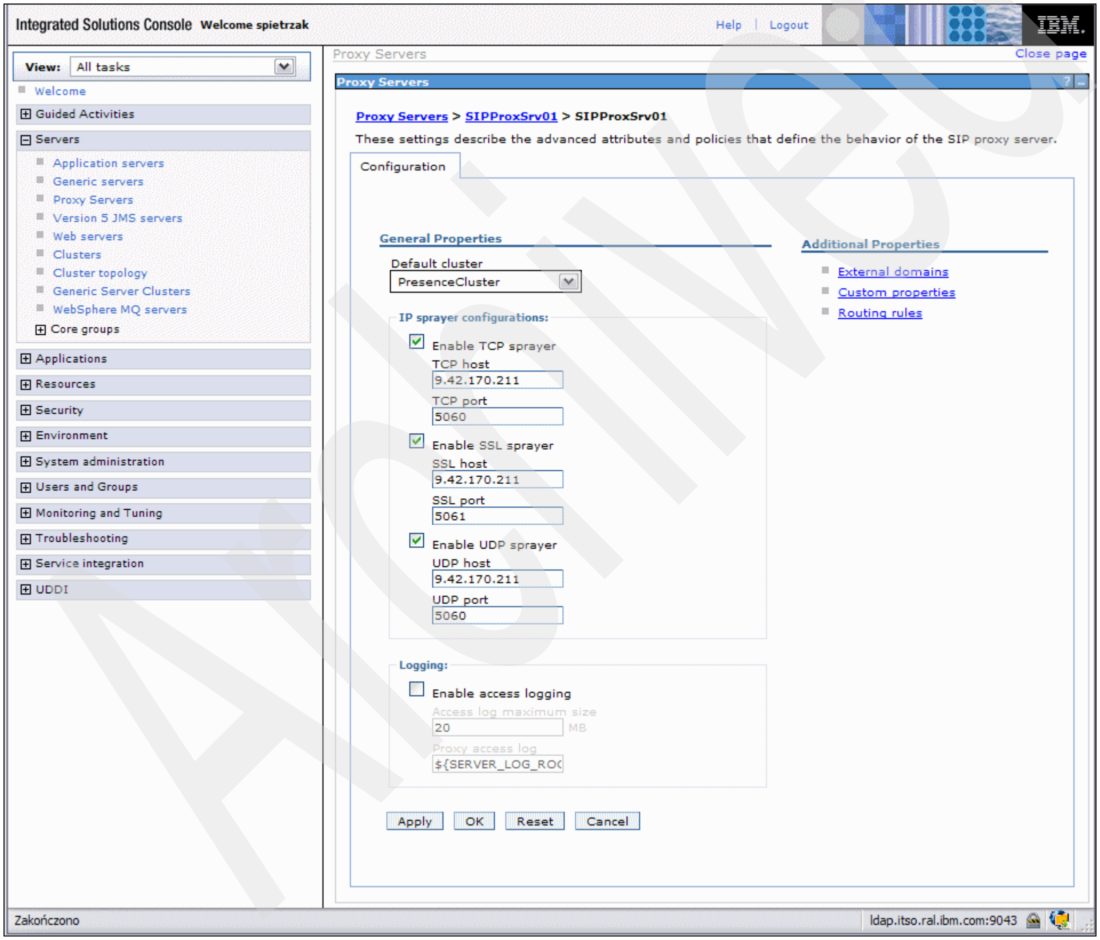


Figure 4-10 WebSphere Application Server Proxy Server - Step 5

Note: We plan to grow our environment for high availability; therefore, the Load Balancer in front of the Proxy Servers is necessary to spray the traffic to the number of proxies. If you are not going to use Load Balancer at this stage, you do not have to configure sprayer parameters for proxy.

4.2 Setting Up the Presence Server databases

In this section, we describe the process for creating database instances that the Presence Server uses. We create two databases. The main database is PSDB, and it contains all of the configuration and runtime data that the Presence Server uses. The second database, PSUR, stores Usage Records.

To create the databases and tables for Presence Server, we use scripts that are delivered with the Presence Server installation image. After the databases are created, we configure the WebSphere JDBC Resources to make the databases accessible from the WebSphere Application Server infrastructure.

4.2.1 Creating the PSDB database

To create the PSB database:

1. Login to the database server, NodeDB, as a user with database administration privileges, in our case it is db2inst1.
2. Switch to the temporary directory on the NodeDB machine.
3. Unpack the installation tar archive to a temporary directory.

   Example 4-10 shows the files for the Presence Server databases.

Example 4-10 Files for the Presence Server databases

```
[root@db2 ~]# su - db2inst1
[db2inst1@db2 ~]$ cd /usr/src/itso-lab/
[db2inst1@db2 itso-lab]$tar -xf /mnt/IMS_62/IBMWebSpherePresenceServer.tar
[db2inst1@db2 itso-lab]$ ls -la
total 24
drwxrwxrwx  5 root     root    4096 Sep 23 13:54 .
drwxr-xr-x  5 root     root    4096 Sep 23 13:19 ..
drwxr-xr-x  3 db2inst1 db2grp1 4096 Sep 16 11:46 installableApps
drwxr-xr-x  3 db2inst1 db2grp1 4096 Sep 16 11:46 lib
drwxr-xr-x  3 db2inst1 db2grp1 4096 Sep 16 11:46 properties
```

4. Go to the directory with the database scripts for the Presence database. Example 4-11 on page 106 provides scripts for the Presence database.

Example 4-11 Scripts for the Presence database

```
[db2inst1@db2 presence]$ pwd
/usr/src/itso-lab/installableApps/presence/scripts/dbScripts/presence
```

5. Edit the RunConfigDB2.sh file.
6. Update values for parameters that are listed in Table 4-1.

Table 4-1 PSDB - parameters

Parameter	New value	Description
DBSERVER	db2.itso.ral.ibm.com	Host name of the database server.
DBPORT	50000	Port under which the database is listed
DBNAME	PSDB	Name of the Presence database
DBALIAS	PSDB	Alias by which the Presence database is available
DBLOCALE	US	DB2 Locale code
DBINSTANCE	db2inst1	Name of the database instance
DBINSTANCPW	<password>	Password for the database instance
DBUSER	db2inst1	User ID with administration privileges on the database
DBUSERPW	<password>	Password for the User ID
DBDIR	/home/db2inst1	Presence database home directory
DBCREATE	TRUE	Boolean value that specifies whether the database is recreated

7. Run the edited script, as shown in Example 4-12. During this process many messages appear from the script. Some of them can contain information about errors. If there are errors, it does not necessarily mean that script was wrong.

Example 4-12 Running the RunConfigDB2.sh script

```
[db2inst1@db2 presence]$ ./RunConfigDB2.sh

...
```

```
Creating server database.
SQL1031N  The database directory cannot be found on the indicated file system.
SQLSTATE=58031
SQL1031N  The database directory cannot be found on the indicated file system.
SQLSTATE=58031
SQL1027N  The node directory cannot be found.
DB20000I  The CREATE DATABASE command completed successfully.
DB20000I  The CATALOG TCPIP NODE command completed successfully.
DB21056W  Directory changes may not be effective until the directory cache is
refreshed.
SQL1005N  The database alias "PSDB" already exists in either the local
database directory or system database directory.

...

./ConfigDB2.sh is complete.
```

8. Verify that the tables in the PSDB database were created correctly, as shown in Example 4-13.

Example 4-13 Verification of the created database PSDB

```
[db2inst1@db2 presence]$ db2 connect to PSDB

   Database Connection Information

 Database server        = DB2/LINUX 9.1.2
 SQL authorization ID   = DB2INST1
 Local database alias   = PSDB

[db2inst1@db2 presence]$ db2 list tables

Table/View                      Schema          Type  Creation time
------------------------------- --------------- ----- --------------------------
CONFIG                          DB2INST1        T     2007-09-23-15.21.04.151156
FULLDOC                         DB2INST1        T     2007-09-23-15.21.04.499508
MANAGING_SERVERS                DB2INST1        T     2007-09-23-15.21.04.781121
PRESENCE_AUTH_RULES             DB2INST1        T     2007-09-23-15.21.05.224669
PUBLISH                         DB2INST1        T     2007-09-23-15.21.04.295852
WATCHERS                        DB2INST1        T     2007-09-23-15.21.04.996934

  6 record(s) selected.

[db2inst1@db2 presence]$
```

4.2.2 Creating the Presence Usage Records database

To create the PSUR database:

1. Go to the directory that contains the database scripts for the Presence Usage Records database, as shown in Example 4-14.

Example 4-14 Usage Records script location

```
[db2inst1@db2 presence]$ pwd
/usr/src/itso-lab/installableApps/presence/scripts/dbScripts/usageRecords
```

2. Prepare information values for the parameters, as listed in Table 4-2.

Table 4-2 PSUR - parameters

Parameter	Value	Description
DBSERVER	db2.itso.ral.ibm.com	Host name of the database server.
DBPORT	50000	Port under which the database is listing
DBNAME	PSUR	Name of the Presence database
DBALIAS	PSUR	Alias by which the Presence database is available
DBLOCALE	US	DB2 Locale code
DBINSTANCE	db2inst1	Name of the database instance
DBINSTANCPW	<password>	Password for the database instance
DBUSER	db2inst1	User ID with administration privileges on the database
DBUSERPW	<password>	Password for the User ID
DBDIR	/home/db2inst1	Presence database home directory
DBCREATE	TRUE	Boolean value that specifies whether the database is recreated
LOCAL	TRUE	Boolean value that specify whether the database is on the local host or on the remote host

3. Run the command in Example 4-15 on page 109 using the parameters from Table 4-2.

Example 4-15 PSUR database creation

```
./crtsrvDb2.sh db2.itso.ral.ibm.com 50000 PSUR PSUR \
US db2inst1 <password> db2inst1 <password> UsageDbDb2.ddl TRUE TRUE
```

4. Verify that the required database table was created correctly, as shown in Example 4-16.

Example 4-16 Verification of the PSUR database

```
[db2inst1@db2 usageRecords]$ db2 connect to PSUR

   Database Connection Information

 Database server        = DB2/LINUX 9.1.2
 SQL authorization ID   = DB2INST1
 Local database alias   = PSUR

[db2inst1@db2 usageRecords]$ db2 list tables

Table/View                      Schema          Type  Creation time
------------------------------- --------------- ----- --------------------------
USAGERECORDS                    DB2INST1        T     2007-09-23-15.48.50.476091

1 record(s) selected.
```

Creating authentication alias for the Presence databases

In this section, we create authentication alias in WebSphere Application Server, which are used to authenticate for access to the Presence Server database.

To create authentication alias for the Presence Server database:

1. Login to the WebSphere Application Server Console.
2. From the navigation pane, select **Security** → **Secure Administration, Applications, and infrastructure**.
3. Under Authentication, expand **Java Authentication and Authorization Service**, and click **J2C authentication data**, as shown in Figure 4-11 on page 110.

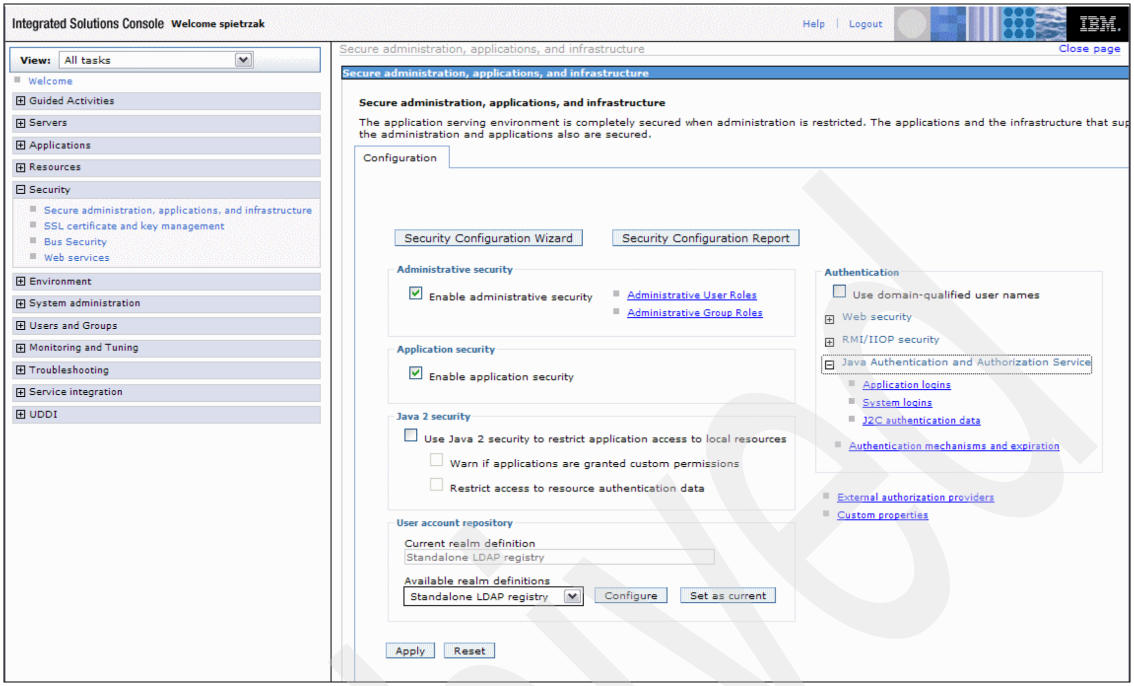


Figure 4-11 Authentication alias for Presence DB - Step 1

4. Click **New**.
5. Enter the following parameters, as shown in Figure 4-12 on page 111:
   - Alias: **PresenceDBAlias**
   - User ID: **db2inst1**
   - Password: **<db_password>**
6. Click **OK**.
7. Save the changes to the master configuration.

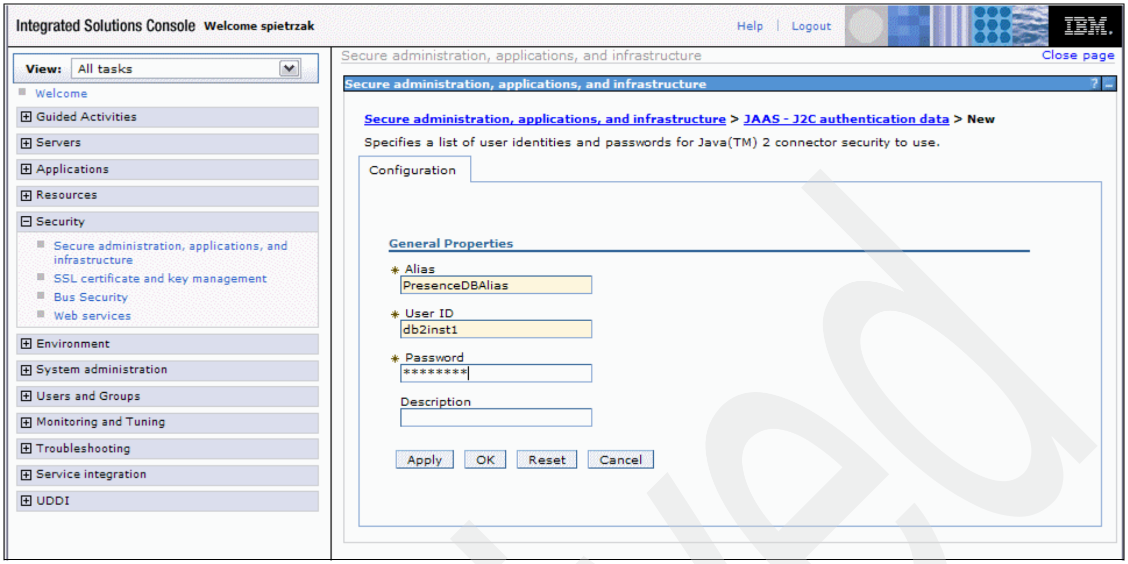


Figure 4-12 Authentication alias for Presence DB - Step 2

4.2.3 Configuring the Presence Server database data source

In this section, we configure the data source jdbc/db that connects to the Presence Server PSDB database.

1. From the navigation pane, select **Resources** → **JDBC** → **JDBC Providers**.
2. Click the JDBC Provider **DB2 Universal JDBC Driver Provider**, as shown in Figure 4-13 on page 112.

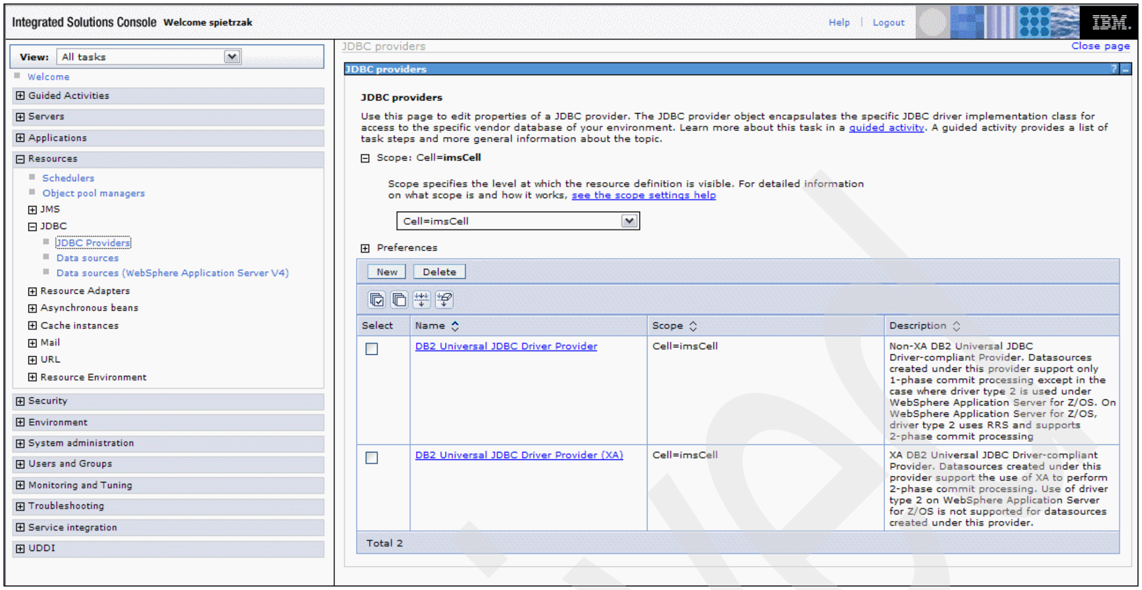


Figure 4-13 Presence DB data source - Step 1

3. Under Additional Properties, click **Data Sources**. See Figure 4-14 on page 113.

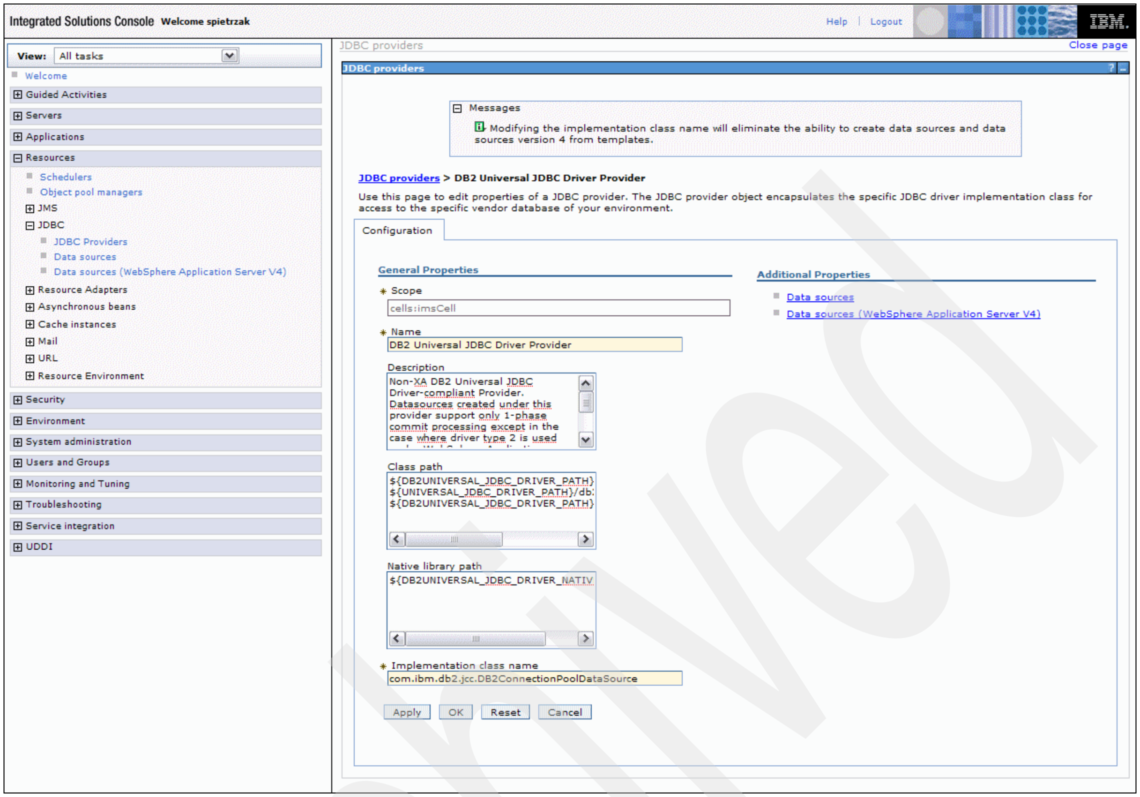


Figure 4-14 Presence DB data source - Step 2

4. Click **New**, and enter the following values, as shown in Figure 4-15 on page 114:
   - JNDI name **jdbc/db**
   - Component managed authentication alias **PresenceDBAlias**
5. Click **Next**.

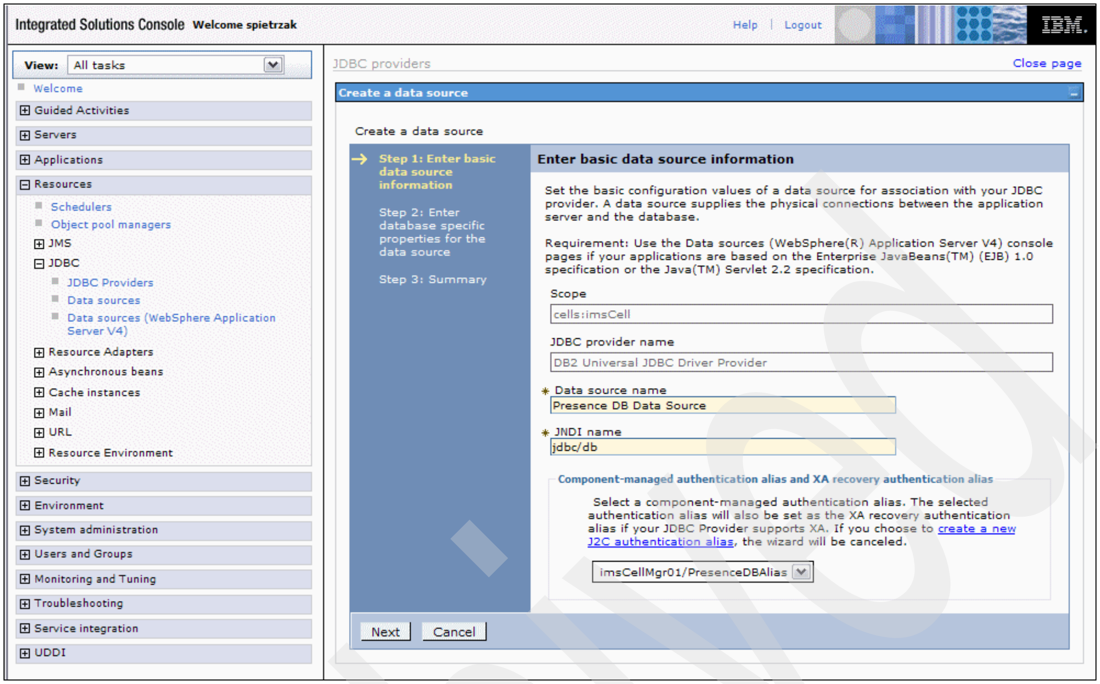


Figure 4-15 Presence DB data source - Step 3

6. On the next page, enter the following database parameters, as shown in Figure 4-16 on page 115:
   - Database name: **PSDB**
   - Driver type: **4**
   - Server name: **db2.itso.ral.ibm.com**
   - Port number: **50000**
7. Click **Next**.

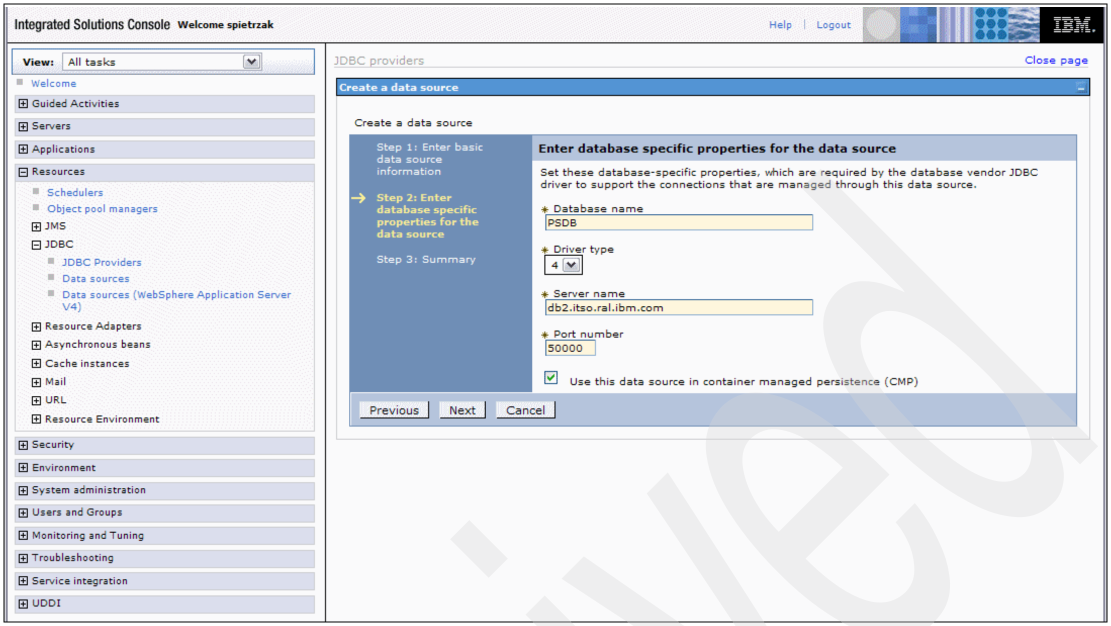


Figure 4-16 Presence DB data source - Step 4

8. Click **Finish**.
9. Save the changes to the master configuration.

4.2.4 Configuring the Presence Server Usage Report data source

In this section, we configure the data source jdbc/ur that connects to the Presence Server PSUR database:

1. From the navigation pane, select **Resources** → **JDBC** → **JDBC Providers**.
2. Click the JDBC Provider **DB2 Universal JDBC Driver Provider**, as shown in Figure 4-17 on page 116.

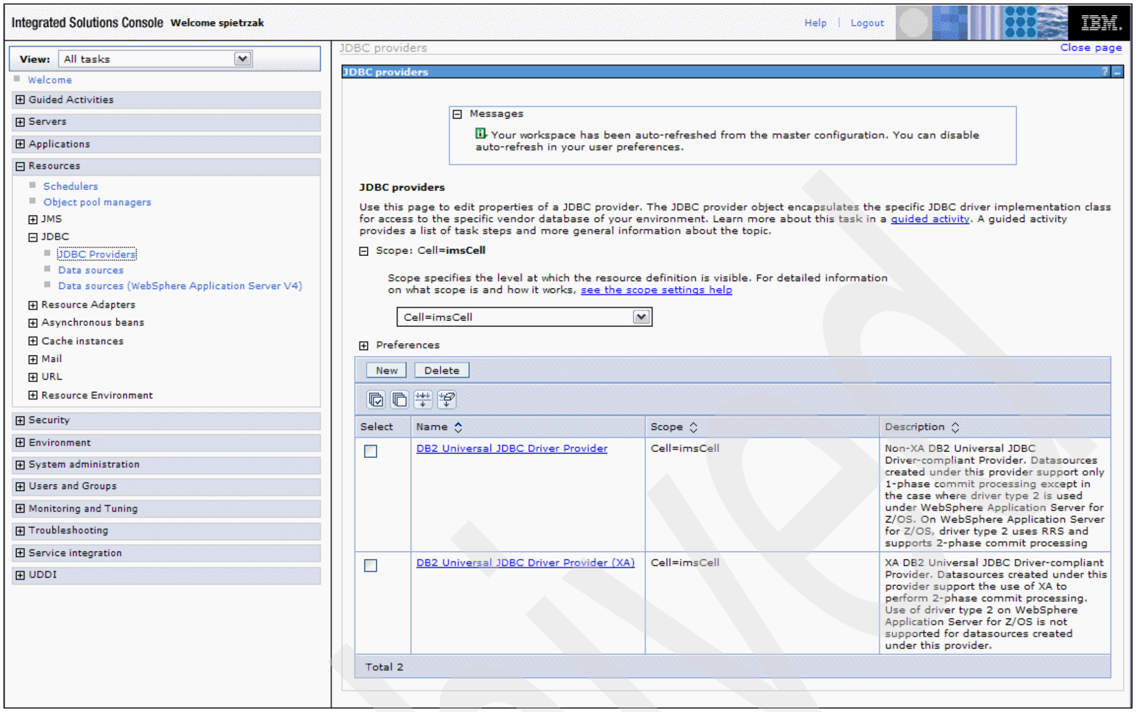


Figure 4-17 Presence DB data source - Step 5

3. Under Additional Properties, click **Data Sources**, as shown in Figure 4-18 on page 117.

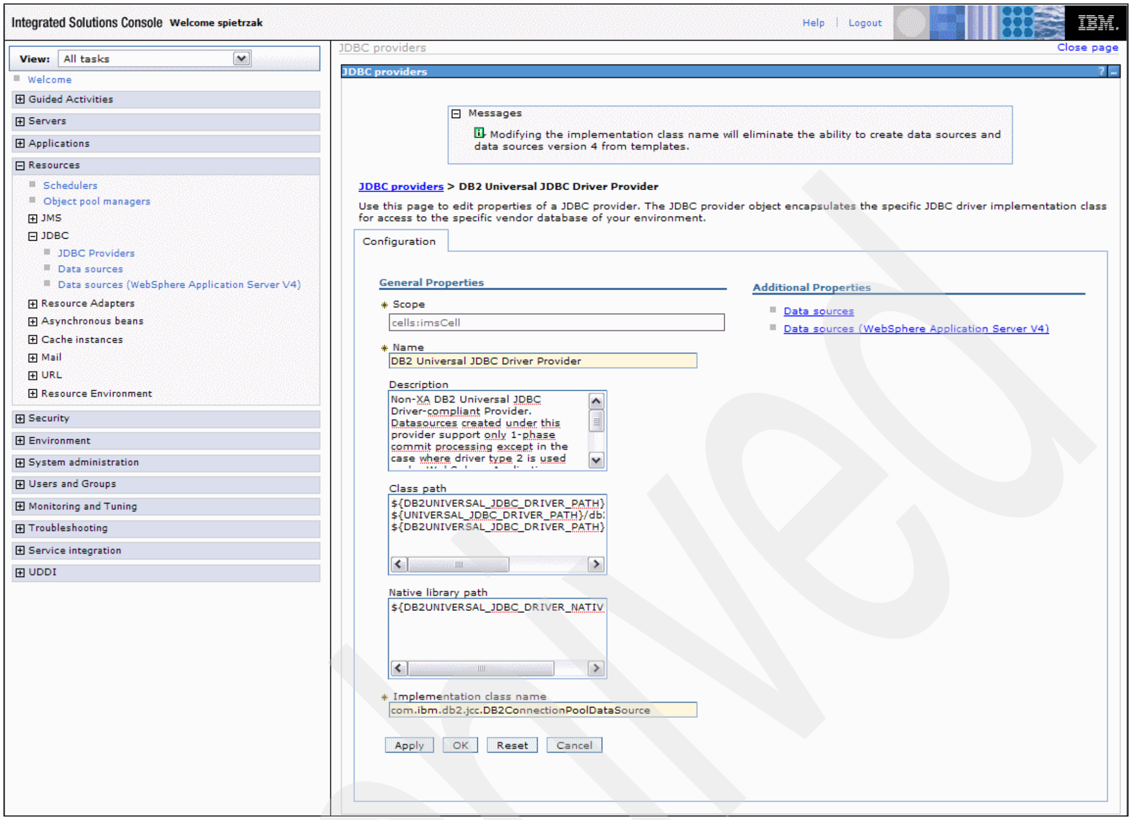


Figure 4-18 Presence DB data source - Step 6

4. Click **New**, and enter the following values, as shown in Figure 4-19 on page 118:
   - JNDI name: **jdbc/ur**
   - Component managed authentication alias: **PresenceDBAlias**
5. Click **Next**.

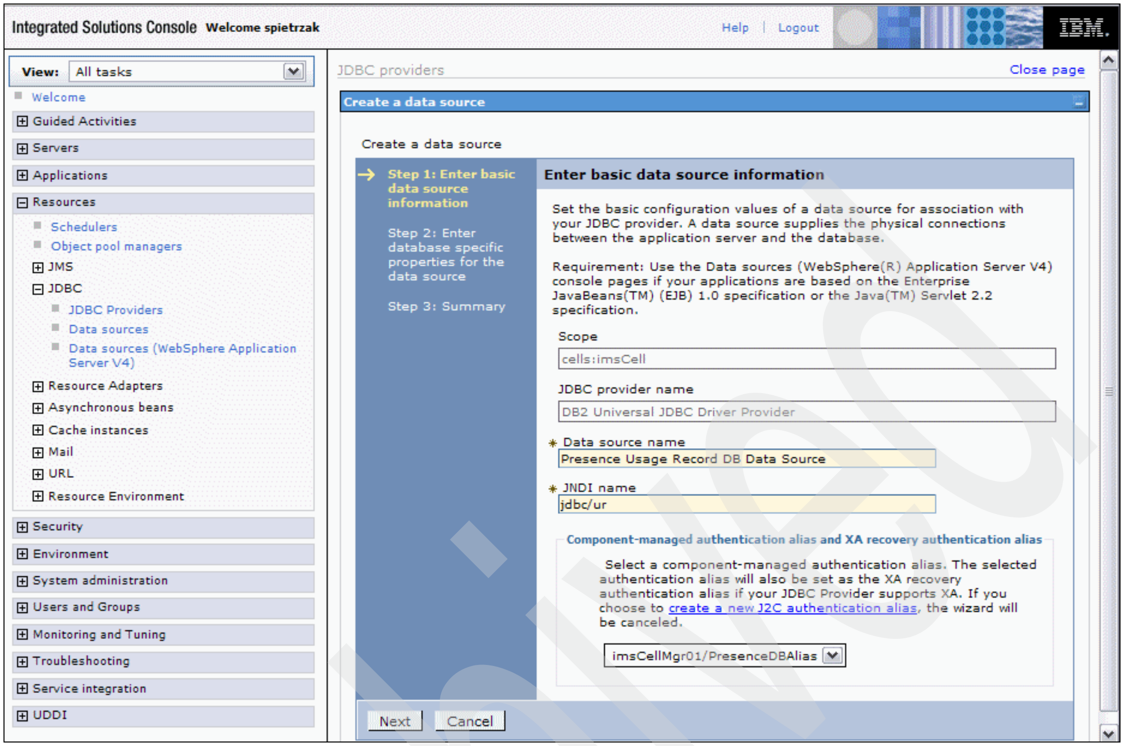


Figure 4-19 Presence DB data source - Step 7

6. Enter the following database parameters, as shown in Figure 4-20 on page 119:
   - Database name: **PSUR**
   - Driver type: **4**
   - Server name: **db2.itso.ral.ibm.com**
   - Port number: **50000**
7. Click **Next**, and verify that the parameters are correct on the Summary page (Figure 4-20 on page 119).
8. Click **Finish**, and save changes to the master configuration.

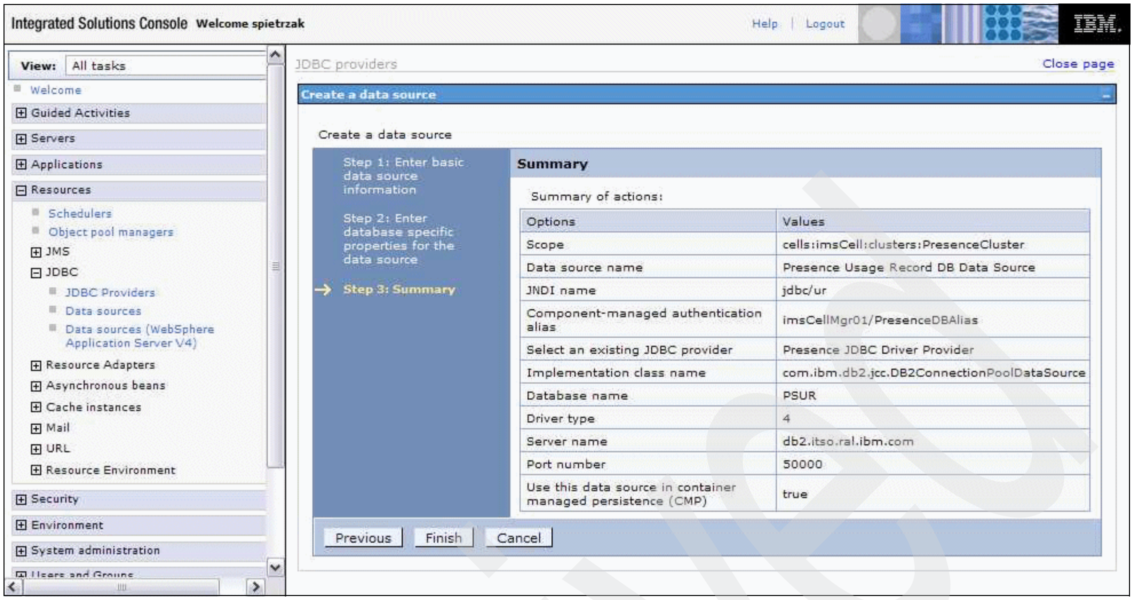


Figure 4-20 Presence DB data source - Step 8

4.3 Preparing the integration infrastructure

Because WebSphere Presence Server is designed to work in a clustered environment, Service Integration Bus and JMS are used to communicate across the Cluster.

4.3.1 Creating Service Integration Bus

To prepare the Service Integration Bus for the Presence Server:

1. From the navigation pane, click **Service Integration** → **Busses** → **New**, and perform the following actions:
   a. Type **PresenceBus** as the name for the new bus, select the **Bus security** option, as shown in Figure 4-21 on page 120.
   b. Click **Next**, and then click **Finish** on the Summary page.

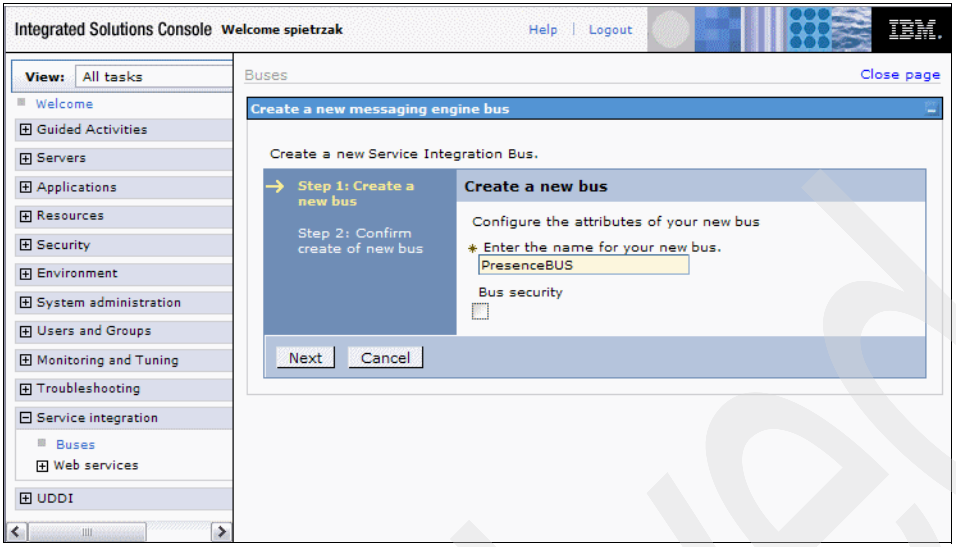


Figure 4-21 Service Integration Bus - Step 1

2. Go to **Busses** → **PresenceBus** → **Bus Member**, click **New**, and perform the following actions, as shown in Figure 4-22:
   a. Select **Server**. From the pull-down menu, select **PresNode01:PresSrv01**.
   b. Select **File store** as type of messages store.
   c. Click **Next**.

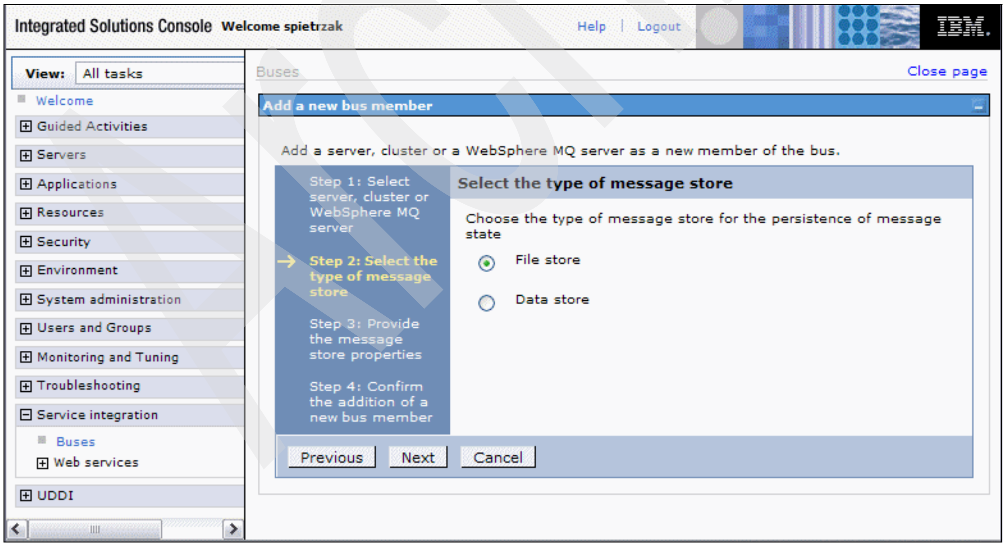


Figure 4-22 Service Integration Bus - Step 2

d. On Figure 4-23, accept the default values, and click **Next**.
e. On the Summary page, click **Finish**.
f. Save the changes to the master repository.

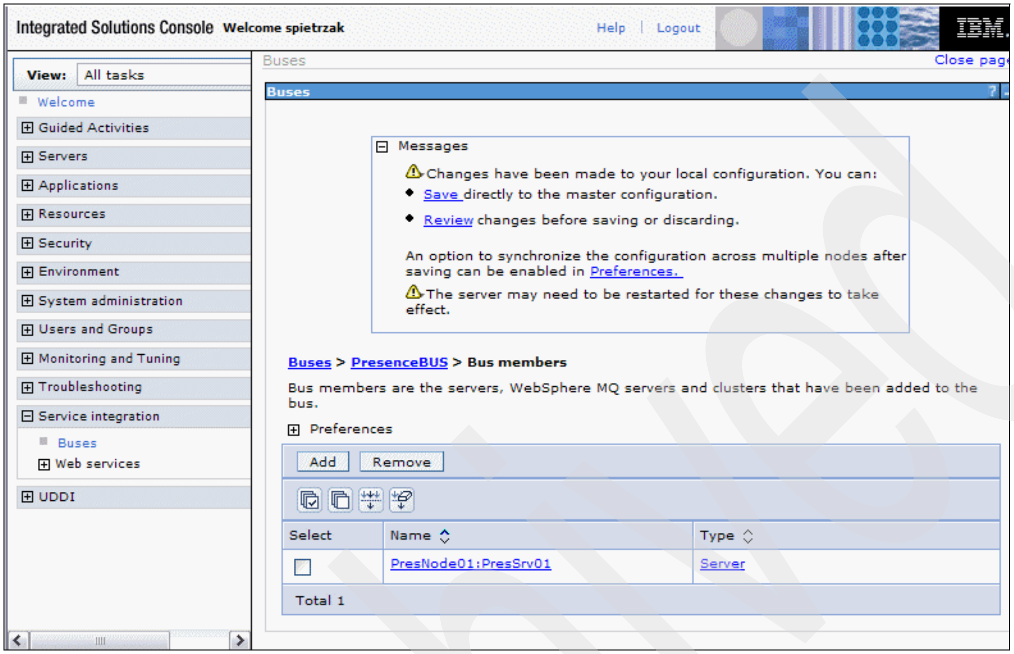


Figure 4-23 Service Integration Bus -Step 3

4.3.2 Creating the JMS Topic Connection Factory

1. From the navigation pane, click **Resources** → **JMS** → **JMS Provider**, and perform the following actions, as shown in Figure 4-24 on page 122:

   a. From the pull-down menu, select **Cluster=PresenceCluster**.
   b. From the JMS providers list, click **Default messaging provider**.

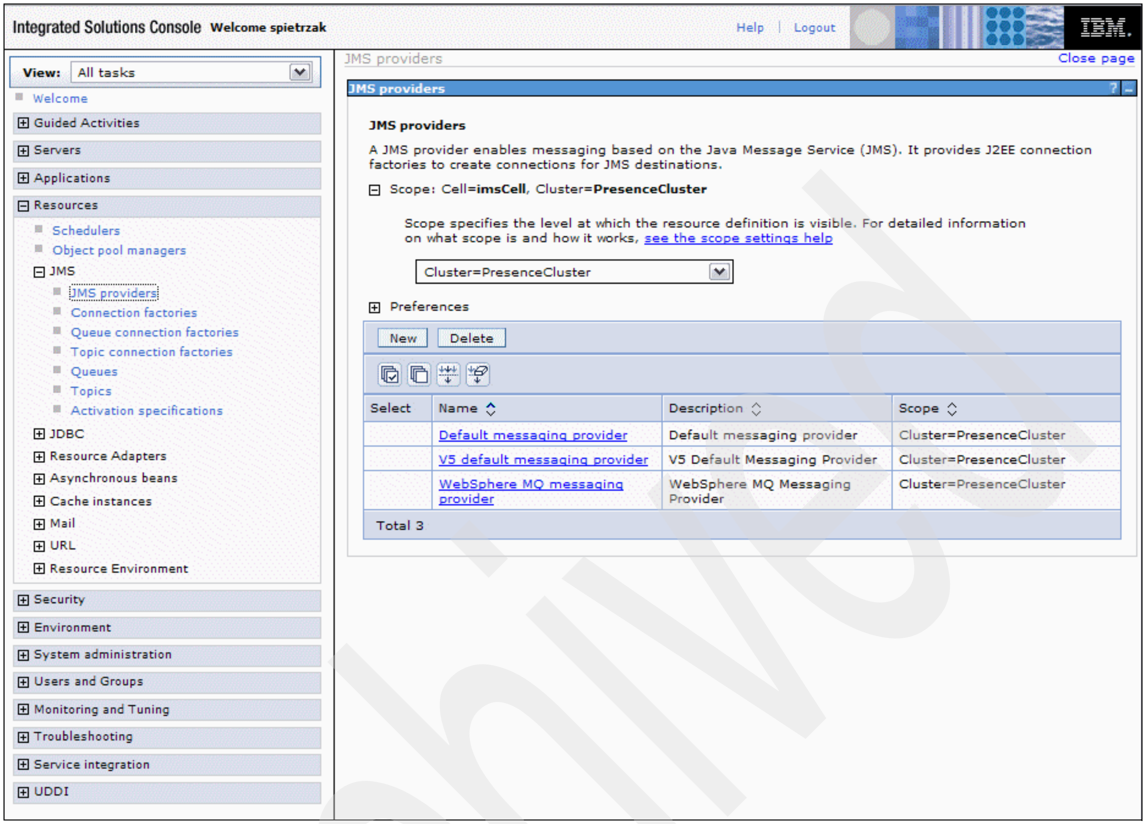


Figure 4-24 JMS Topic Connection Factory - Step 1

2. Under Additional Properties click **Topic Connection Factory**, as shown in Figure 4-25 on page 123.

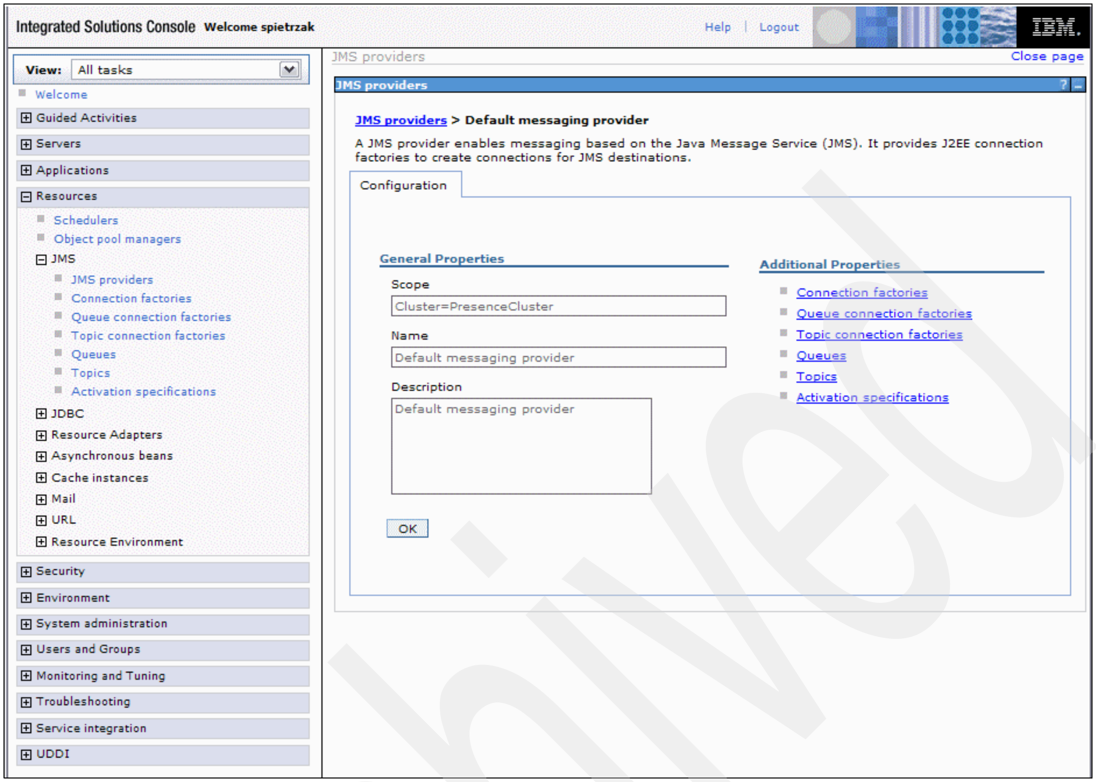


Figure 4-25 JMS Topic Connection Factory - Step 2

3. Click **New**.
4. On the General Properties page (Figure 4-26 on page 124), enter the following parameters:
   - Name: **PresenceTCF**
   - JNDI Name: **jms/presenceTCF**
   - Bus name: **PresenceBUS**
   - Connection proximity: **Server**

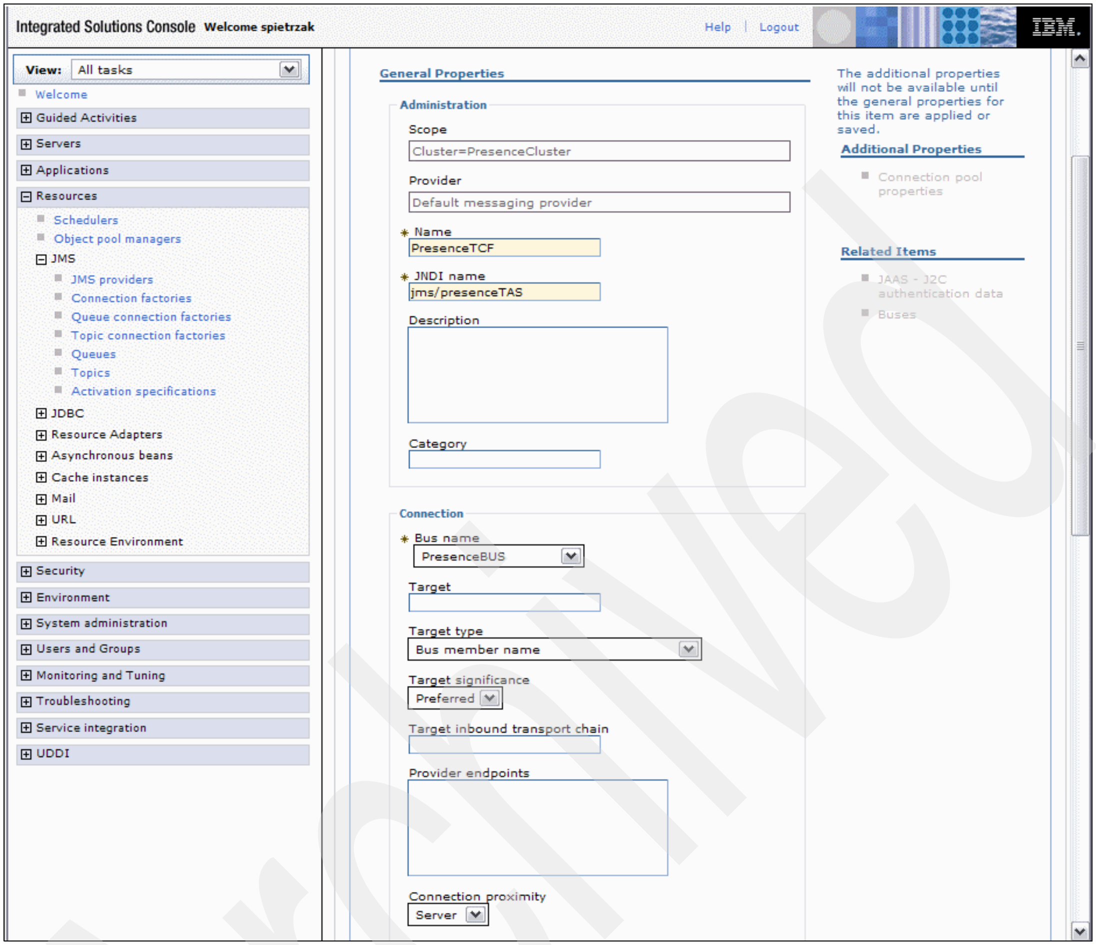


Figure 4-26 JMS Topic Connection Factory - Step 3

5. Click **OK**.
6. Save changes to the master repository.

4.3.3 Creating the JMS Topic

1. From the navigation pane, click **Resources** → **JMS** → **JMS Provider**, and complete the following actions, as shown in Figure 4-27 on page 125:
   a. From the pull-down menu, select **Cluster=PresenceCluster**.
   b. From the JMS providers list, click **Default messaging provider**.

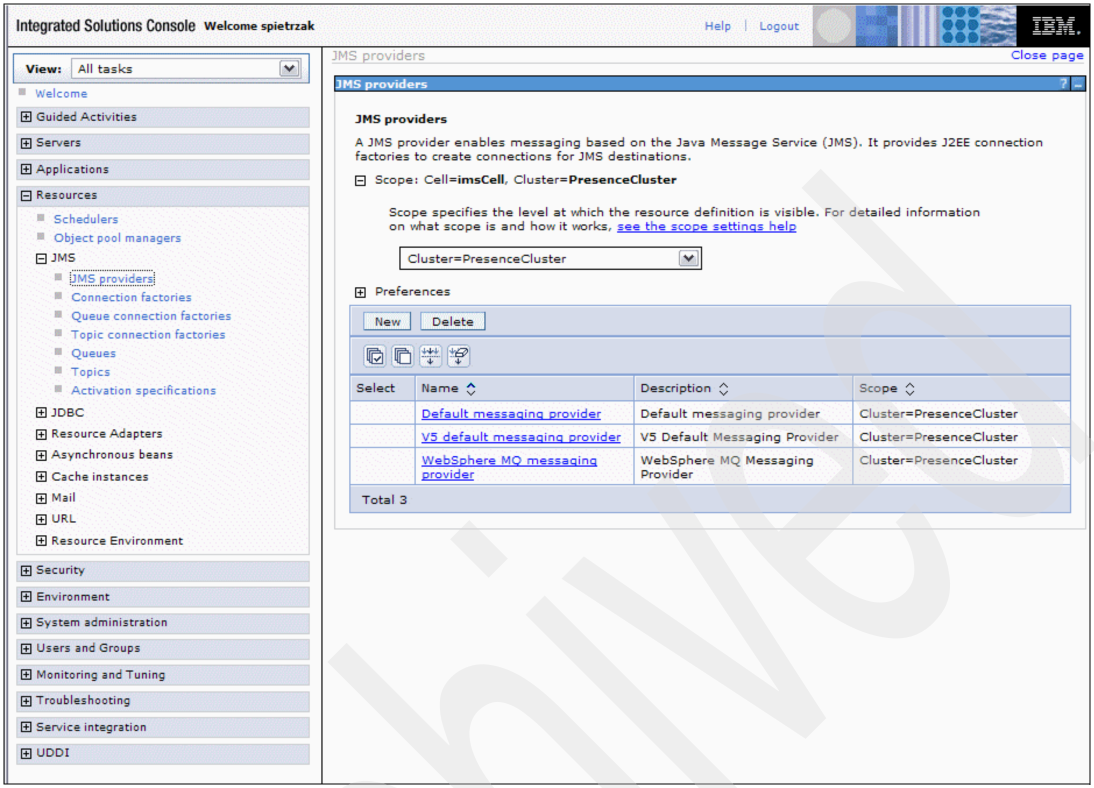


Figure 4-27 JMS Topic - Step 1

2. Under Additional Properties, click **Topics**, as shown in Figure 4-28 on page 126.

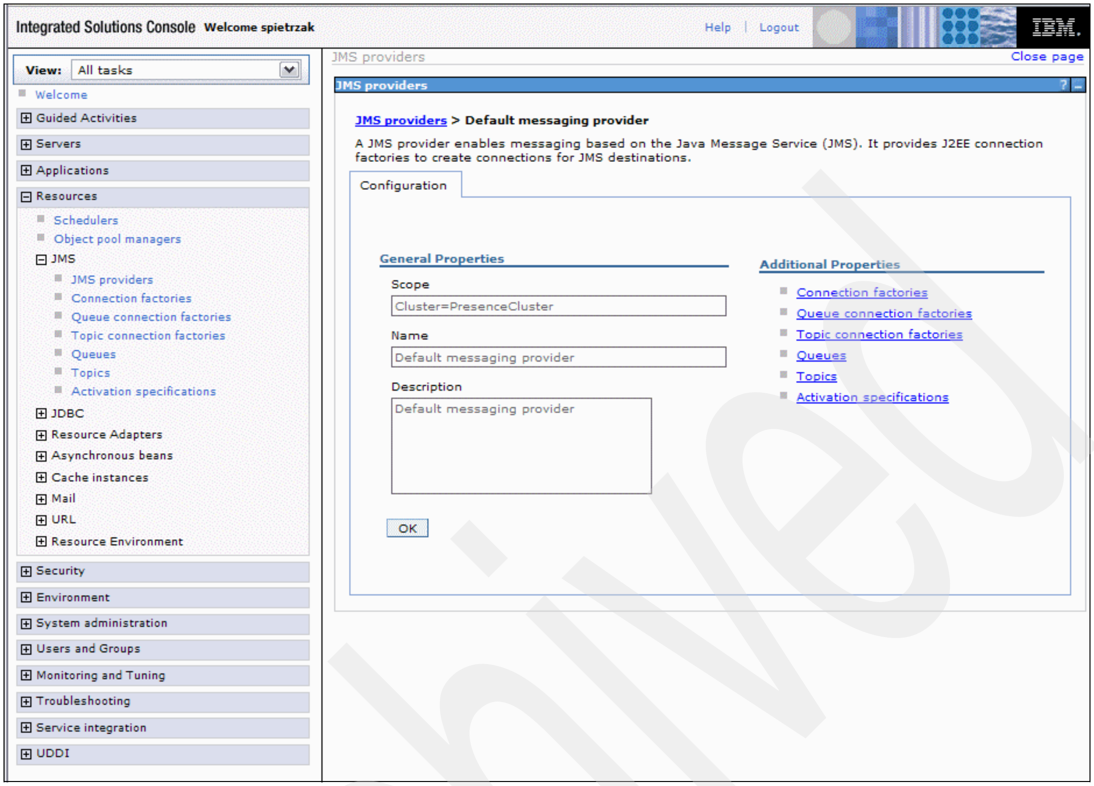


Figure 4-28 JMS Topic -Step 2

3. Click **New**.
4. On the General Properties page (Figure 4-29 on page 127) enter the following parameters:
   - Name: **PresenceT**
   - JNDI Name: **jms/presencePublishT**
   - Bus name: **PresenceBUS**
   - Topic space: **Default.Topic.Space**

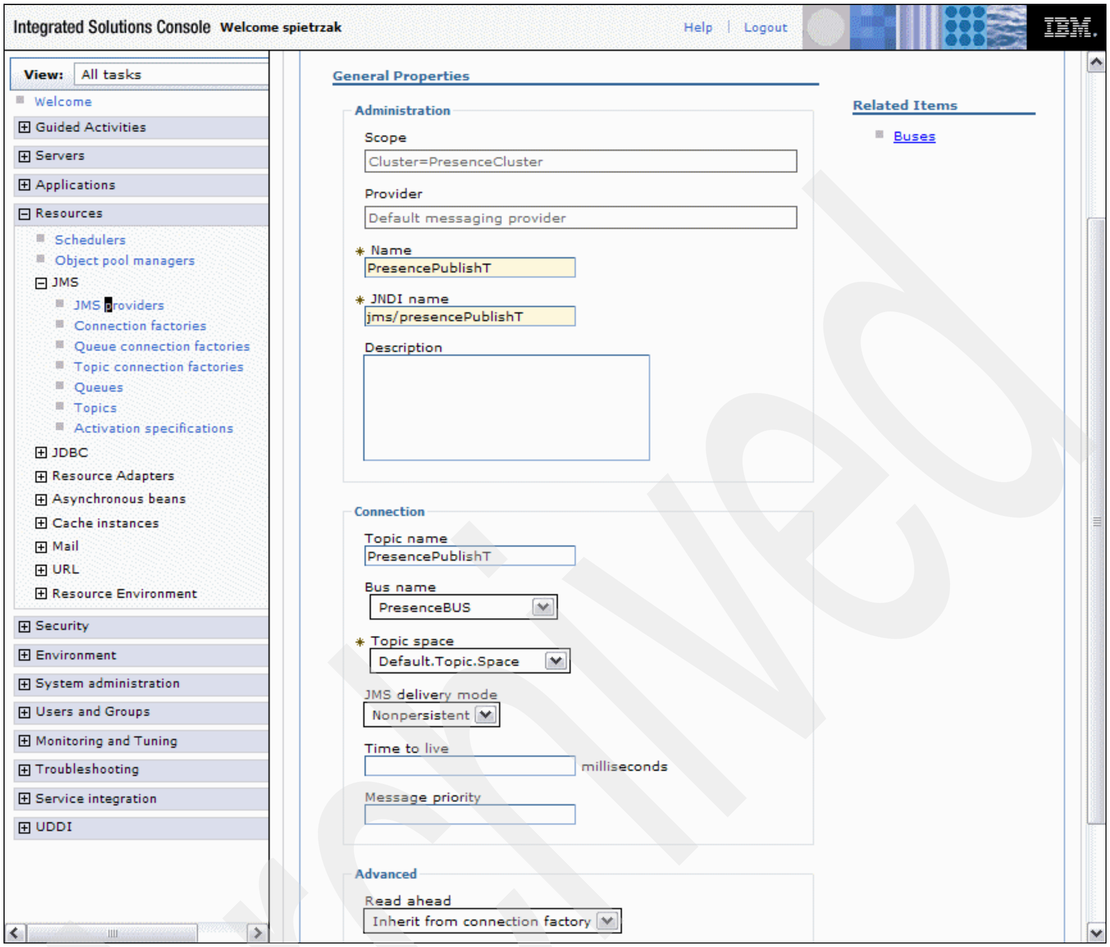


Figure 4-29 JMS Topic - Step 3

5. Click **OK**.
6. Save changes to the master repository.

4.3.4 Creating the JMS Activation specifications

1. From the navigation pane, click **Resources** → **JMS** → **JMS Provider**, and complete the following tasks, as shown in Figure 4-30 on page 128:
   a. From the pull-down menu, select **Cluster=PresenceCluster**.
   b. From the JMS providers list, click **Default messaging provider**.

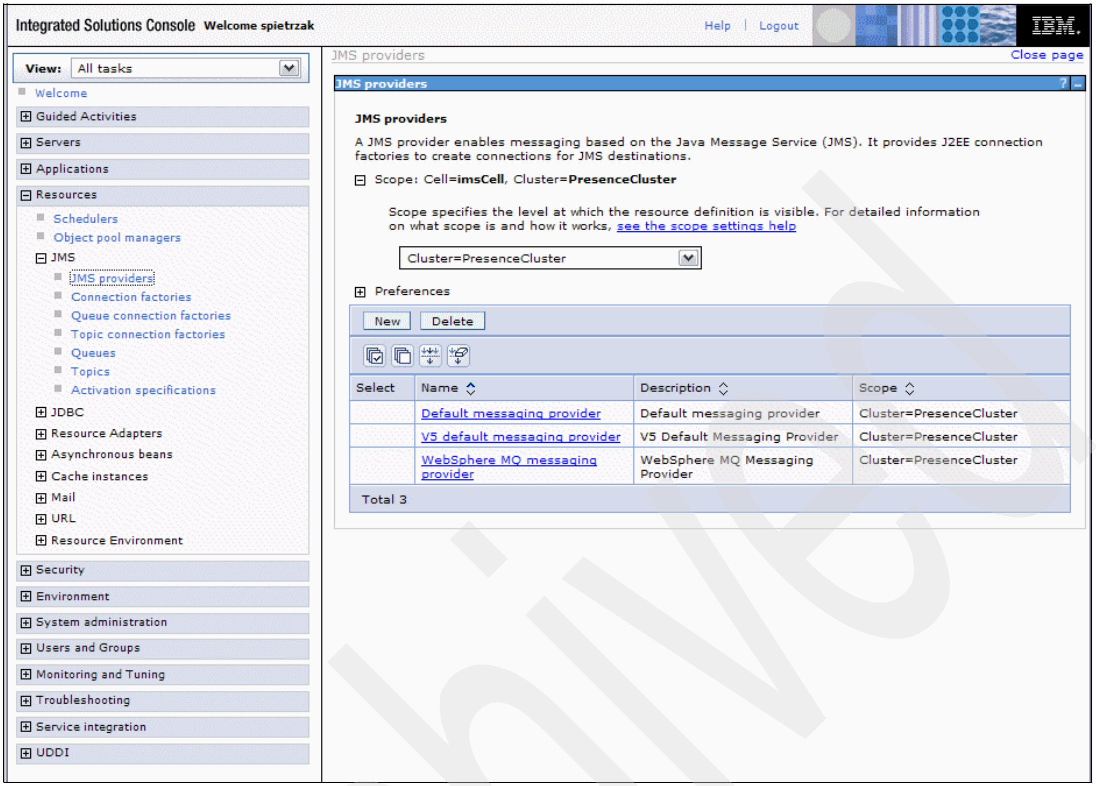


Figure 4-30 JMS Activation specification - Step 1

2. Under Additional Properties, click **Activation Specifications**, as shown in Figure 4-31 on page 129.

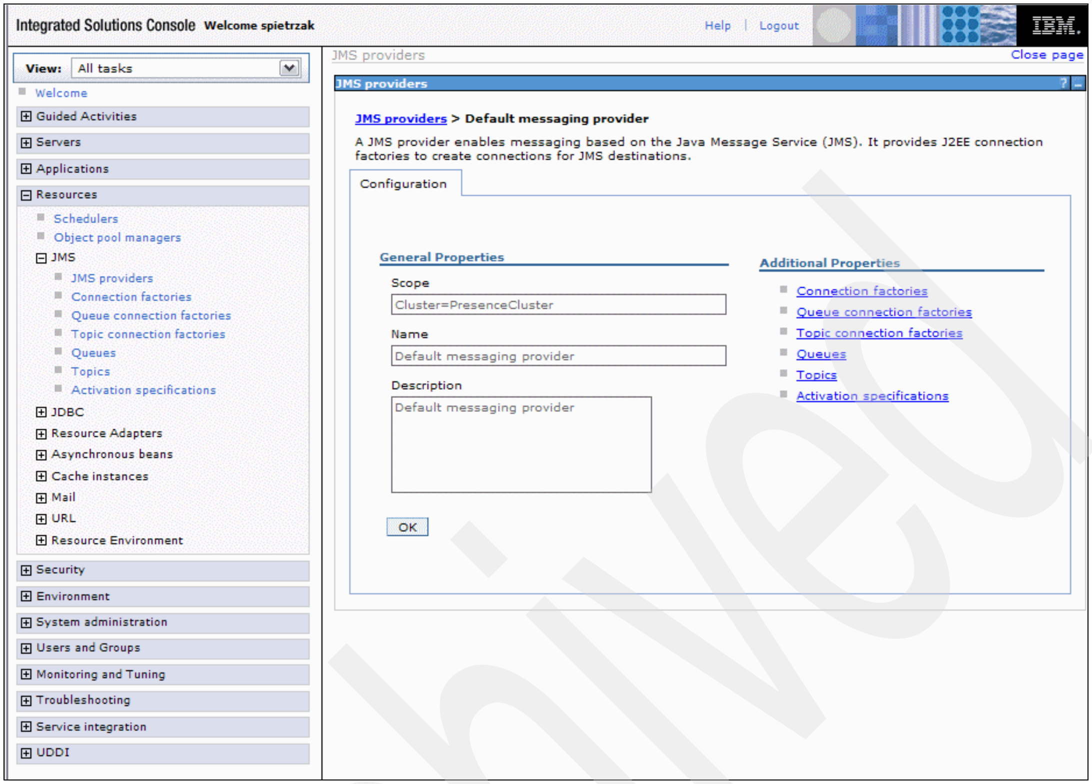


Figure 4-31 JMS Activation specification - Step 2

3. Click **New**.
4. On the General Properties page, enter the following parameters, as shown in Figure 4-32 on page 130:
   - Name: **PresenceActivationSpec**
   - JNDI Name: **jms/tas**
   - Destination type: **topic**
   - Destination JNDI name: **jms/presencePublishT**
   - Bus name: **PresenceBUS**
   - Acknowledge mode: **Auto-acknowledge**
   - Subscription durability: **nondurability**

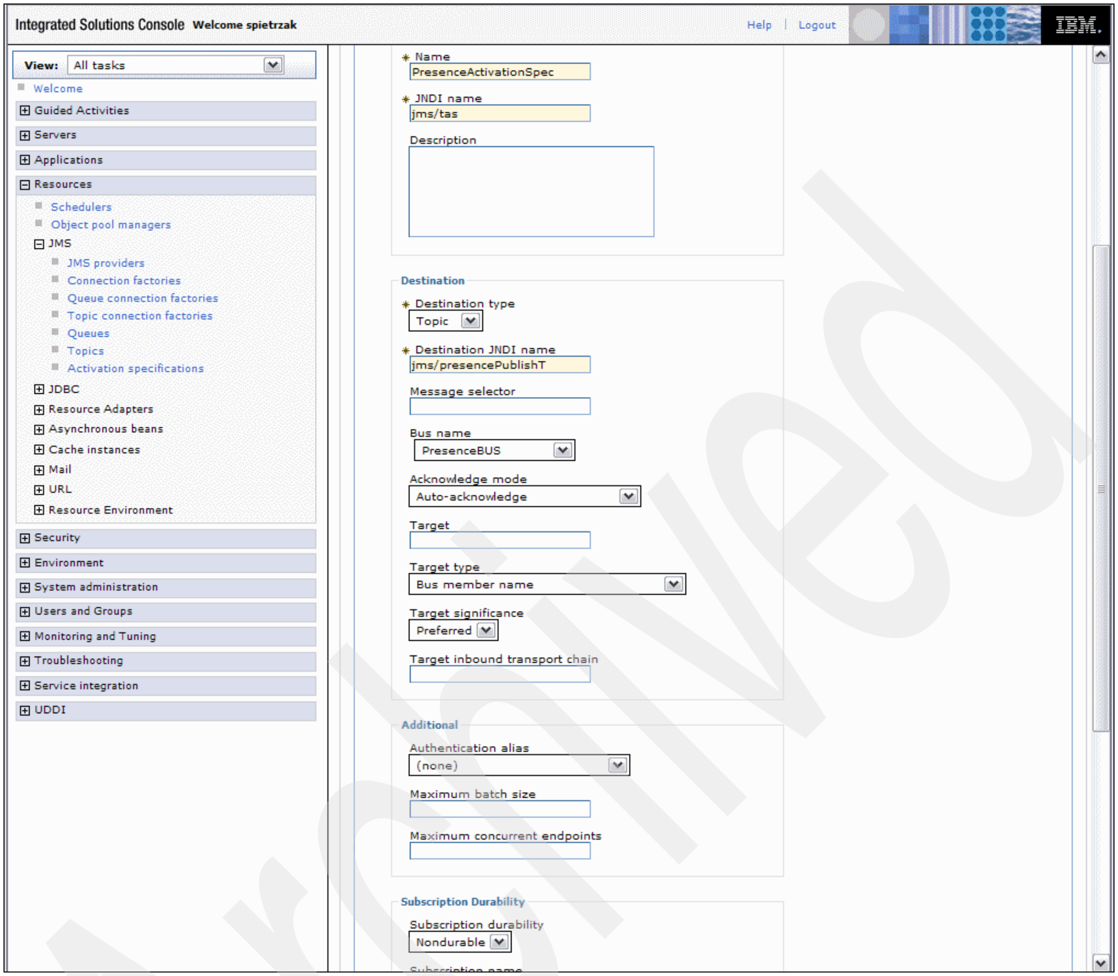


Figure 4-32 JMS Activation specification - Step 3

5. Click **OK**.
6. Save changes to the master repository.

4.4 Environment verification

We are now ready to deploy the Presence Server, but before we can start, we must validate the environment. The tests in the next sections ensure that the environment is ready for the Presence Server installation.

4.4.1 Restarting the WebSphere Application Server

Before you deploy the Presence Server, we recommend that you restart imsCell components. To restart the WebSphere Application Server:

1. Stop WebSphere Application Server infrastructure components:
   a. Login to WebSphere Application Server Console as a user with administration privileges:
   b. From the navigation pane, click **Servers** → **Application Servers**.
   c. Stop all of the running application servers.
   d. From the navigation pane, click **System administration** → **Nodes**.
   e. Stop all of the working nodes except the imsCellMgr01 node.
   f. From the navigation pane, click **System administration** → **Deployment manager**.
   g. Click **Stop** to stop the Deployment Manager.
2. Start the Deployment Manager, nodes, and Presence Servers:
   a. Login to the MngNode as root.
   b. Go to the Deployment Manager profile directory, and run Deployment Manager process, as shown in Example 4-17.

Example 4-17 Run Deployment Manager

```
[root@ldap Dmgr01]# pwd
/opt/IBM/WebSphere/AppServer/profiles/Dmgr01
[root@ldap Dmgr01]# cd bin/
[root@ldap bin]# ./startManager.sh
```

   c. Verify that the Deployment Manager successfully started by checking the file SystemOut.log. It should not contain any errors.
   d. Login to the PresNode01 as root.
   e. Run the Presence Application Server PresSrv01. Before you start the node synchronize it using the commands in Example 4-18.

Example 4-18 Start PresSrv01 Application Server

```
[root@pres1 bin]# pwd
/opt/ibm/WebSphere/AppServer/profiles/PresNode01/bin
[root@pres1 bin]# ./syncNode.sh ldap -username wasadmin -password <password>
...
[root@pres1 bin]# ./startNode.sh
...
```

f. Verify that the PresSrv01 successfully started by checking the SystemOut.log file. It should not contain any errors.

> **Note:** In the previous examples, name LDAP represents the short host name of MngNode. If the short name for MngNode is not defined, use a Fully Qualified Name, for example, “ldap.itso.ral.ibm.com”.

4.4.2 Testing data source connections

To test data source connections:

1. Login to the WebSphere Application Server Console.
2. From the navigation pane, click **Resources** → **JDBC** → **Data sources**.
3. Select the following data sources from the list, as shown in Figure 4-33, and then click **Test connection**, as shown in Figure 4-34 on page 133:
   – jdbc/db
   – jdbc/ur

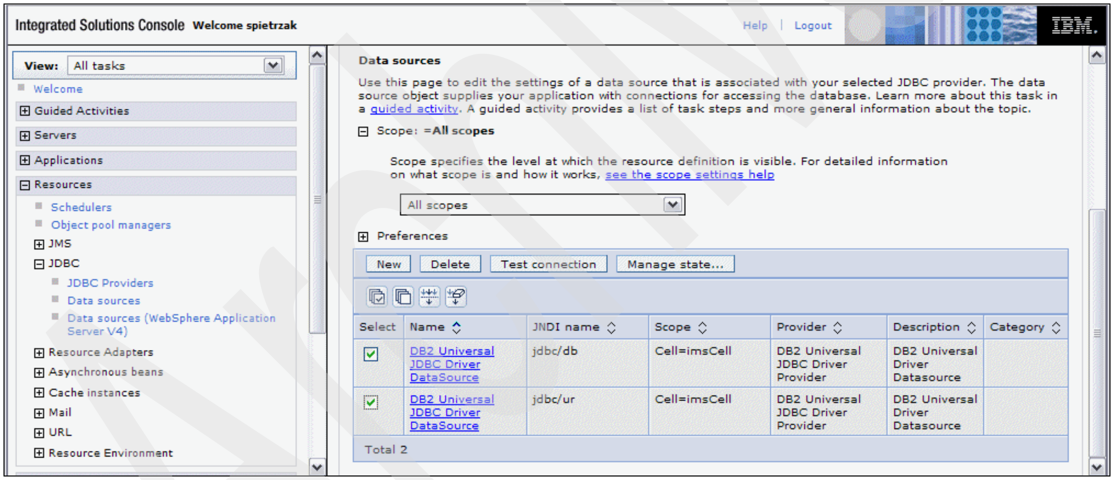


Figure 4-33 Verification - data sources- Step1

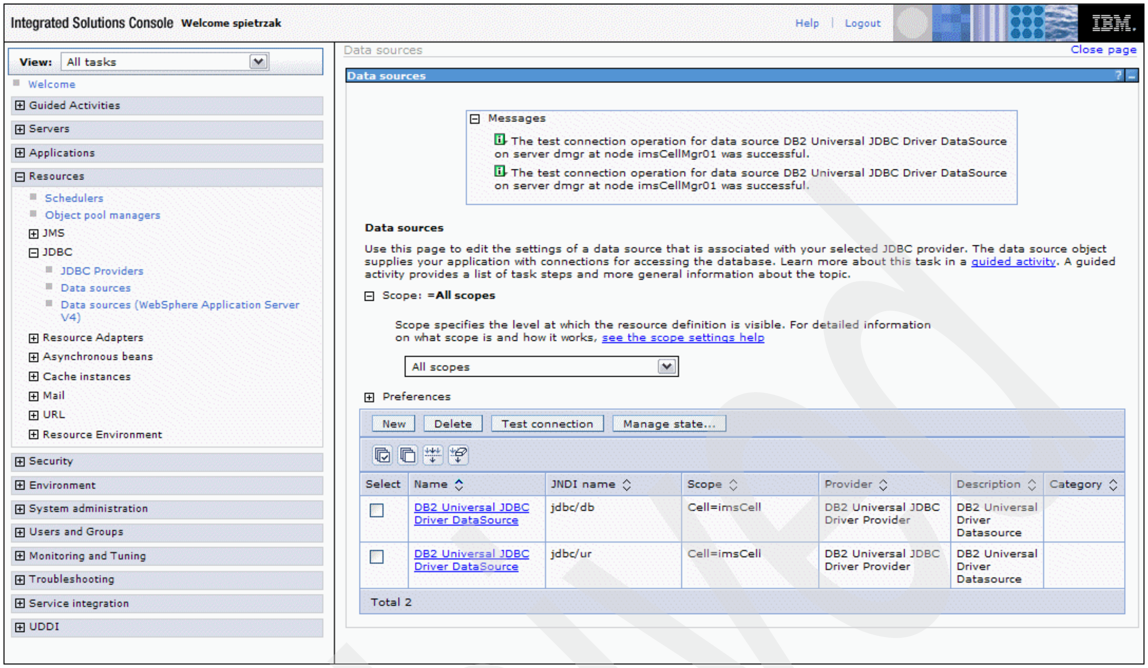


Figure 4-34 Verification - data sources - Step 2

Test of all data sources should be successfully completed.

4.4.3 Verifying the Service Integration Bus

To verify that the Service Integration Bus started after the infrastructure restart:

1. Logon to the WebSphere Application Server console.
2. From the navigation pane, select **Service integration** → **Busses**.
3. Click **PresenceBUS** to see the details, as shown in Figure 4-35 on page 134.

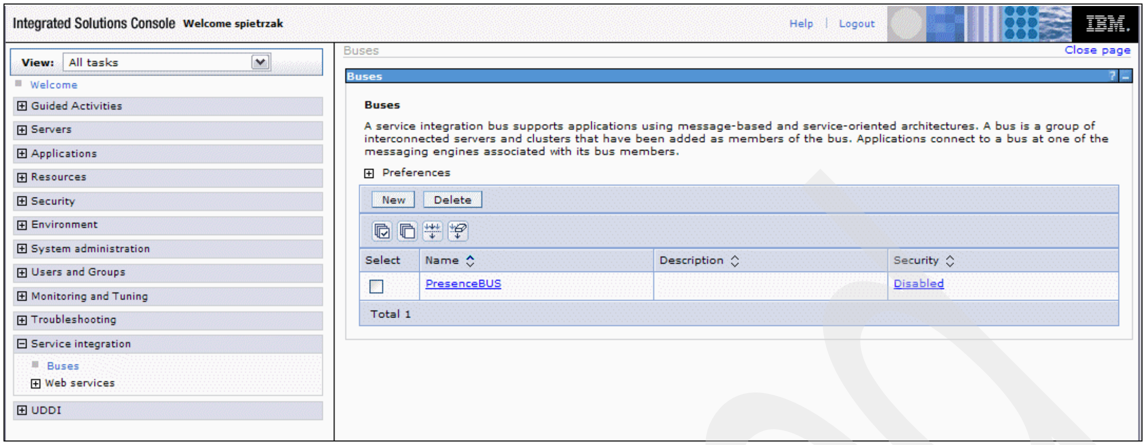


Figure 4-35 Verification - buses - Step 1

4. Under Topology, click **Messaging engines**, as shown in Figure 4-36 on page 135.

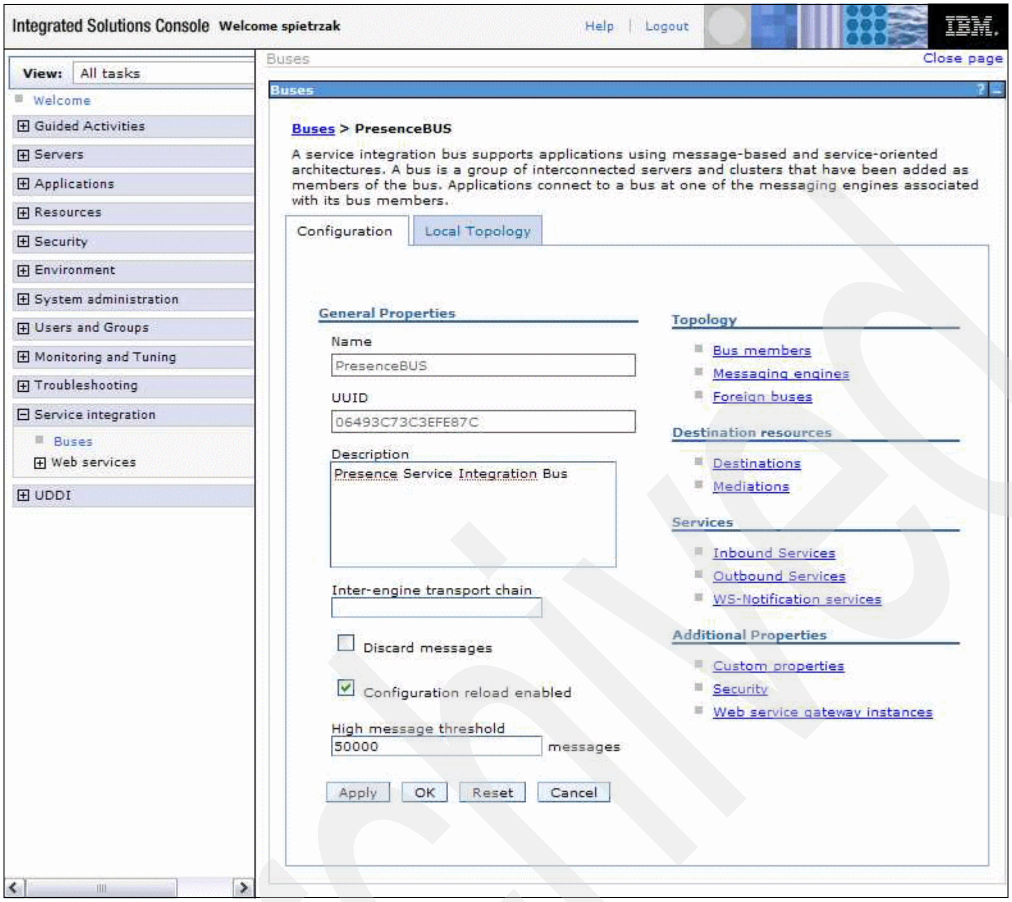


Figure 4-36 Verification - buses - Step 2

5. Verify that the messaging engine is running with the status at started, as shown in Figure 4-37 on page 136.

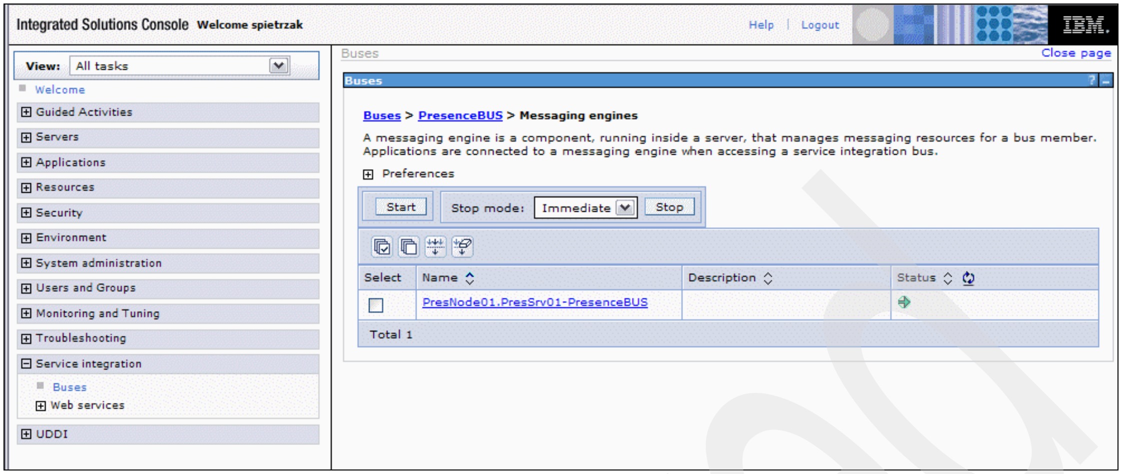


Figure 4-37 Environment verification - PresenceBus engine

4.4.4 Verifying logs files

Verify that the log files in Table 4-3 do not contain any errors.

Table 4-3 Logs files important for Presence Server

Node name	Directory	File name
MngNode	/opt/IBM/WebSphere/AppServer/profiles/Dmgr01/logs/dmgr	SystemOut.log
MngNode	/opt/IBM/WebSphere/AppServer/profiles/Dmgr01/logs/dmgr	SystemErr.log
PresNode01	/opt/ibm/WebSphere/AppServer/profiles/PresNode01/logs/PresSrv01	SystemOut.log
PresNode01	/opt/ibm/WebSphere/AppServer/profiles/PresNode01/logs/PresSrv01	SystemErr.log

4.5 Presence Server deployment

Deploying Presence Server is a common process of deploying Enterprise Application in WebSphere Application Server. In this section, we describe the deployment process.

4.5.1 Copying Presence Server's binary files

Perform following steps on the Deployment Manager Node (MngNode) and on each node in the cluster PresenceCluster, in this case, the PresNode01 node.

To copy Presence Server's binary files:

1. Log in as root for machine LDAP (MngNode).
2. Unpack the Presence Server installation files, as shown in Example 4-19.

Example 4-19 Unpack Presence Server binaries

```
[root@ldap AppServer]# pwd
/opt/ibm/WebSphere/AppServer
[root@ldap AppServer]# tar -xf /mnt/IMS_62/IBMWebSpherePresenceServer.tar
```

3. Verify that the files in Example 4-20 are available.

Example 4-20 Presence Server binary file's verification

```
[root@ldap AppServer]# ls -l ./lib/ext/
total 56
-rwxrwxrwt  1 bin daemon 18701 Sep 16 11:46 common-statscache.jar
-rwxrwxrwt  1 bin daemon 28208 Sep 16 11:49 presence.server.pmi.jar

[root@ldap AppServer]# ls -l ./properties/version/Presence.*
-rwxrwxrwt  1 bin daemon 272 Sep 16 11:49 Presence.product
-rwxrwxrwt  1 bin daemon 223 Sep 16 11:49 Presence.server.component

[root@ldap AppServer]# ls -l
./properties/version/nif/backup/presence.server/files.list
-rwxrwxrwt  1 bin daemon 109 Sep 16 11:49
./properties/version/nif/backup/presence.server/files.list

[root@ldap AppServer]# ls -l ./installableApps/presence/
total 1256
drwxrwxrwx  3 bin daemon    4096 Sep 16 11:49 client
drwxrwxrwx  2 bin daemon    4096 Sep 16 11:49 javadoc
drwxrwxrwx  4 bin daemon    4096 Sep 16 11:46 scripts
-rwxrwxrwt  1 bin daemon 1260010 Sep 16 11:49 WebSpherePresenceServerEAR.ear

[root@ldap AppServer]# ls -l ./installableApps/presence/scripts/config/
total 32
-rwxrwxrwt  1 bin daemon 1757 Sep 16 11:46 ConfigurationParams.txt
-rwxrwxrwt  1 bin daemon 4152 Sep 16 11:46 SystemConfiguration.xml
-rwxrwxrwt  1 bin daemon 8281 Sep 16 11:46 UpdateConfiguration.jar
```

4. Perform steps 1 through 3 on the PresNode01 machines.

4.5.2 Deploying the Presence Server enterprise application

To deploy the Presence Server enterprise application:

1. Logon to the WebSphere Application Server Console as a user with administration privileges.
2. From the navigation panel, click **Applications** → **Install New Application**.
3. Choose **Remote file system**, and click **Browse** to locate the WebSpherePresenceApplicationEAR.ear archive.
4. Go through the following directories: **imsCellMgr01** → **/** → **opt** → **IBM** → **WebSphere AppServer** → **installableApps** → **presence**.
5. Select **WebSpherePresenceServerEAR.ear**, and click **OK**, as show in Figure 4-38.

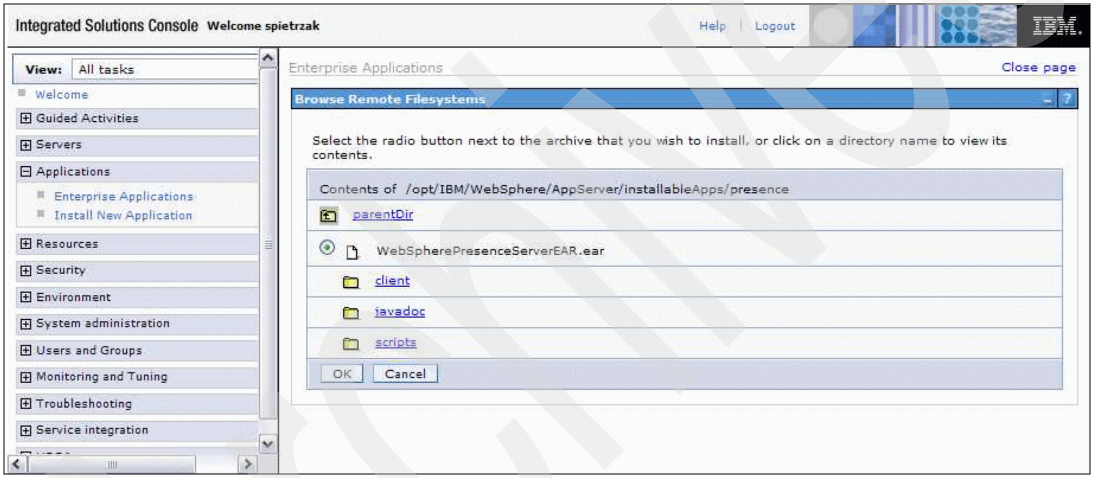


Figure 4-38 Presence Server deployment - browse remote file systems

6. Click **Next**, as shown in Figure 4-39 on page 139.

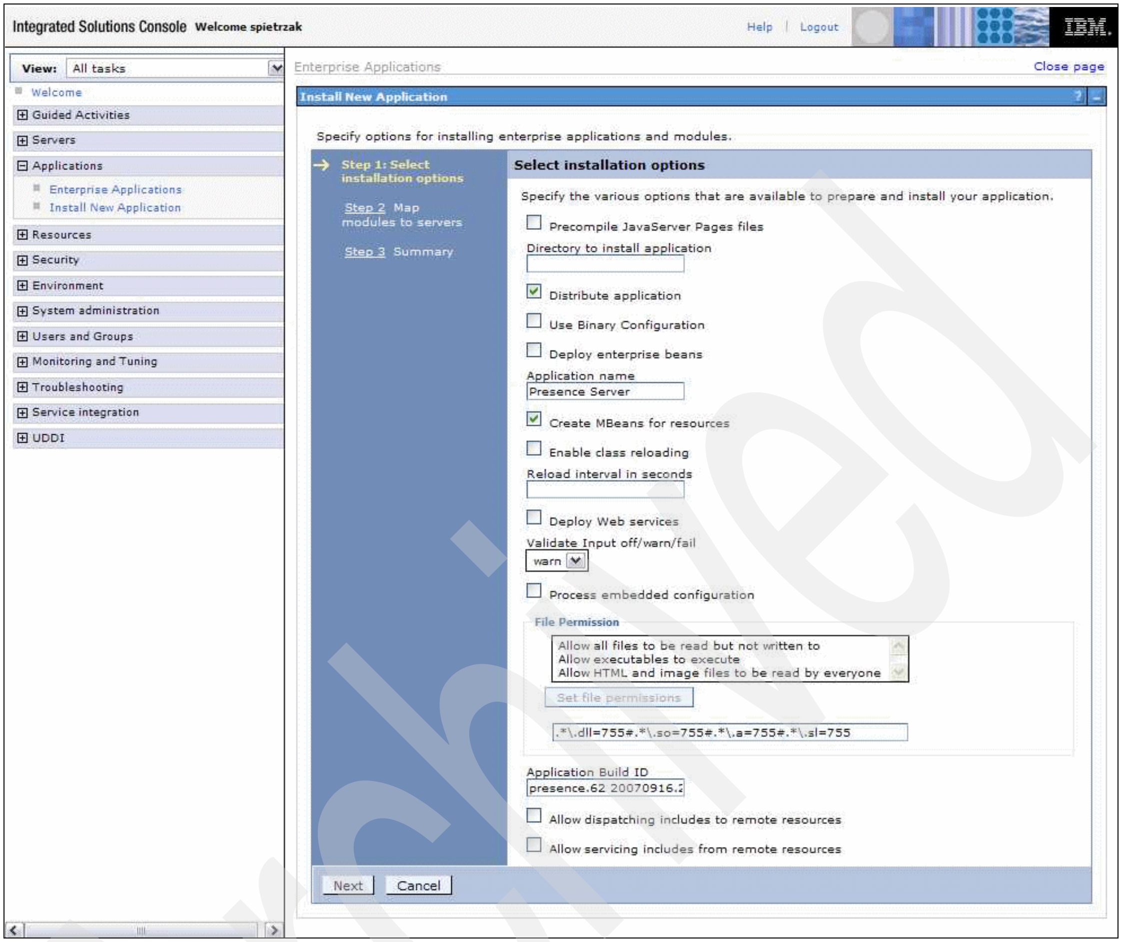


Figure 4-39 Presence Server deployment - Step 1

7. Map Presence Server's modules to the PresenceCluster, as shown in Figure 4-40 on page 140 and Figure 4-41 on page 140:
   a. From the Cluster and Servers, select WebSphere:cell=imsCell,cluster=PresenceCluster.
   b. Select all of the modules from the list.
8. Click **Apply**.

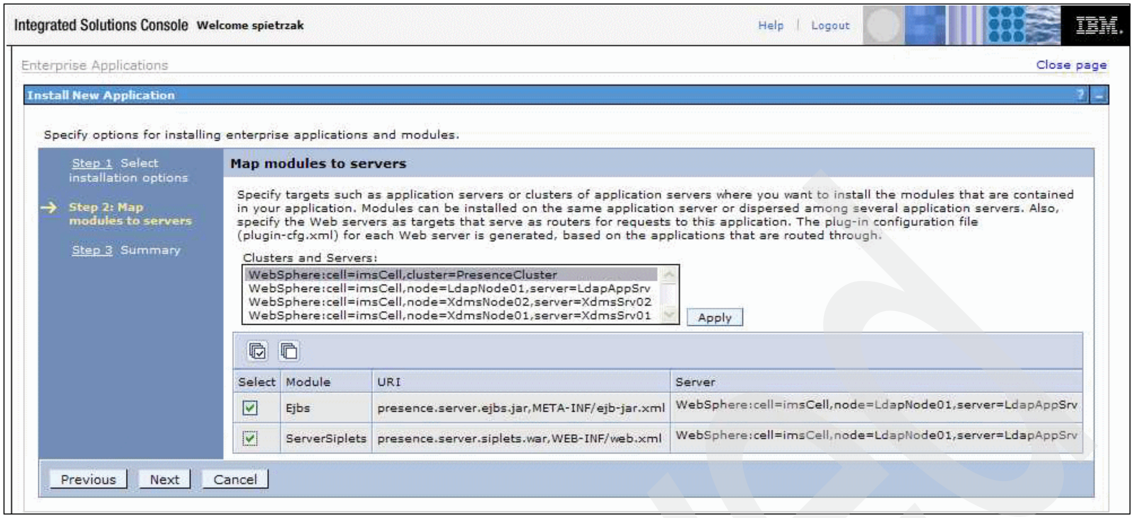


Figure 4-40 Presence Server deployment - Step 2a

9. Click **Next**.

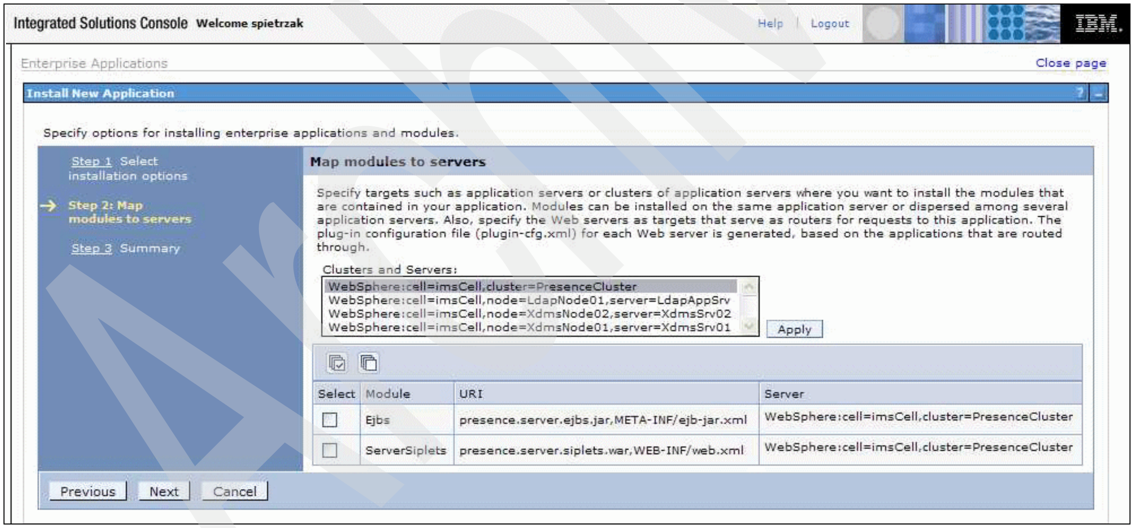


Figure 4-41 Presence Server deployment - Step 2b

10. Verify the information on the Summary page, as shown in Figure 4-42 on page 141, and click **Finish**.

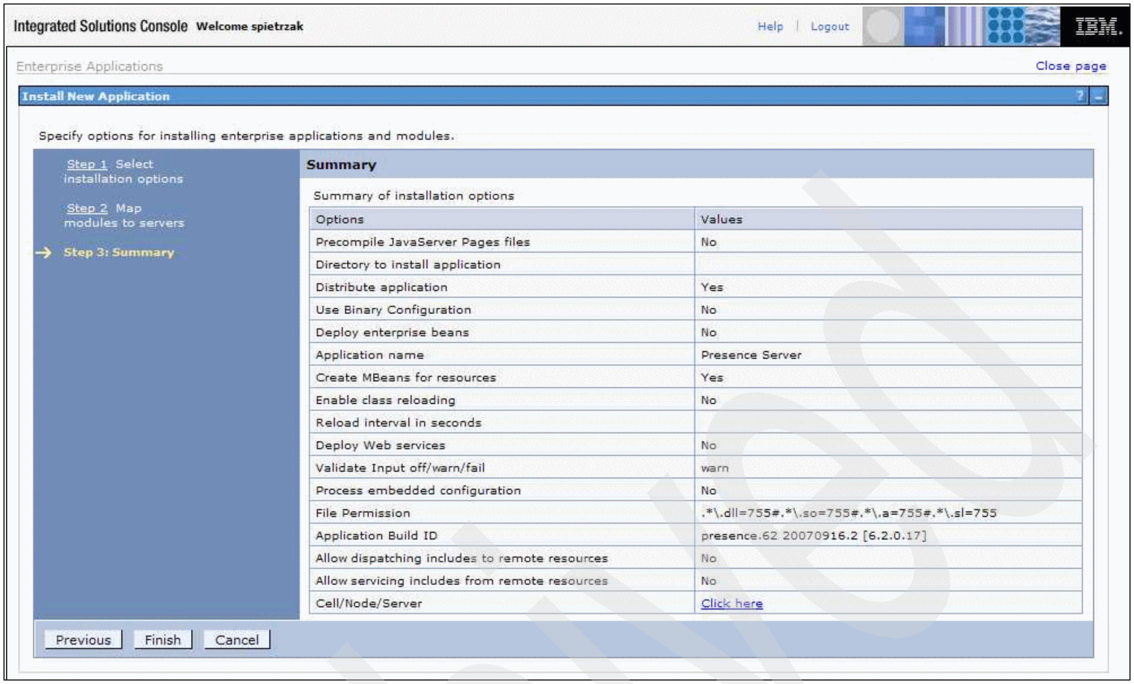


Figure 4-42 Presence Server deployment - Step 3

11. Save the changes to the master repository, which is shown in Figure 4-43 on page 142.

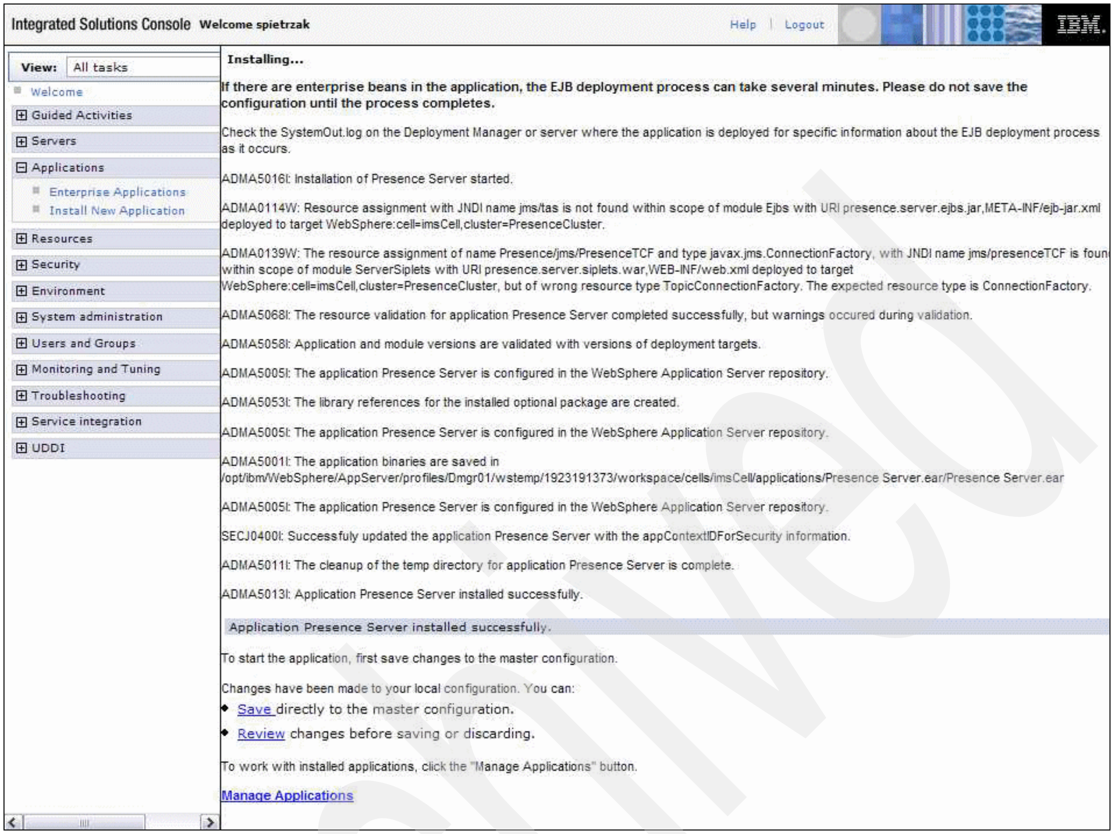


Figure 4-43 Presence Server deployment - Step 4

4.5.3 Restarting the Presence Server Cluster

After the application is installed, you need to restart Presence Cluster. To restart the Presence Cluster:

1. From the navigation pane, click **Servers** → **Clusters**.
2. From the list, select Presence Cluster, and click **Stop**.
3. After the Cluster stops, select it once again.
4. Click **Start**.
5. Login to PresNode01 server and go to the directory with server's logs. Example 4-21 on page 143 shows the Presence Server's logs on server pres1.

Example 4-21   Presence Server's logs on server pres1

```
[root@pres1 PresSrv01]# pwd
/opt/IBM/WebSphere/AppServer/profiles/PresNode01/logs/PresSrv01
```

6. In the SystemOut.log file, check for errors, as shown in Example 4-22.

Example 4-22   Errors from the Presence Server during the first start

```
[9/25/07 14:55:28:328 EDT] 00000055 PresenceTimer I
#--------------------- Presence Server Report ---------------------


# 1. Build Version
#Presence Server Build Number is : 'presence.62 20070916.2 [6.2.0.17]'
#
#--------------------- ERROR REPORT ---------------------
# System is NOT initialized rejecting all requests, reason: 'Initable
com.ibm.presence.configuration.jmx.PresenceJMXManager failed init process'
#--------------------- ERROR REPORT ---------------------
[9/25/07 14:55:28:331 EDT] 00000055 PresenceTimer I   CWSPS0202I: Presence Server
Build Number : presence.62 20070916.2 [6.2.0.17]
```

4.6 Presence Server configuration

After installation, the Presence Server is ready to be used as a standalone application. Additional configuration is required to integrate the Presence Server with the XDMS Shared List and XDMS Presence Rules servers. We describe the additional configuration in Chapter 7, "Presence Server and XDMS integration configuration" on page 267.

4.6.1 Presence Server base configuration

To configure the base Presence Server installation:

1. Login as a root to the LDAP server (MngNode).
2. Go to the following directory:
   /opt/IBM/WebSphere/AppServer/installableApps/presence/scripts/config
3. Open the ConfigurationParams.txt file for editing.
4. Set up the parameters in Example 4-23 on page 144.

Example 4-23 Connection parameters used to configure Presence Server

```
cfg.system=/opt/IBM/WebSphere/AppServer/installableApps/presence/scripts/config/Syste
mConfiguration.xml
username=db2inst1
password=<db_passwprd>
dbConnectionString=jdbc:db2://db2.itso.ral.ibm.com:50000/PSDB
dbDriver=com.ibm.db2.jcc.DB2Driver
```

5. Save and close the file.
6. Open the SystemConfiguration.xml file for editing.
7. Set up the parameters in Example 4-24.

Example 4-24 Base configuration of Presence Server

```
<jmxListener user="wasadmin" password="<password>"/>
```

> **Note:** See the configuration file that we provide the in additional materials section for this Redbooks publication:
> /scripts/PresenceServerConfiguraiton/SystemConfiguration_base.xml

8. Save and close the file.
9. Run the command in Example 4-25.

Example 4-25 Presence Server configuration

```
[root@ldap config]# ./configure.sh SystemConfiguration_base.xml
File loader created.
Property file loaded.
Property file parsed.
There are 1 elements to be configured.
XML file name:
/opt/IBM/WebSphere/AppServer/installableApps/presence/scripts/config/SystemConfigurat
ion.xml
XML source loaded from
/opt/IBM/WebSphere/AppServer/installableApps/presence/scripts/config/SystemConfigurat
ion.xml
DB manager created
Connecting with user: db2inst1
Connecting to: jdbc:db2://db2.itso.ral.ibm.com:50000/PSDB
DB connection created.
DB ResultSet created
Storing com.ibm.presence.configuration.SystemConfiguration in progress..
DB table is not empty. Updating the entry with key =
com.ibm.presence.configuration.SystemConfiguration .
```

```
SQL executed.
DB connection closed
XML stored in DB.
Done.
-------------------------------------------------------------------------------
```

4.6.2 Starting the Presence Server application

After you change the Presence Server configuration, you must restart the Presence Server application:

1. From the navigation pane, click **Applications** → **Enterprise Applications**.
2. Select **PresenceCluster**, and click **Stop**.
3. Check in the SystemOut.log file for each server in the Presence Cluster that are stopped.
4. Select **PresenceCluster**, and click **Start**.
5. In the SystemOut.log file, verify that the Presence Servers started correctly, as shown in Figure 4-26.

Example 4-26 Presence Server start up log

```
[9/25/07 15:25:30:822 EDT] 0000002f ApplicationMg A   WSVR0221I: Application started: 
Presence Server
[9/25/07 15:25:30:943 EDT] 0000001e SessionContex I 
DRSSessionContext:memberIsActivated - elected primary invalidator for default_hosts
iplets
[9/25/07 15:25:46:838 EDT] 00000033 PresenceTimer I   Initialization completed 
successfully
[9/25/07 15:25:46:844 EDT] 00000033 PresenceTimer I 
#--------------------- Presence Server Report ---------------------
# 1. Build Version
#Presence Server Build Number is : 'presence.62 20070916.2 [6.2.0.17]'
#
# 2. Starting System Initialization
# com.ibm.presence.init.StartupManager took 10526 millisecond
# com.ibm.presence.configuration.jmx.PresenceJMXManager took 0 millisecond
# Default init took 164 millisecond
# com.ibm.presence.pmi.PresencePMIManager took 8 millisecond
# com.ibm.presence.monitor.JMXManager took 8 millisecond
# com.ibm.presence.util.ServiceLocator took 0 millisecond
# com.ibm.presence.management.ServerIdGenerator took 0 millisecond
# com.ibm.presence.configuration.PartialConfiguration took 0 millisecond
# com.ibm.presence.authorization.AuthorizationListsManager took 0 millisecond
```

```
# com.ibm.presence.aggregators.AggregatorManager took 16 millisecond
# com.ibm.presence.externalsource.SipExternalSourceInstaller took 0 millisecond
# com.ibm.presence.winfo.WatcherInfoManager took 0 millisecond
# com.ibm.presence.charging.ChargingManager took 26 millisecond
# com.ibm.presence.rls.RLSManager took 34 millisecond
# com.ibm.presence.authorization.AuthorizationManager took 0 millisecond
# com.ibm.presence.authorization.PresenceRulesManager took 0 millisecond
#
# 3. Presence Server Configuration Check
#Test Name: Configuration - Aggregator Configuration
#                                                Test Result:pass
#Test Name: Configuration - Presence Aggreator configuration
#                                                Test Result:pass
#Test Name: DB 1 - JDBC jndi lookup
#                                                Test Result:pass
#Test Name: DB 2 - Datasource connection
#                                                Test Result:pass
#Test Name: DB 3 - Table structure
#                                                Test Result:pass
#Test Name: DB 4 - Table structure (Managing)
#                                                Test Result:pass
#Test Name: DB 5 - Table structure (Authorization)
#                                                Test Result:pass
#Test Name: JMS 1 - TopicConnectionFactory lookup
#                                                Test Result:pass
#Test Name: JMS 2 - Topic lookup
#                                                Test Result:pass
#Test Name: JMS 3 - Topic Session Creation
#                                                Test Result:pass
#Test Name: ServiceLocator - External Source Listener Bean
#                                                Test Result:pass
#Test Name: ServiceLocator - JMS Publisher
#                                                Test Result:pass
#Test Name: ServiceLocator - Presence Services Bean
#                                                Test Result:pass
#Test Name: ServiceLocator - Presence Storage Bean
#                                                Test Result:pass
#Test Name: ServiceLocator - Register External Source Bean
#                                                Test Result:pass
#Test Name: ServiceLocator - Register Services Bean
#                                                Test Result:pass
[9/25/07 15:25:46:848 EDT] 00000033 PresenceTimer I   CWSPS0202I: Presence Server
Build Number : presence.62 20070916.2 [6.2.0.17]
```

5

XML Data Management Server installation

In this chapter, we describe the steps to install and configure the IBM XML Document Management Servers (XDMS) in the base scenario environment.

This chapter contains:

5.1 Preparing to install XDMS

The minimum required configuration for XDMS involves installing the Shared List and Presence Rules enterprise applications on separate WebSphere Application Server clusters, with each cluster containing one WebSphere Application Server federated node with one Application Server in the node.

> **Note:** We use separate clusters for Shared List and Presence Rules components of XDMS in the base environment because this allows for the environment to be scaled in the Scalability phase.

The installation and configuration of XDMS that we describe in this Chapter assumes that all of the pre-requisite software is installed and configured, as documented in 3.3, “Preparing the base environment” on page 64, before starting the installation of XDMS. The following steps are involved in the installation and configuration of XDMS Server. The rest of the chapter explains each of these steps in greater detail:

1. Configure WebSphere Application Server infrastructure:
   - Create the XDMS Cluster.
   - Create the Converged Proxy Server.
   - Configure the User and Group.
   - Prepare the Trust Association Interceptors.
2. Create the databases:
   - Shared List databases
   - Presence Rules database
   - Service Integration (SI) Bus databases
   - Usage record tables
   - Configuring of the data sources
3. Configure the XDMS SI Bus and JMS messaging:
   - Configure the Shared List Service Integration Bus.
   - Configure the Presence Rules.
4. Deploy the XDMS Server application:
   - Deploy the Shared List application.
   - Deploy the Presence Rules application.

Figure 5-1 on page 149 shows an overview of the XDMS installation.

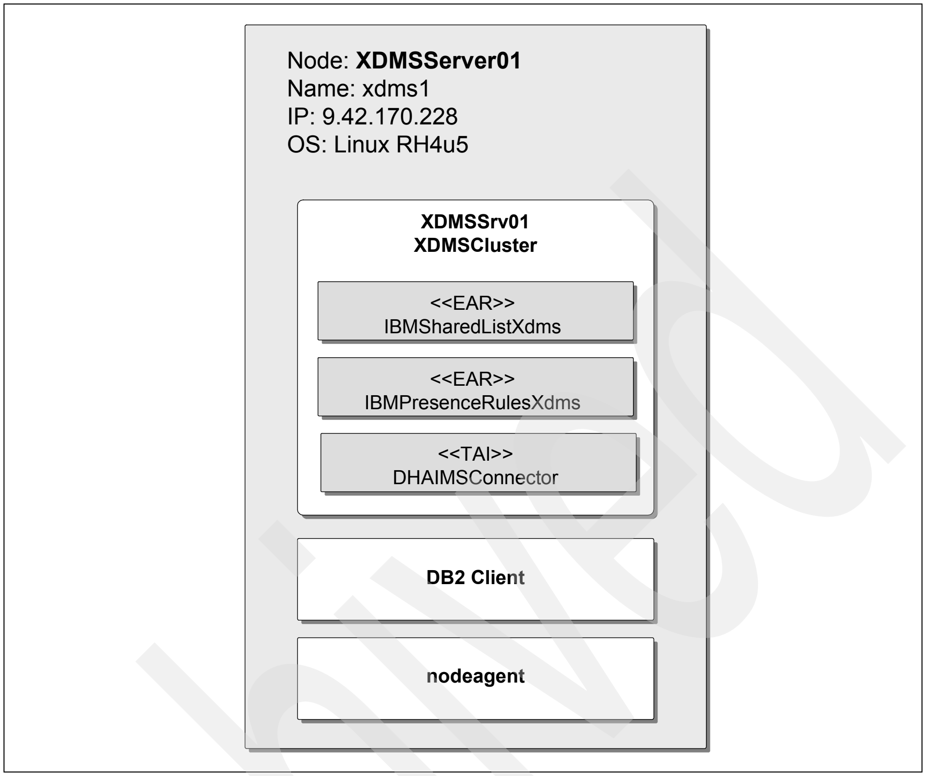


Figure 5-1 Overview of XDMS installation

5.2 Configuring the WebSphere Application Server infrastructure

Preparing the WebSphere Application Server infrastructure for XDMS deployment requires that you create and configure the following WebSphere Application Server artifacts:

- XDMS Cluster creation
- User and Group
- The Trust Association Interceptors

5.2.1 Creating the XDMS Cluster

The XDMS Server provides Shared List and Presence Rules services through two applications:

- Shared List enterprise application
- Presence Rules enterprise application

We recommend that you deploy these two applications in a clustered environment and that you have each application run in a separate cluster. The goal is to scale each application accordingly, independent of the other.

Creating Shared List Clusters

The Shared List Cluster hosts the XDMS Shared List application. For the base scenario environment, the cluster is configured with a single server. In this section, we explain how to create the Shared List Cluster:

1. From the Integration Solutions Console of the Deployment Manager, in the Integrated Solution Console, click the **Servers Clusters** link.
2. To create a new cluster, click the **New** button, as shown in Figure 5-2 on page 151.

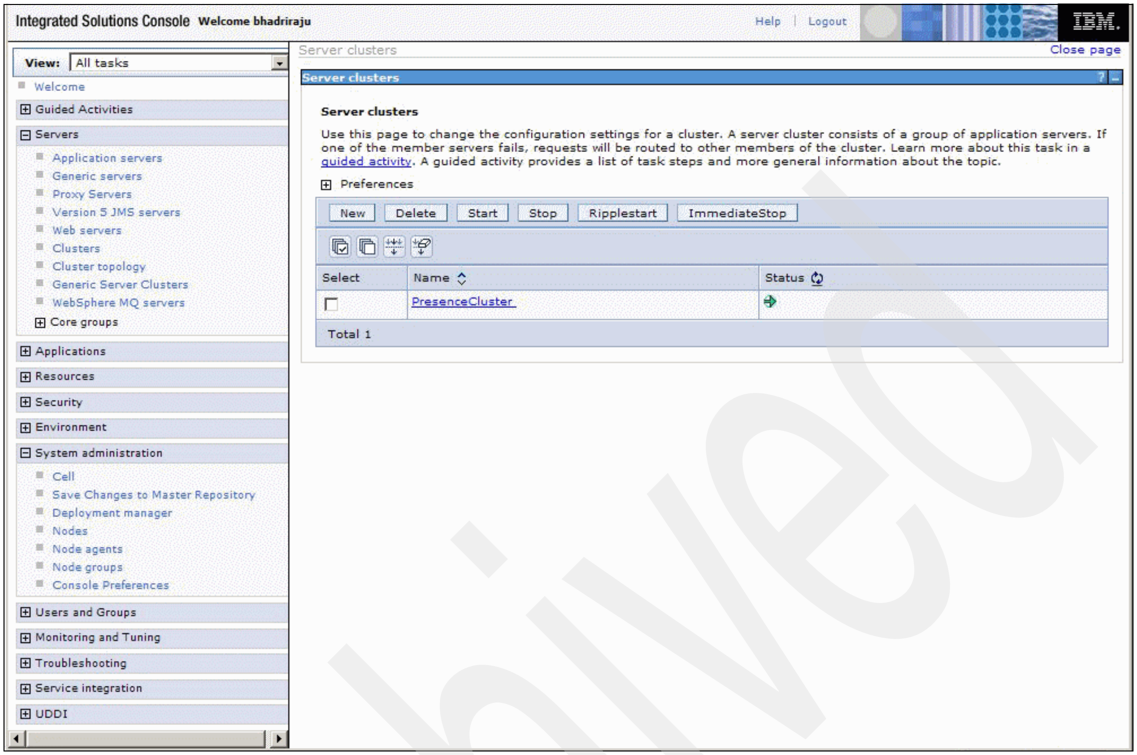


Figure 5-2 Shared List Cluster Creation: Clusters view in the Integrated Solutions Console (ISC)

3. In the Cluster name field, type the name **ListCluster**.
4. Select the **Configure HTTP session memory-to-memory replication** option, as shown in Figure 5-3 on page 152, and click **Next**.

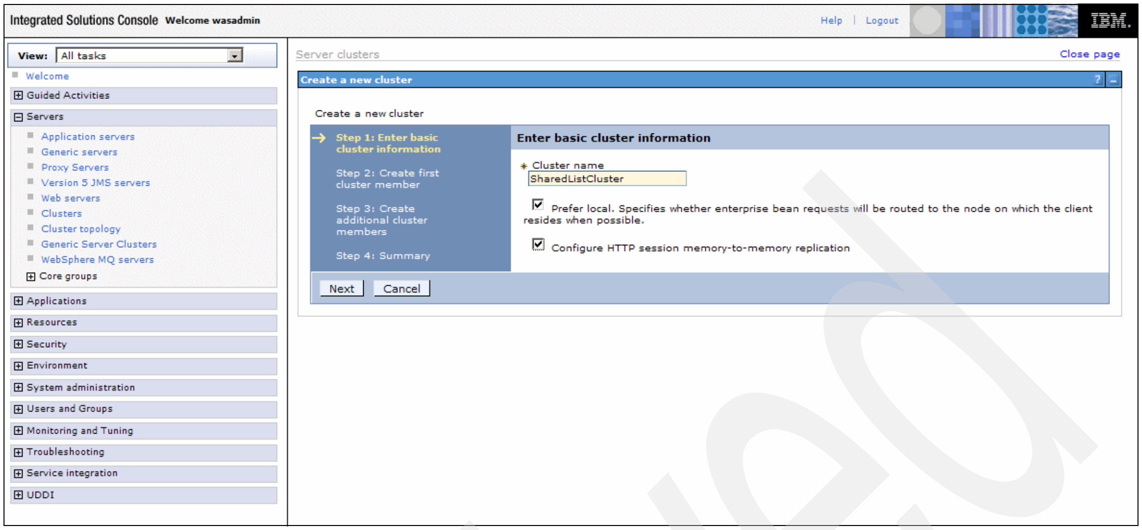


Figure 5-3 SharedListCluster creation: Basic cluster creation information

5. In the Create first cluster member panel, enter the following information, as shown in Figure 5-4 on page 153:
   a. In the Member name field, type **XdmsSrv01**.
   b. In the Select node field, select **XdmsNode01(ND 6.1.0.11)**.

   Click **Next**.

6. In the Select basic for first cluster member panel, enter the following information, as shown in Figure 5-4 on page 153:
   a. Select the **Create the member using an application server template** option.
   b. From the pull-down menu, select **default**.

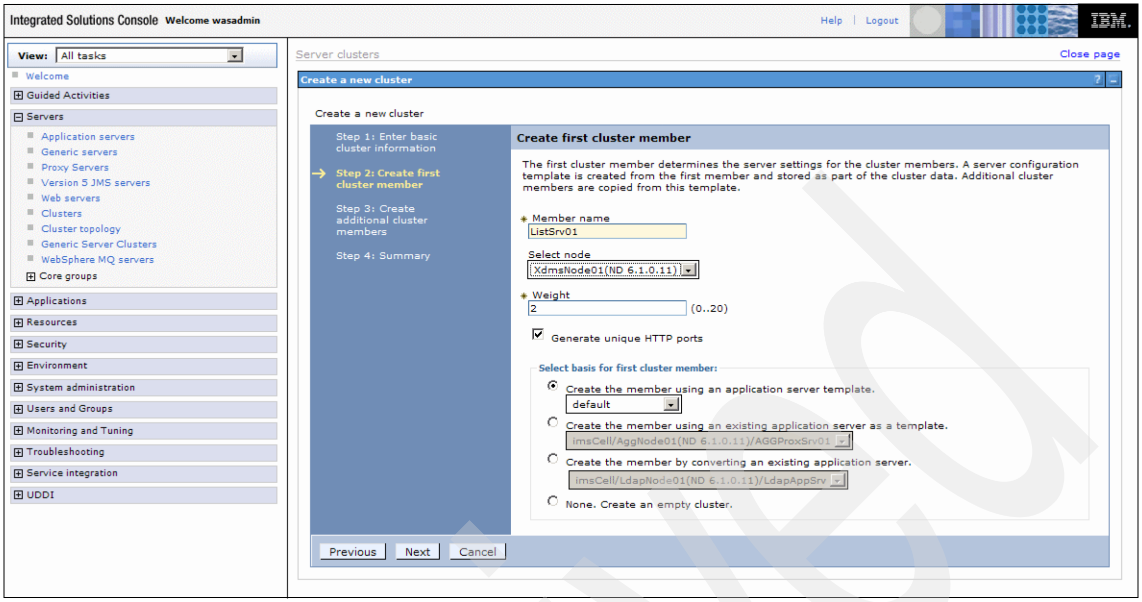


Figure 5-4 SharedListCluster Creation: Create first cluster member panel

7. On the **Create a New Cluster - Summary** panel, click **Finish** to complete the Shared List Cluster creation process.
8. In the Clusters view, click **Save** to store the master configuration, as shown in Figure 5-5 on page 154.

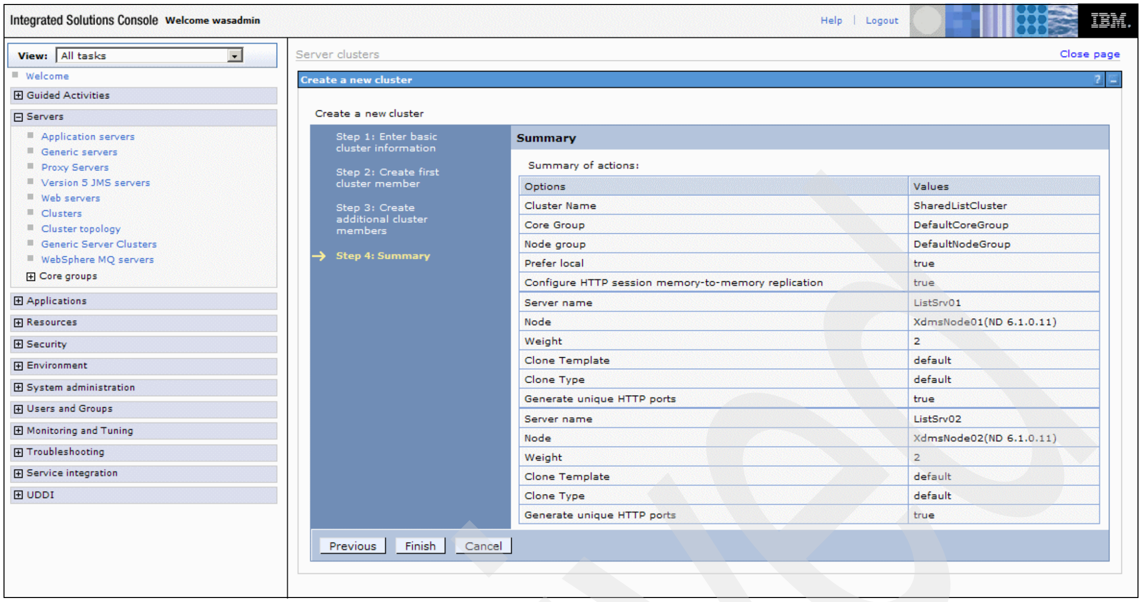


Figure 5-5 SharedListCluster creation: Final cluster creation panel

Creating the Presence Rules Cluster

The Presence Rules Cluster hosts the Presence Rules application of XDMS. For the base environment, we create a single cluster with one server for installing the Presence Rules application. In this section, we explain how to create the Presence Rules Cluster:

1. In the Clusters panel of the Integrated Solutions Console:
   a. Click the **New** button to create a new cluster.
   b. For Cluster name, enter the name **RulesCluster**.
   c. Select the **Configure HTTP session memory-to-memory replication** option, and click **Next**, as shown in Figure 5-6 on page 155.

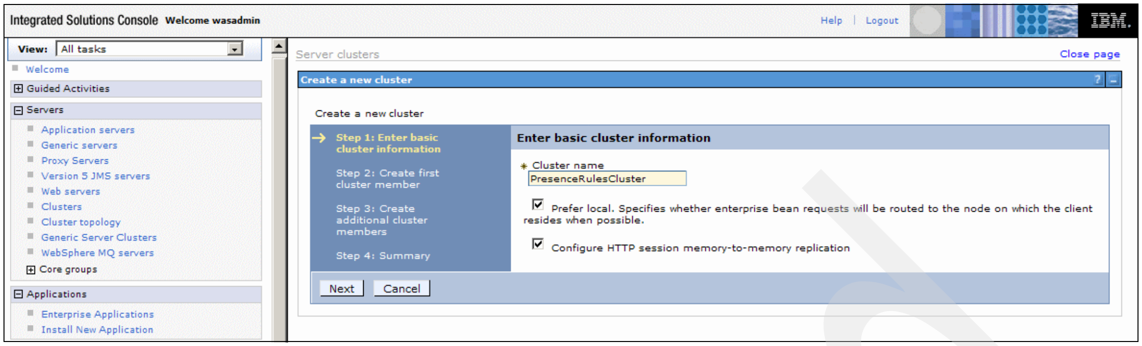


Figure 5-6 Presence Rule Cluster creation: Basic cluster creation information

2. On the Create first cluster member panel, as shown in Figure 5-7:
   a. In the Member name field, type **XdmsSrv01**.
   b. In the Select node field, select **XdmsNode01(ND 6.1.0.11)**.
   c. In the Select basic for first cluster member section, select the options:
      - **Create the member using an application server template**
      - **Default**
3. Click **Next**.

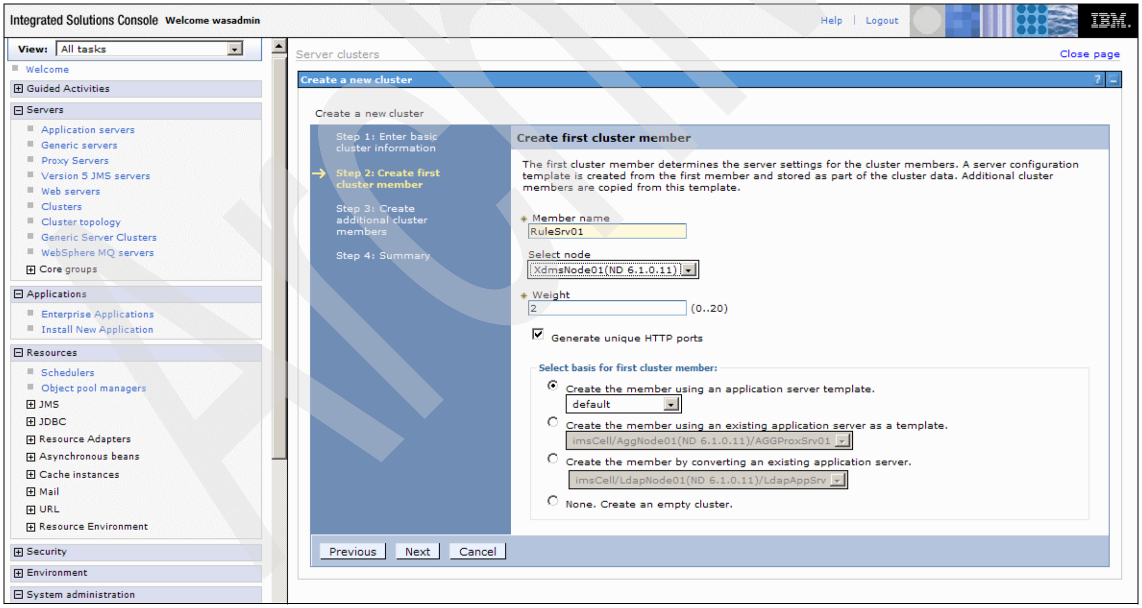


Figure 5-7 PresenceRulesCluster Creation: Create first cluster member panel

4. Click **Finish** to complete the creation of the Presence Rules Cluster, as shown in Figure 5-8.

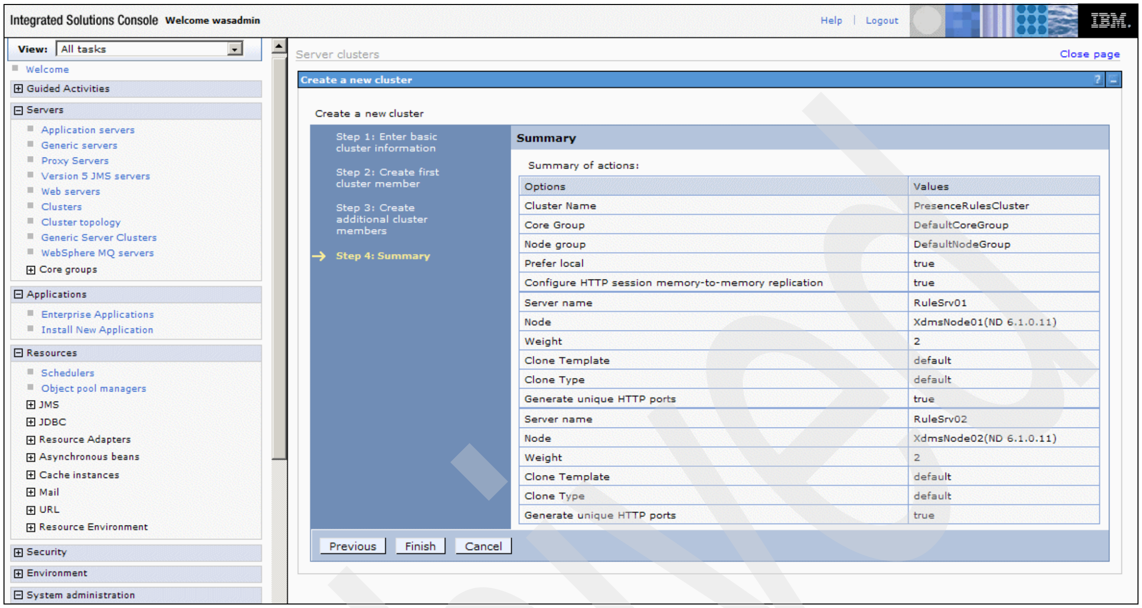


Figure 5-8 PresenceRulesCluster creation: Final cluster creation panel

5. In the **Servers Clusters** panel, click **Save** to save the master configuration, as shown in Figure 5-9 on page 157.

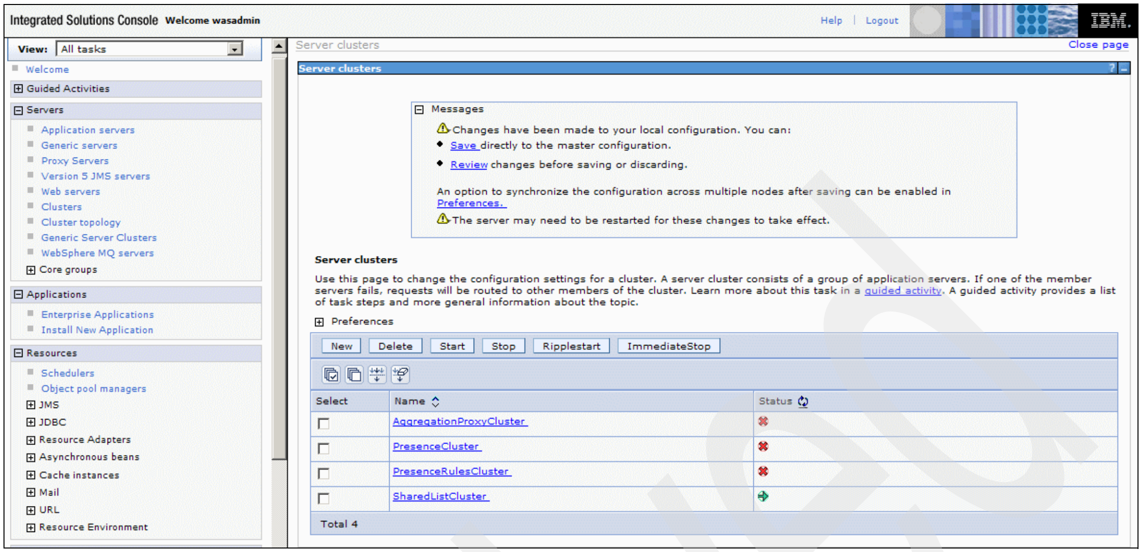


Figure 5-9 ISC ready to save master configuration for Presence Rules Cluster creation

Verifying the Presence Rules and Shared List clusters

To verify that the PresenceRulesCluster and SharedListCluster clusters are created:

1. In the Clusters view, select the clusters.
2. Click **Start** to start the cluster.

You should see both the clusters started, as shown in Figure 5-10 on page 158.

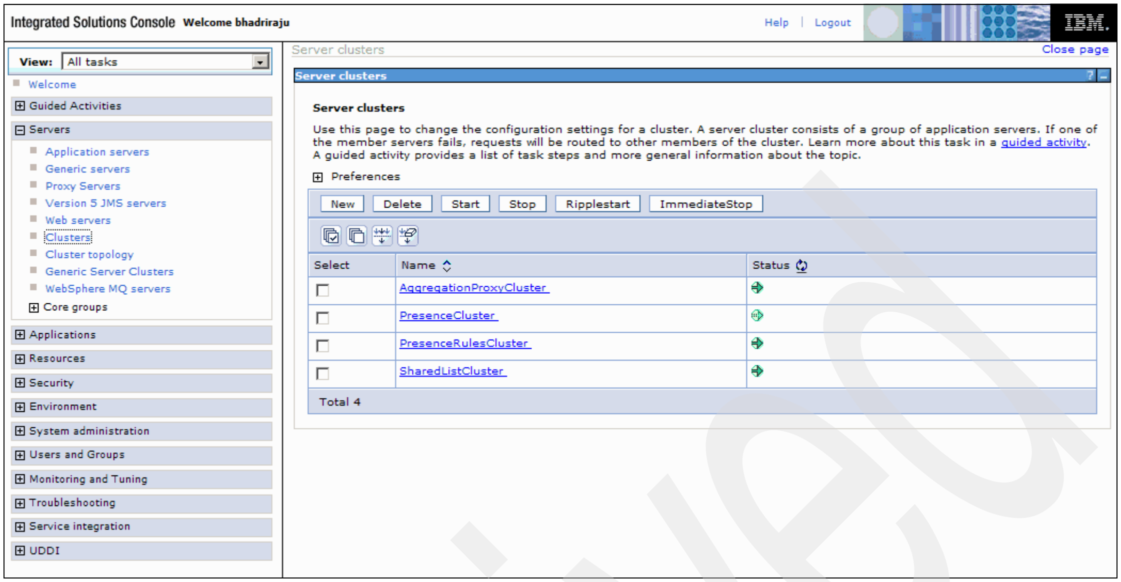


Figure 5-10 Shows the PresenceRulesCluster and SharedListCluster started

> **Note:** Before you start the cluster, make sure that all the nodes in the cluster are started. If the nodes are not started, the cluster does not start and presents the following warning:
>
> ```
> Warning: Cluster member XdmsSrv01 will not be started because the
> Node Agent on node XdmsNode01 is not active. Cluster members can be
> started individually from the cluster member collection panel.
> ```

5.3 Creating the converged Proxy Server

In this section, we create and configure the converged Proxy Server for XDMS. To create the Proxy Server:

1. In the Integrated Solutions Console of the WebSphere Application Server Deployment Manager:
   a. Click **Servers** → **Proxy Servers**.
   b. Click the **New** button to create and enter the information for a new Proxy Server.

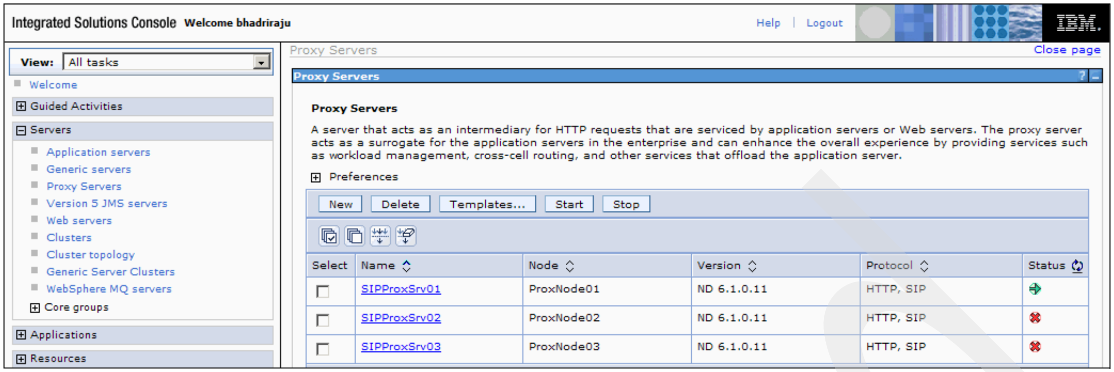


Figure 5-11 Proxy Servers

2. In the Create a new Proxy Server entry panel, enter the following information, as shown in Figure 5-12:
   a. Select the right WebSphere Application Server node that you want this Proxy Server to be created on. For the base environment, we dedicated a WebSphere Application Server node for it which is “ProxNode04”.
   b. In the Server Name field entry, enter **SIPProxSrv04** for the name of the Proxy Server, click **Next**.

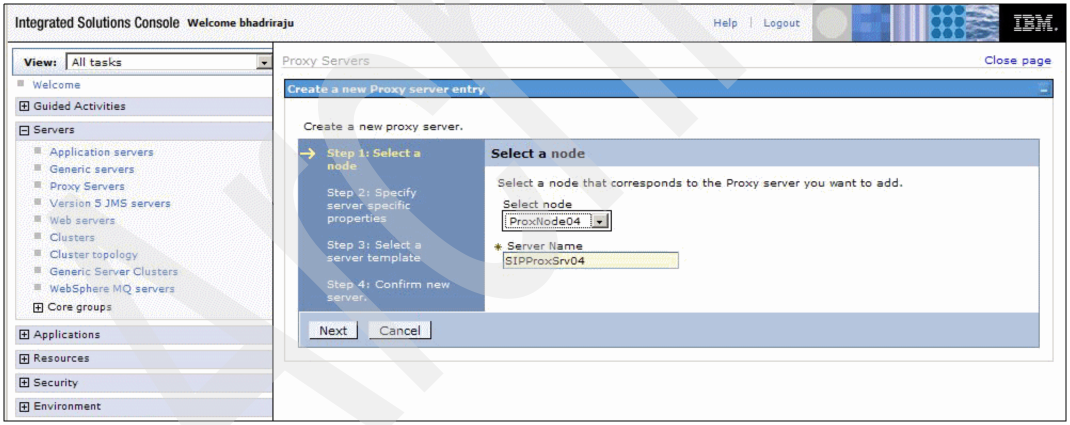


Figure 5-12 XDMS Converged Proxy Server creation for XDMS - Select a node

3. In the Specify server specific properties panel, accept the defaults, and click **Next**, as shown in Figure 5-13 on page 160.

Note: Note here that both HTTP and SIP protocols are checked which mean that the proxy will support both these protocols making it a converged proxy. The Generate unique ports check box creates unique ports for each protocol. To view all the ports created for a server in WebSphere Application Server, from the Integration Solutions Console, select **Proxy Servers** → **SIPProxSrv04** →**Ports**.

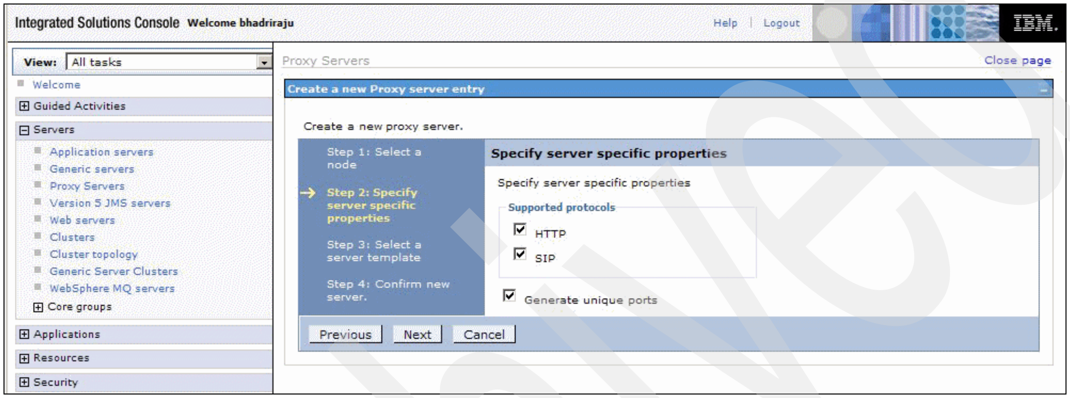


Figure 5-13 XDMS Converged Proxy Server creation for XDMS - Specify server specific properties

4. In the Select a server template panel, accept all defaults.
5. On the Confirm new server panel, click **Finish**, as shown in Figure 5-14.

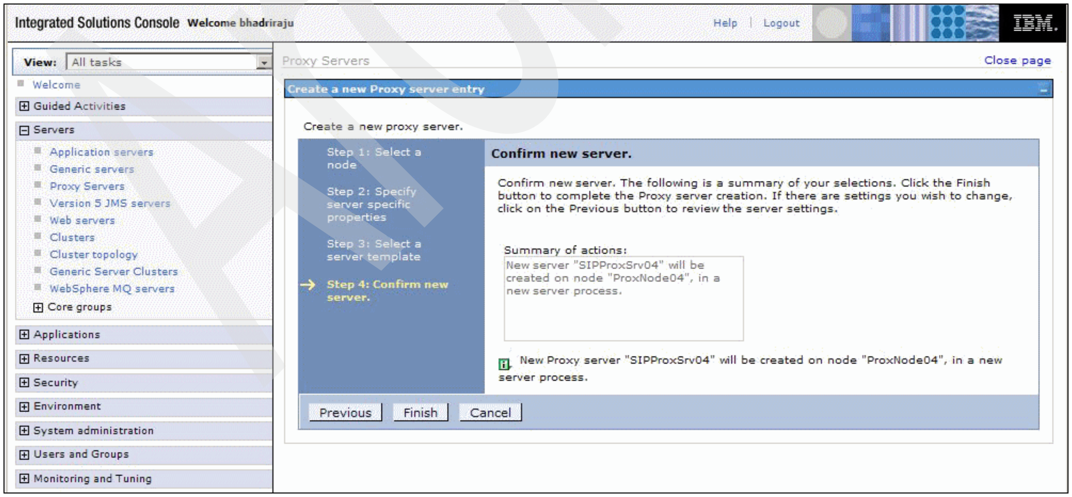


Figure 5-14 XDMS Converged Proxy Server creation for XDMS - Confirm new Server panel

6. In the Proxy Server panel, click **Save** to save the master configuration of the WebSphere Application Server, as shown in Figure 5-15.

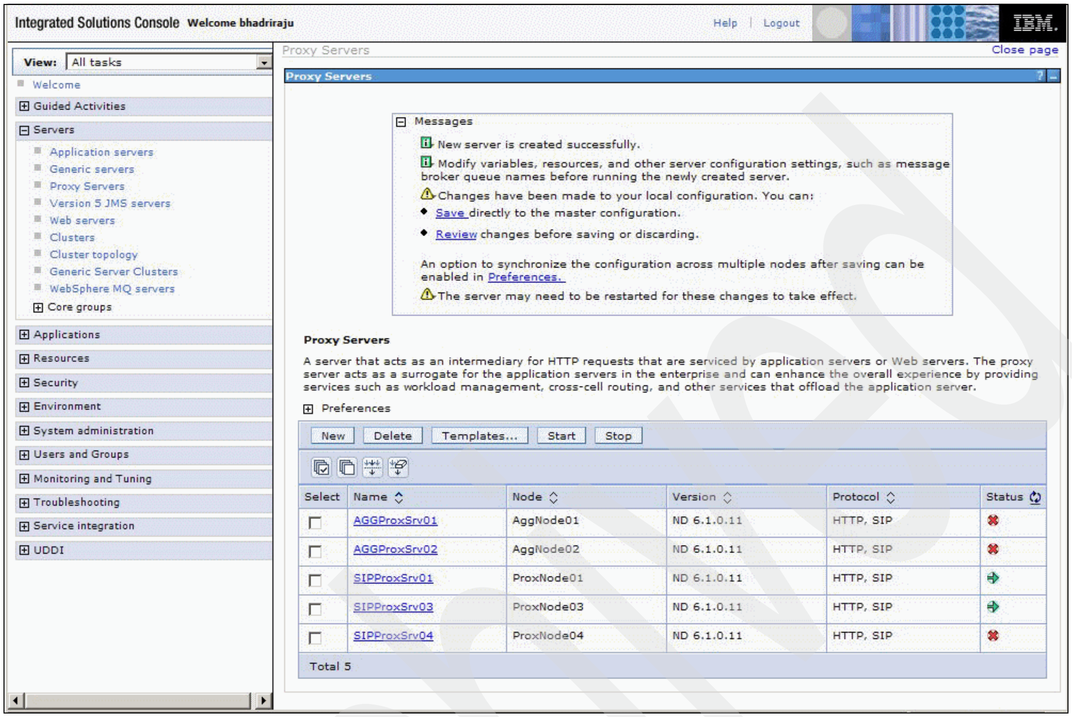


Figure 5-15 XDMS converged Proxy Server creation for XDMS - Proxy Servers panel

> **Note:** To verify if the Proxy Server that we just created was successful started, look for the Proxy Server `SIPProxSrv04` in the Integrated Solutions Console, from the Proxy Servers view.

5.4 User and Groups configuration

XDMS requires that a user with administrator and configuration privileges manage the XDMS documents that are global to all users that are defined in XDMS, for example white and black lists. For this purpose, we create a *superadmin* user with these privileges. When the Presence Server or XDMS aggregation proxy communicate with XDMS for these documents, they need to use the superadmin user account and password to manipulate the global documents. We created the superadmin user and the configuration as part of setting up the base environment in Chapter 3, “Overview of the base environment” on page 59.

See section, 3.3.2, “WebSphere Application Server security configuration” on page 69.

5.4.1 WebSphere Application Server security configuration

The XDMS Server requires that you turn on application-level security in WebSphere Application Server. We did the security configuration for the base environment when we set up the base environment in Chapter 3, “Overview of the base environment” on page 59. See section, 3.3.2, “WebSphere Application Server security configuration” on page 69.

5.4.2 Unpacking the XDMS install package

To unpack the XDMS install package, we need to stop all of the servers and all of the nodes in the XDMS Cluster, which includes the Deployment Manager that manages the XDMS Cluster. To unpack the XDMS install package:

1. Stop the servers using the command in Example 5-1.

Example 5-1 Command to stop the servers

```
/was_profile_root/bin/stopServer.sh server_name -username user_name and
-password password
was_profile_root :- /opt/IBM/WebSphere/AppServer/profiles/AppSrv01/
```

2. Stop all of the nodes in the cluster using the command in Example 5-2.

Example 5-2 Command to stop all the nodes in the cluster

```
/was_profile_root/bin/stopNode.sh -username user_name and -password
password
```

3. Stop the Deployment Manager, which manages the XDMS Cluster, using the command in Example 5-3.

Example 5-3 Command to stop the Deployment Manager

```
was_profile_root/bin/stopManager.sh -username user_name and -password
password
```

Note: The user name represents the WebSphere Application Server Administrator user ID, and the password is the password for that user ID.

4. Unpack the install tar file IBMXdmsInstallPackage_6.2.0.tar for XDMS in the *wasroot - /opt/IBM/WebSphere/ApplicationServer* directory on all the nodes that are in the XDMS Cluster and the Deployment Manager. This step is important because it copies the *com.ibm.ws.xdms_6.2.0.jar* into the plug-ins folder under wasroot.
5. Make sure that the *com.ibm.ws.xdms_6.2.0.jar* exists in the plug-in folder.
6. Start the Deployment Manager, all nodes, and all servers in all nodes:
   a. To start the Deployment Manager, use the command in Example 5-4.

Example 5-4 Command for starting the Deployment Manager

```
/was_profile_root/bin/startManager.sh -username user_name and -password
password
```

   b. To start the node, use the commands in Example 5-5.

Example 5-5 Command for starting the node

```
/was_profile_root/bin/startNode.sh -username user_name and -password
password
```

   c. To start the servers, use the commands in Example 5-6

Example 5-6 Command for starting the servers

```
/was_profile_root/bin/startServer.sh server_name -username user_name
and -password password
```

Note: *Before* you start the server, start the nodes that contain the servers.

5.4.3 Preparing the HTTP and SIP Trust Association Interceptor

The HTTP and SIP Trust Association Interceptors (TAI) are installed to make the IMS domain a trusted domain that intercepts the SIP and HTTP traffic into the domain for authentication. We installed and configured the TAI for the base environment when we set up the base environment in 3.3.2, “WebSphere Application Server security configuration” on page 69.

5.5 Creating the databases

In this section, we cover the steps for creating and configuring the various databases that XDMS needs. The XDMS databases store the various XML documents, such as Resource Lists, RLS documents, Presence Rules, and Usage Records.

5.5.1 Creating the DB2 database for the XDMS Shared List

To create the DB2 database for the XDMS Shared List application:

1. Copy the IBMSharedListXdms.sh file, which is shipped with the XDMS install package, to the DB2 server (NodeDB).

   > **Note:** The IBMSharedListXdms.sh file is located in the XDMS install package, in the following folder:
   >
   > `/installableApps/xdms/scripts/dbScripts/xdms`

2. On the DB2 server, login as a DB2 administrative user, which is normally db2inst1.
3. Run the IBMSharedListXdms.sh script using the following options

   ```
   ./IBMSharedListXdms.sh <XDMS> <db2inst1> <password>
   ```

4. Connect to the XDMS database to make sure that the database was successfully created, as shown in Figure 5-15 on page 161.

```
[db2inst1@db2 xdms]$ db2 connect to XDMS

   Database Connection Information

 Database server        = DB2/LINUX 9.1.2
 SQL authorization ID   = DB2INST1
 Local database alias   = XDMS

[db2inst1@db2 xdms]$ db2 list tables

Table/View                      Schema          Type  Creation time
------------------------------- --------------- ----- --------------------------
RESOURCELISTS                   DB2INST1        T     2007-09-25-16.48.43.818847
RESOURCELISTSACLS               DB2INST1        T     2007-09-25-16.48.45.441941
RLSSERVICES                     DB2INST1        T     2007-09-25-16.48.39.665586
RLSSERVICESACLS                 DB2INST1        T     2007-09-25-16.48.42.189048

  4 record(s) selected.

[db2inst1@db2 xdms]$
```

Figure 5-16 Shared List database creation: Tables creation verification

5.5.2 Creating the DB2 database for the Service Integration Bus

In this section, we provide steps for creating the DB2 database for the XDMS Services Integration Bus:

1. Login as a DB2 super user, db2inst1, using `su - db2inst1`.
2. Run the command **`./createXdmsSIBus.sh XDMSSIB db2inst1`** ***`password.`***

   The createXdmsSIBus.sh script is shipped with the XDMS install package in the following directory:
   `/opt/IBM/WebSphere/AppServer/installableApps/xdms/scripts/dbScripts/xdms`
3. Run db2 to connect to the XDMSSIB user, db2inst1, using *password*.

   Figure 5-17 shows the execution of steps 1 and 2, which you can use to verify that the script in step 1 ran successfully.

```
[db2inst1@db2 xdms]$ ./createXdmsSIBus.sh XDMSSIB db2inst1 zaq12wsx
DB20000I  The CREATE DATABASE command completed successfully.
09/26/2007 09:42:11     0   0   SQL1064N  DB2STOP processing was successful.
SQL1064N  DB2STOP processing was successful.
09/26/2007 09:42:13     0   0   SQL1063N  DB2START processing was successful.
SQL1063N  DB2START processing was successful.
[db2inst1@db2 xdms]$ db2 connect to XDMSSIB user db2inst1 using zaq12wsx

   Database Connection Information

 Database server        = DB2/LINUX 9.1.2
 SQL authorization ID   = DB2INST1
 Local database alias   = XDMSSIB

[db2inst1@db2 xdms]$
```

Figure 5-17 Service Integration Bus database creation: panel showing database created

5.5.3 Creating the DB2 database for Usage Records

To create the DB2 database for XDMS Usage Records:

1. Login as the DB2 super user db2inst1. Using **su - db2inst1**
2. Run `crtsrvDb2.sh` with the following arguments, as shown in Figure 5-18 on page 166:

   ```
   ./crtsrvDb2.sh db2.itso.ral.ibm.com 50000 XDMSUR XDMSUR US db2inst1
   zaq12wsx db2inst1 zaq12wsx ./UsageDbDb2.ddl  TRUE
   ```

Note: The crtsrvDb2.sh scripts exists in the following folder:

`/installableApps/xdms/scripts/dbScripts/xdms`

```
[db2inst1@db2 usageRecords]$ ./crtsrvDb2.sh db2.itso.ral.ibm.com 50000 XDMSUR XD
MSUR US db2inst1 zaq12wsx db2inst1 zaq12wsx ./UsageDbDb2.ddl TRUE
Usage: crtsrvDb2.sh dbServer dbPort dbName dbAlias dbLocale
                    dbInstance dbInstancePW dbUser dbUserPW
                    ddlFile dbCreate dbLocal dbNodeName menuflag
These values will be used to create the database.
Each value must be filled in.
The menuflag should be set to TRUE to display the menu.

Press Enter to accept these values and continue, otherwise
  enter the number of the parameter you wish to override.

   1. Database server hostname (dbServer)......:  db2.itso.ral.ibm.com
      Recommended Value: fully qualified host name.
   2. Database server connection port (dbPort).:  50000
   3. Database name (dbName)...................:  XDMSUR
      Recommended Value: XDMSURDB
   4. Database alias (dbAlias).................:  XDMSUR
      Recommended Value: XDMSUR
   5. Database locale (dbLocale)...............:  US
      Recommended Value: US
   6. Database server instance id (dbInstance).:  db2inst1
   7. Database server instance password (dbInstancePW).:  zaq12wsx
   8. Database userid (dbUser).................:  db2inst1
   9. Database userid password (dbUserPW)......:  zaq12wsx
  10. Path and file name of DDL file (ddlFile).:  ./UsageDbDb2.ddl
  11. Database (re)create (dbCreate)...........:  TRUE
      Recommended Value: TRUE
  12. Local Database (dbLocal) ................:  TRUE
      Recommended Value: TRUE
  13. Remote Database Server Node Name (dbNodeName)...:  RDBSRV
      Recommended Value: RDBSRV
```

Figure 5-18 Usage Records database creation: the values assigned to the attributes in the usage records db script

3. After running the script that connects to the XDMSUR database, check for the list of tables to verify that the database and table were created, as shown in Figure 5-19 on page 167.

```
[ b2inst1@db2 usageRecords]$ db2 connect to XDMSUR user db2inst1 using zaq12wsx

   Database Connection Information

 atabase server         = DB2/LINUX 9.1.2
 QL authorization ID    = DB2INST1
 ocal database alias    = XDMSUR

[ b2inst1@db2 usageRecords]$ db2 list tables

T ble/View                       Schema          Type  Creation time
- ------------------------------ --------------- ----- --------------------------
U AGERECORDS                     DB2INST1        T     2007-09-26-10.08.56.222569

  1 record(s) selected.
```

Figure 5-19 Usage Records database creation: Results of the User database creation for XDMS

5.5.4 Configuring data sources

To connect the databases to the XDSM Shared List and Presence Rules applications, we use the following process:

1. Create WebSphere variables.
2. Create the JAAS authentication alias.
3. Create the JDBC provider.
4. Define the data sources.

> **Note:** Make sure that the DB2 client is installed on all of the servers in the cluster where the XDMS Shared List and Presence Rules applications are deployed.

Creating WebSphere variables

We set the WebSphere Application Server variables for the base environment when we set up the base environment in 3.3.1, “Setting WebSphere Application Server variables” on page 67.

Creating the JAAS authentication alias

The authentication alias authenticates data access to the database. To create the authentication alias:

1. From the Integration Solutions Console of the Deployment Manager, select **Secure administration, applications, and infrastructure** → **JAAS - J2C authentication data**, as shown in Figure 5-20 on page 168.

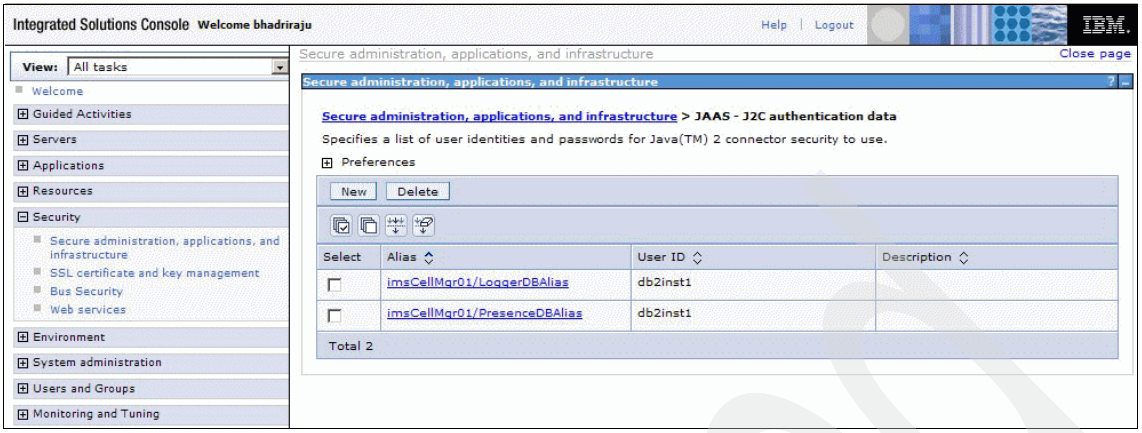


Figure 5-20 Creating JAAS Authentication Alias: J2C authentication alias panel in the ISC

2. Click **New**.
3. Enter the following data for the new authentication alias, as shown in Figure 5-21:
   - Name: **XDMS**
   - User ID: **db2inst1**
   - Password: **<password>**
   - Description: **XDMS Alias**

   Click **OK**.

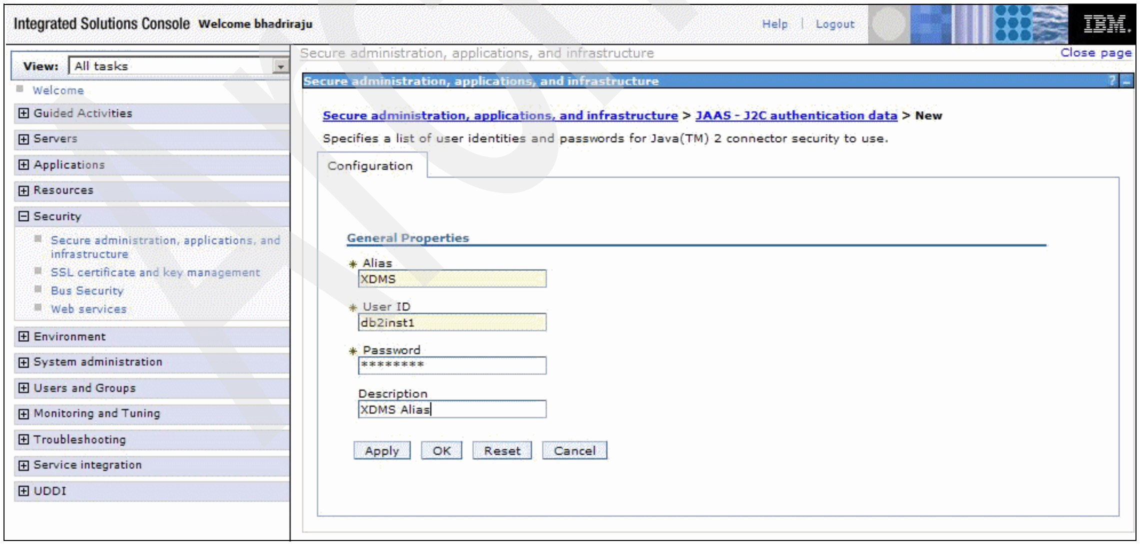


Figure 5-21 Creating JAAS Authentication Alias: panel to enter new authentication data in ISC

4. In the Configuration panel, click **Save** to save the master configuration, as shown in Figure 5-22.

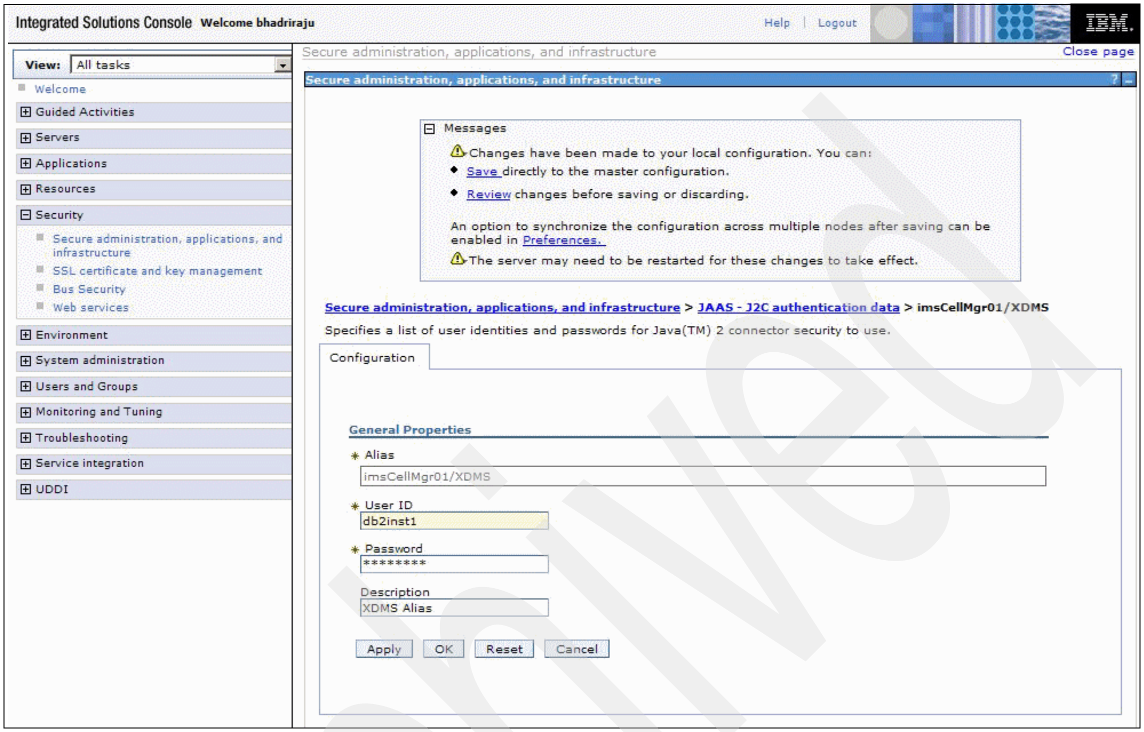


Figure 5-22 Creating the authentication alias: panel to save the authentication alias in ISC

Figure 5-23 on page 170 shows the authentication alias that was created in ISC.

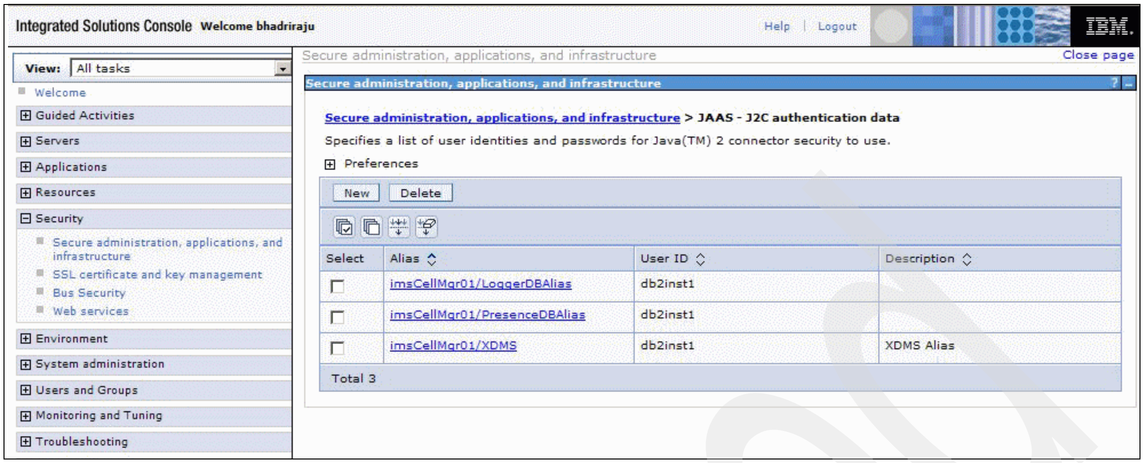


Figure 5-23 Creating authentication alias: Shows authentication alias created in ISC

Defining the data sources

WebSphere Application Server provides a database guided activity to configure the JDBC providers and data sources. Perform the following steps to configure data sources for the Shared List and Presence Rules XDMS databases:

1. In the Integration Solutions Console of the Deployment Manager, click **Guided Activities** as shown in Figure 5-24 on page 171.

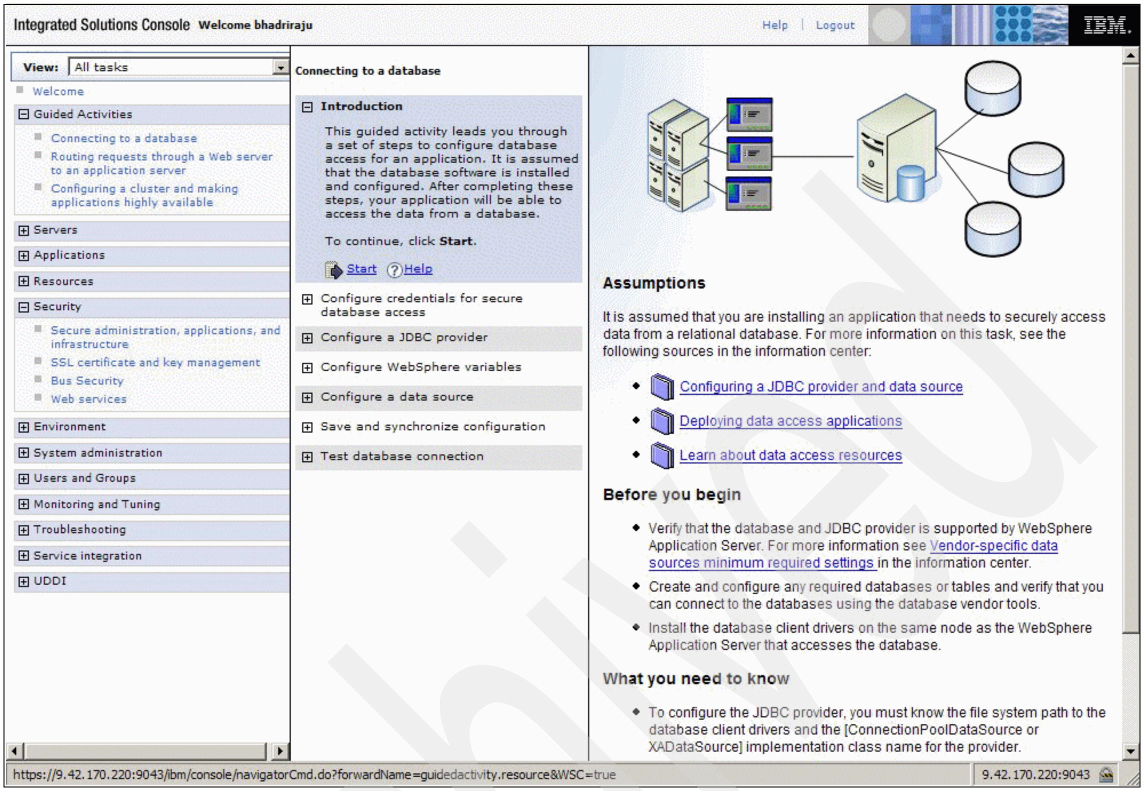


Figure 5-24 JDBC provider creation for Shared List and Presence Rules databases Panel 1

2. Click **Configure a JDBC provider** as shown in Figure 5-25 on page 172.

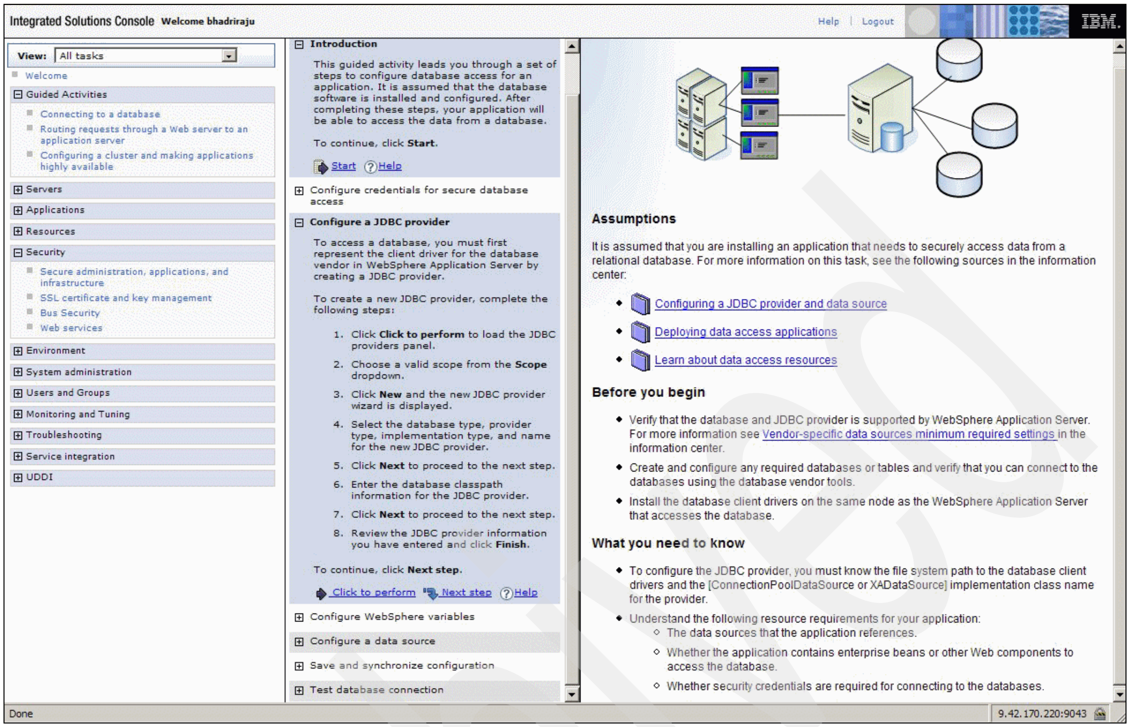


Figure 5-25 JDBC provider creation for Shared List and Presence Rules databases: Panel 2

3. Click the **Click to Perform** link to open the JDBC provider panels, as shown in Figure 5-26 on page 173.

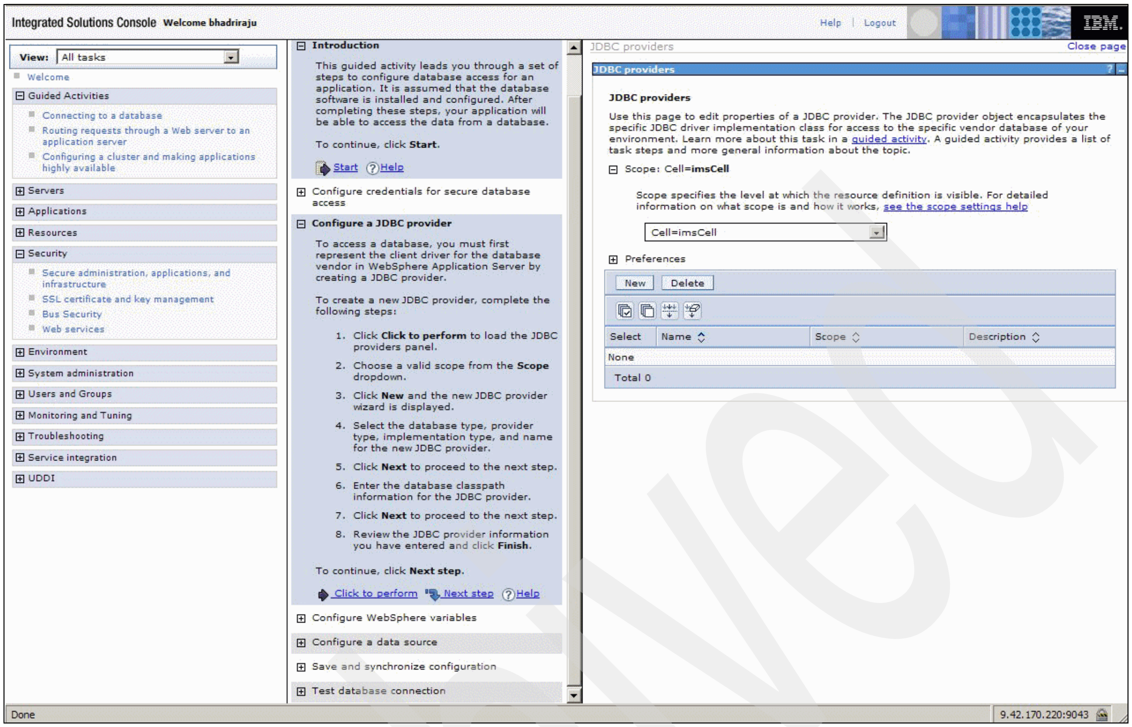


Figure 5-26 JDBC provider creation for Shared List and Presence Rules databases Panel 3

4. Click **New** to create a new JDBC provider with the Scope set to cell: imsCell.
5. Enter the following values as shown in Figure 5-27 on page 174:
   a. Database Type **DB2**
   b. Provider Type **DB2 Universal JDBC Driver Provider**
   c. Implementation Type **XA data source**
   d. Name **DB2 Universal JDBC Driver Provider (XA)**
6. Click **Next**.

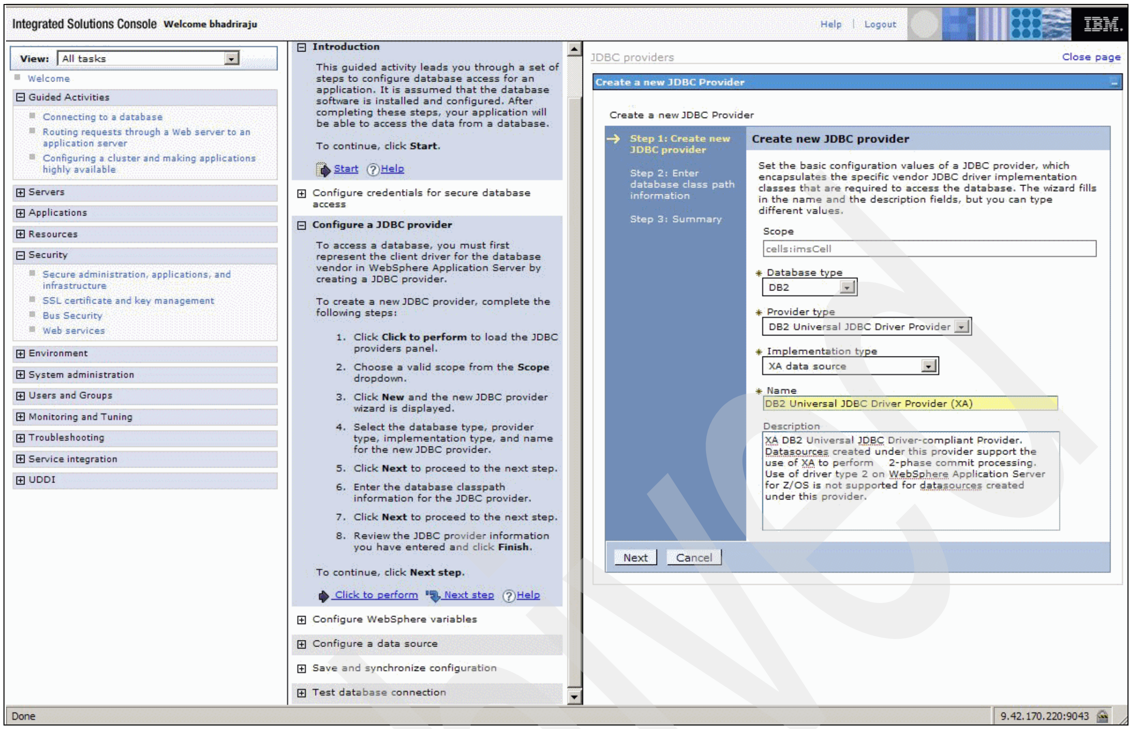


Figure 5-27 JDBC provider creation for Shared List and Presence Rules databases Panel 4

7. In the Enter Database class path information panel enter the following information:

   a. Directory location for db2jcc.jar, db2jcc_license_cisuz.jar, which is saved as WebSphere variable ${DB2UNIVERSAL_JDBC_DRIVER_PATH} entry: **/opt/ibm/db2/V9.1/java**.

8. Click **Next**.

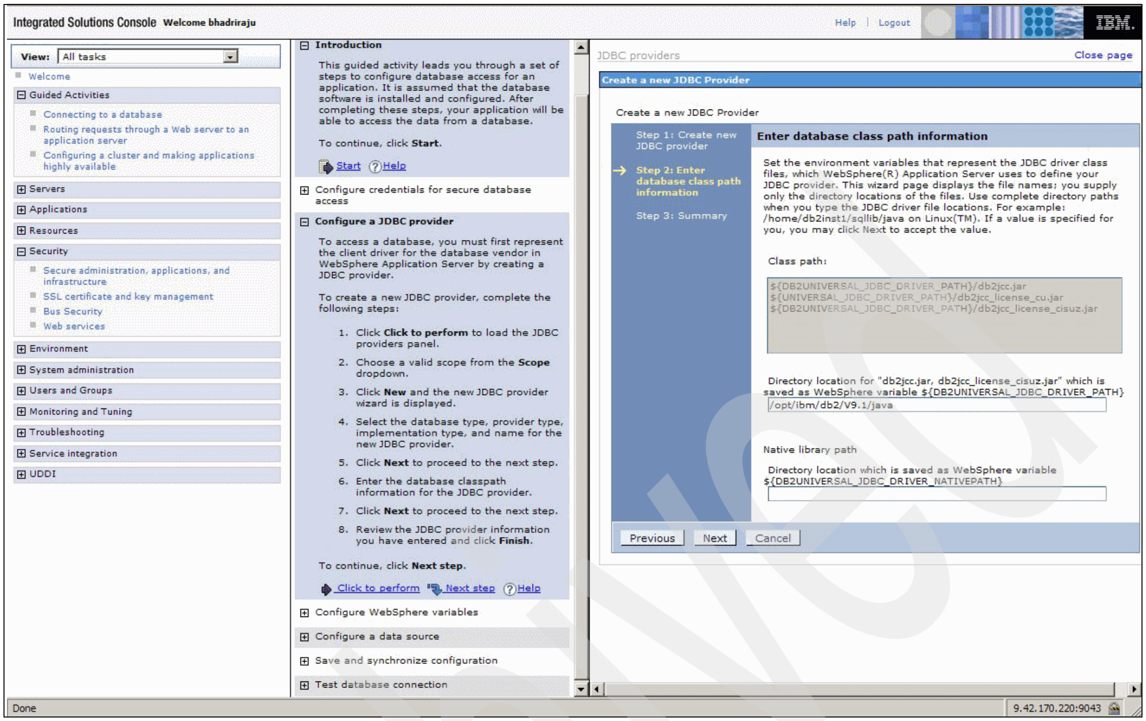


Figure 5-28 JDBC provider creation for Shared List and Presence Rules: Enter database class path information

9. On the Summary panel, click **Finish** to complete the creation of the JDBC provider, as shown in Figure 5-28.

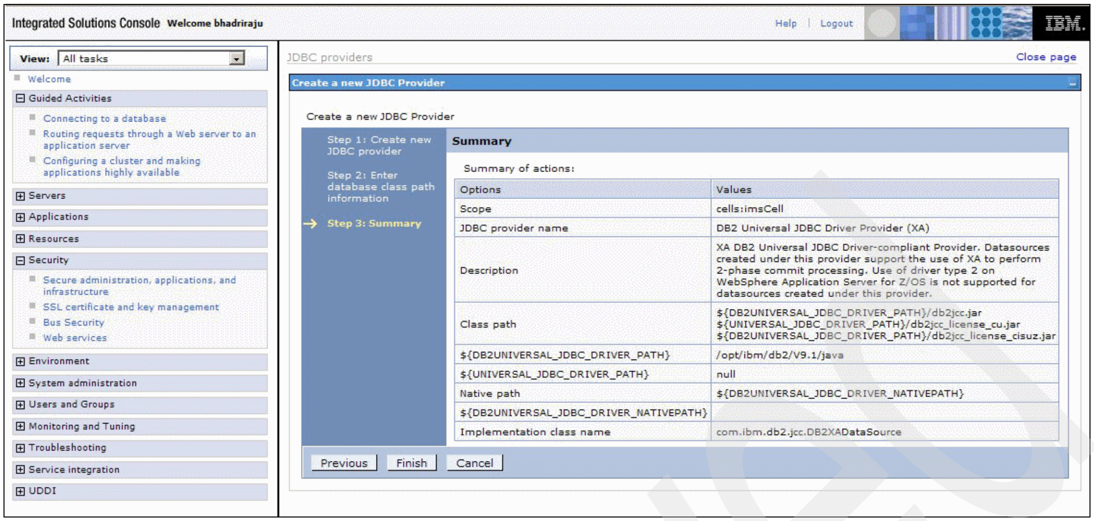


Figure 5-29 JDBC provider creation for Shared List and Presence Rules databases: Summary panel

10. In the JDBC providers panel, click **Save** to save the master configuration, as shown in Figure 5-29.

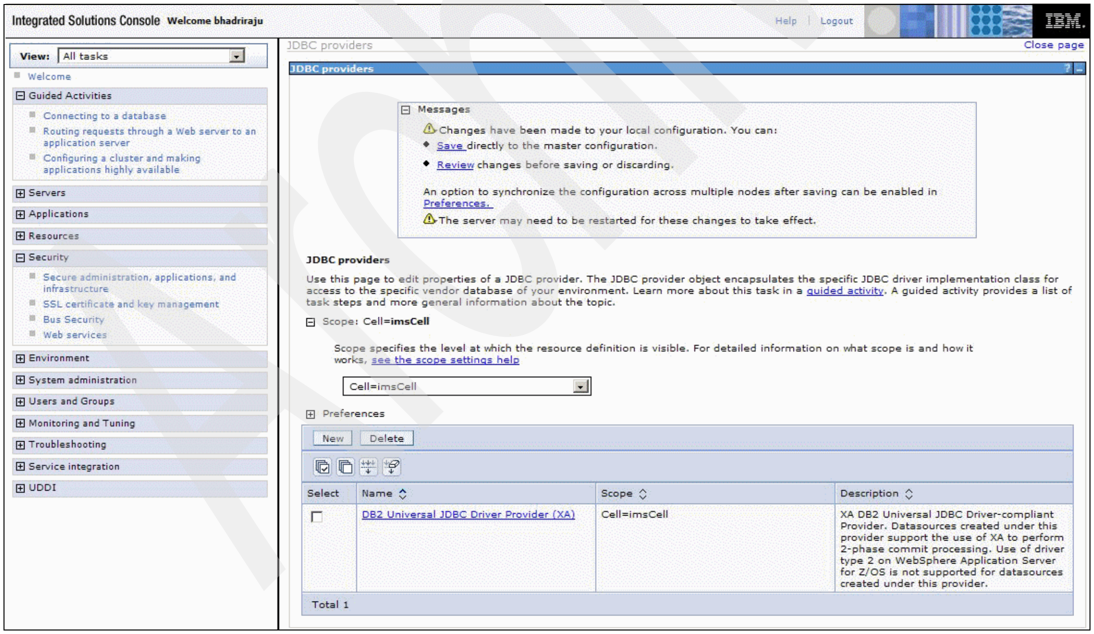


Figure 5-30 JDBC provider creation for Shared List and Presence Rules databases: Save panel

11. Click the JDBC provider **DB2 Universal JDBC Driver Provider** to open its Additional Properties, as shown in Figure 5-31.

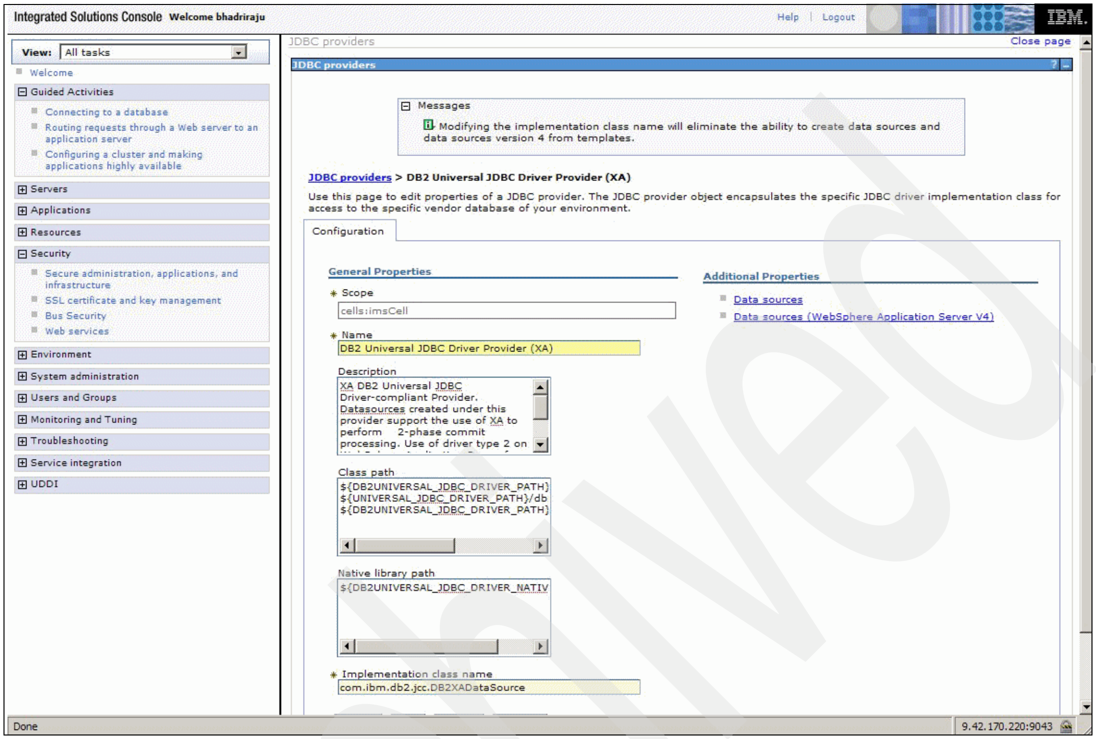


Figure 5-31 Data source creation for Shared List and Presence Rules databases: Panel 1

12. In the JDBC providers panel, under Additional Properties, click **Data sources** to configure the data sources for the DB2 Universal JDBC Driver Provider (XA) JDBC provider, as shown in Figure 5-32 on page 178.

13. Click **New** to create a data source.

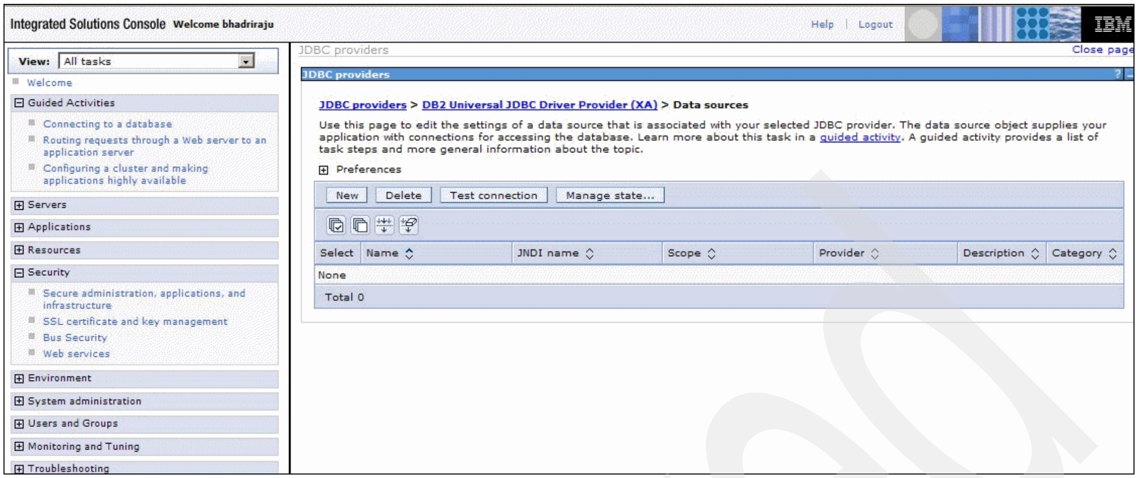


Figure 5-32 Data source creation for Shared List and Presence Rules databases: Panel 2

14. To create a data source for the XDMS databases, enter the following values in the Enter basic data source information panel, as shown in Figure 5-33 on page 179:

    – Date Source Name: **DB2 Universal JDBC Driver DataSource**

    – JNDI Name: **jdbc/xdms**

    – Component-managed authentication alias and XA recovery authentication alias: **imsCellMgr01/XDMS**

    Click **Next**.

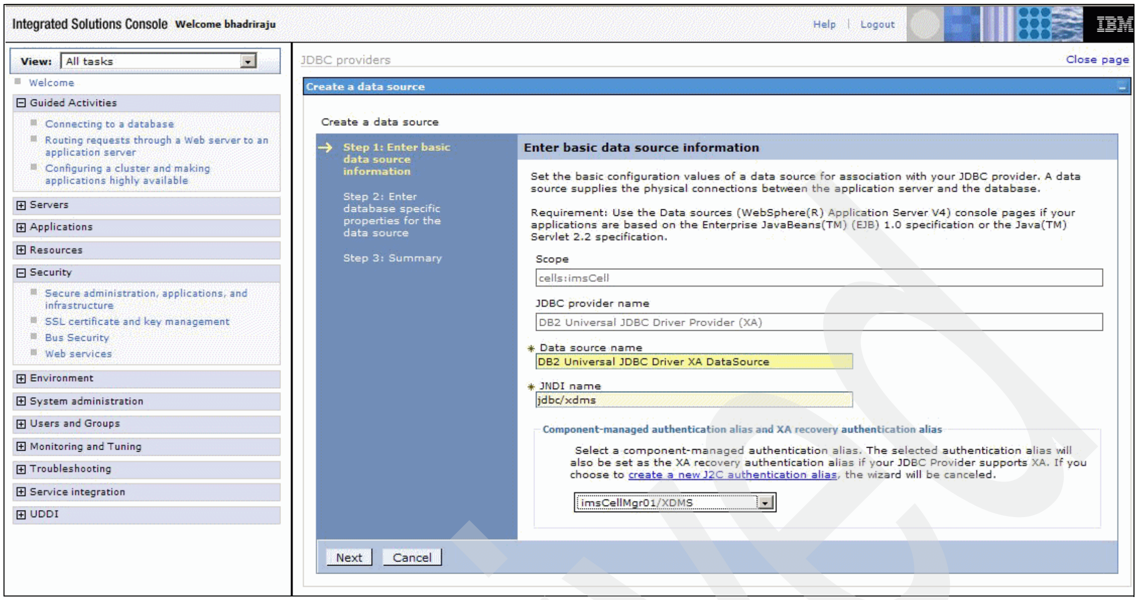


Figure 5-33 Data source creation for Shared List and Presence Rules databases: Panel 3

15. In the Enter database specific properties for the data source panel, enter the following values, as shown in Figure 5-34:
    - Database name: **XDMS**
    - Server name: **db2.itso.ral.ibm.com**
    - Accept the defaults for the other values

    Click **Next**.

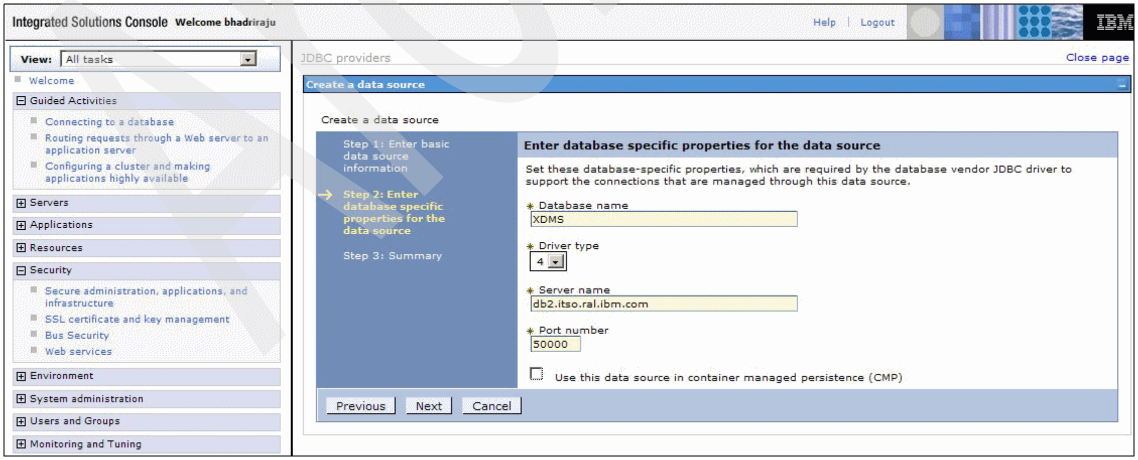


Figure 5-34 Data source creation for Shared List and Presence Rules databases: Panel 4

16.On the Summary panel, click **Finish** to complete the data source creation.

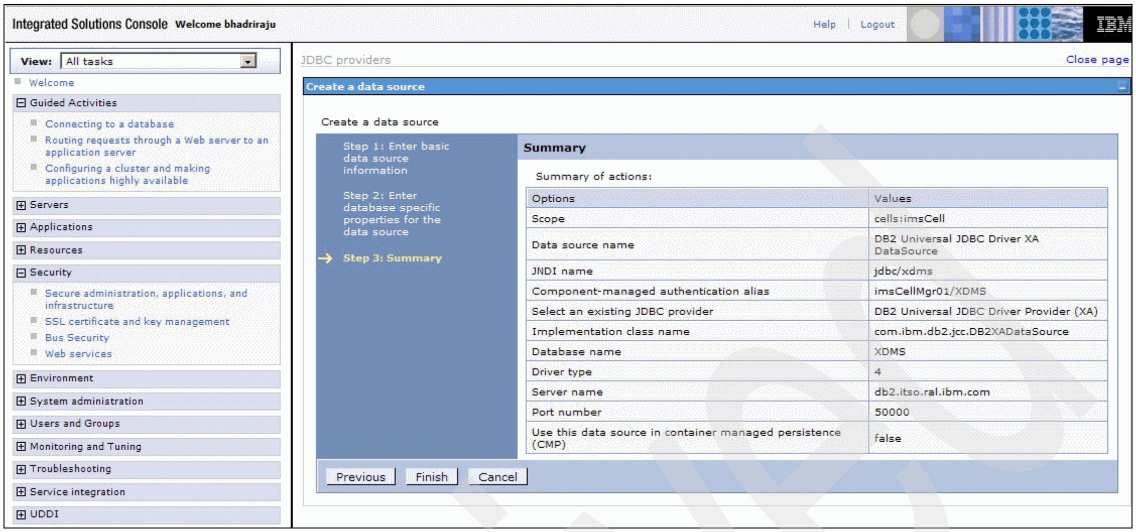


Figure 5-35 Data source creation for Shared List and Presence Rules databases: Panel 5

17.Click **Save** to save the master configuration, as shown in Figure 5-36.

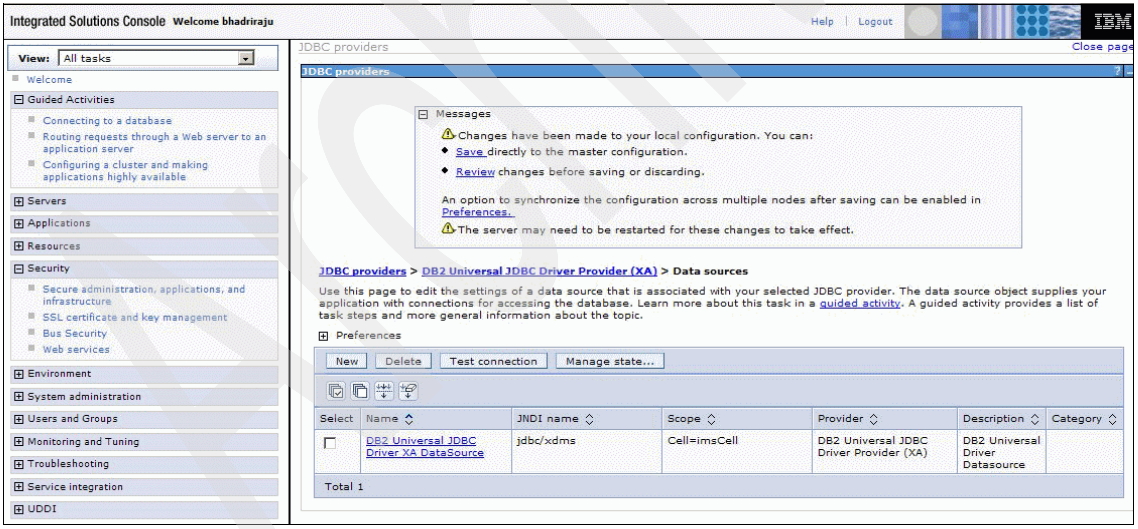


Figure 5-36 Data source creation for Shared List and Presence Rules databases: Panel 6

18.Select the data source that you just created, and click **Test Connection** to verify the connection to the database, as shown in Figure 5-37 on page 181.

19. If the connection test is successful, the following message should appear, as shown in Figure 5-37:

```
The test connection operation for data source DB2 Universal JDBC
Driver XA DataSource on server dmgr at node imsCellMgr01 was
successful.
```

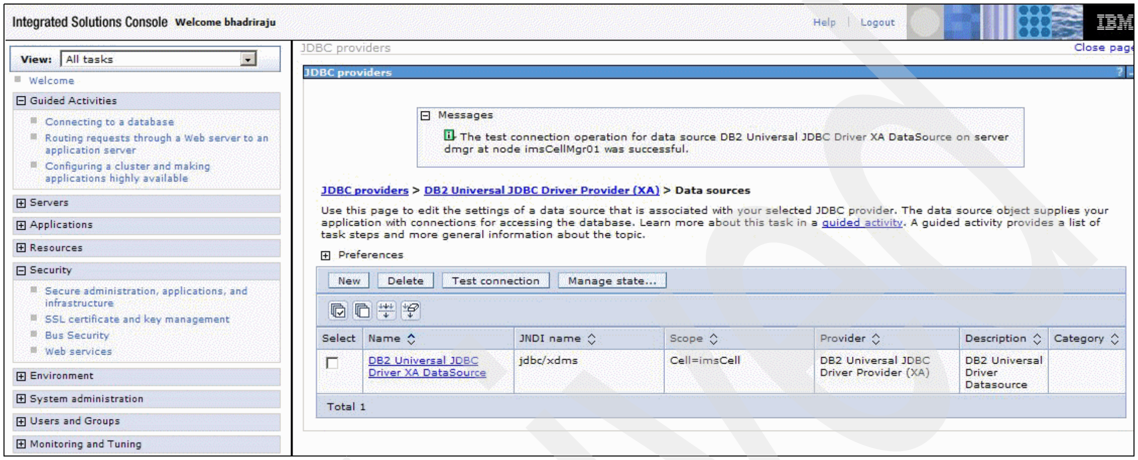


Figure 5-37 Data source creation for the XDMS database: Testing the connection to database

Defining the data source for the SIB database

In this section, we provide steps to create data source for the Service Integration Bus data source.

The process to define the data source for the SIB database is two-fold:

1. Repeat steps 1-21 in the previous section “Defining the data sources” on page 170.
2. Use the values that you defined in “Defining the data sources” on page 170 to perform steps 5 and 6, as well as steps 14, 15, 16, 17.

To define the data source for the SIB database:

1. Perform steps 5 and 6, from “Defining the data sources” on page 170, using the following data. Enter the following values, as shown in Figure 5-38 on page 182:
   - Database Type: **DB2**
   - Provider Type: **DB2 Universal JDBC Driver Provider**
   - Implementation Type: **Connection pool data source**
   - Name: **DB2 Universal JDBC Driver Provider**

   Click **Next**.

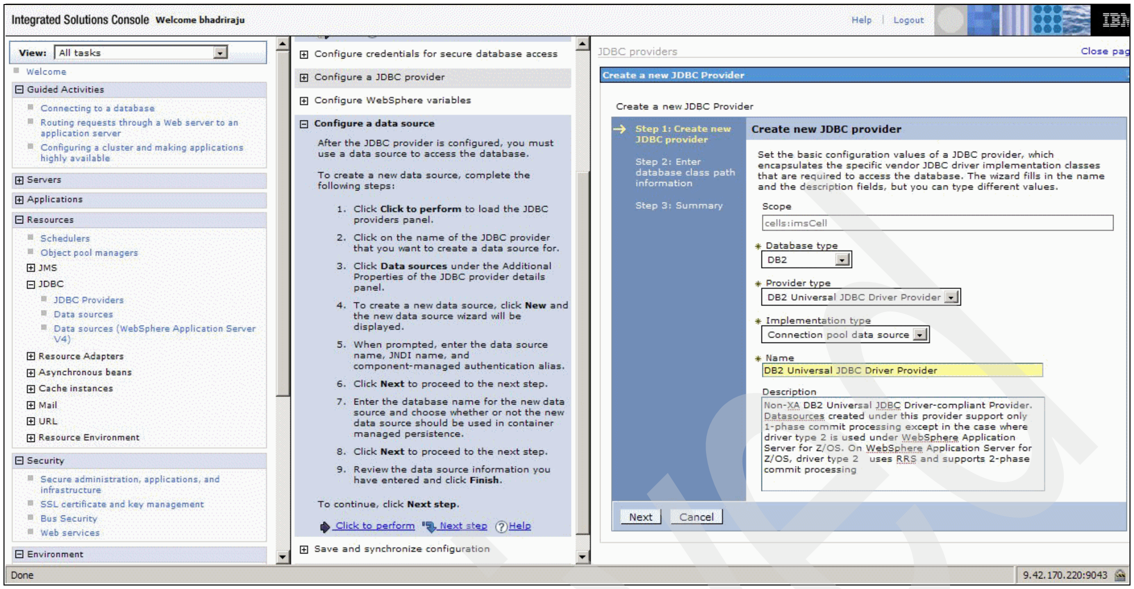


Figure 5-38 Creating JDBS provider for Service Integration Bus database

2. Perform steps 14 and 15, from "Defining the data sources" on page 170, using the following data, and enter the following values as shown in Figure 5-39 on page 183:

   – Date Source Name: **DB2 Universal JDBC Driver DataSource**

   – JNDI Name: **jdbc/xdmssib**

   – Component-managed authentication alias and XA recovery authentication alias: **imsCellMgr01/XDMS**

   Click **Next**.

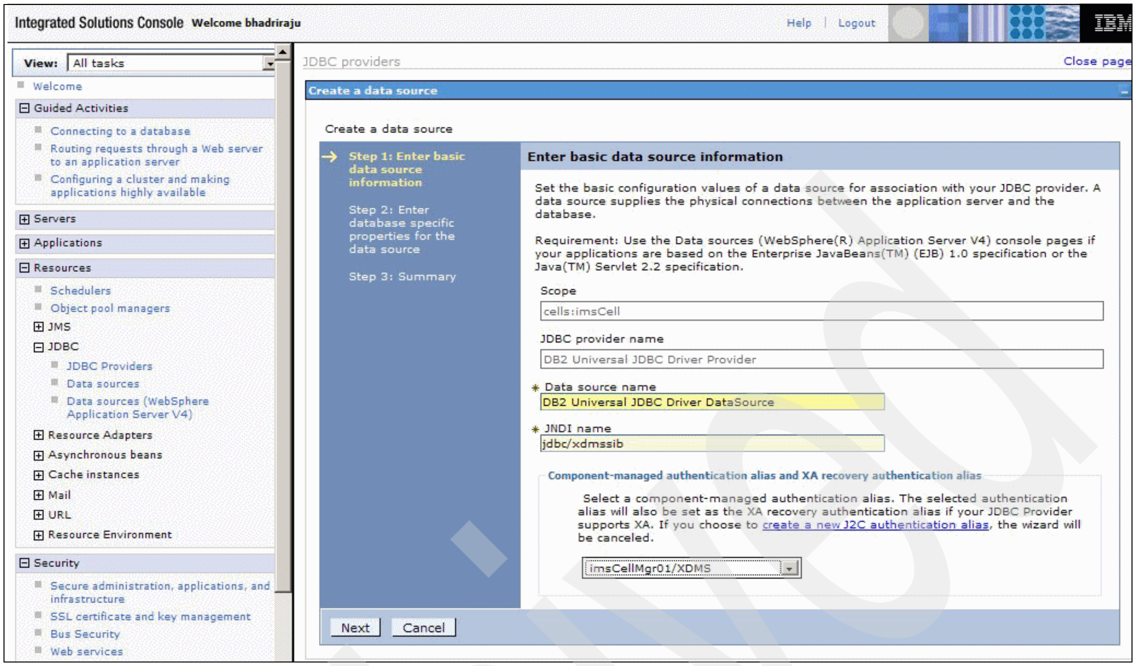


Figure 5-39 Data source creation for Service Integration Bus: Basic data source information panel

3. Perform steps 16 and 17, from "Defining the data sources" on page 170, using the following data, and enter the values, as shown in Figure 5-40:
   - Database name: **XDMSSIB**
   - Server name: **db2.itso.ral.ibm.com**

   Click **Next**.

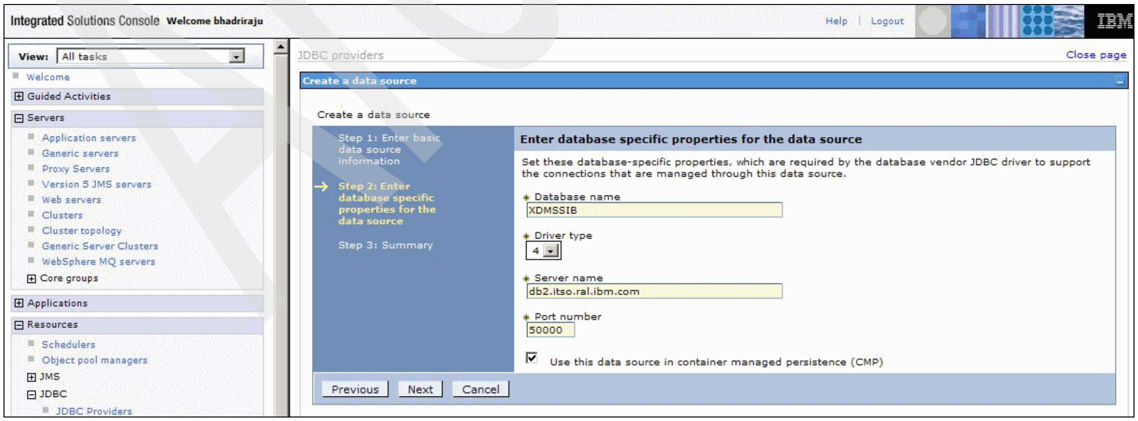


Figure 5-40 Data source creation for Service Integration Bus: Database specific properties for data store

4. Select the data source that you created, and click **Test Connection** to verify the connection to the database.
5. If the connection test is successful, the following message is displayed:

```
The test connection operation for data source DB2 Universal JDBC
Driver XA DataSource on server dmgr at node imsCellMgr01 was
successful
```

Connecting to the Usage Record data source

In this section, we cover the steps to create the data source for the Usage Records data source. Repeat steps 1-21 in the section “Defining the data sources” on page 170, and use the values in this section to perform steps 5 and 6, 14 and 15, and 16 and 17.

1. Perform steps 5 and 6 using the following values that you must enter, as shown in Figure 5-41:
   - Database Type: **DB2**
   - Provider Type: **DB2 Universal JDBC Driver Provider**
   - Implementation Type: **Connection pool data source**
   - Name: **DB2 Universal JDBC Driver Provider**

   Click **Next**.

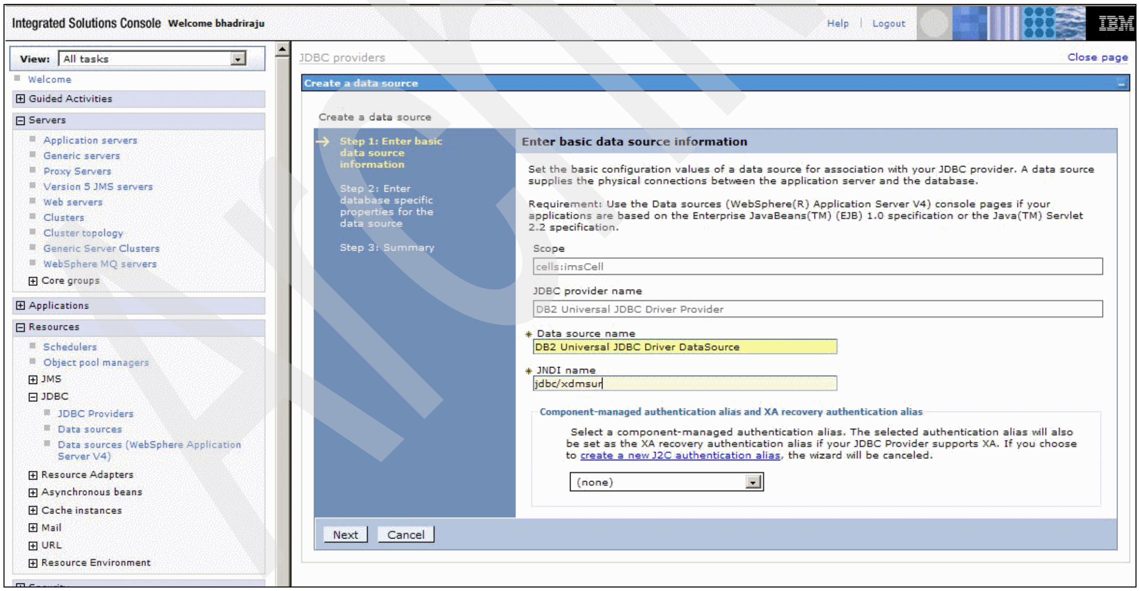


Figure 5-41 Data source creation for Usage Records: Basic data source information

2. Perform steps 14 and 15 using the following values that you must enter:
   - Date Source Name: **DB2 Universal JDBC Driver DataSource**
   - JNDI Name: **jdbc/xdmsur**
   - Component-managed authentication alias and XA recovery authentication alias: **imsCellMgr01/XDMS**

   Click **Next**.
3. Perform steps 16 and 17 using the following data using the following values that you must enter:
   - Database name: **XDMSUR**
   - Server name: **db2.itso.ral.ibm.com**

   Click **Next**.
4. Select the data source that you previously created, and click **Test Connection** to verify the connection to the database.
5. If the connection test is successful, the following message is displayed:

   ```
   The test connection operation for data source DB2 Universal JDBC
   Driver XA DataSource on server dmgr at node imsCellMgr01 was
   successful
   ```

5.6 Configuring the Service Integration Bus and JMS

Messaging engines use a message store to keep its operating information and message recovery. In a clustered environment, the message store is the Service Integration Bus database that we created earlier in the installation process. A Service Integration Bus and JMS must be installed on each XDMS Cluster, in our case, the Shared List and Presence Rules Clusters.

We provide two scripts to create and delete Service Integration Bus and JMS artifacts:

- createXDMSMessaging.py

  This script creates the Service Integration Bus and JMS artifacts that the IBM WebSphere XML Document Management Servers (XDMS) component uses. This script does not currently enable security.
- deleteXDMSMessaging.py

  This script removes Service Integration Bus and JMS artifacts that the IBM WebSphere XML Document Management Servers (XDMS) component uses.

5.6.1 Configuring the Service Integration Bus and JMS for Shared List

You must create and configure the Service Integration Bus for the Share List Cluster:

1. Go the Deployment Manager bin directory, which is /opt/IBM/WebSphere/AppServer/bin
2. Enter and run the following command:

   ```
   ./wsadmin -username superadmin -password password -lang jython -f
   ./installableApps/xdms.scripts/createXDMSMessaging.py
   -busDataSourceJndi jdbc/xdmssib busDataStoreSchema LISTSIB -cluster
   SharedListCluster, -busName SharedListBus
   ```

   Figure 5-42 on page 187 shows the result of running this command.

```
[root@ldap bin]# ./wsadmin.sh -username wasadmin -password zaq12wsx -lang jython
 -f /mnt/IMS_62_Builds/XDMS\ -\ 20070924_2046/installableApps/xdms/scripts/
dbScripts/  jmsScripts/ uninstall/
[root@ldap bin]# ./wsadmin.sh -username wasadmin -password zaq12wsx -lang jython
 -f /mnt/IMS_62_Builds/XDMS\ -\ 20070924_2046/installableApps/xdms/scripts/jmsSc
ripts/createXDMSMessaging.py -busDataSourceJndi jdbc/xdmssib -busDataStoreSchema
 LISTSIB -cluster SharedListCluster -busName SharedListBus
WASX7209I: Connected to process "dmgr" on node imsCellMgr01 using SOAP connector
;  The type of process is: DeploymentManager
WASX7303I: The following options are passed to the scripting environment and are
 available as arguments that are stored in the argv variable: "[-busDataSourceJn
di, jdbc/xdmssib, -busDataStoreSchema, LISTSIB, -cluster, SharedListCluster, -bu
sName, SharedListBus]"

****************************************************************

        Licensed Materials - Property of IBM
        5724-O05
        (c) Copyright IBM Corp. 2007 All Rights Reserved.
        * Trademark of International Business Machines

****************************************************************

XDMS messaging configured for cluster SharedListCluster
Adding SIBus SharedListBus .......

    XDMS Service Integration Bus added

Adding Cluster SharedListCluster as a member of SIBus SharedListBus .......

schemaName = LISTSIB.

SharedListCluster added as a cluster member of SIBus SharedListBus

Adding Destination SharedListCluster to SIBus SharedListBus .......

  Destination SharedListCluster added to SIBus SharedListBus

Adding JMS Connection Factory xdmsTCF to Node .......

    JMS Connection Factory xdmsTCF added to cluster SharedListCluster

Adding JMS Topic xdmsPublishT to cluster SharedListCluster .......

Adding JMS Activation Spec XDMSActivationSpec to cluster SharedListCluster .....
..

    JMS Activation Spec XDMSActivationSpec added to cluster SharedListCluster

XDMS messaging configured for cluster SharedListCluster
```

Figure 5-42   Service Integration bus creation and configuration for SharedListCluster

5.6.2 Configuring the Service Integration Bus and JMS for Presence Rules

You must create and configure the Service Integration for the Presence Rules Cluster:

1. Go the Deployment Manager bin directory, which is /opt/IBM/WebSphere/AppServer/bin
2. Enter and run the following command:

```
./wsadmin -username superadmin -password password -lang jython -f
./installableApps/xdms.scripts/createXDMSMessaging.py
-busDataSourceJndi jdbc/xdmssib busDataStoreSchema RULESSIB -cluster
PresenceRulesCluster, -busName PresenceRulesBus
```

   Figure 5-43 on page 189 shows the result of running this command.

```
[root@ldap bin]# ./wsadmin.sh -username wasadmin -password zaq12wsx -lang jython
 -f /mnt/IMS_62_Builds/XDMS\ -\ 20070924_2046/installableApps/xdms/scripts/jmsSc
ripts/createXDMSMessaging.py -busDataSourceJndi jdbc/xdmssib -busDataStoreSchema
 RULESSIB -cluster PresenceRulesCluster -busName PresenceRulesBus
WASX7209I: Connected to process "dmgr" on node imsCellMgr01 using SOAP connector
;  The type of process is: DeploymentManager
WASX7303I: The following options are passed to the scripting environment and are
 available as arguments that are stored in the argv variable: "[-busDataSourceJn
di, jdbc/xdmssib, -busDataStoreSchema, RULESSIB, -cluster, PresenceRulesCluster,
 -busName, PresenceRulesBus]"

****************************************************************

        Licensed Materials - Property of IBM
        5724-O05
        (c) Copyright IBM Corp. 2007 All Rights Reserved.
        * Trademark of International Business Machines

****************************************************************

XDMS messaging configured for cluster PresenceRulesCluster
Adding SIBus PresenceRulesBus .......

    XDMS Service Integration Bus added

Adding Cluster PresenceRulesCluster as a member of SIBus PresenceRulesBus ......
.

schemaName = RULESSIB.

PresenceRulesCluster added as a cluster member of SIBus PresenceRulesBus

Adding Destination PresenceRulesCluster to SIBus PresenceRulesBus .......

  Destination PresenceRulesCluster added to SIBus PresenceRulesBus

Adding JMS Connection Factory xdmsTCF to Node .......

    JMS Connection Factory xdmsTCF added to cluster PresenceRulesCluster

Adding JMS Topic xdmsPublishT to cluster PresenceRulesCluster .......

Adding JMS Activation Spec XDMSActivationSpec to cluster PresenceRulesCluster ..
.....

    JMS Activation Spec XDMSActivationSpec added to cluster PresenceRulesCluster

XDMS messaging configured for cluster PresenceRulesCluster
```

Figure 5-43 Service Integration bus creation and configuration for PresenceRulesCluster

5.7 Deploying the Shared List application

Create the following items before you deploy the Shared List enterprise application into the ShareListCluster:

- XDMS database and Shared List tables
- Data source for XDMS Shared List

To deploy the Shared List enterprise application:

1. Launch the Integrated Solutions Console (ISC), and Select **Application** → **Enterprise Applications**, as shown in Figure 5-44.

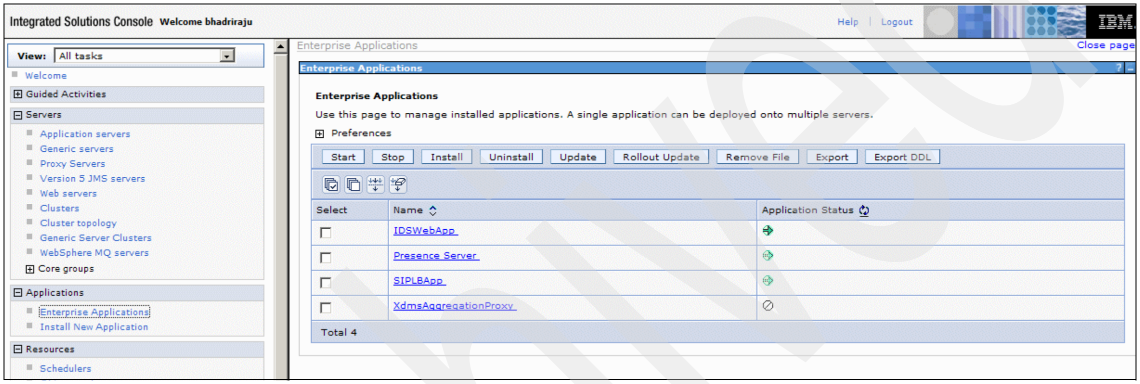


Figure 5-44 Shared List application deployment: Enterprise Applications panel in ISC

2. Click the **Install** button, and from the Local or Remote file system, select the IBMSharedListXdms.ear file by clicking the **Browse** button, as shown in Figure 5-45 on page 191.
3. Click **Next**.

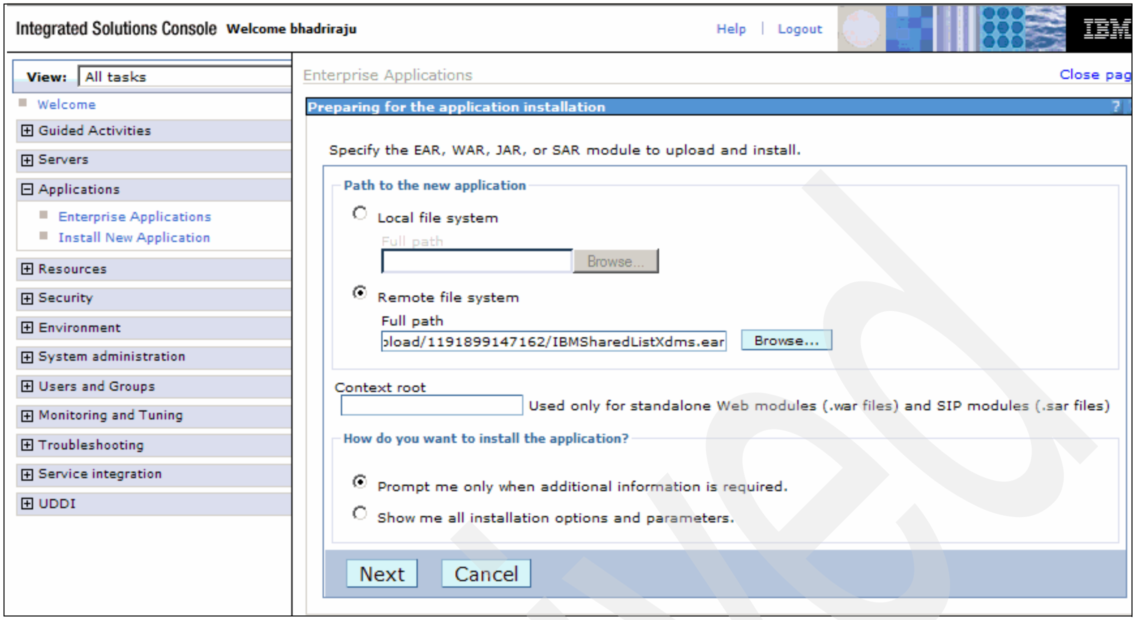


Figure 5-45 Shared List application deployment: Preparing for the application installation in ISC

4. On the Select Install Options panel, accept all defaults, as shown in Figure 5-46 on page 192.
5. Click **Next**.

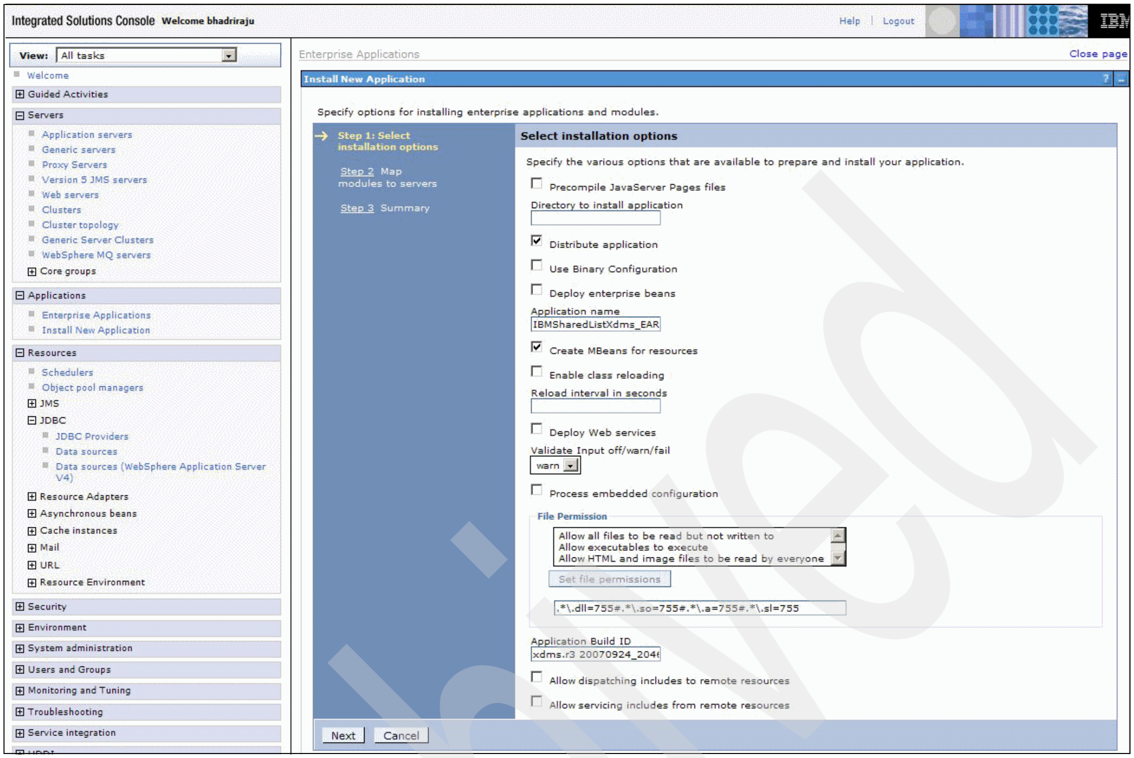


Figure 5-46 Shared List application deployment: Select Install Options pane in ISC

6. In the Map Modules to Servers panel, select all of the modules from the list, and from the Clusters and Servers list, select **SharedListCluster**, as shown in Figure 5-47 on page 193.
7. Click **Apply**.
8. Click **Next**.

   This action should deploy the Shared List application to the ShareListCluster as, shown in Figure 5-47 on page 193.

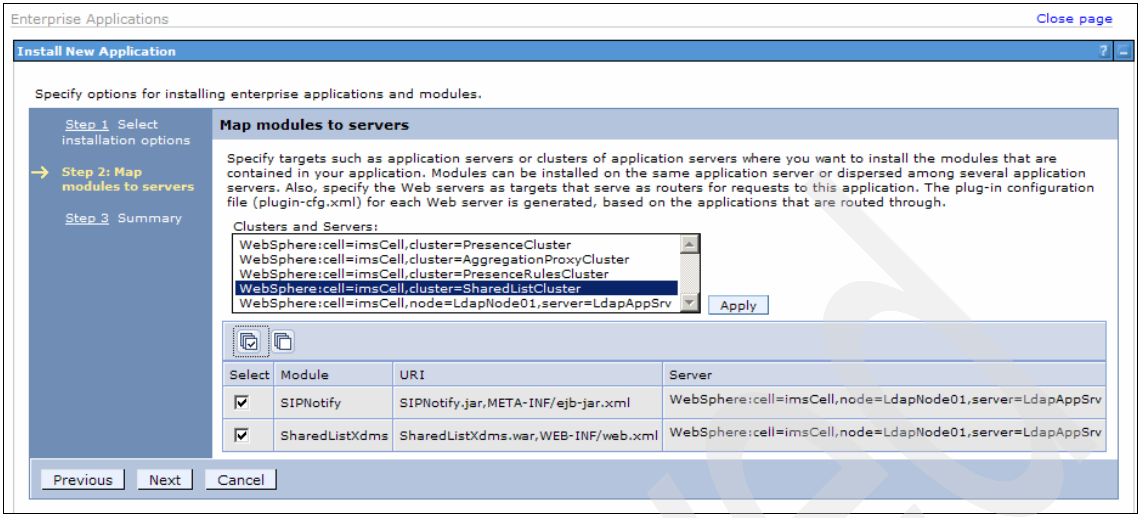


Figure 5-47 Shared List application deployment: Map modules to servers panel in ISC

9. Click **Finish** on the last panel to complete the deployment of the Shared List application, as shown in Figure 5-48.

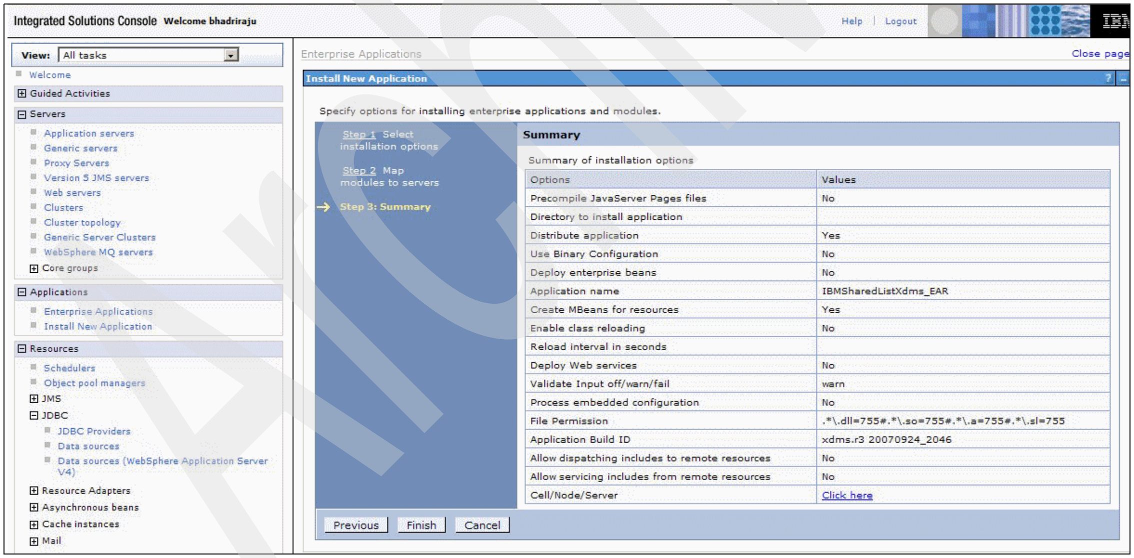


Figure 5-48 Shared List application deployment: Final panel in ISC

10. Click **Save** to save the master configuration, as shown in Figure 5-49 on page 194.

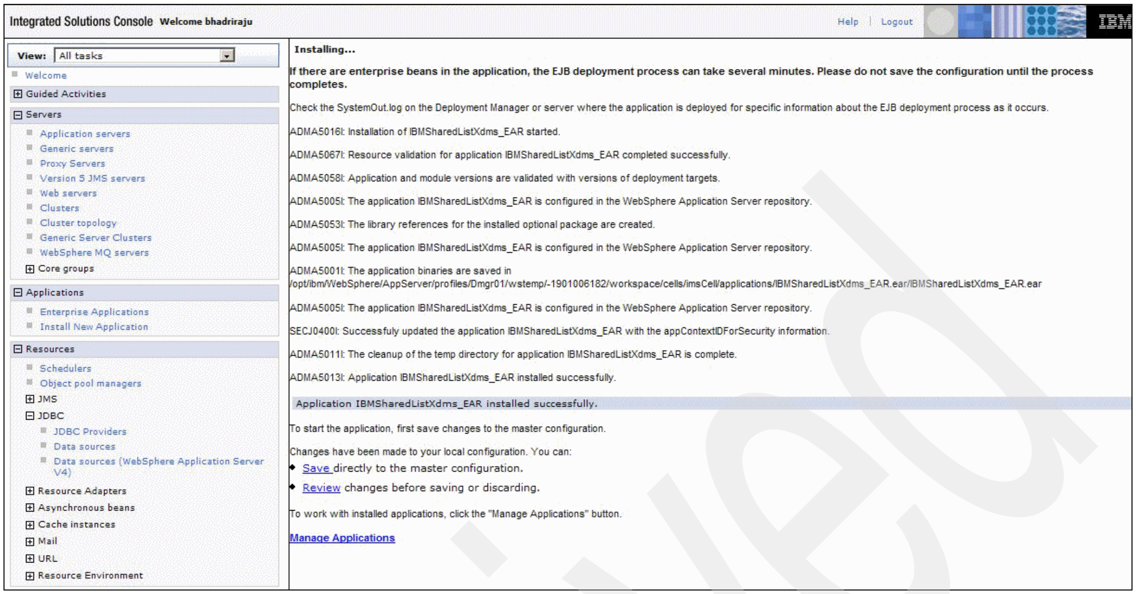


Figure 5-49 Shared List application deployment: Final panel in ISC

5.7.1 Configuring the Shared List XDMS resource provider

To configure the Shared List XDMS resource provider:

1. Click **Resources Resource Environment** → **Resource Environment Providers**.
2. Select the **Cluster=SharedListCluster** scope from the pull-down menu.
3. Click the Resource Provider Environment for the global ibm-xdms properties, and set the superadmin and xcapRoot, as shown in Figure 5-50 on page 195:
   a. Click **ibm-xdms**.
   b. Under Additional Properties, click **Custom Properties**, and you will see the following properties:
      - superAdminUser - A user configured with the super-admin role
      - superAdminPassword- Password for the user who is defined as the superAdminUser

        Change the superAdminPassword to be `password`.
      - xcapRoot - The XCAP Root of the Aggregation Proxy, if it is installed or the XCAP Root of the local server
   c. Update the value of superAdminPassword to be <**password>**, and accept the default values for the other two properties.

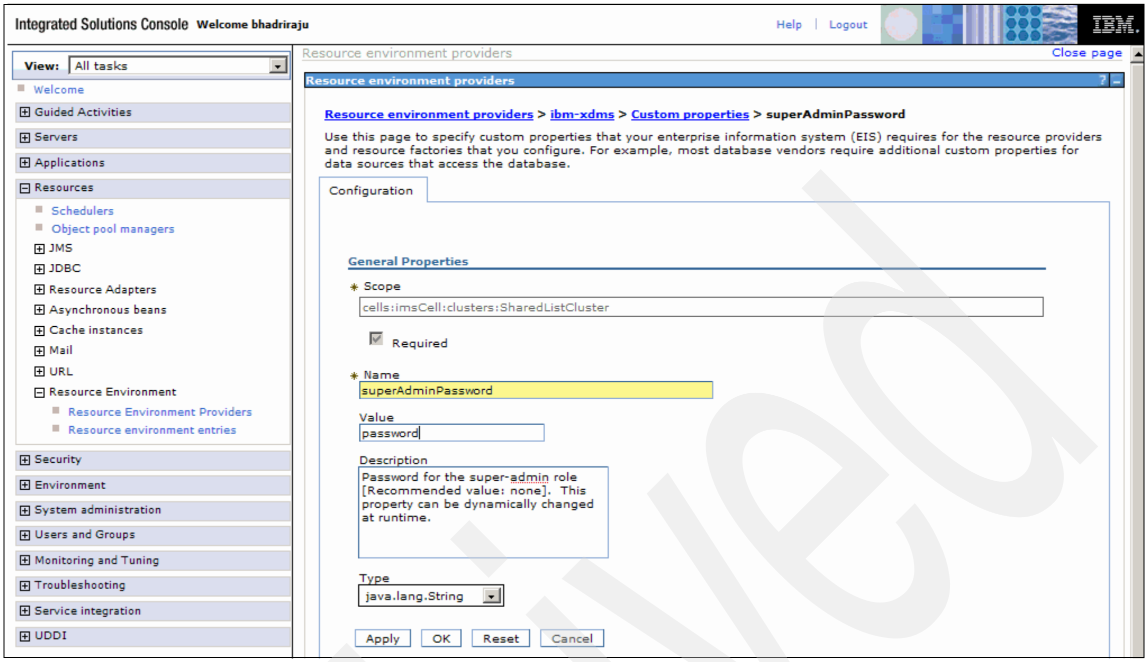


Figure 5-50 SharedList application resource environment provider configuration

4. If you have not kept the defaults for database creation for XDMS, perform this configuration step. Go to the Resource Provider Environment for each AUID, and configure the data source and table name:

   a. Click an AUID, such as resource-lists, rls-services, com.ibm.resource-lists-acls, or com.ibm.rls-services-acls.

   > **Note:** At a minimum, these two properties, db!default!datasource and db!default!table, must match the properties that are gathered at the end of the Creating the database section. The properties reference the WebSphere AUID variables for data source and data table.

   b. Start the ShareListCluster, and make sure it starts all the servers and the applications in the cluster.

5.8 Deploying the Presence Rules application

In this section, we cover the deployment of Presence Rules enterprise application to the Presence Rules Cluster. Perform the following steps to deploy the Presence Rules application:

1. Launch the Integrated Solutions Console (ISC), and Select **Application** → **Enterprise Applications**, as shown in Figure 5-51.

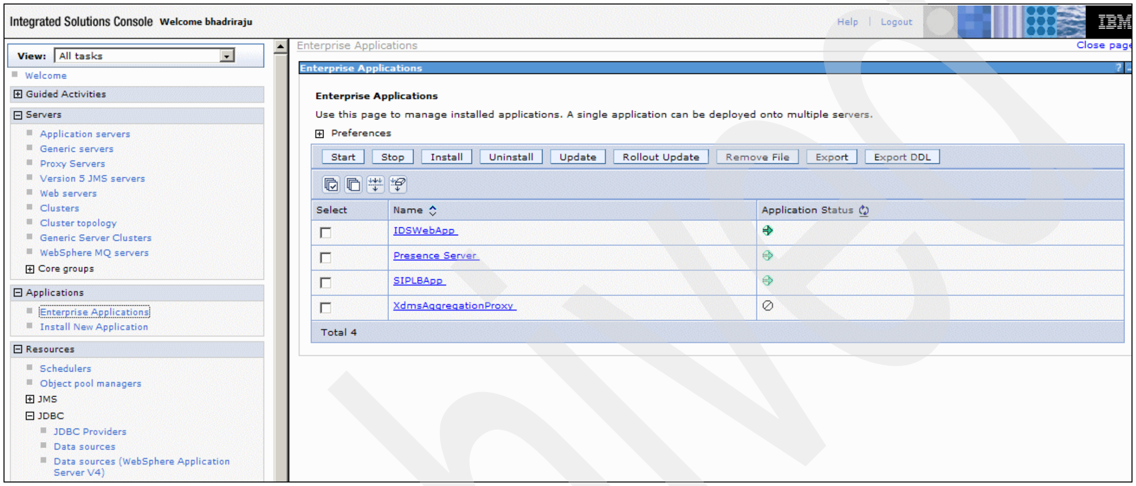


Figure 5-51 Presence Rules application deployment: Enterprise Applications panel in ISC

2. Click the **Install** button, and select IBMPresenceRulesXdms.ear file from the Local or Remote file system by clicking the **Browse** button, as shown in Figure 5-52 on page 197.
3. Click **Next**.

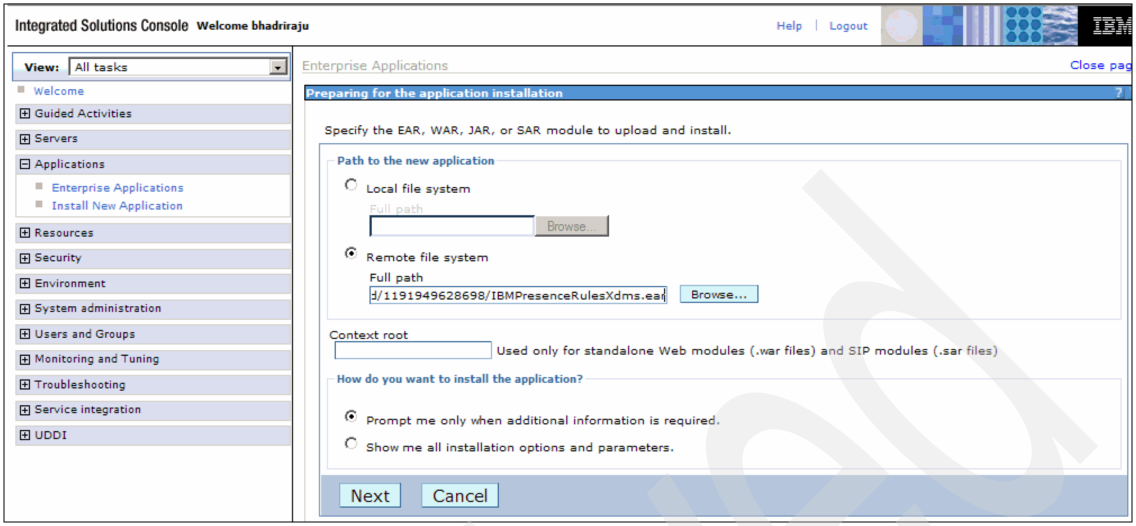


Figure 5-52 PresenceRules application deployment: Preparing for the application installation in ISC

4. In the Select Install Options pane, accept the defaults, as shown in Figure 5-53 on page 198, and click **Next**.

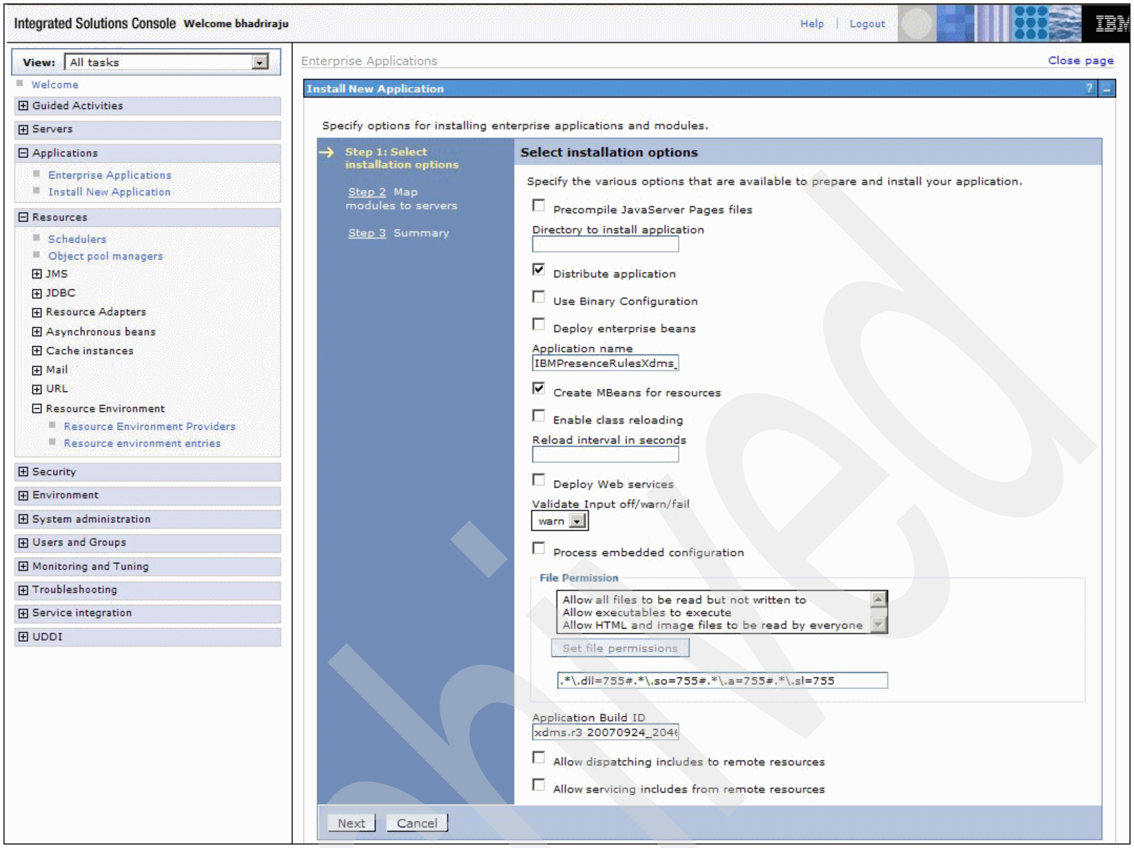


Figure 5-53 Presence Rules application deployment: Select Install Options pane in ISC

5. In the Map Modules to Servers panel, check all of the modules from the list, select the **PresenceRulesCluster** from the **Clusters and Servers** list, click **Apply**, as shown in Figure 5-54 on page 199.
6. Click **Next**, which deploys the Presence Rules application to the PresenceRulesCluster.

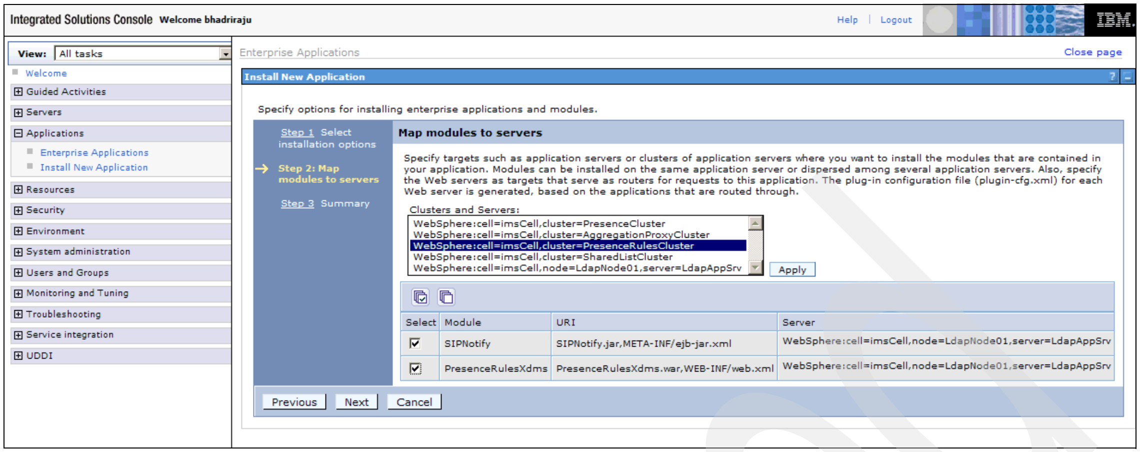


Figure 5-54 Presence Rules application deployment: Map modules to servers pane in ISC

7. On the last panel, click **Finish** to complete the deployment of the Shared List application, as shown in Figure 5-55.

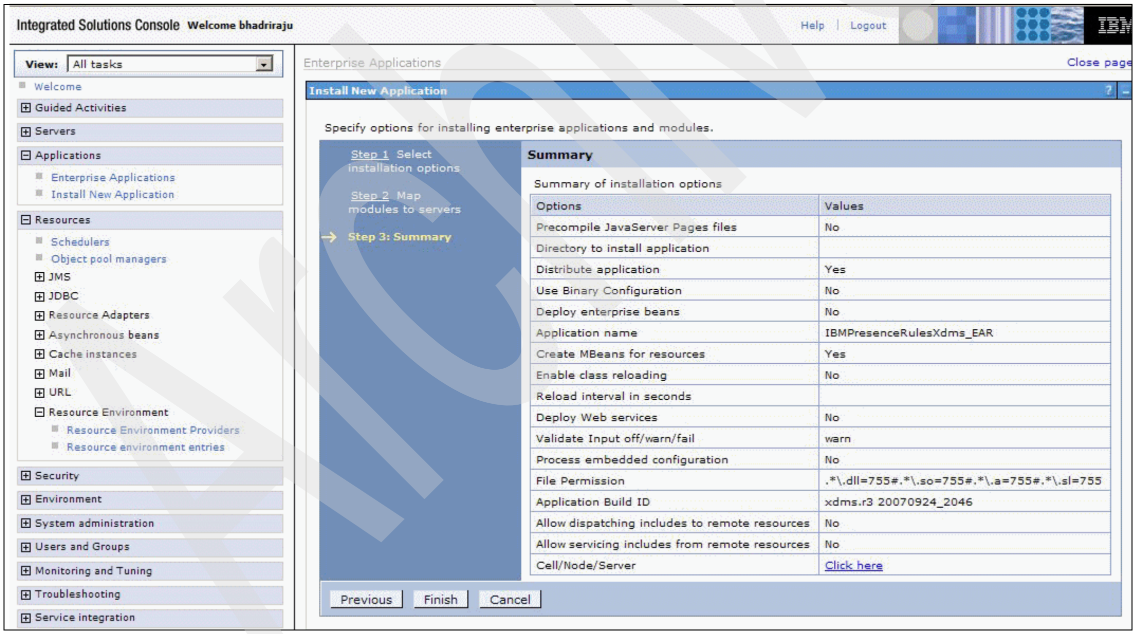


Figure 5-55 Presence Rules application deployment: Final panel in ISC

8. Click **Save** to save the master configuration in the final application deployment panel of the ISC, as shown in Figure 5-56 on page 200.

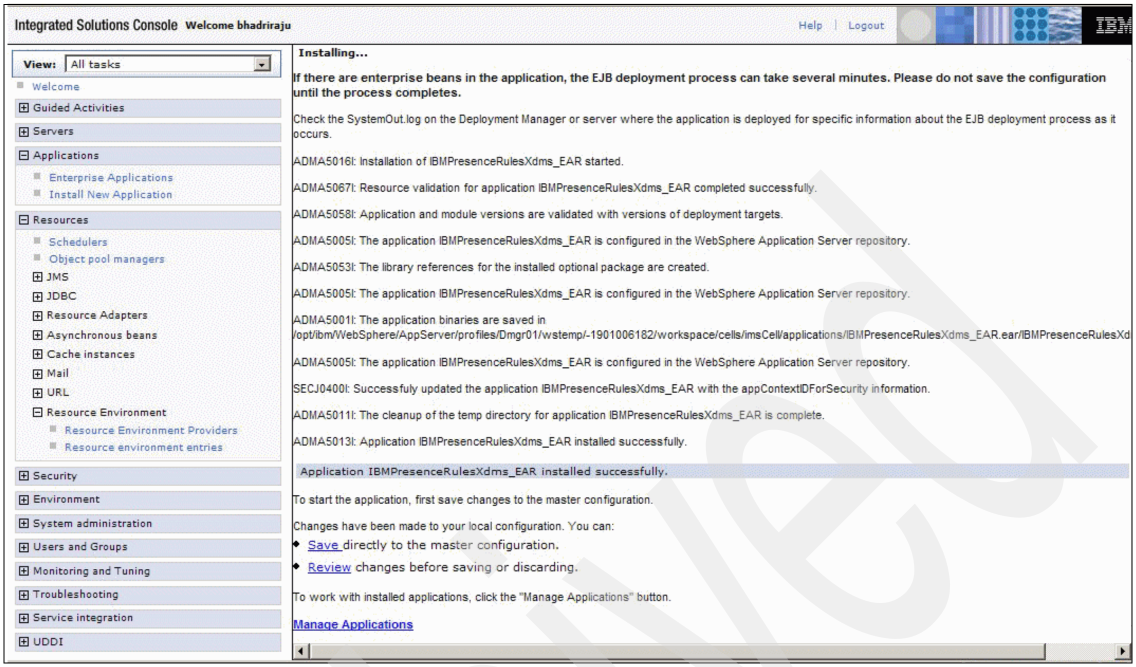


Figure 5-56 Presence Rules application deployment: Final panel in ISC

5.8.1 Configuring the Presence Rules XDMS resource providers

If you did not use the default configurations, you must configure the resource provider for the Presence Rules XDMS:

1. In the Integrated Solutions Console, click **Resources** → **Resource Environment** → **Resource Environment Providers**.
2. Select the **Cluster=PresenceRulesCluster** scope in the pull-down menu.
3. Click the **Resource Provider Environment** for the global ibm-xdms properties, and set the superadmin and xcapRoot:
   a. Click **ibm-xdms**.
   b. Under Additional Properties, click **Custom Properties**, and the following properties are displayed:
      - superAdminUser - A user configured with the super-admin role
      - superAdminPassword- Password for the user defined as superAdminUser

> **Note:** Change the superAdminPassword to be `password`.

   - xcapRoot - The XCAP Root of the Aggregation Proxy if installed or the XCAP Root of the local server

   c. Update the value of superAdminPassword to be **password**, and accept the default values for the other two properties.

4. If you did not accept the defaults, configure your custom properties here. Go to the Resource Provider Environment for each AUID, and configure the data source and table name:

   a. Click an AUID, such as org.openmobilealliance.pres-rules or com.ibm.pres-rules-acls.

   b. Under additional properties, click **Custom Properties**.

> **Note:** At a minimum, these two properties, db!default!datasource and db!default!table, must match the properties that are gathered at the end of the Creating the database section. The properties reference the WebSphere AUID variables for the data source and data table.

This completes the installation of the XDMS component for the base scenario environment.

6

Aggregation proxy installation

In this chapter, we describe the steps to install and configure the Aggregation proxy in the base scenario environment infrastructure.

This chapter contains:

6.1 Preparing to install the Aggregation Proxy

WebSphere Application Server is loaded on the servers that are designated for the Aggregation Proxy. The designated IP addresses are 9.42.170.216 and 9.42.170.217. The WebSphere Application Server is prepared to work on RedHat Enterprise Server v4.5. Figure 6-1 shows the details of the WebSphere Application Server.

```
Name:           IBM Update Installer for WebSphere Software
Version:        6.1.0.11
Platform:       linux
Architecture:   ia32
Build Release:  WAS61.UPDI
Build Level:    fep0734.33
Build Date:     08/30/2007
```

Figure 6-1 WebSphere Application Server version details

Additional configuration prerequisites are:

- The node created for the Aggregation proxy is federated to Deployment Manager cell.
- The required version of WebSphere Application Server is 6.1.0.11. To verify the required information:
  a. In the navigation panel of the Integrated Solutions Console, click **Welcome** and the page is updated with the link to WebSphere Application Server.
  b. Click **WebSphere Application Server** to get further information.
- The user repositories must be configured.

> **Important:** Aggregation proxy is not configured for the scenario of shared XDMS situation, but for the objective of fulfilling the needs of simple XDMS and single source of client as a resource list enabler using HTTP. The configuration scope covers only the Basic Routing.

Figure 6-2 on page 205 provides an overview of the Aggregation Proxy installation.

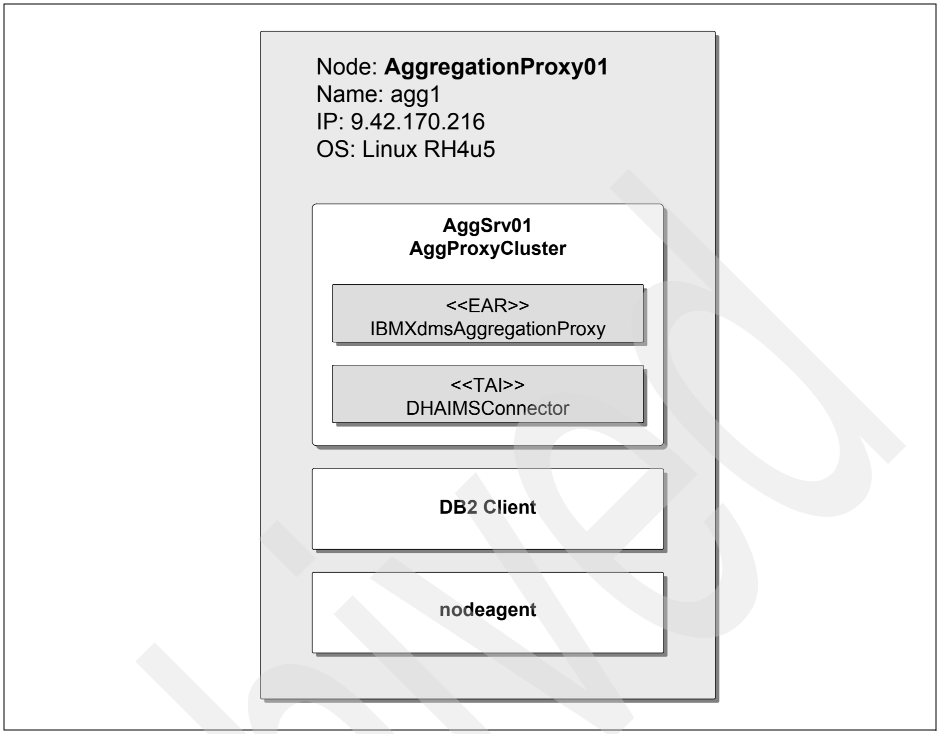


Figure 6-2 Overview of the Aggregation Proxy installation

6.2 Security configuration

Use the Deployment Manager security setting options to create role-based security. Access the Deployment Manager through http://9.42.170.220:9043/ibm/console

To get into the security settings:

1. In the navigation panel of the Integrated Solutions Console, click **Security**.
2. Click **Secure administration, applications, and Infrastructure**, and the page is updated.
3. Select the **Enable application security** option, as shown in Figure 6-3 on page 206.
4. Click **Web security** to enable the treeview on the available list.

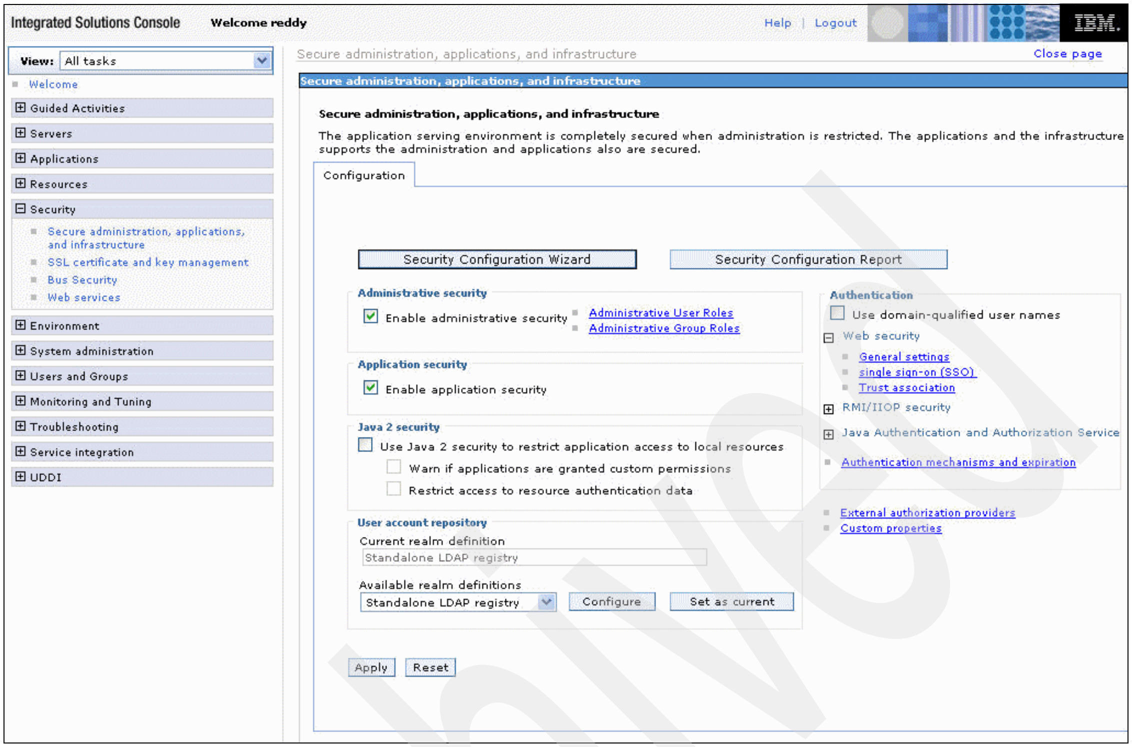


Figure 6-3 Select Web security

After you select the General settings, the page updates. Use the next steps to create the Web authentication:

1. Select the **Authenticate only when the URI is protected** option.
2. Select the **Use available authentication data when an unprotected URI is accessed** option, as shown in Figure 6-4 on page 207.
3. Click **Apply**, and then click **Save**, to save the changes to the master configuration.

This completes the general security definition configuration.

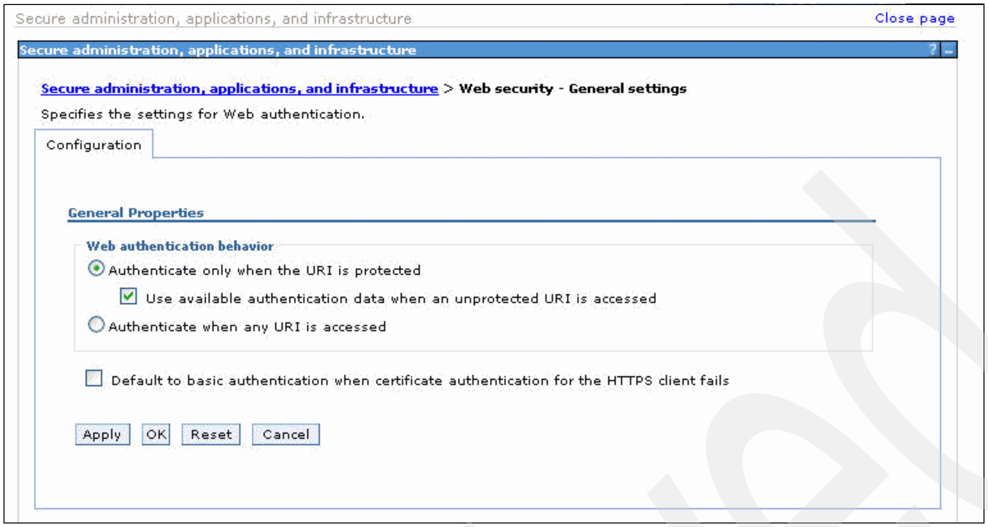


Figure 6-4 Web security - general settings

6.3 Installing the Aggregation Proxy interceptor

1. From the navigation panel in the Integrated Solution Console, click **Security**.
2. Click **Secure administration, applications, and Infrastructure**.
3. Click **Web security** to enable the treeview on the available list, as shown in Figure 6-5 on page 208.

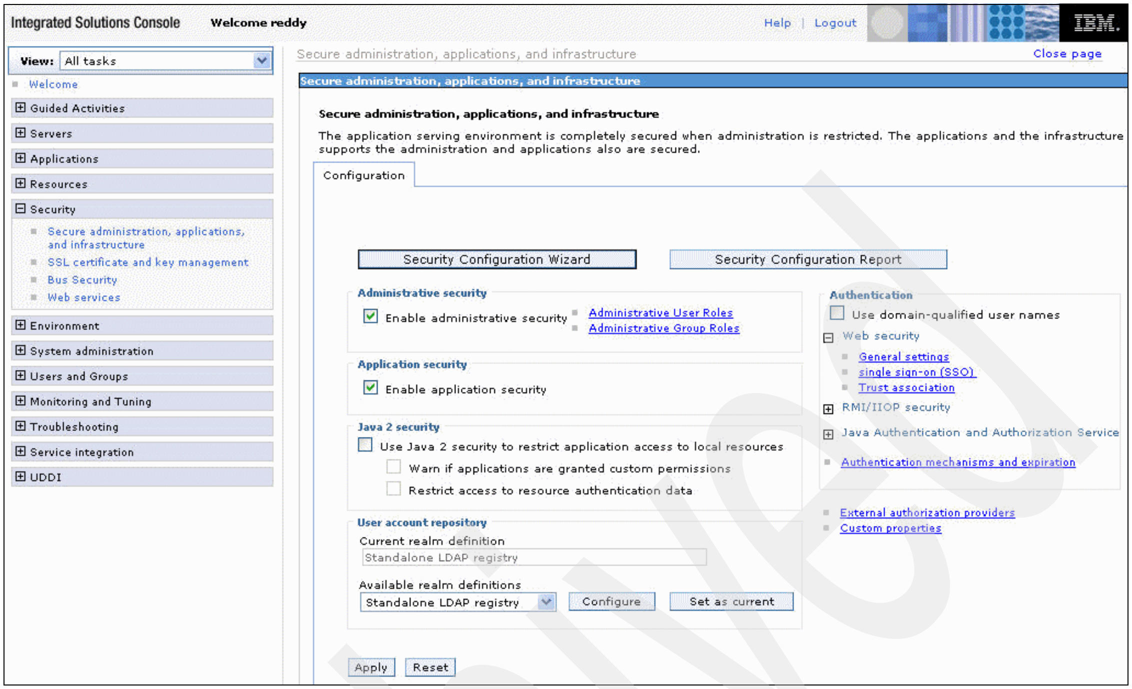


Figure 6-5 Secure administration, applications, and Infrastructure - configuration

4. Click **Trust association**, which updates the page for the Trust association configuration.
5. Under General Properties, select the **Enable the trust association** option, and under Additional Properties, click the Interceptors hyperlink, as shown in Figure 6-6 on page 209.
6. Click **Apply,** which saves the changes to the master configuration.

Figure 6-6 Trust association - general properties

7. From **Enable trust association**, repeat the steps, and click **Interceptors**, as shown in Figure 6-7.

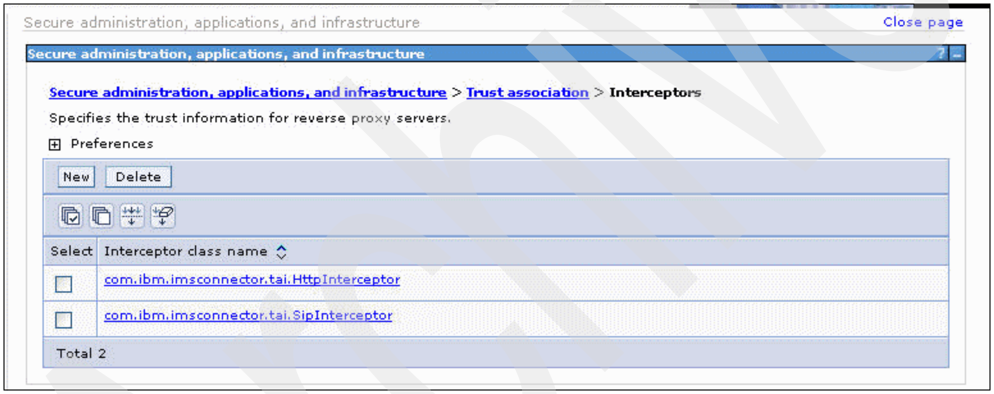


Figure 6-7 Interceptors

The page updates with the available interceptors. You are required to create HTTP interceptor. Next, we create a new interceptor, and add the new Interceptor com.ibm.glm.http.security.tai.HttpDigestTAI.

8. Click **New**.
9. In the Interceptor class name field, type `com.ibm.glm.http.security.tai.HttpDigestTAI`, and click **Apply**, as shown in Figure 6-8.

Figure 6-8 Create a new Interceptor

As shown in Figure 6-9 on page 210, you just created a new entry of the specified class name as an additional interceptor to the list.

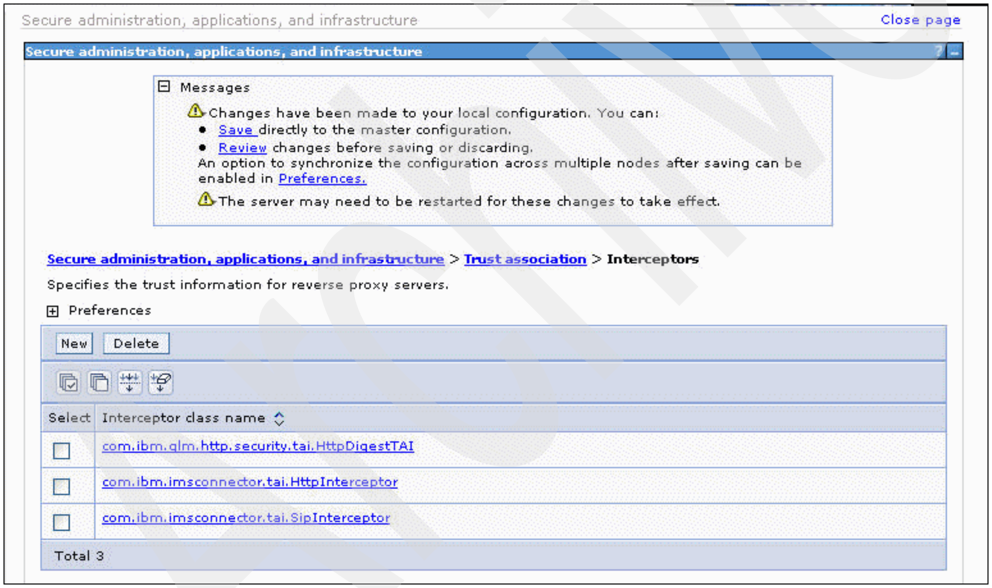


Figure 6-9 List showing new Interceptor

10. Select the **com.ibm.glm.http.security.tai.HttpDigestTAI** option to define the custom properties.

11. Click the **com.ibm.glm.http.security.tai.HttpDigestTAI** link to open the page for creating new custom properties, as shown in Figure 6-10 on page 211.

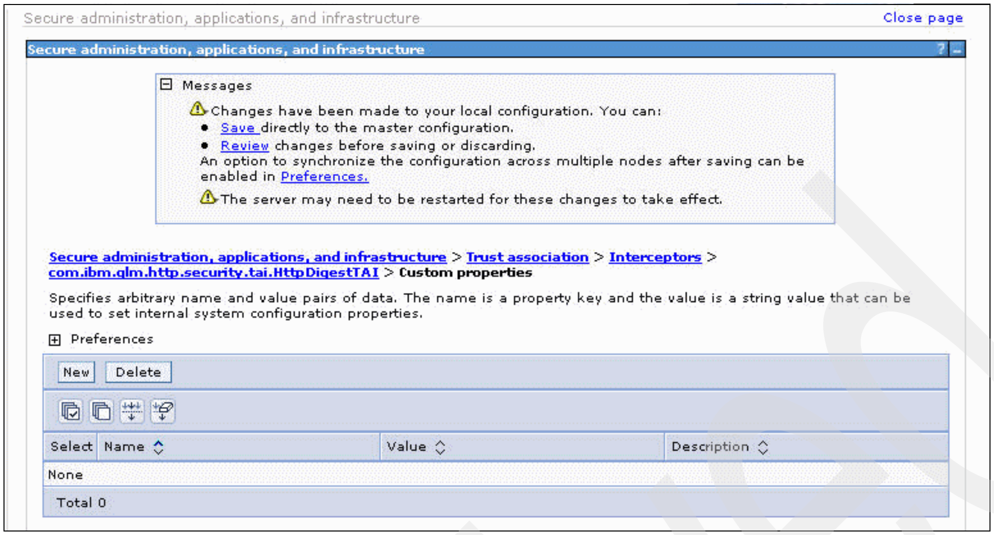


Figure 6-10 HttpDigestTAI custom properties

12. Click **New**, and enter the following parameters, as shown in Figure 6-11 on page 212:
    a. In the Name field, type `LdapAuth`.
    b. In the Value field, type `cn=root`.
13. Click **OK** to update the custom properties for LdapAuth.

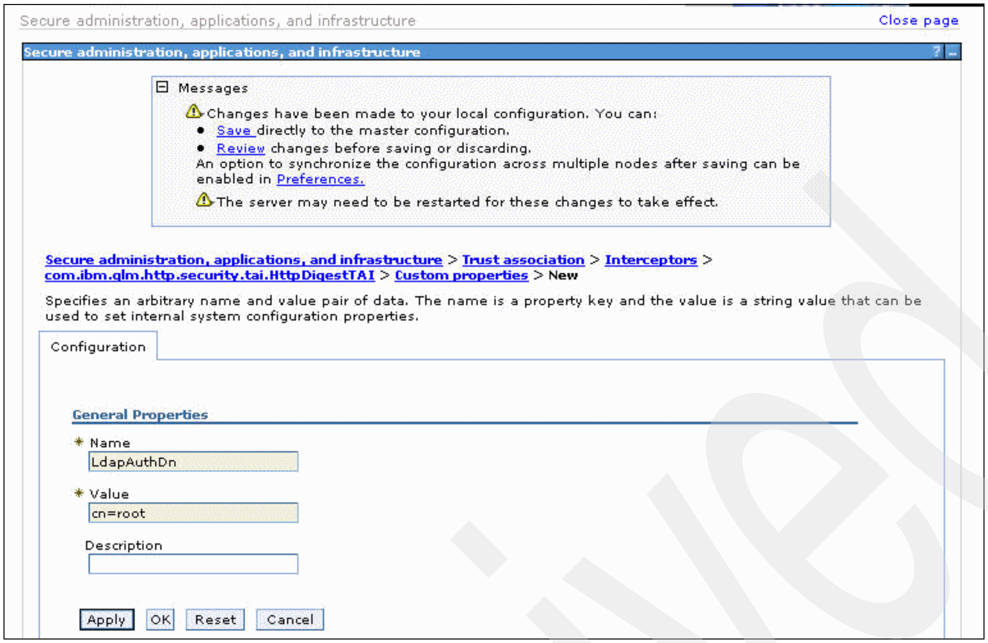


Figure 6-11 LdapAuthDn custom property

Now, we create and add new custom properties for LdapAuthPw.

14.Click **New**, and enter the following parameters, as shown in Figure 6-12 on page 213:

a. In the Name field, type `LdapAuthPw`.

b. In the Value field, type the actual password.

15.Click **OK**.

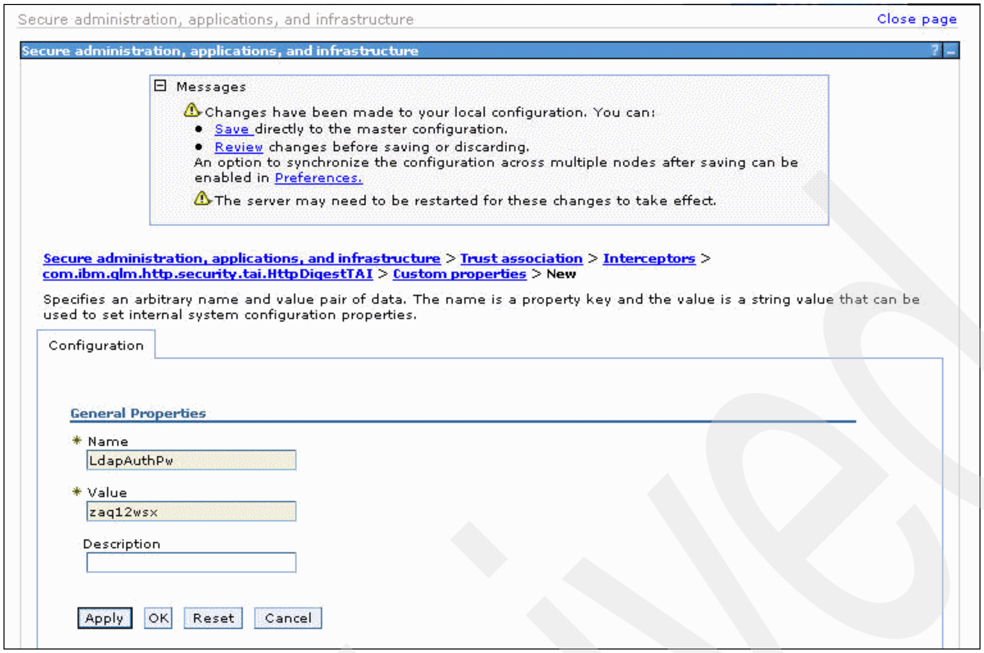


Figure 6-12 LdapAuthPw custom property

Now we create and add a new custom property for LdapBaseDn.

16. Click **New**, and enter the following parameters, as shown in Figure 6-13 on page 214:

    a. In the Name field, type `LdapBaseDn`.

    b. In the Value field, type `cn=itso`.

17. Click **OK**.

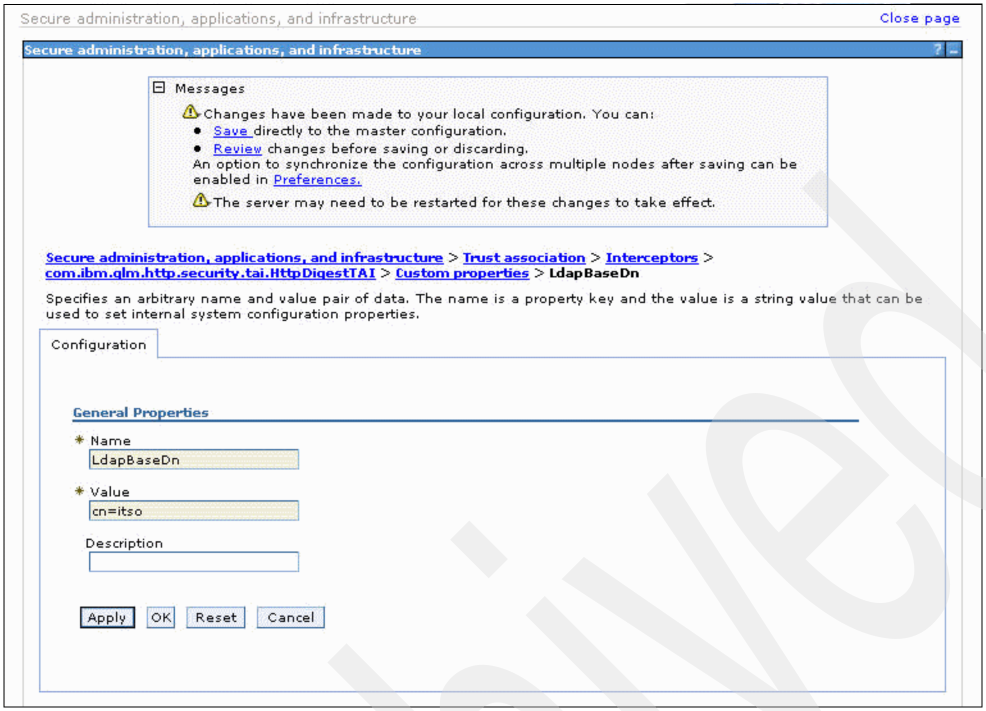


Figure 6-13 LdapBaseDn custom property

Create and add a new custom property for LdapHost.

18. Click **New**, and enter the following parameters, as shown in Figure 6-14 on page 215:

    a. In the Name field, type `LdapHost`.

    b. In the Value field, type `ldap.itso.ral.ibm.com`.

19. Click **OK**.

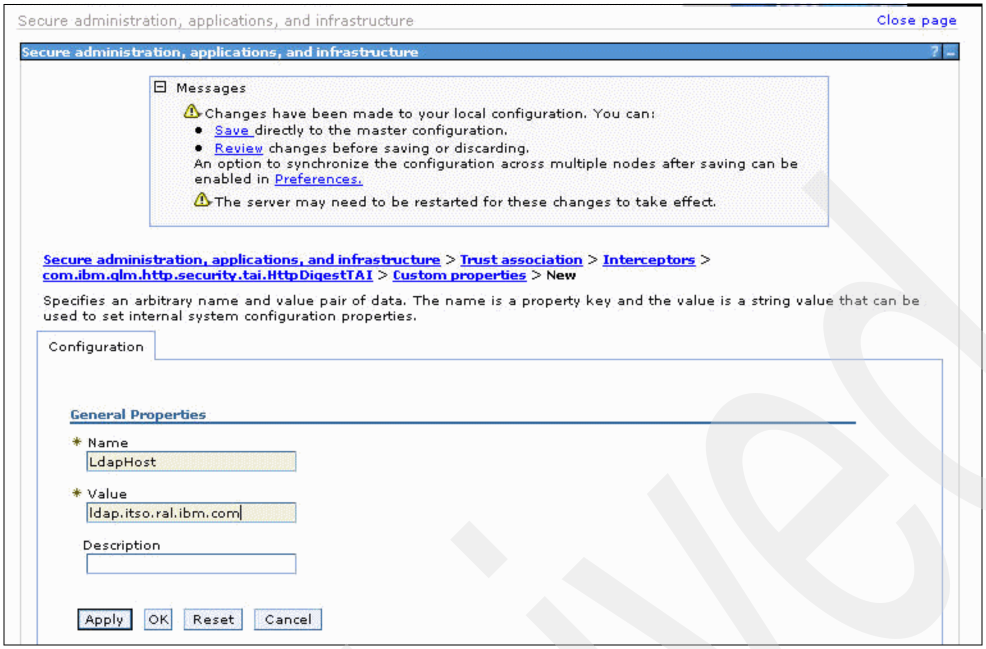


Figure 6-14 LdapHost custom property

Create and add the new custom property, LdapPort.

20.Click **New**, and enter the following parameters, as shown in Figure 6-15 on page 216:

a. In the Name field, type `LdapPort`.

b. In the Value field, type `389`.

21.Click **OK**.

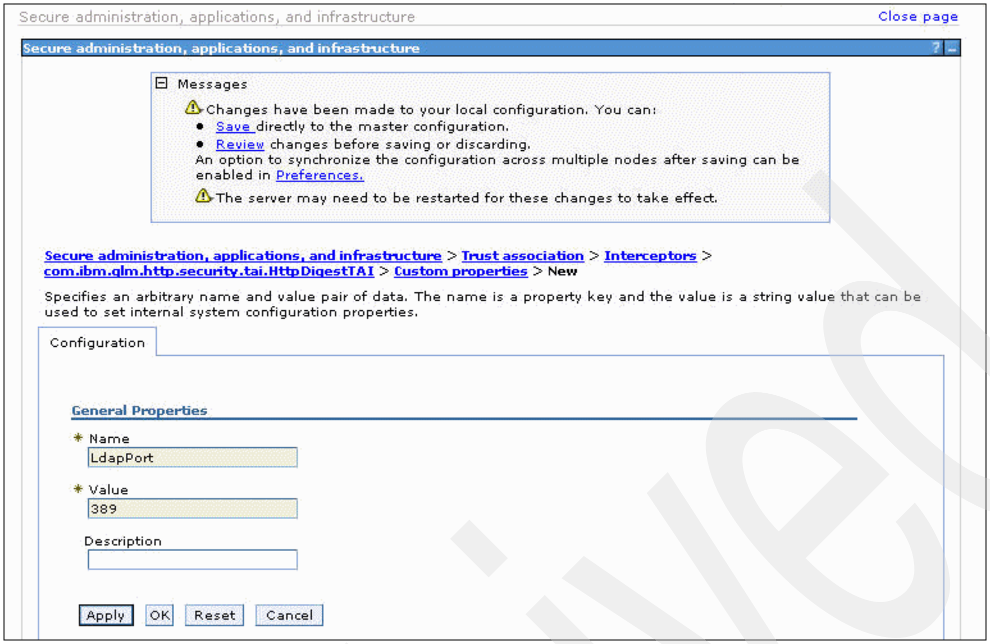


Figure 6-15 LdapPort custom property

Create and add the new custom property, LdapUserFilter.

22. Click **New**, and enter the following parameters, as shown in Figure 6-16 on page 217:

   a. In the Name field, type `LdapUserFilter`.

   b. In the Value field, type `(&(|(cn=%v)(uid=%v))(objectclass=person))`.

23. Click **OK**.

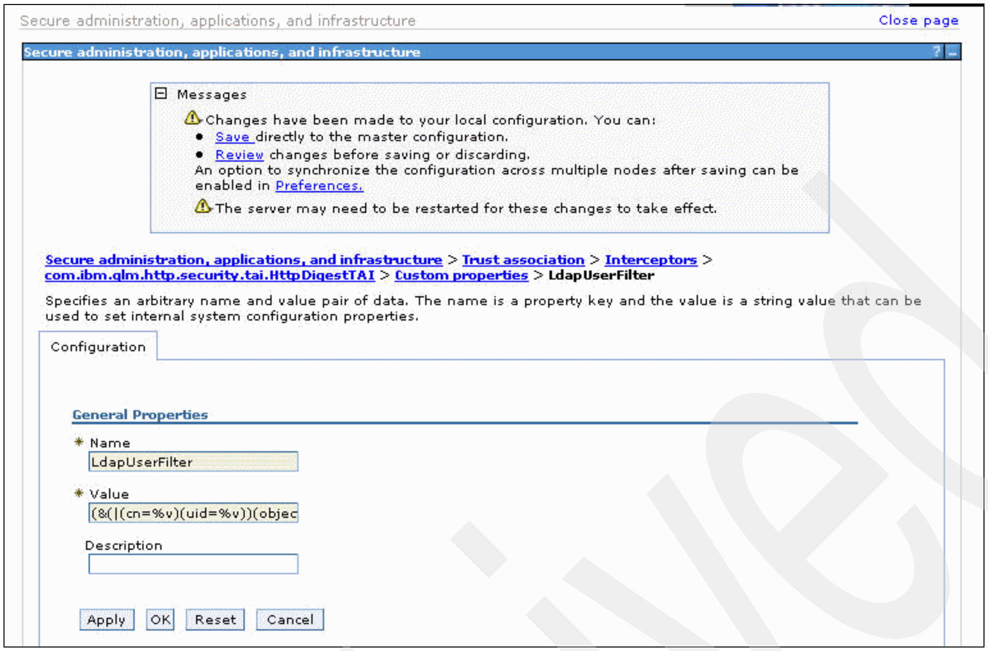


Figure 6-16 LdapUserFilter custom property

Create and add the new custom property, RetryCount.

24. Click **New**, and enter the following parameters, as shown in Figure 6-17 on page 218:

   a. In the Name field, type `RetryCount`.

   b. In the Value field, type `3`.

25. Click **OK**.

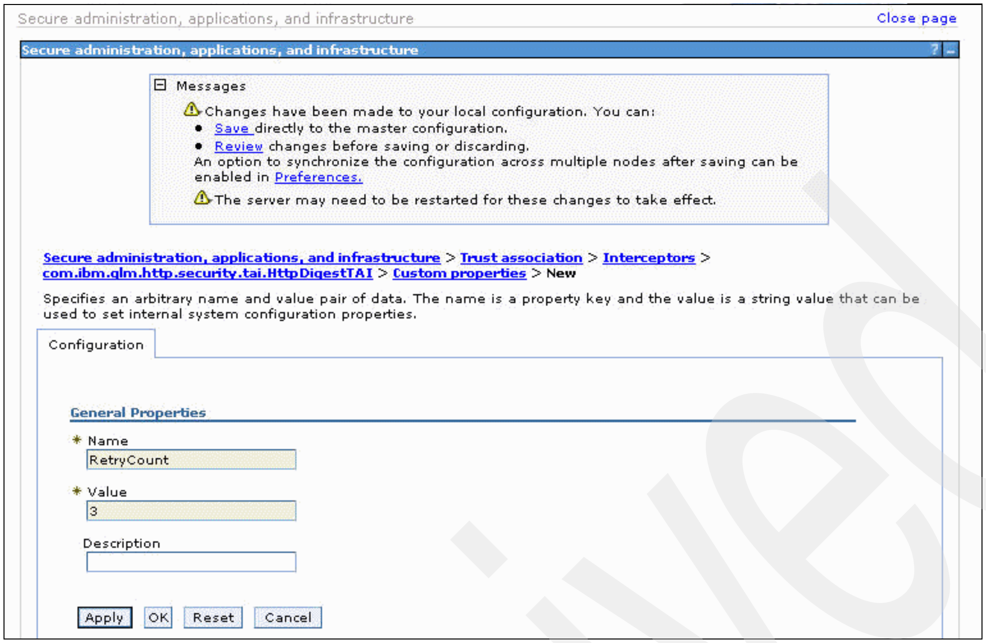


Figure 6-17 RetryCount custom property

Create and add the new custom property, auth.int.enable.

26. Click **New**, and enter the following parameters, as shown in Figure 6-18 on page 219:

   a. In the Name field, type `auth.int.enable`.

   b. In the Value field, type `false`.

27. Click **OK**.

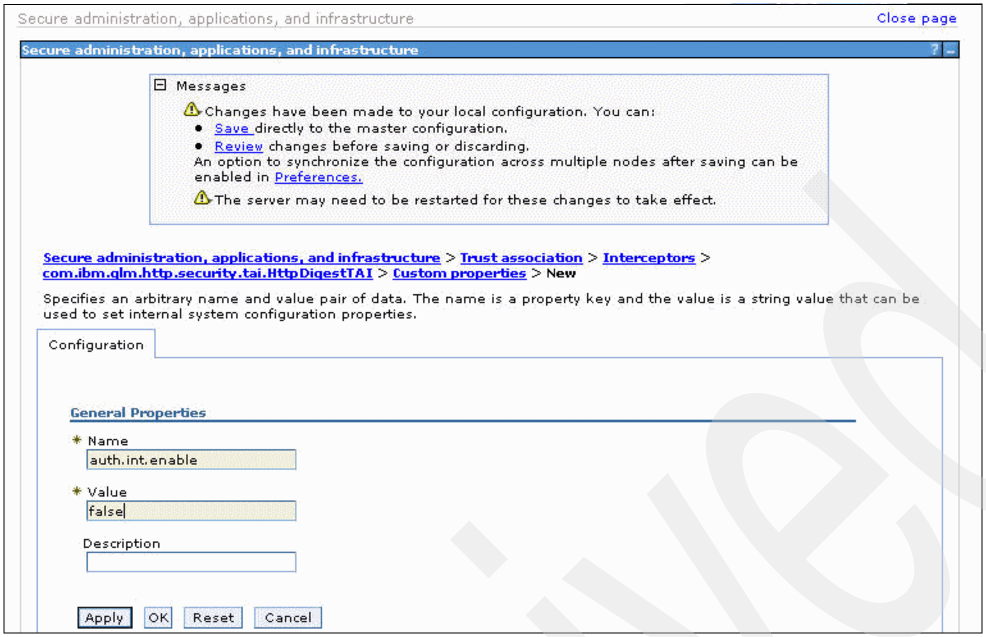


Figure 6-18 uth.int.enable custom property

28. Click **Save**, to save the changes to the master configuration, as shown in Figure 6-19 on page 220.

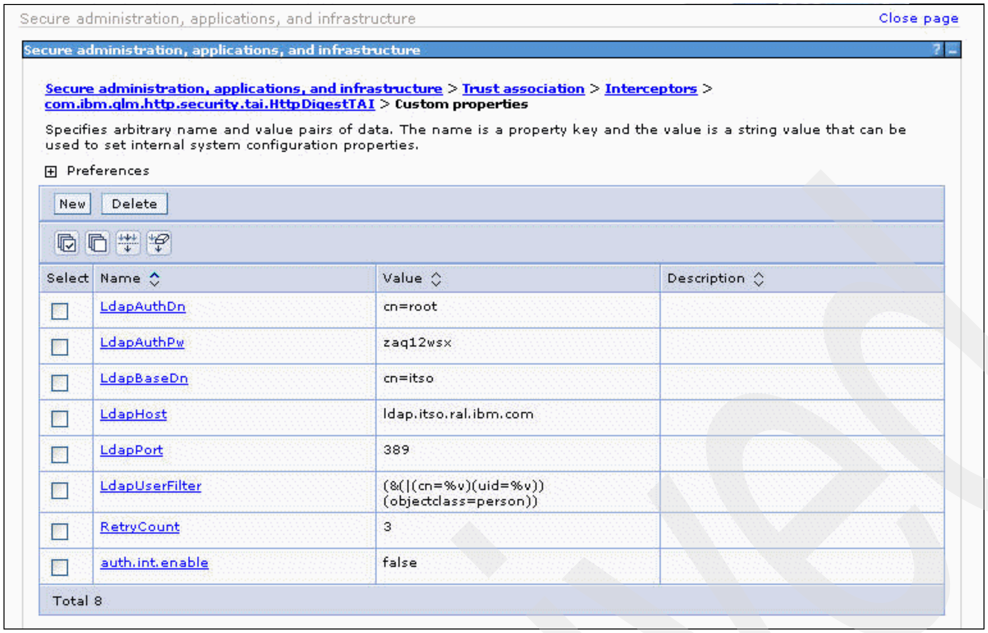


Figure 6-19 List of custom properties

6.4 Creating the Aggregation Proxy Cluster

To create the Aggregation Proxy Cluster:

1. In the Integrated Solutions Console navigation panel, click **Servers** → **Clusters**. Figure 6-20 is displayed.
2. Click **New**.

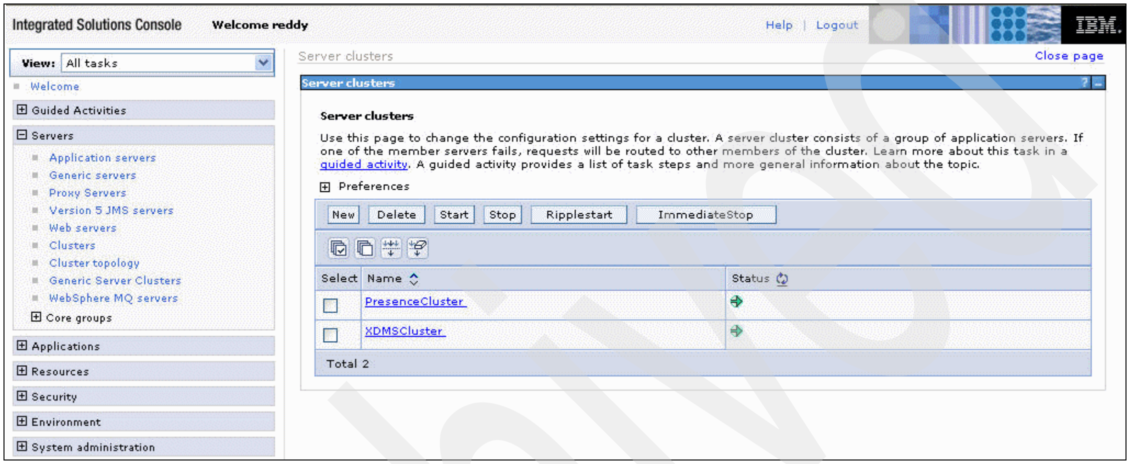


Figure 6-20 Servers selected

3. On the new cluster page, enter the following parameters, as shown in Figure 6-21:

   a. In the Cluster name field, type `AggregationProxyCluster`.

   b. Select the **Prefer local** option, and then click **Next**.

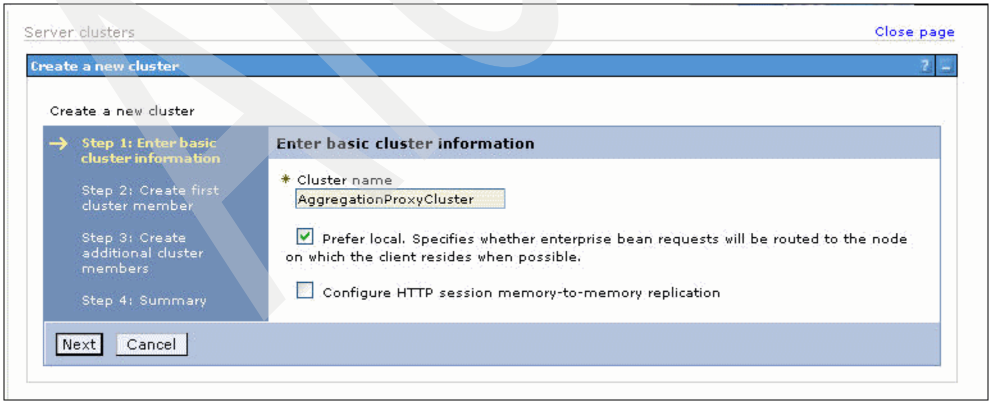


Figure 6-21 Cluster basic information

4. On the Create first cluster member page, Figure 6-22, enter the following parameters:
   a. In the Member name field, type `AGGProxySvr01`.
   b. In the Select node field, select **AggNode01(ND 6.1.0.11)**.
   c. In the Weight field, type `2`.
   d. Select the **Generate unique HTTP ports** option.
   e. From the Create the member using an Application Server template pull-down menu, select **default**.
5. Click **Next**.

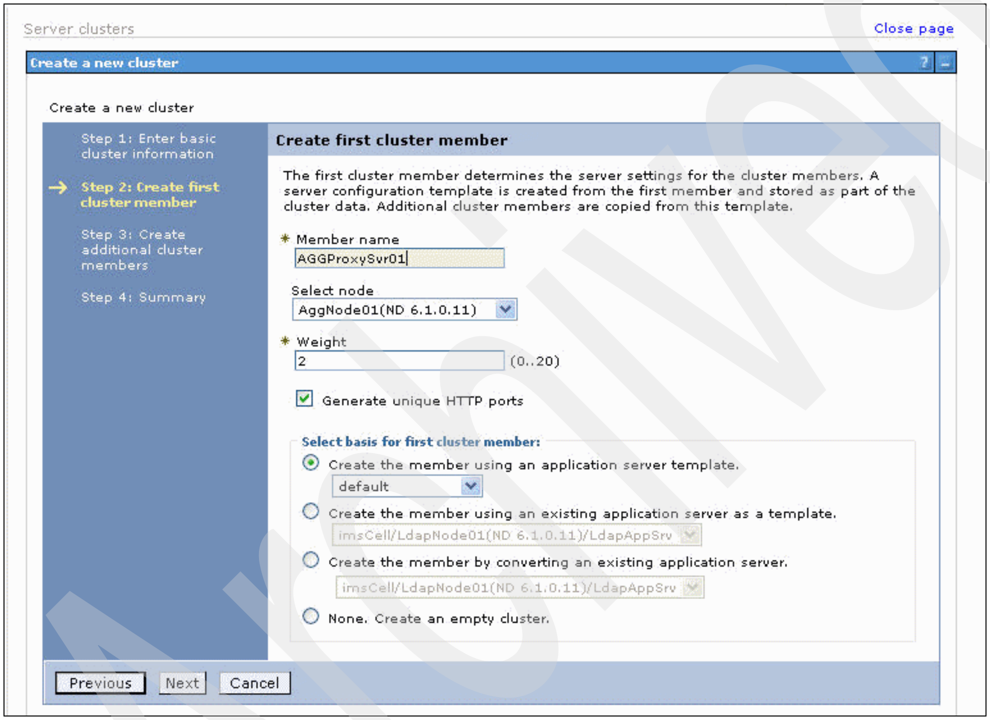


Figure 6-22 Create first cluster member

6. There are no additional cluster members in the base scenario environment. On the Create additional cluster members page, Figure 6-23 on page 223, click **Next**.

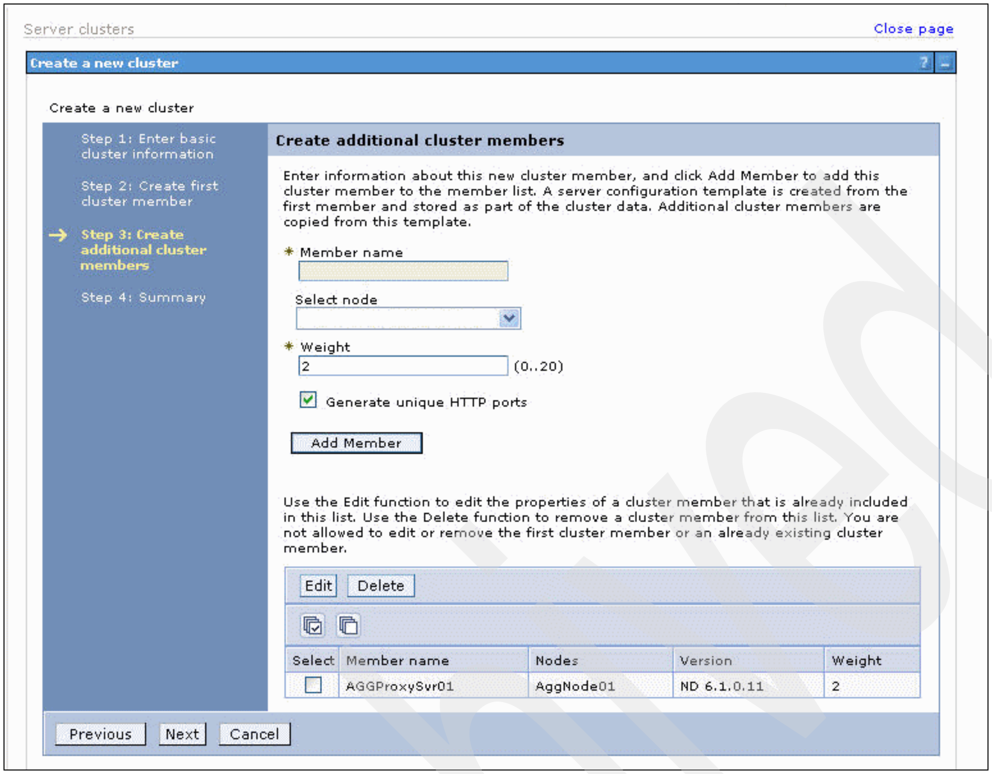


Figure 6-23 Create additional cluster members page

7. On the Create a New Cluster - Summary page, Figure 6-24 on page 224, click **Finish**.

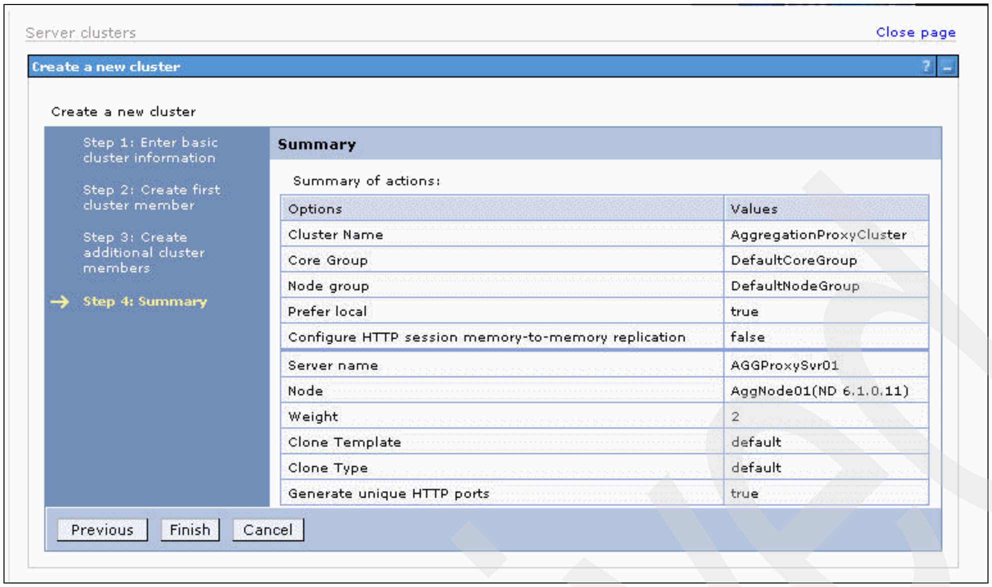


Figure 6-24 Create a new clusters summary page

8. Click **Save**, to save changes to the master configuration, as shown in Figure 6-25.

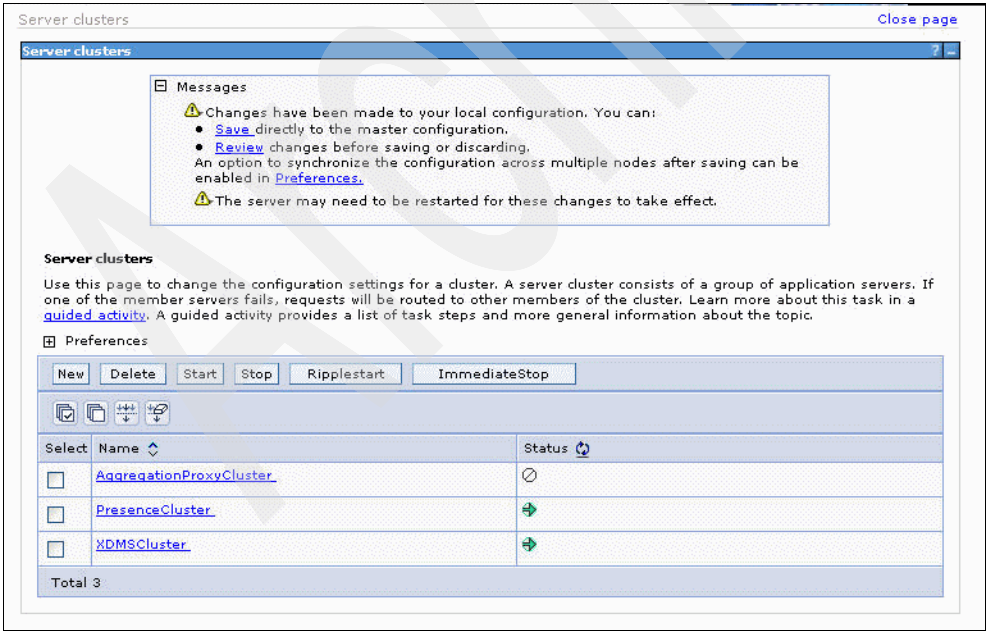


Figure 6-25 Server clusters page showing the AggregationProxyCluster

6.5 Verifying the AggregationProxyCluster

To verify the AggregationProxyCluster cluster that we created in the previous section:

1. In the Clusters view, select the **AggregationProxyCluster** option, as shown in Figure 6-26.
2. Click **Start** to start the cluster.

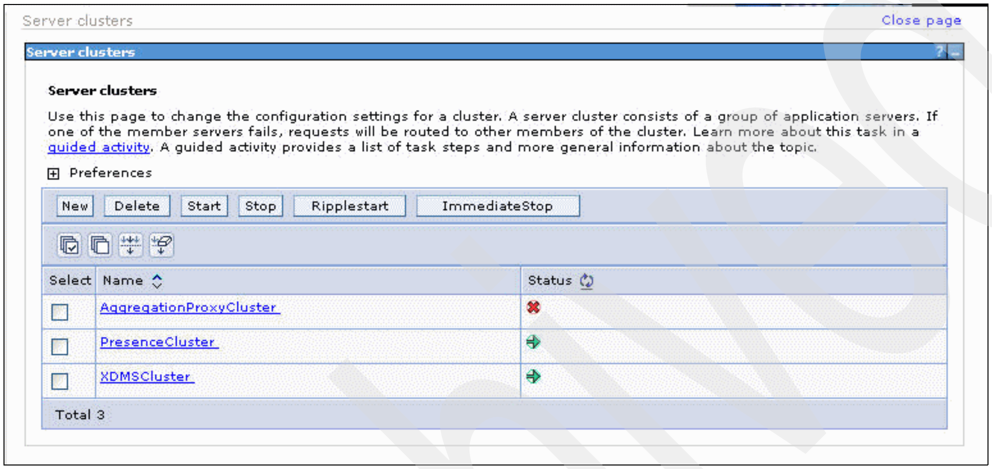


Figure 6-26 Session Stopped status

After started, the Integrated Solutions Console refreshes the page, and displays the status of this cluster as started, which we show in Figure 6-27.

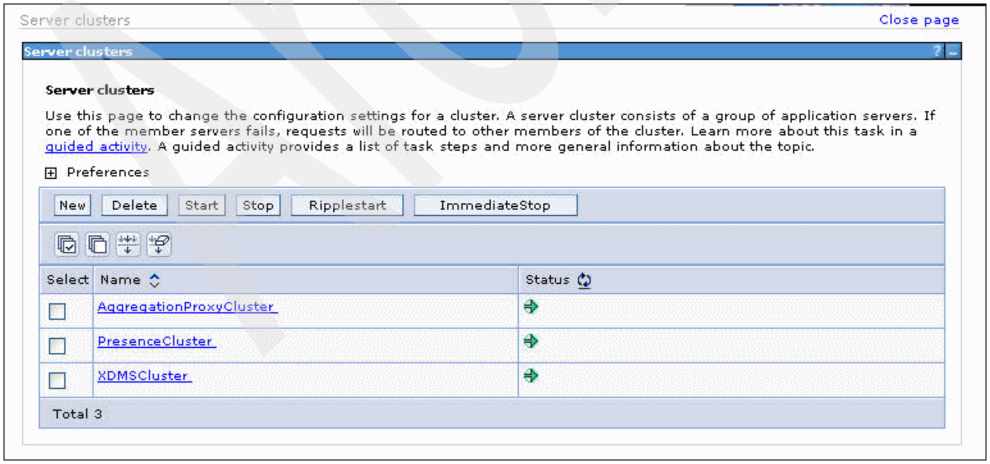


Figure 6-27 Shows the AggregationProxy is started

6.6 Deploying the Aggregation Proxy application

The process to deploy the Aggregation Proxy enterprise application is:

1. Install the XdmsAggregationProxy application.
2. Configure the XdmsAggregationProxy.
3. Verify Resource environment entries.

6.6.1 Installing the XdmsAggregationProxy application

To install the XdmsAggregationProxy application:

1. Launch the Integrated Solutions Console (ISC), and Select **Application** → **Enterprise Applications**. The page is refreshed with the available list of applications.

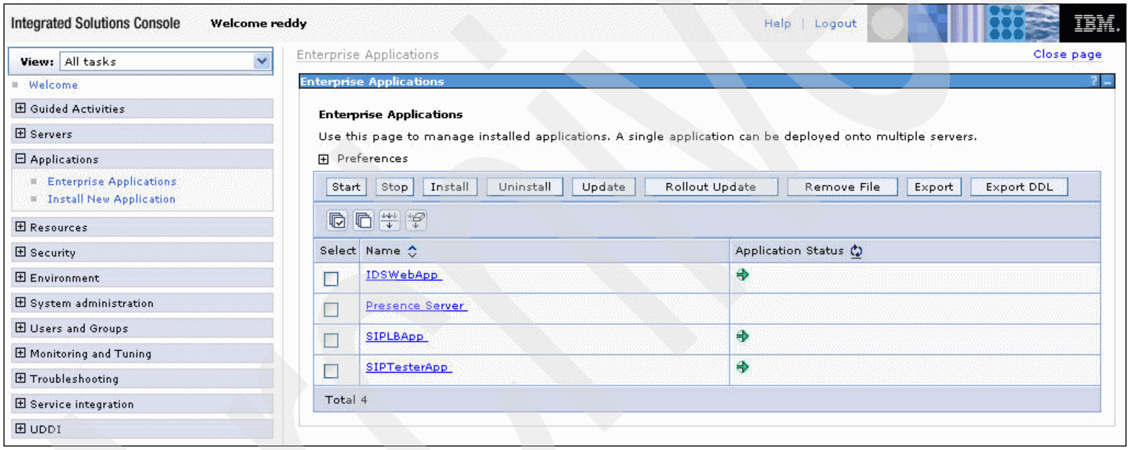


Figure 6-28 Enterprise Applications

2. Install from the host, and locate the EAR file by browsing the directory. Select the file from the local file system.

> **Tip:** Preferably, the EAR file is loaded from the server where the Deployment Manager is available.

3. Click **Install** to install a new application.
4. On the file system, select the location of the EAR file, as shown in Figure 6-29 on page 227.

5. Select the **Remote file system** option, as shown in Figure 6-30 on page 228, and click the **Browse** button to locate the node from where the file can be loaded.
6. Click **LdapNode01**, which is the local system for the Deployment Manager.

> **Note:** Make sure that the XDMS binaries are loaded on to this system before you attempt to get the EAR file on this server.

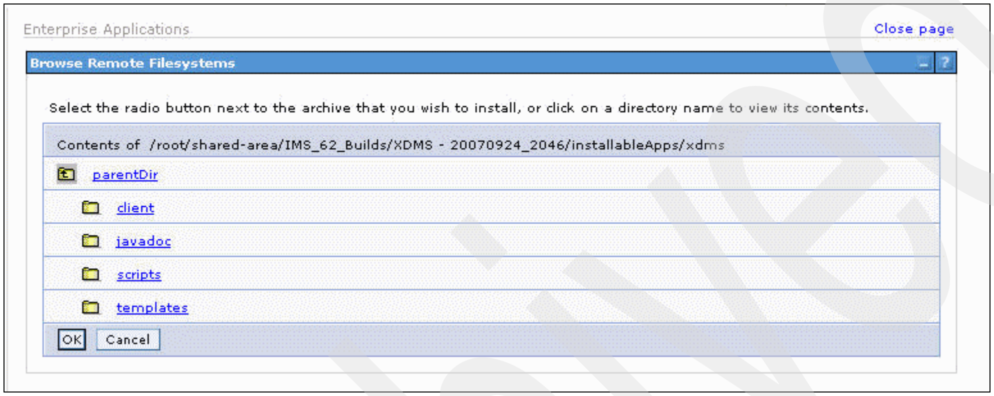


Figure 6-29 The required EAR

7. Ensure that the Full path is available with the loadable EAR file, and click **Next**, as shown in Figure 6-30 on page 228.

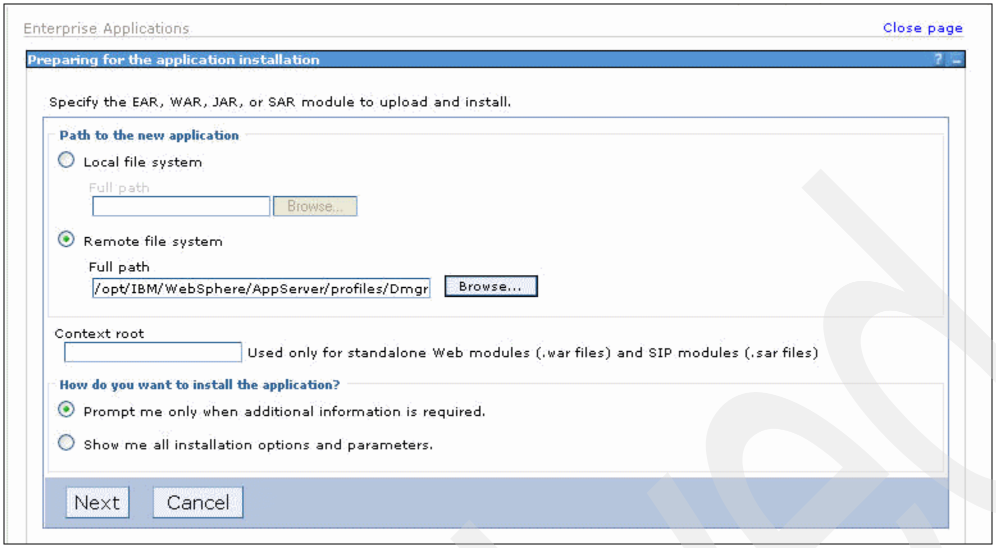


Figure 6-30 Remote application EAR

8. On the Select installation options page, Figure 6-31 on page 229, perform the following tasks:
   a. In the Application name field, type **XdmsAggregationProxy**.
   b. Accept the defaults for the remaining fields on the page.
9. Click **Next**.

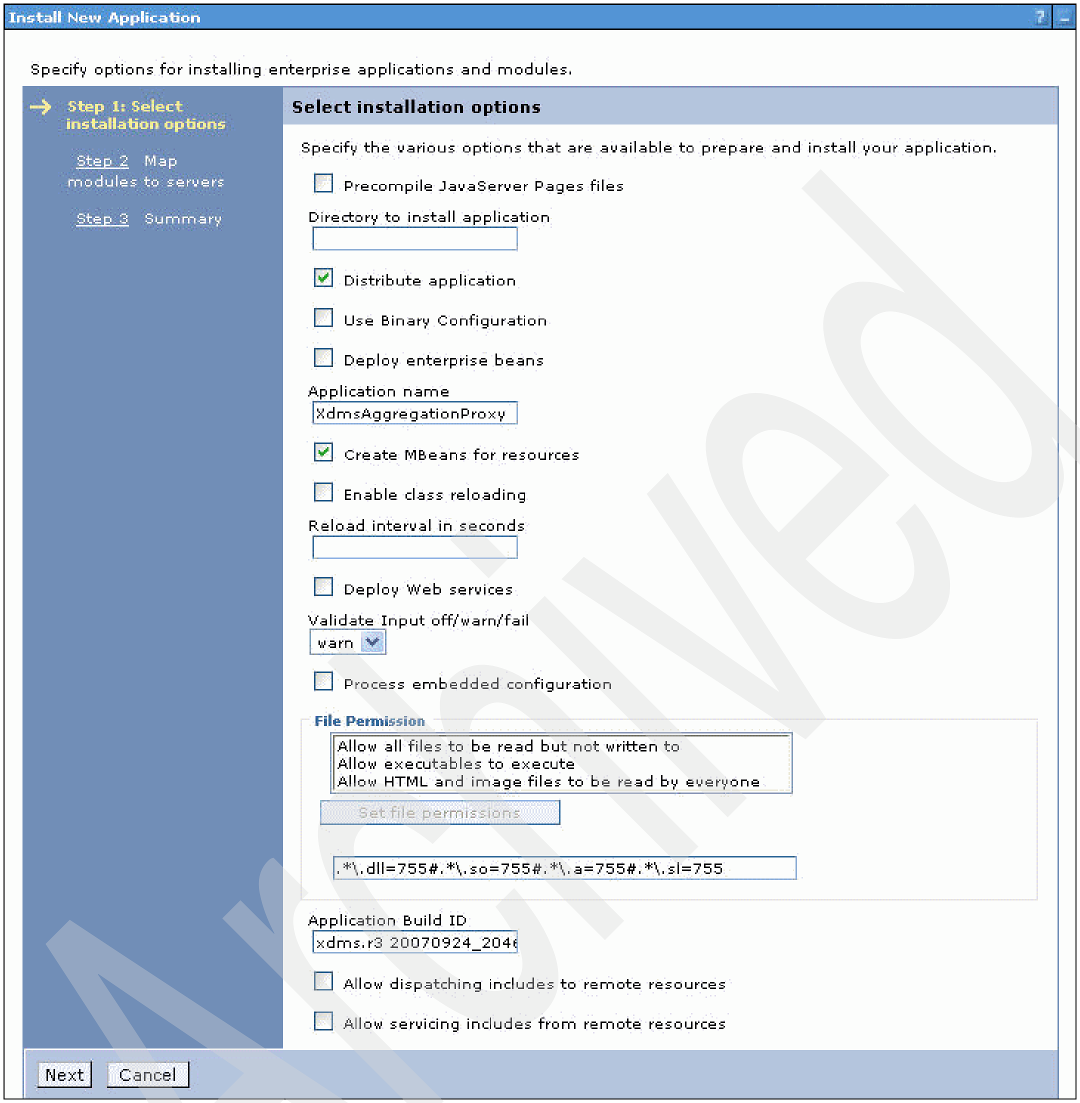


Figure 6-31 Install options selected

10. On the Map modules to servers page, Figure 6-32 on page 230, perform the following tasks:

    a. In the Clusters and Servers section, select **WebSphere:cell=imsCell,cluster=AggregationProxyCluster**.

    b. Select the **AggsProxyWeb** module option, and click **Apply** to update the changes, as shown in Figure 6-32 on page 230.

11.Click **Next**.

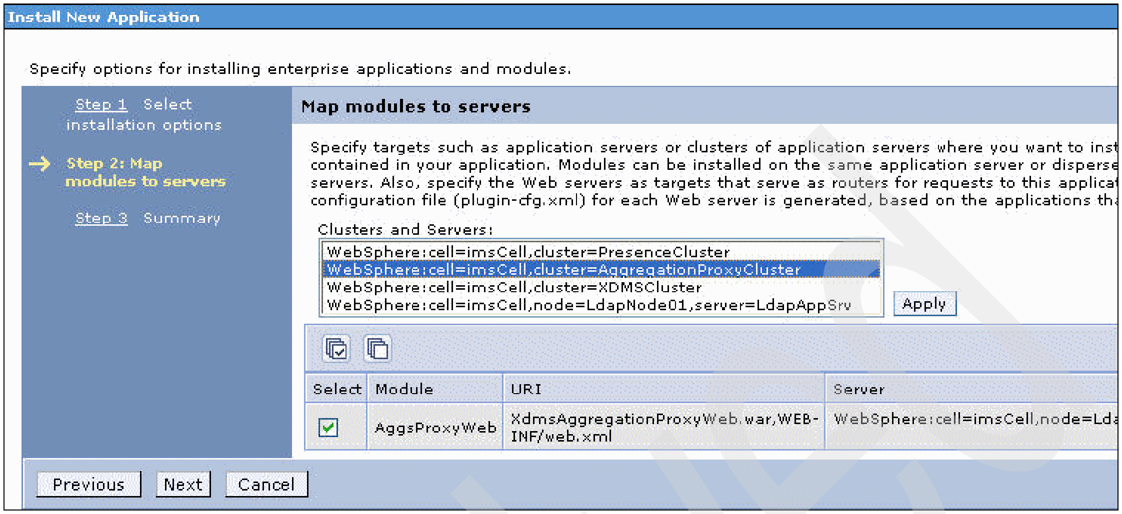


Figure 6-32 Map to the required module

12.On the Summary page, Figure 6-33 on page 231, verify the configurations for the XdmsAggregationProxy, and click **Finish** to accept the configurations.

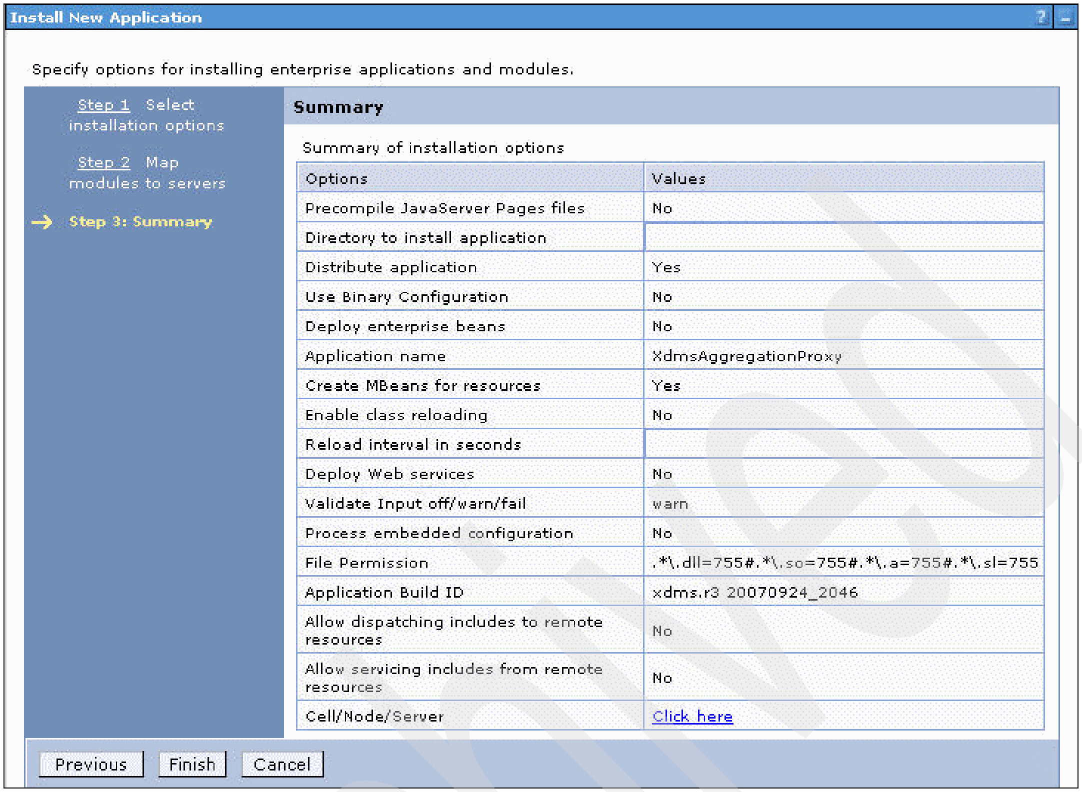


Figure 6-33 Configuration summary

13. Click **Save**, to save the changes to the master configuration. Example 6-1 shows the log for saving the changes to the master configuration.

Example 6-1 The log for saving to the master configuration

```
ADMS0200I: The configuration synchronization started for cell.
ADMS0202I: Automatic synchronization mode is disabled for node: ProxNode01.
ADMS0202I: Automatic synchronization mode is disabled for node: AggNode01.
ADMS0202I: Automatic synchronization mode is disabled for node: LdapNode01.
ADMS0202I: Automatic synchronization mode is disabled for node: XdmsNode01.
ADMS0202I: Automatic synchronization mode is disabled for node: PresNode01.
ADMS0202I: Automatic synchronization mode is disabled for node: ProxNode04.
ADMS0201I: The configuration synchronization started for node: ProxNode01.
ADMS0201I: The configuration synchronization started for node: AggNode01.
ADMS0205I: The configuration synchronization completed successfully for node:
ProxNode01.
ADMS0203I: The automatic synchronization mode is enabled for node: ProxNode01.
ADMS0201I: The configuration synchronization started for node: LdapNode01.
```

```
ADMS0205I: The configuration synchronization completed successfully for node:
AggNode01.
ADMS0201I: The configuration synchronization started for node: XdmsNode01.
ADMS0203I: The automatic synchronization mode is enabled for node: AggNode01.
ADMS0201I: The configuration synchronization started for node: PresNode01.
ADMS0205I: The configuration synchronization completed successfully for node:
PresNode01.
ADMS0203I: The automatic synchronization mode is enabled for node: PresNode01.
ADMS0207I: Node Synchronization state for node: LdapNode01 - initiate time:
2007.09.27 at 04:10:59:997 EDT complete time: 2007.09.26 at 04:11:00:480 EDT result:
Complete Update occurred
ADMS0205I: The configuration synchronization completed successfully for node:
LdapNode01.
ADMS0207I: Node Synchronization state for node: XdmsNode01 - initiate time:
2007.09.27 at 04:10:46:390 EDT complete time: 2007.09.26 at 04:10:49:900 EDT result:
Complete Update occurred
ADMS0205I: The configuration synchronization completed successfully for node:
XdmsNode01.
ADMS0207I: Node Synchronization state for node: ProxNode04 - initiate time:
2007.09.27 at 04:09:06:520 EDT complete time: 2007.09.26 at 04:09:06:782 EDT result:
Complete Update occurred
ADMS0205I: The configuration synchronization completed successfully for node:
ProxNode04.
ADMS0208I: The configuration synchronization complete for cell.
```

6.6.2 Configuring the XdmsAggregationProxy

To configure the XdmsAggregationProxy:

1. From the Integrated Solutions Console, select **Application** → **Enterprise Applications**. The page is refreshed with the available list of applications.
2. Click **XdmsAggregationProxy**, as shown in Figure 6-34 on page 233.

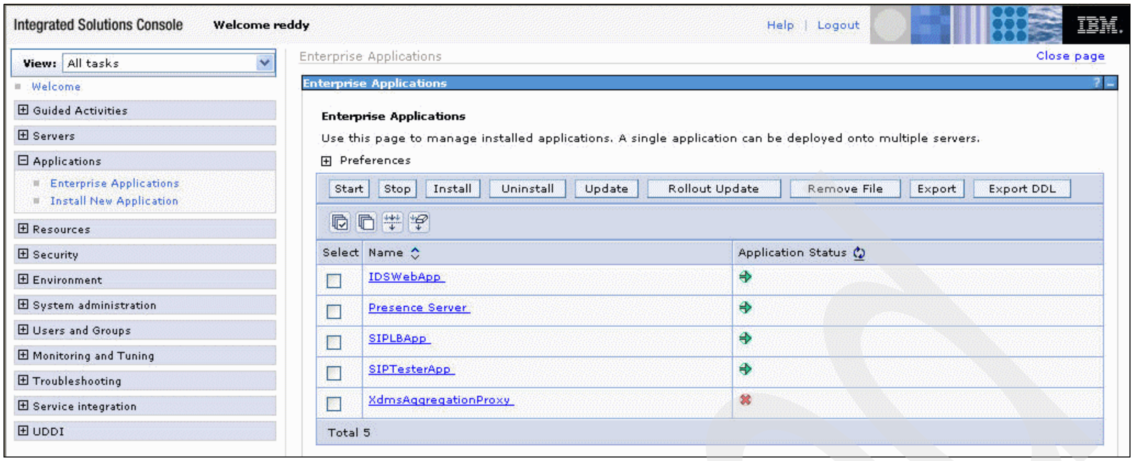


Figure 6-34 Summary of installed applications

3. On the XdmsAggregationProxy application configuration page, perform the following tasks, as shown in Figure 6-35 on page 234:
   a. From the Application reference validation field’s pull-down menu, select **Issue Warnings**.
   b. Click the **Security role to user/group mapping** link.

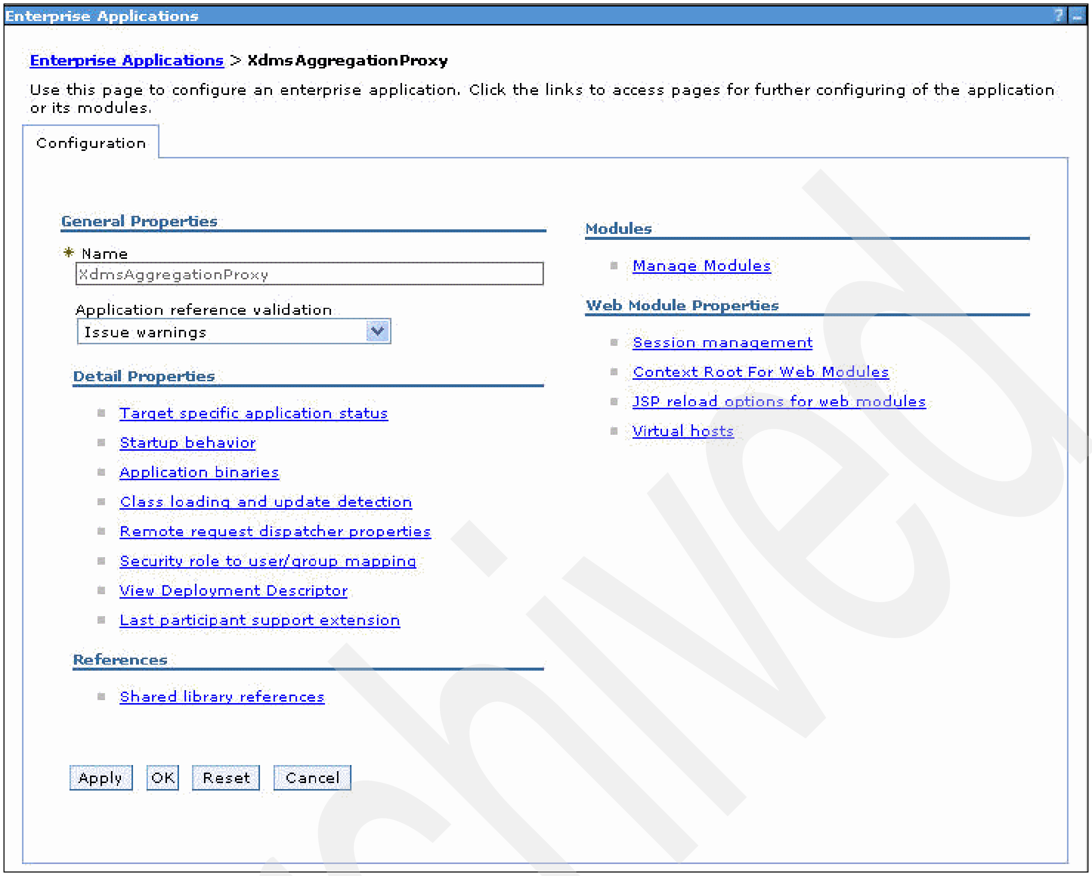


Figure 6-35 XdmAggregationProxy properties configuration

4. On the Security role to user/group mapping page, perform the following tasks, as shown in Figure 6-36 on page 235:
   a. Select the **All authenticated?** option.
   b. Select the **All_Role** option.
   c. Click **Save**, to save changes to master configuration.

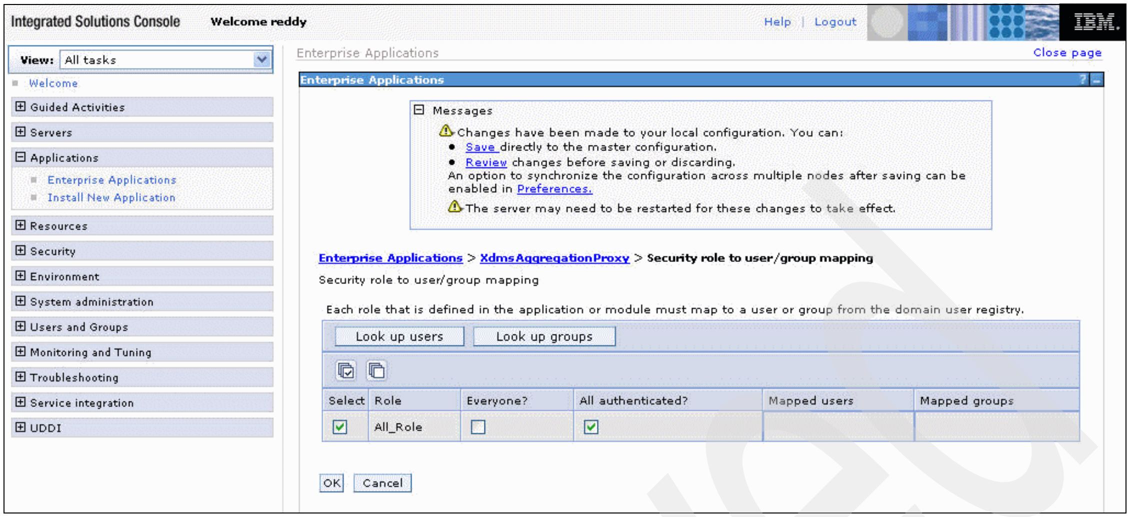


Figure 6-36 XdmAggregationProxy Security role to user/group mapping

> **Attention:** Restart all nodes, clusters, applications, and the Deployment Manager to see these values updated.

6.6.3 Verifying Resource environment entries

To verify Resource environment entries:

1. From the Integrated Solutions Console, select **Resources** → **Resource environment entries**.
2. Click the **AggProxyREPResEnvEntry** link, which we show in Figure 6-37 on page 236.

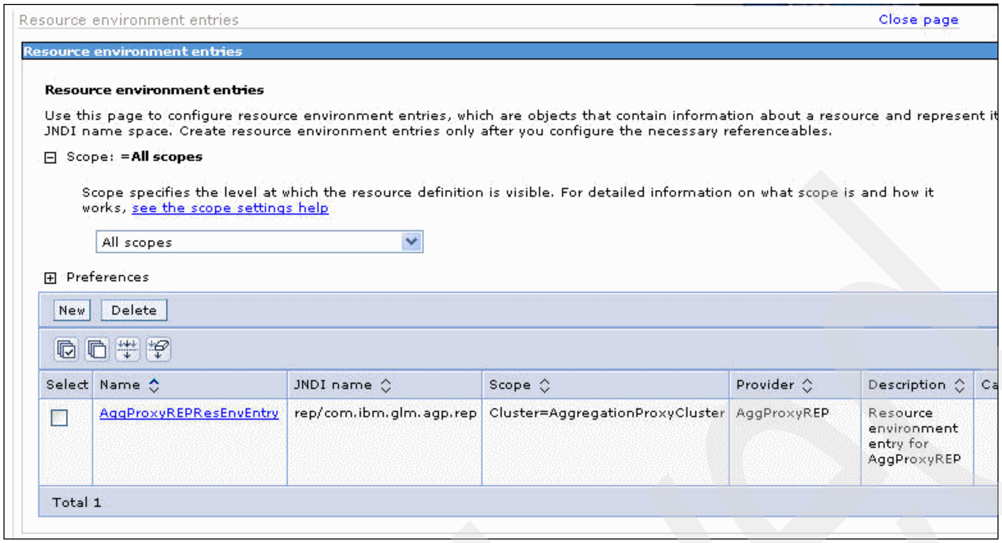


Figure 6-37 AggProxyREPResEnvEntry Resource environment entries

3. On the AggProxyREPResEnvEntry Resource environment entries configuration page, verify the configuration, and click **OK** if the entries are acceptable, as shown in Figure 6-38 on page 237.

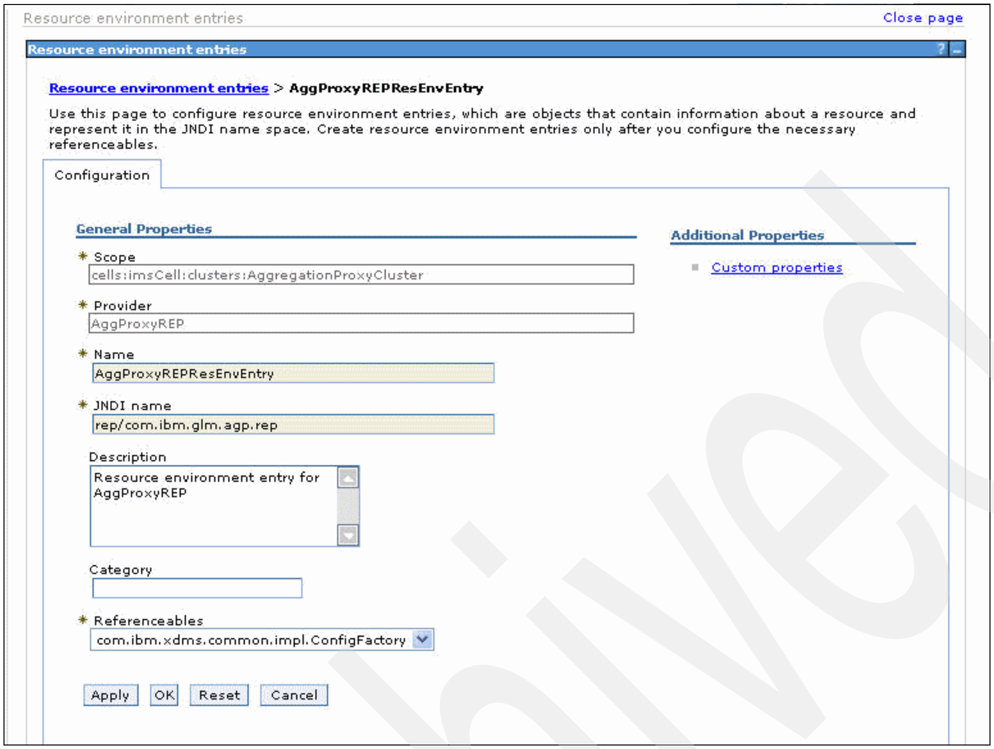


Figure 6-38 AggProxyREPResEnvEntry Resource environment entries configuration

6.7 Ports configuration

We configure the ports to verify the virtual host address that is assigned to the Aggregation Proxy to hold valid port definitions for HTTP port to accept incoming requests.

To configure ports:

1. From the Integrated Solutions Console, **Applications** → **Enterprise Applications**.
2. Select the **XdmsAggregationProxy** application, as shown in Figure 6-39 on page 238.

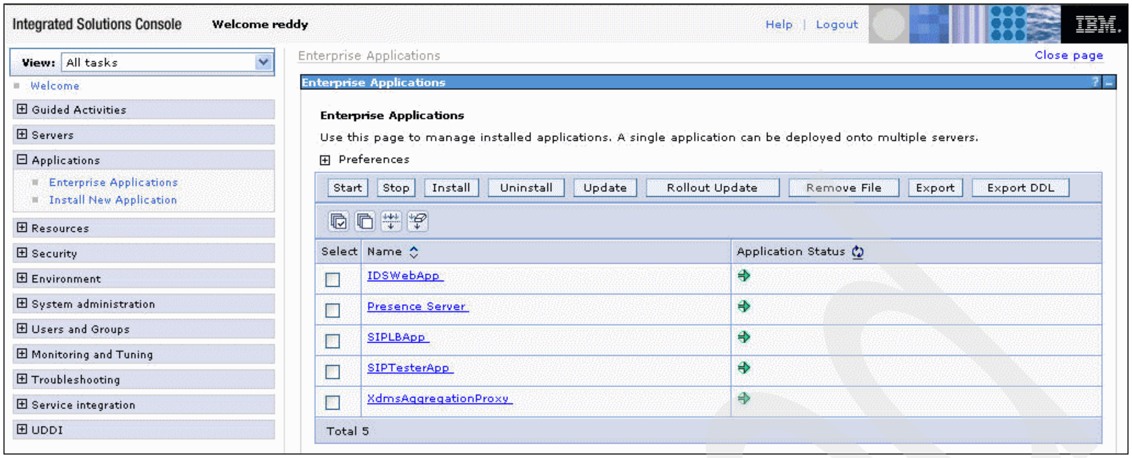


Figure 6-39 List of deployed applications

3. Under Web Module Properties, select **Virtual hosts**, as shown in Figure 6-40.

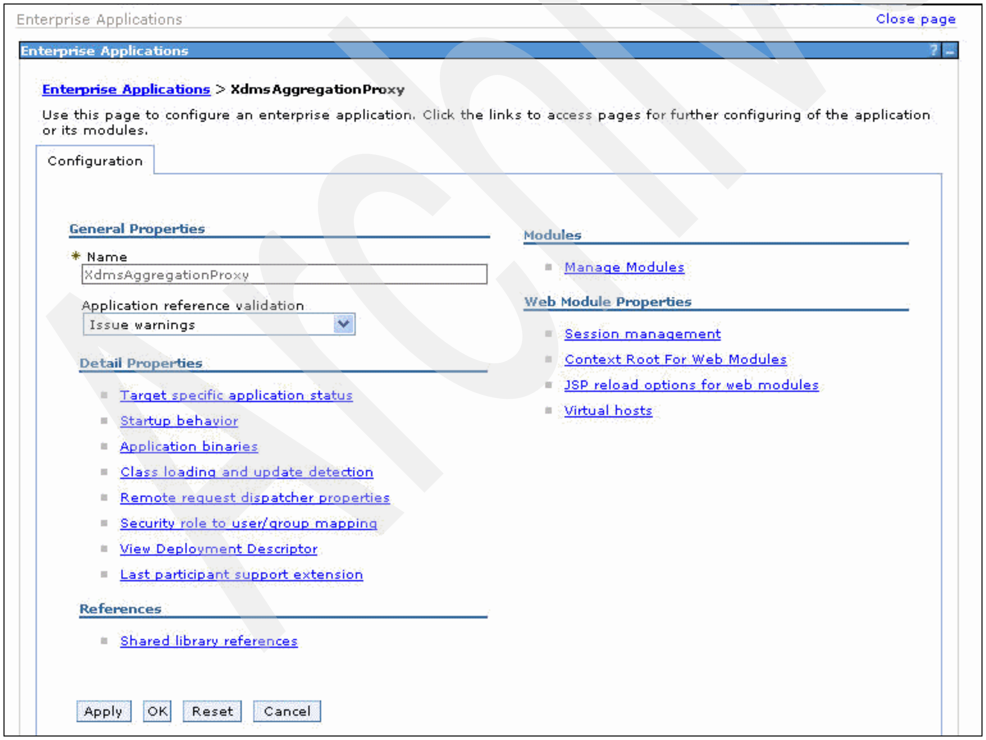


Figure 6-40 XdmsAggregationProxy configuration

4. Select the **AggsProxyWeb** option.
5. From the Virtual host field pull-down menu, select **aggproxy_host**, and click **OK**, as shown in Figure 6-41.

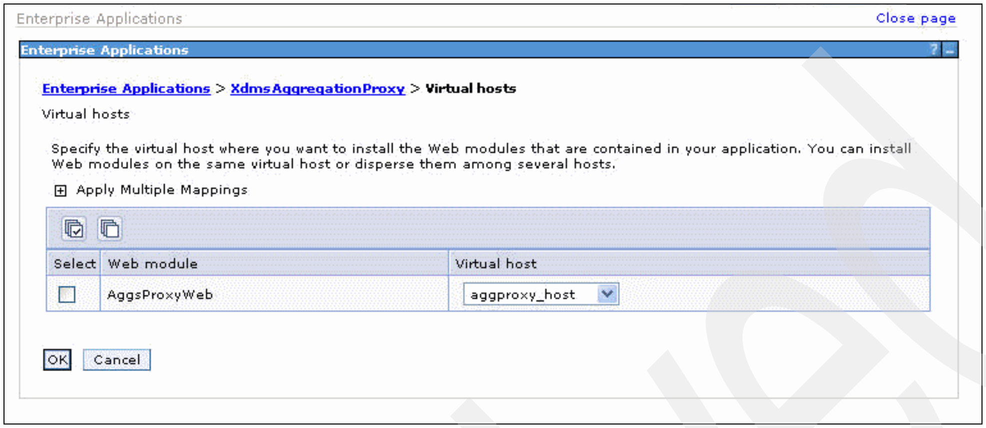


Figure 6-41 Virtual hosts configuration

6. Click **Save**, to save changes to the master configuration, as shown in Figure 6-42 on page 240.

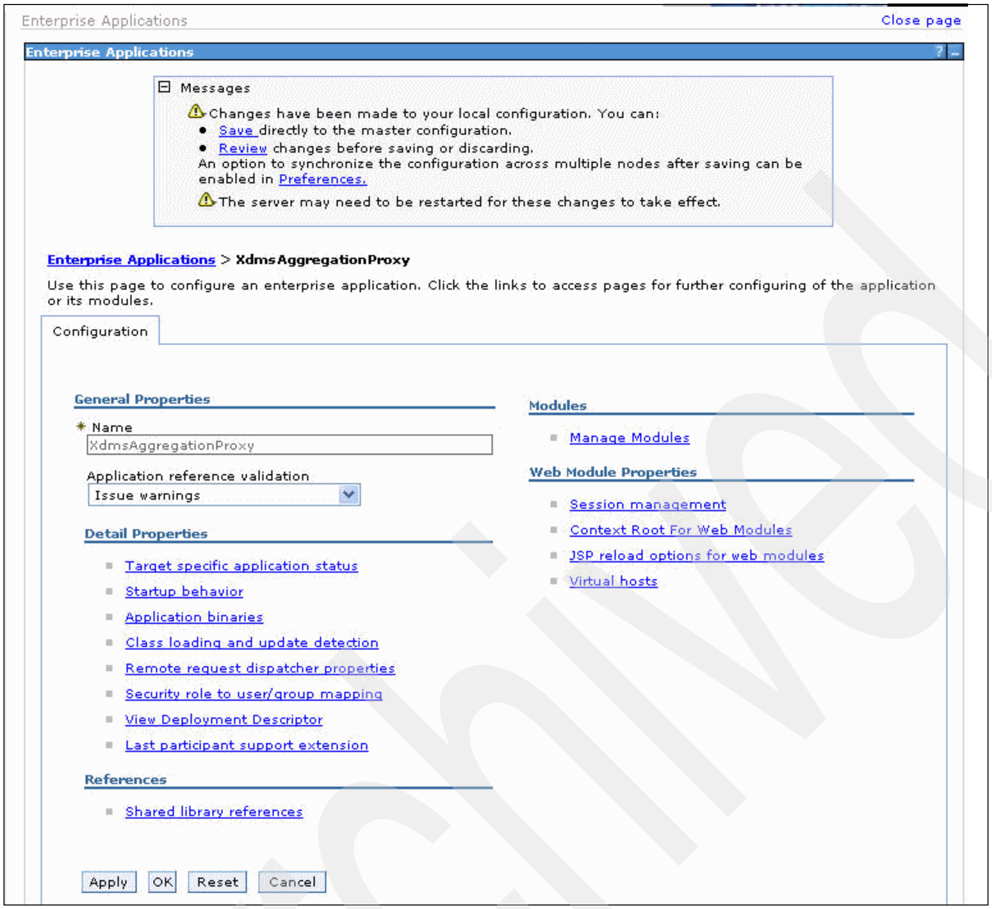


Figure 6-42 Save changes to the master configuration

7. From the Integrated Solutions Console, **Servers** → **Clusters**, and select **AggregationProxyCluster**, as shown in Figure 6-43 on page 241.

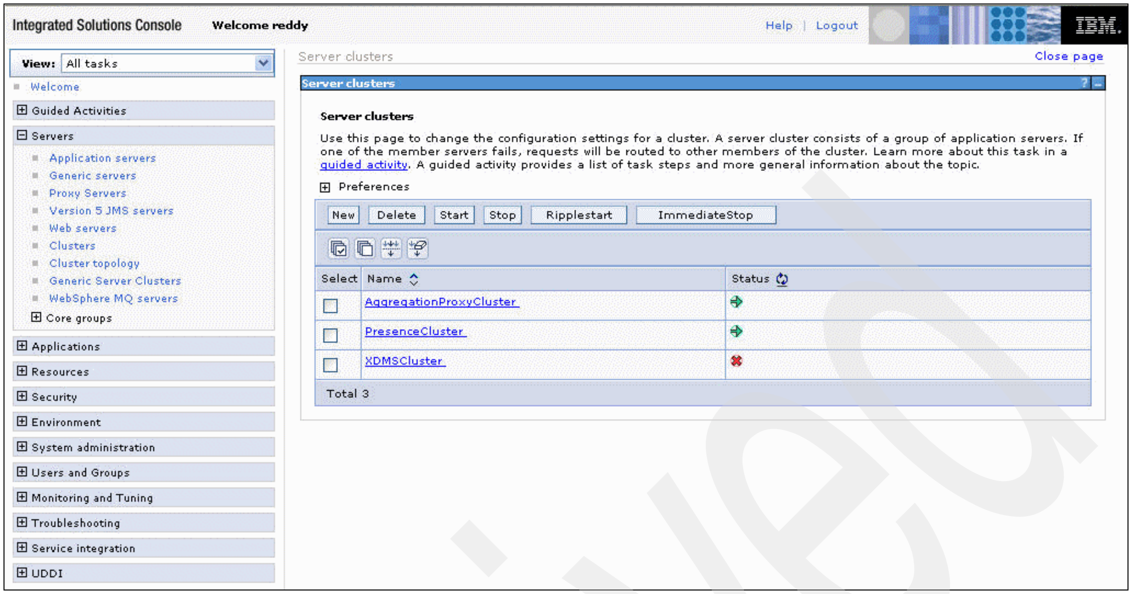


Figure 6-43 Configured clusters in the environment

8. Under Additional Properties section, Figure 6-44, select **Cluster members**.

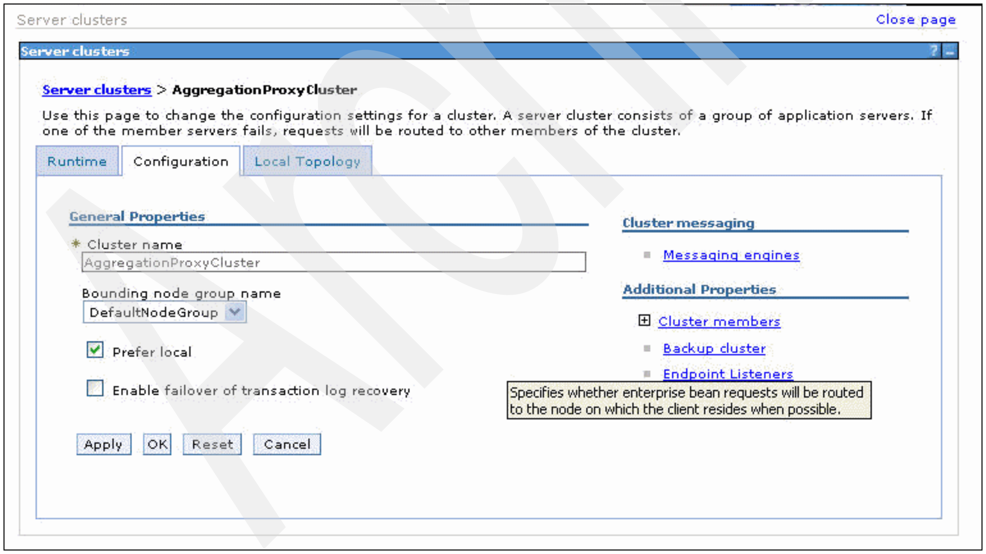


Figure 6-44 AggregationProxyCluster configuration page

9. Select **AGGProxSrv01**, shown in Figure 6-45.

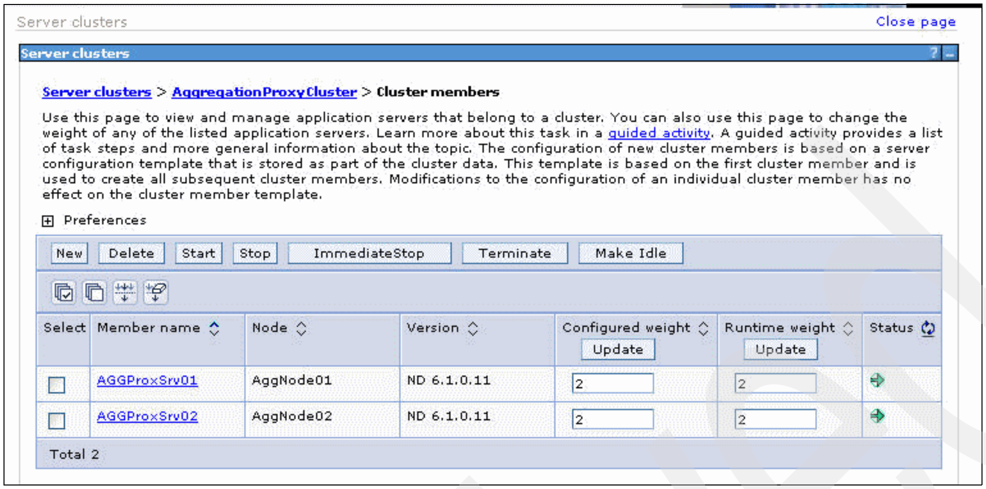


Figure 6-45 AggregationProxyCluster cluster members

10. Under the Communications section, Figure 6-46 on page 243, select **Ports**.

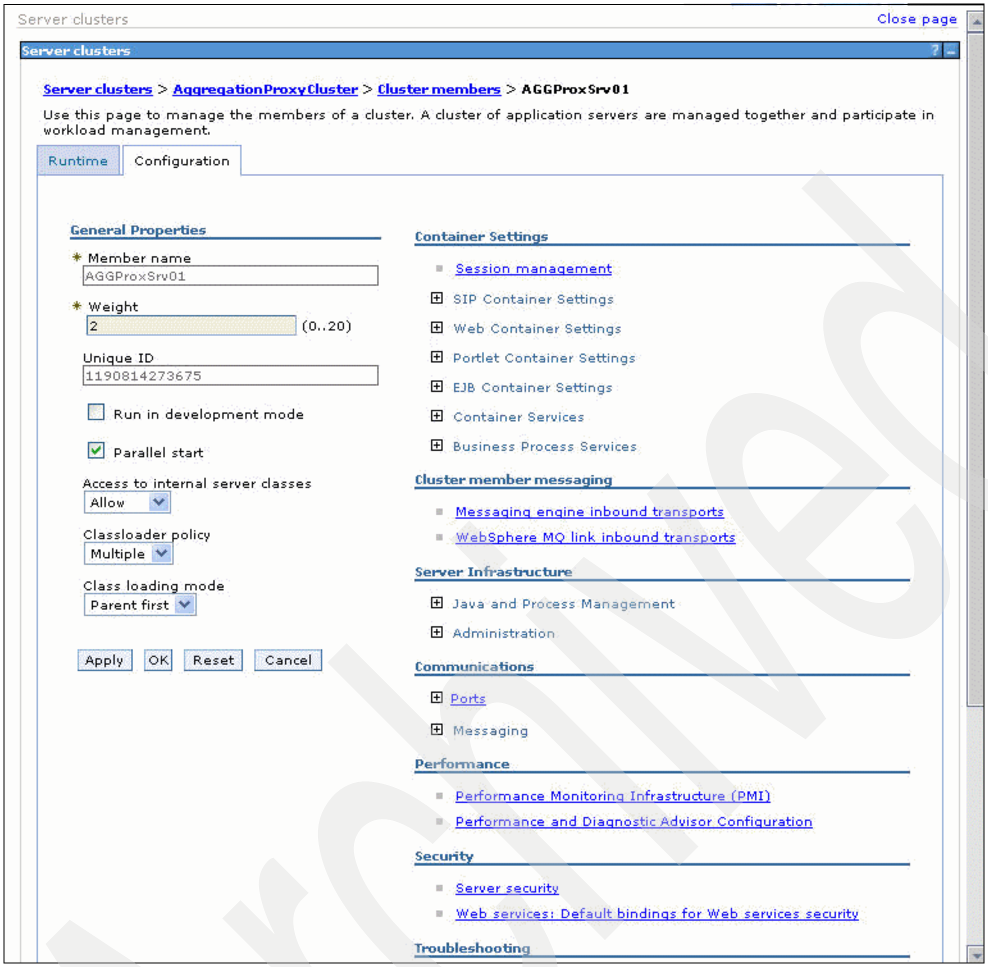


Figure 6-46 AGGProxSrv01 configuration

11. In Figure 6-47 on page 244, verify the port numbers settings.
12. If you need to modify a port number, select the Port name to modify the setting, which effects the required changes in configuration.

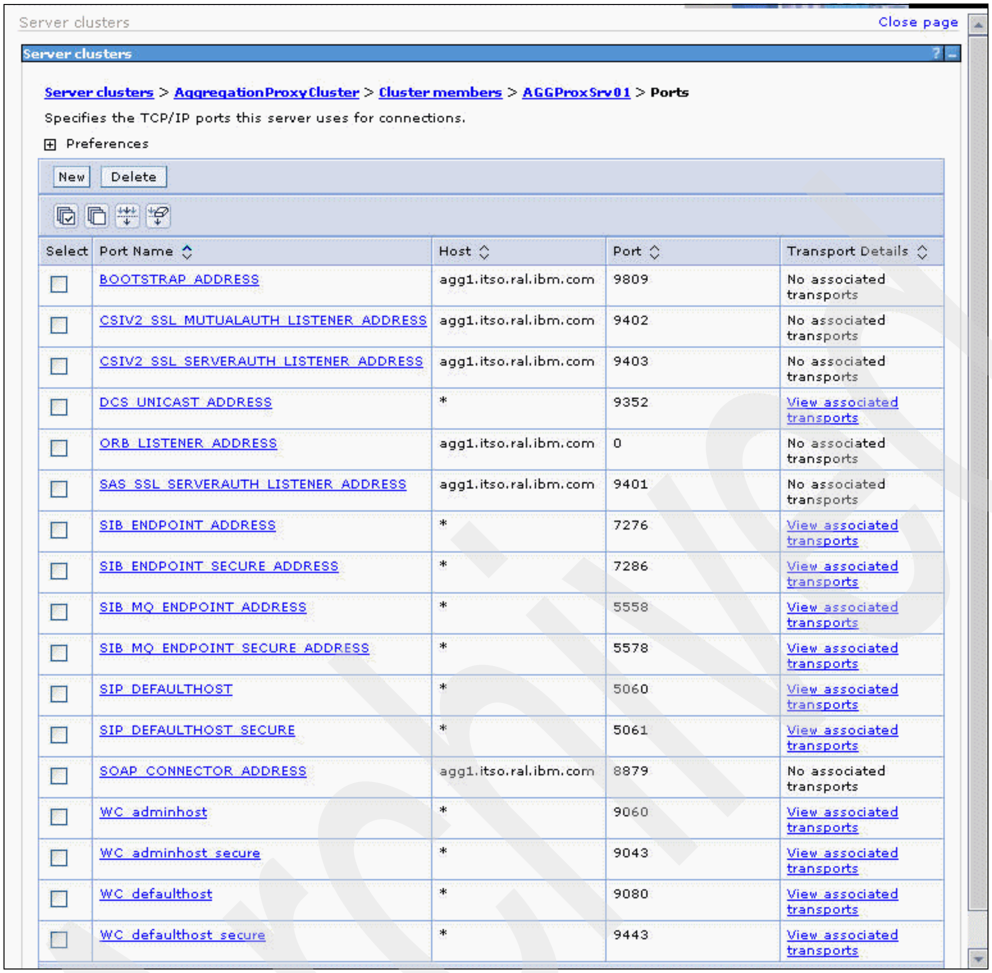


Figure 6-47 Ports

6.7.1 Adding a new Virtual Host

1. From the Integrated Solutions Console, Figure 6-48 on page 245, select **Environment** → **Virtual Hosts**, and then click **New** to add a new Virtual host.

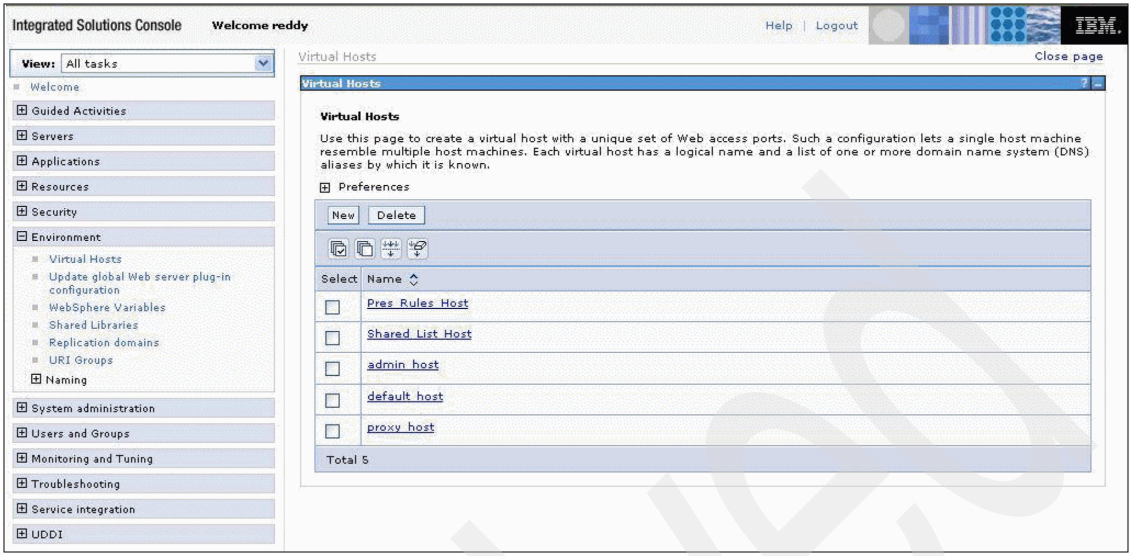


Figure 6-48 Available Virtual Hosts

2. On the new Virtual Host configuration page, Figure 6-49, in the Name field, type **aggproxy_host**, and click **OK**.

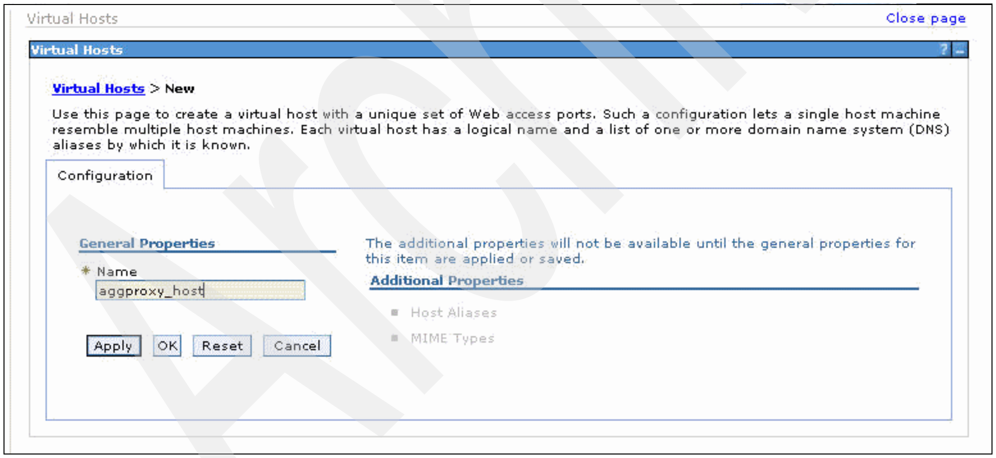


Figure 6-49 New aggproxy_host properties

3. In Figure 6-50 on page 246, select the **aggproxy_host** link.

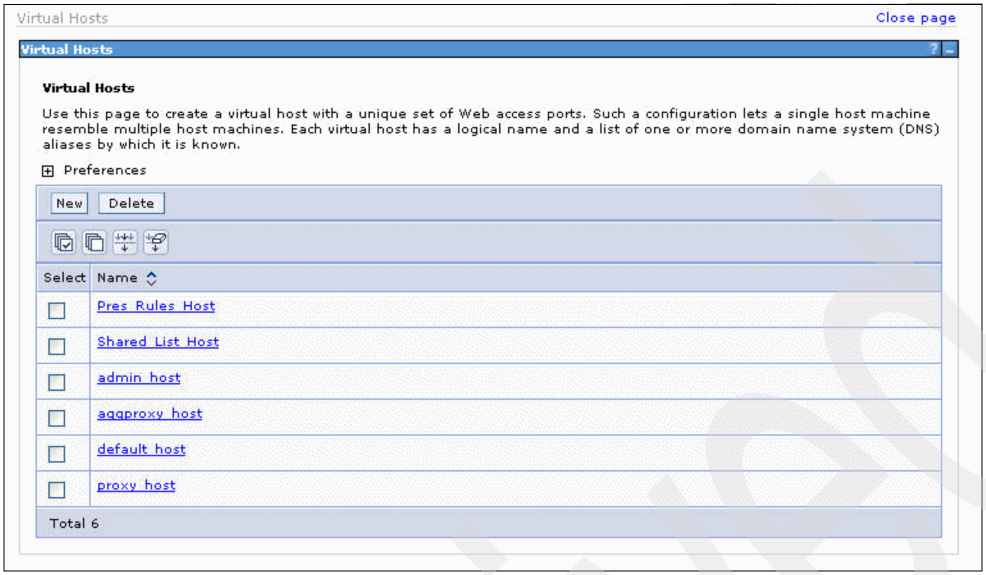


Figure 6-50 List of Virtual Hosts

4. Under the Additional Properties section, Figure 6-51, click **Host Aliases**.

Figure 6-51 Selected the aggproxy_host

5. Click **New** to create a new Host Alias, as shown in Figure 6-52 on page 247.

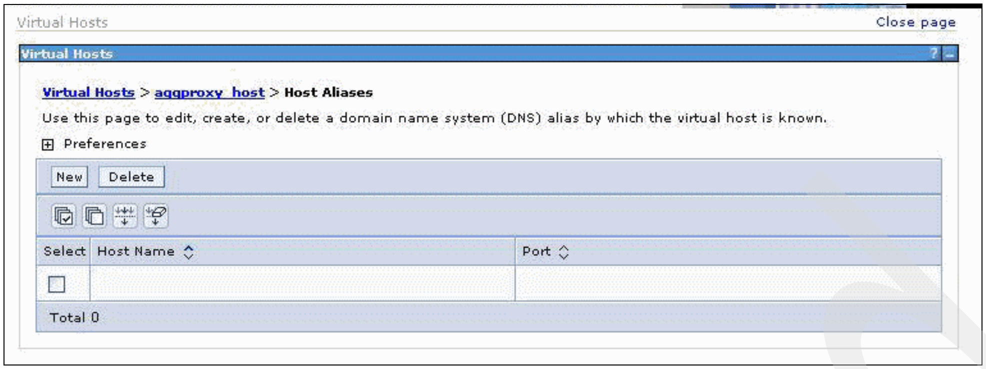


Figure 6-52   Aggproxy_host host aliases page

6. On the new Host Aliases page, Figure 6-53, enter the following parameters, and then click **OK**:

   a. In the Host name field, type *.

   b. In the Port field, type **9080**.

Figure 6-53   New Host Alias properties

7. Repeat steps 5 and 6, and enter the following properties, as shown in Figure 6-54 on page 248:

   a. In the Host name field, type *.

   b. In the Port field, type **80**.

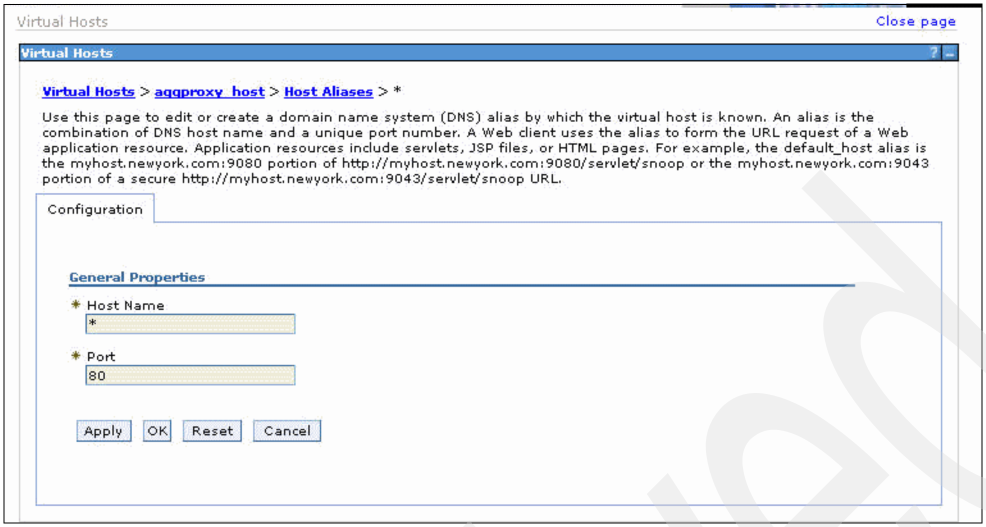


Figure 6-54 New Host Alias properties

8. Repeat steps 5 and 6 with the following properties, as shown in Figure 6-55:
   a. In the Host name field, type **agg1.itso.ral.ibm.com**.
   b. In the Port field, type **9080**.

Figure 6-55 New Host Alias properties

9. Repeat steps 5 and 6 with the following properties, as shown in Figure 6-56:
   a. In the Host name field, type **agg2.itso.ral.ibm.com**.
   b. In the Port field, type **9080**.

Figure 6-56 New Host Alias properties

6.7.2 Proxy setting

We set the Proxy to route the client's XCAP requests to the Aggregation Proxy Cluster. The process for setting the Proxy is:

1. Create a new port.
2. Create a new transport chain.

Creating a new port

To create a new port:

1. From the Integrated Solutions Console, **Servers** → **Proxy Servers**, and click **SIPProxSrv01**, as shown in Figure 6-57 on page 250.

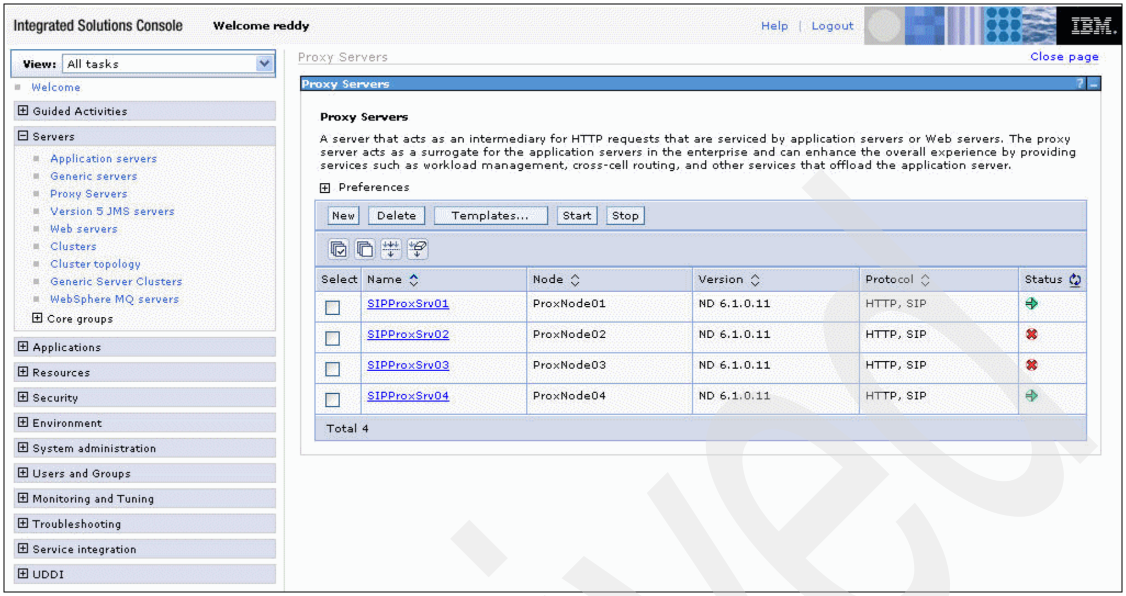


Figure 6-57 List of Proxy Servers

2. In Figure 6-58 on page 251, locate the Communications setting section, and click **Ports** to see list of ports.
3. Verify that the port number for **WC_defaulthost** is different from the port numbers for XCAP request routing.
4. If you need to modify the port number for WC_defaulthost, click the **Detail** button to the right of the list of port numbers.

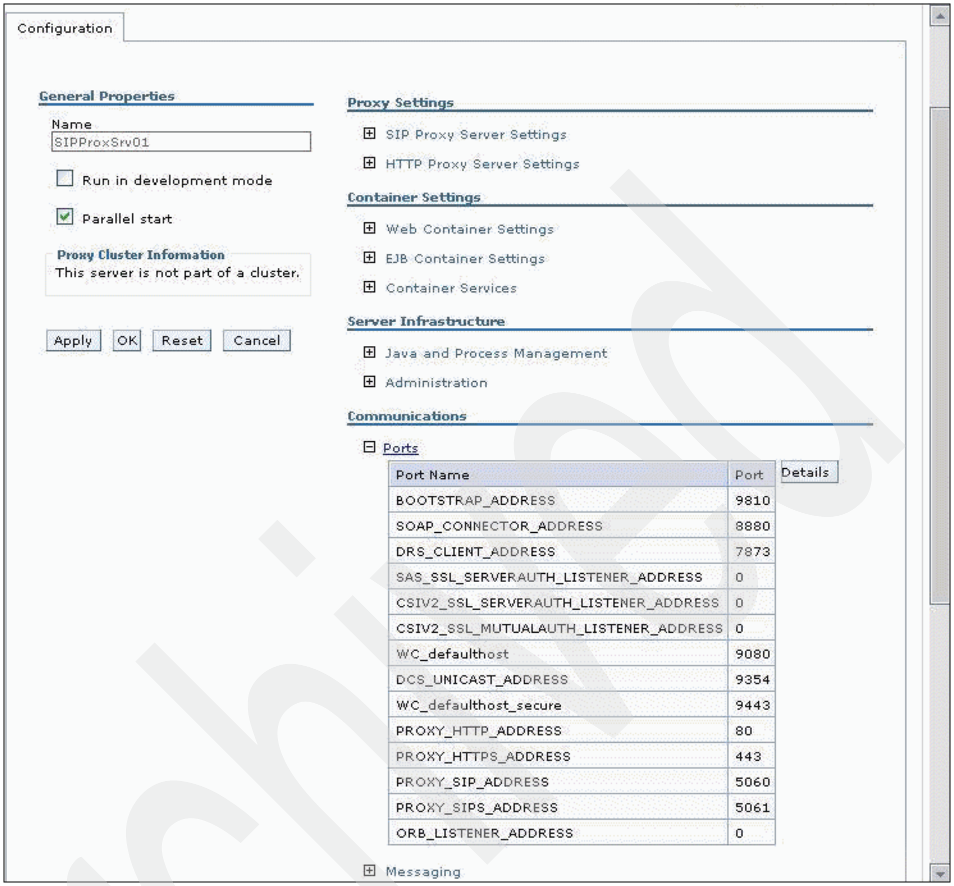


Figure 6-58 SIPProxSrv01 ports and default values

5. Click **WC_defaulthost** to modify the port.
6. On the WC_defaulthost configuration page, Figure 6-59 on page 252, change the port number, in this case, from **9080** to **9088**, and then click **Apply**.

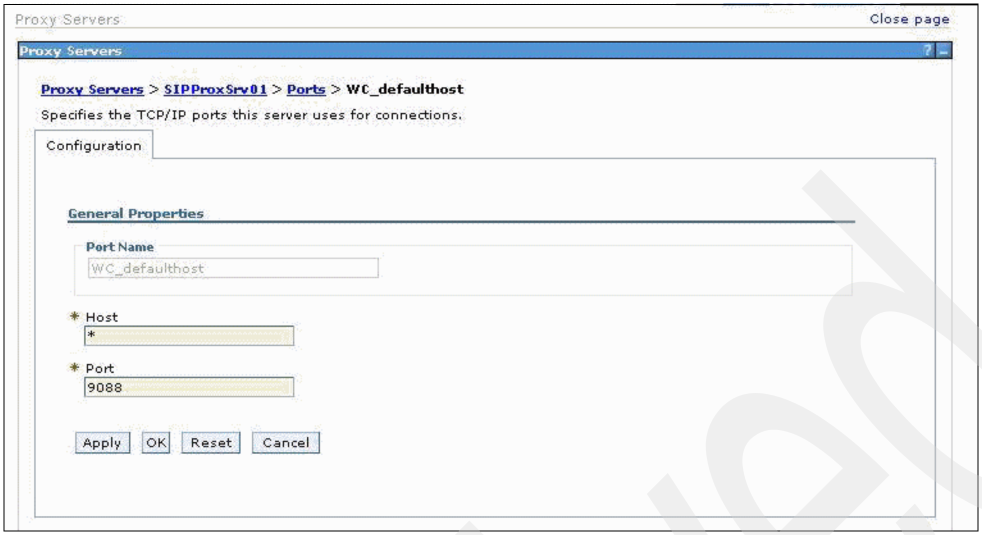


Figure 6-59   Changing WC_defaulthost new port number

Then new WC_defaulthost port number will show on the SIPProxSrv01 port settings page, as shown in Figure 6-60 on page 253.

7. Click the **New** button, Figure 6-60 on page 253, to create a new port for routing XCAP requests to the Aggregation Server.

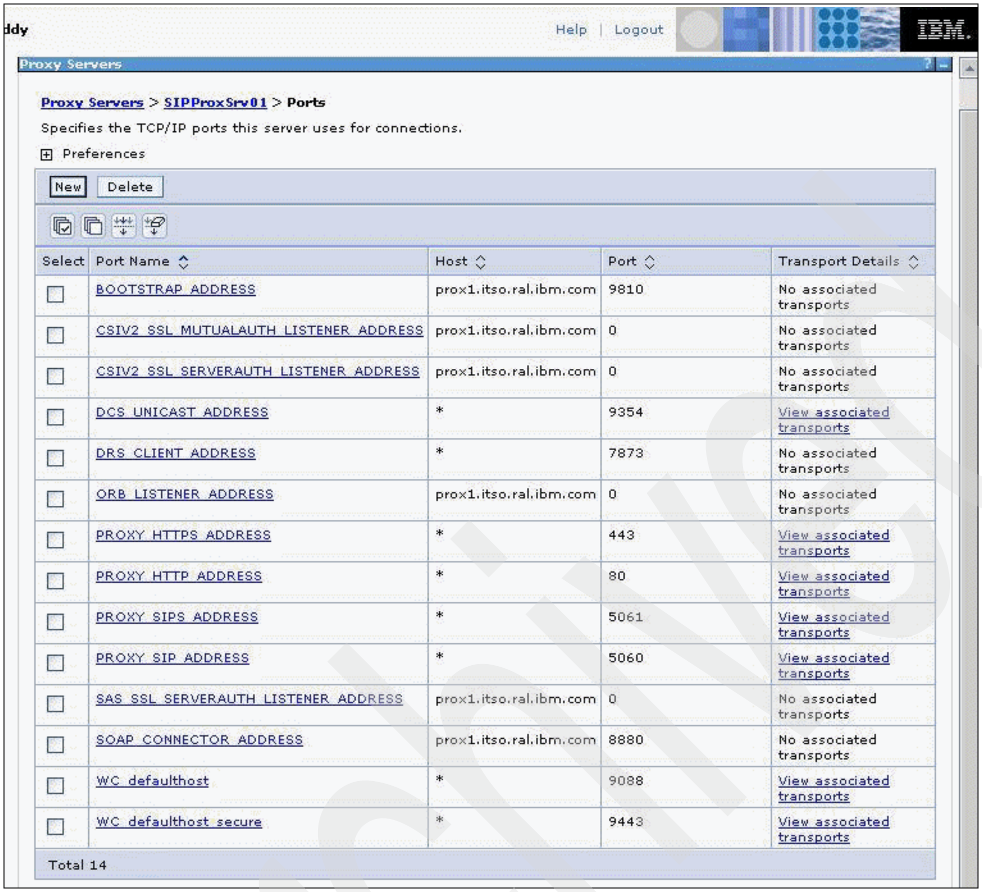


Figure 6-60 List of SIPProxSrv01 port settings port settings

8. On the Port creation page, Figure 6-61 on page 254, do the following:
   a. Select the **User-defined Port** option.
   b. In the Specify the Port name field, type `AGG_PROXY_HTTP_ADDRESS`.
   c. In the Host field, type *.
   d. In the Port field, type **9080**.
   e. Click **Apply**.

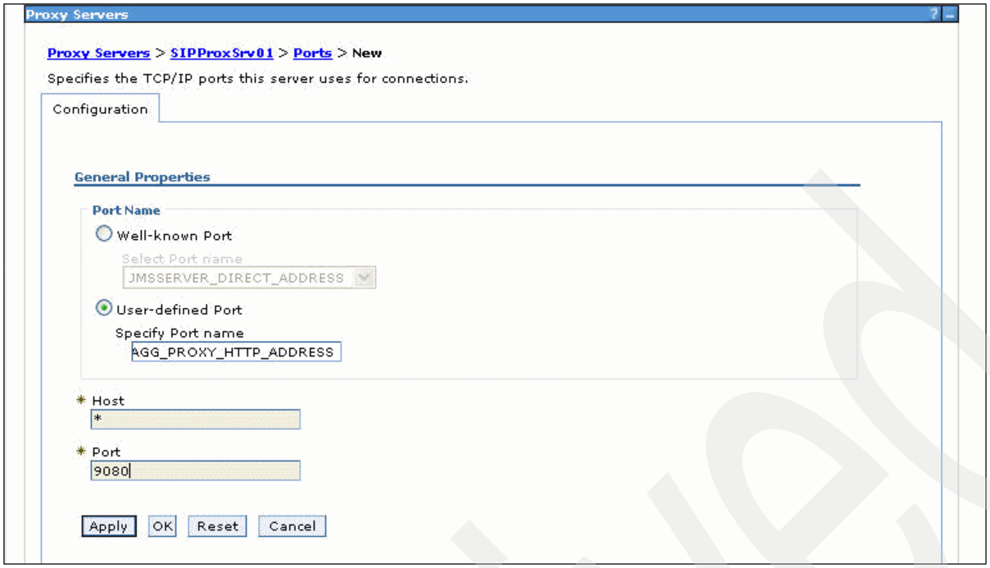


Figure 6-61   New Port properties for routing XCAP requests

9. Click **Save**, to save the changes to the master configuration, as shown in Figure 6-62 on page 255.

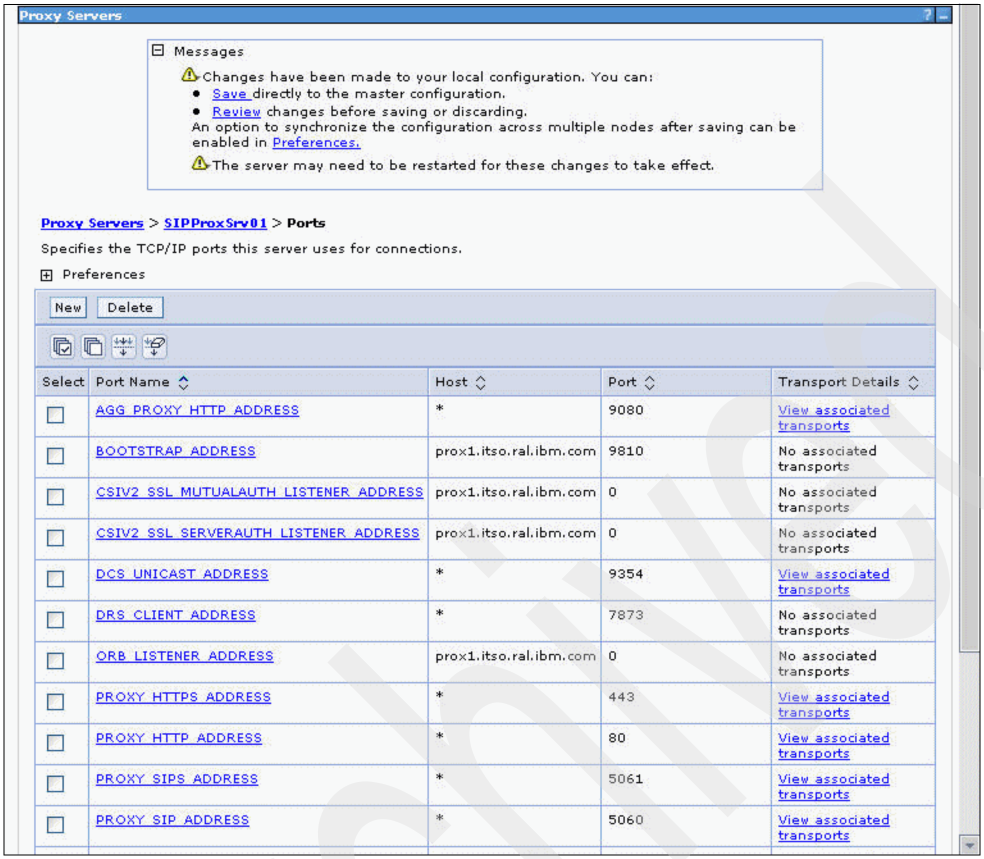


Figure 6-62 Save port changes to the master configuration

Creating a new transport chain

To create the new transport chain:

1. From the Integrated Solutions Console, select **Servers** → **Proxy Servers**, and click **SIPProxSrv01**.
2. Under the Proxy Settings section, Figure 6-63 on page 256, expand **Http Proxy Server Settings**, and click **Proxy Server transports** from the list.

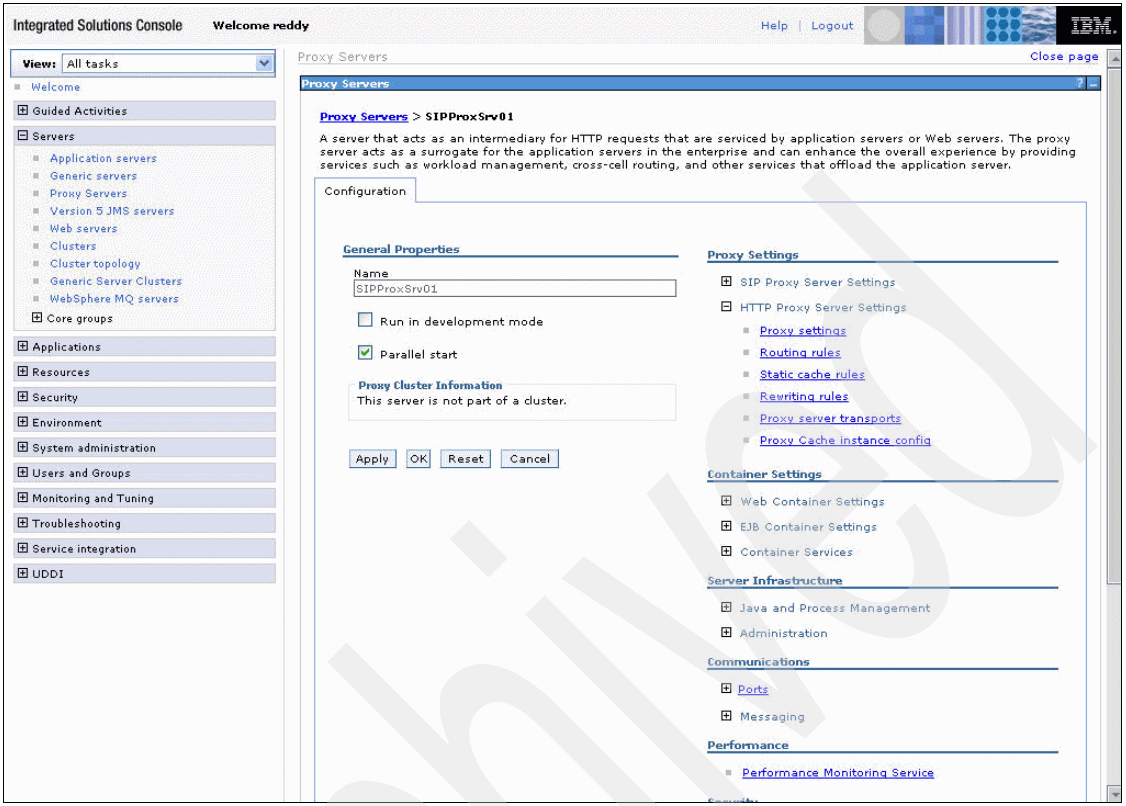


Figure 6-63 List of Proxy Servers

3. Click the **New** button, shown in Figure 6-64, to create a new transport chain for the XCAP requests.

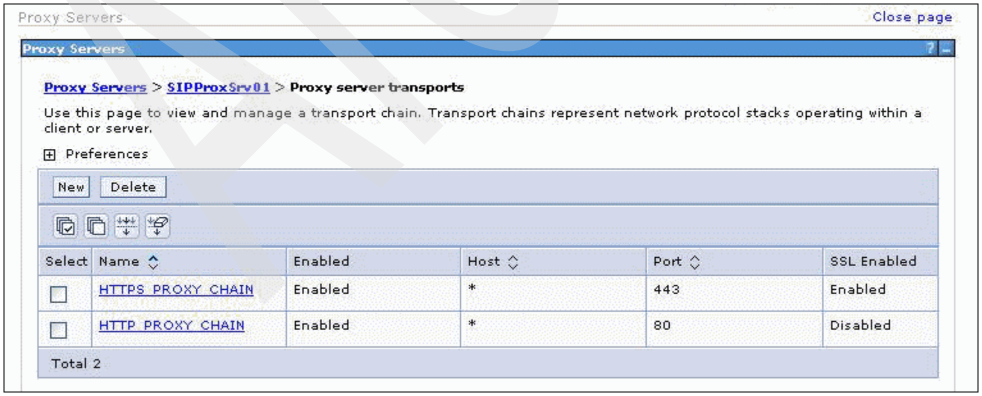


Figure 6-64 SIPProxSrv01 Proxy Server transports

4. On the Transport chain template page, Figure 6-65, enter the following parameters:
   a. In the Transport chain name field, type **HTTP_AGG_PROXY_CHAIN**.
   b. In the Transport chain template field, from the pull-down menu, select **Proxy(templates/chains|proxy-chains.xml#Chain_1)**.
5. Click **Next**.

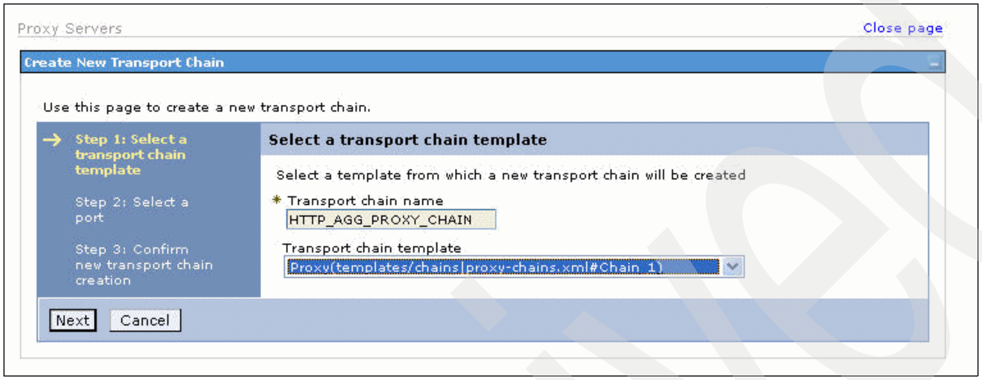


Figure 6-65 Transport chain template selection

6. On the Port selection page, Figure 6-66 on page 258, perform the following tasks:
   a. Select the **Use existing port** option.
   b. From the Existing port field pull-down menu, select **AGG_PROXY_HTTP_ADDRESS**.
7. Click **Next**.

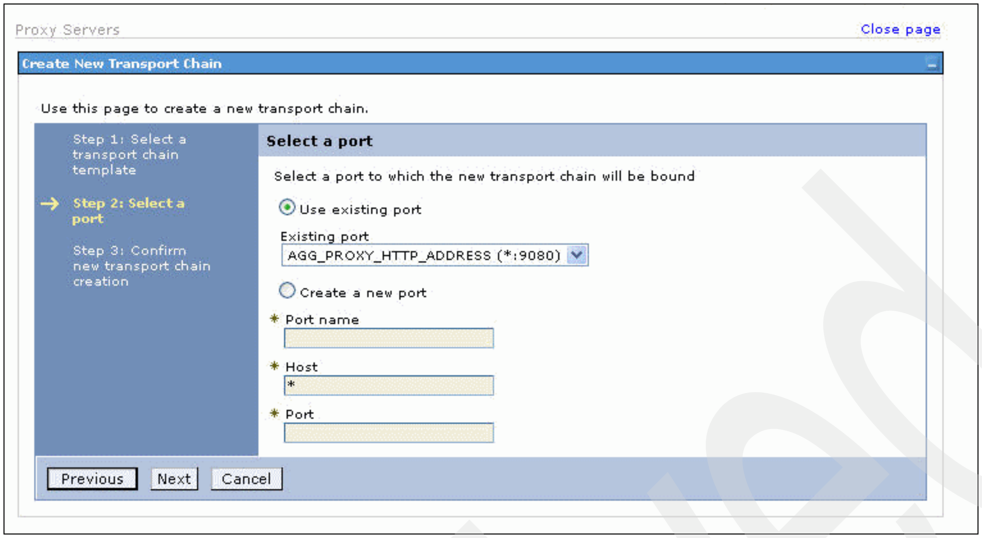


Figure 6-66 Port selection page

8. On the Confirm new transport chain creation page, Figure 6-67, review the new transport chain summary, and then click **Finish** to confirm the creation.

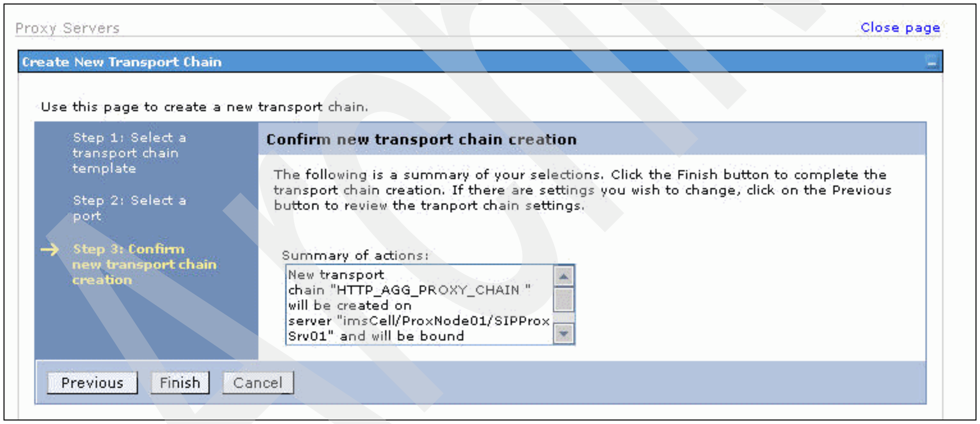


Figure 6-67 New transport chain confirmation page

9. Click **Save**, to save the changes to the master configuration, as shown in Figure 6-68 on page 259.

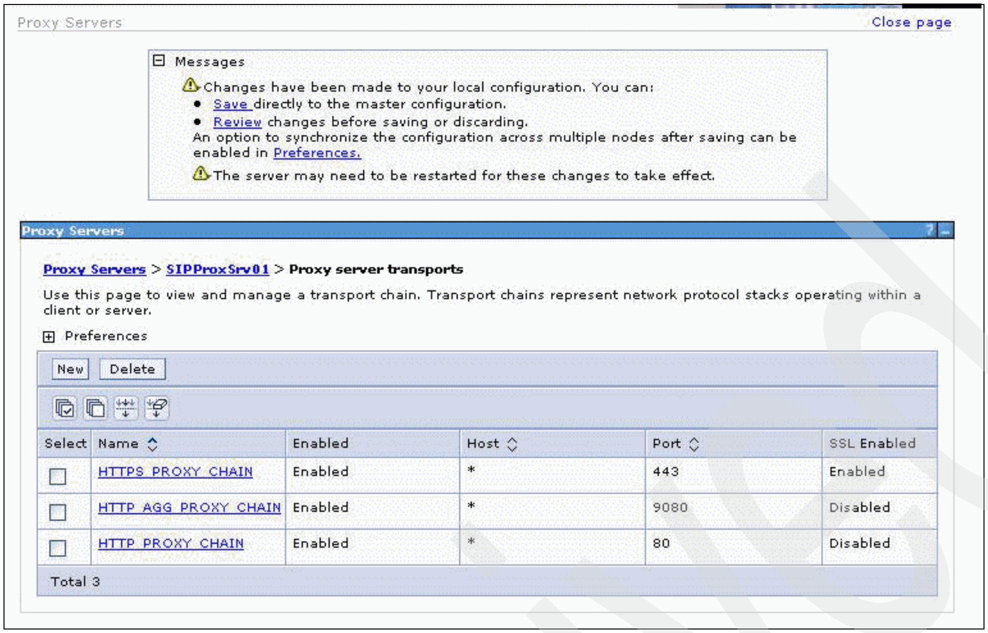


Figure 6-68 Save new transport chain to master configuration

6.8 Basic routing configuration

To configure basic routing:

1. From the Integrated Solutions Console, **Resources** → **Resource Environment Providers**, and select **AggProxyREP**, as shown in Figure 6-69 on page 260.

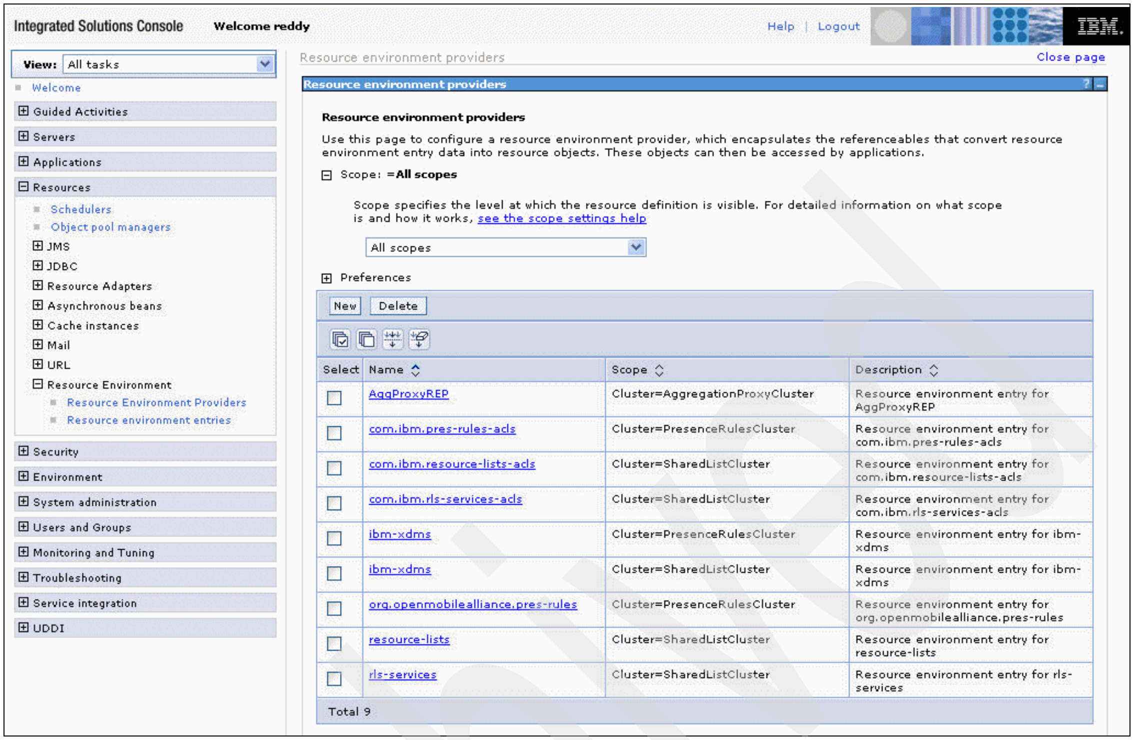


Figure 6-69 Resource environment providers

2. Under the Additional Properties section, Figure 6-70 on page 261, select **Custom properties**.

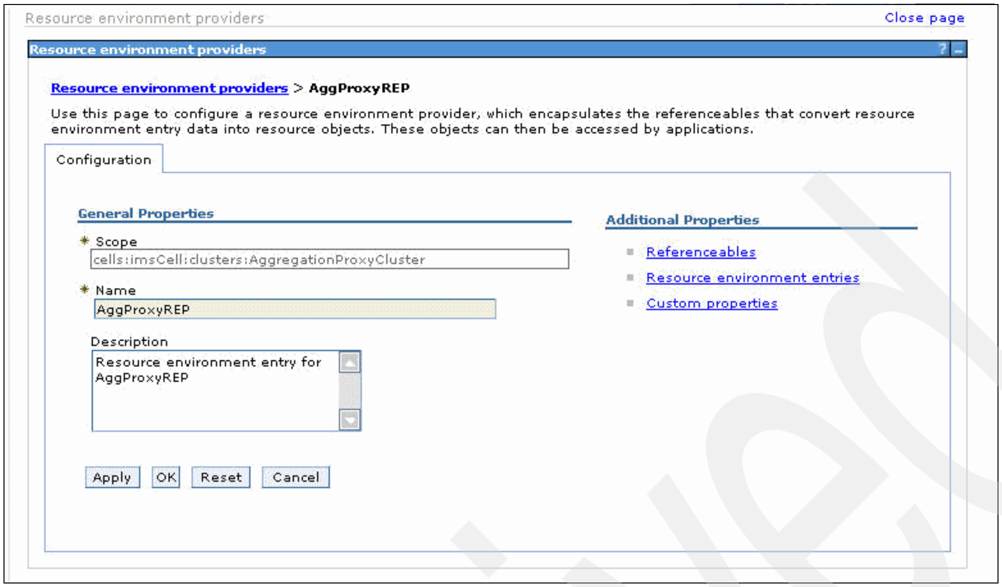


Figure 6-70 AggProxyREP configuration page

3. Select **XDMS_URI**, which we show in Figure 6-71 on page 262.

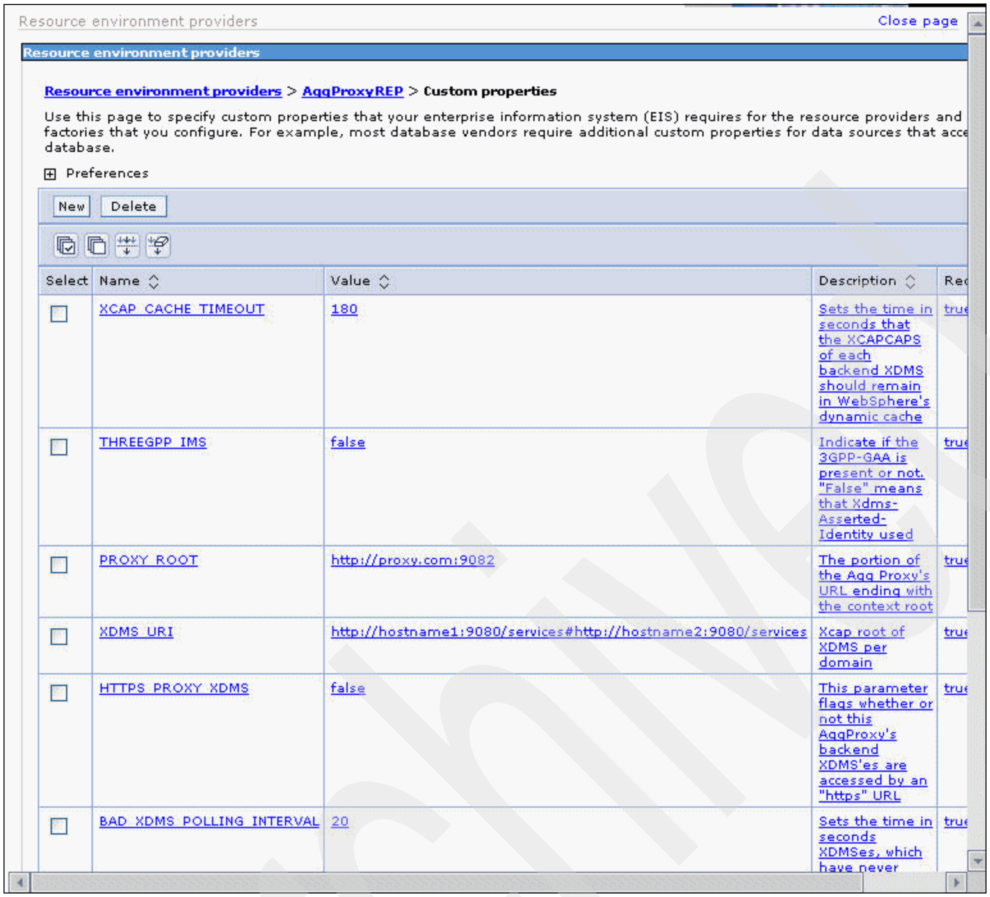


Figure 6-71 AggProxyREP custom properties

4. On the **XDMS_URI** General properties page, Figure 6-72 on page 263, change the Value to `http://xdms.itso.ral.ibm.com:9082#http://xdms.itso.ral.ibm.com:9082`, and then click **OK**.

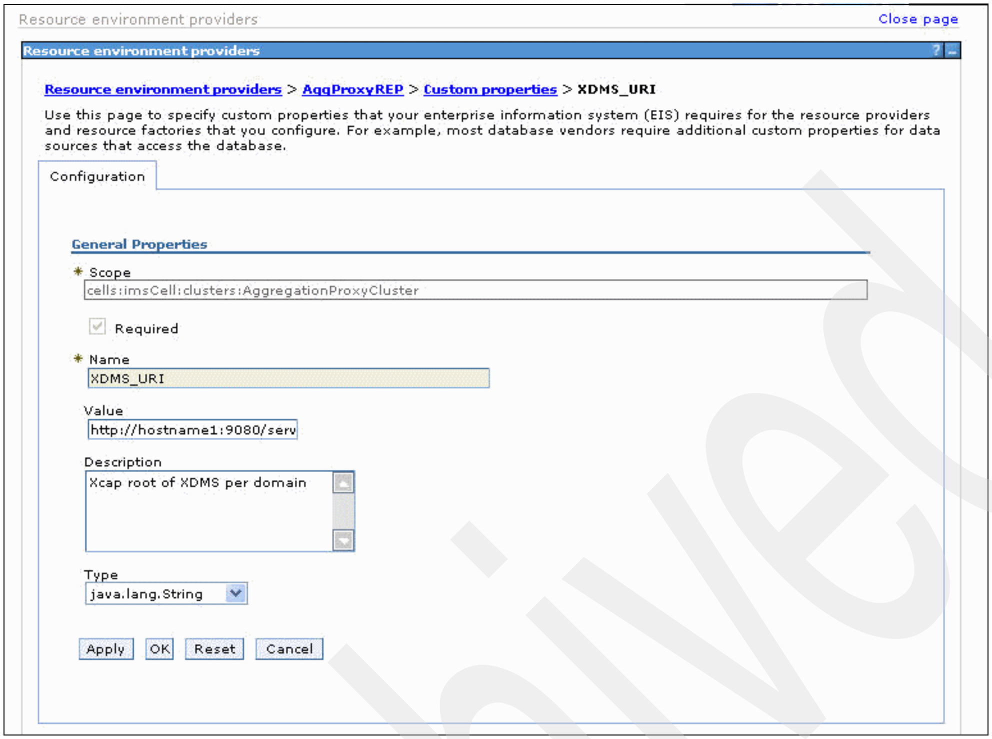


Figure 6-72 XDMS_URI General properties

5. Click **Save**, to save the changes to the master configuration, as shown in Figure 6-73 on page 264.

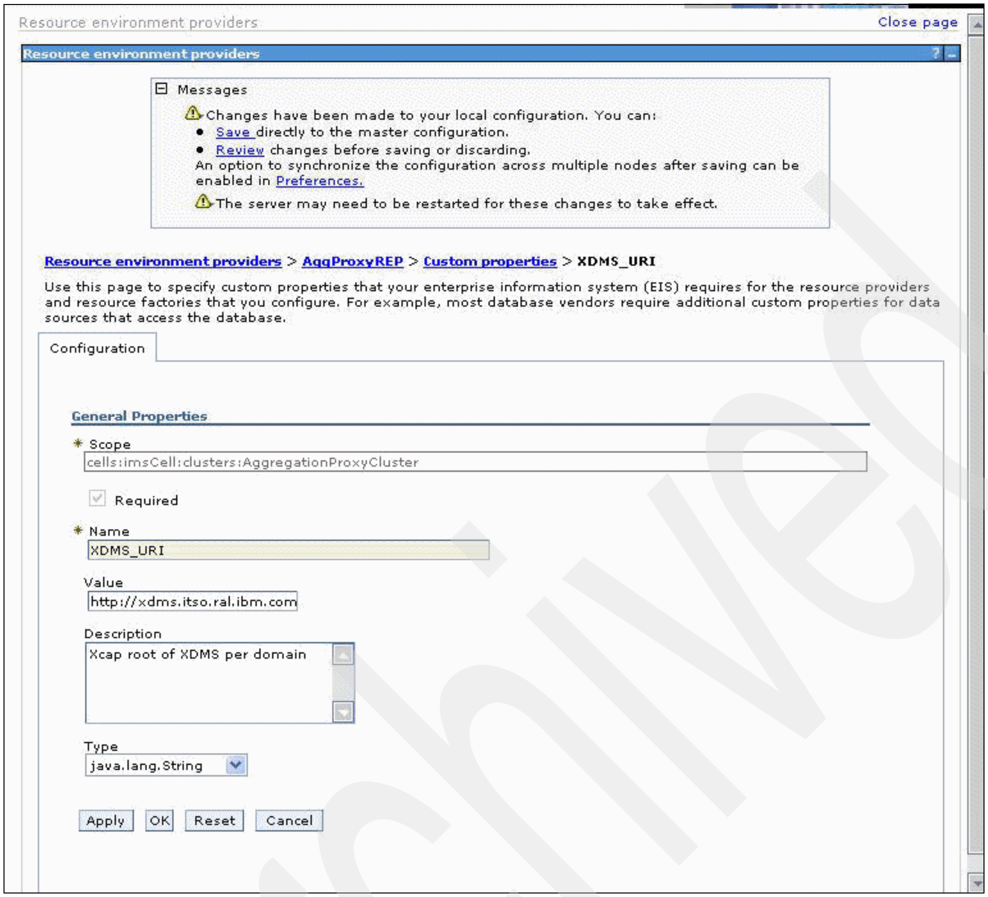


Figure 6-73 Save changes to master configuration

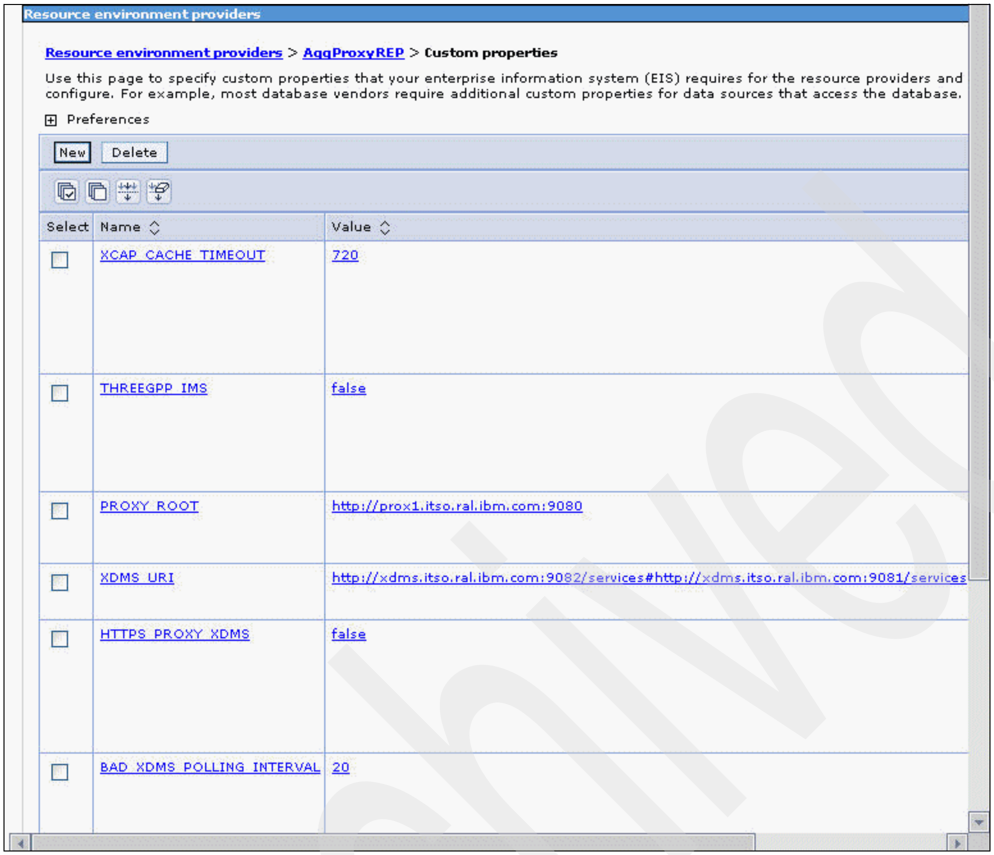


Figure 6-74 Values updated for custom properties

7

Presence Server and XDMS integration configuration

In this chapter, we describe how to configure the SIP/HTTP proxy to split incoming SIP traffic for the XDMS Shared List and Presence Rules servers.

This chapter contains:

7.1 Setting up the environment

We have to configure the SIP/HTTP proxy because in our sample scenario environment, we use the same node for both Shared List and Presence Rules servers. As a result, we have to split the incoming SIP traffic to the Proxy Server to two different Clusters on the same node. To split the traffic, we use different domain names for Shared List (xdms) and Presence Rules (rules). Using these domain names, we create SIP Routing rules on the SIP Proxy Server.

We need two different names for the Shared List and the Presence Rules Servers. So add the names **xdms.itso.ral.ibm.com** and **rules.its.ral.ibm.com** to either the DNS or hosts file, and point them both to the same IP address. Example 7-1 shows the Hosts file configuration.

Example 7-1 Hosts file configuration

```
9.42.170.225          xdms.itso.ral.ibm.com xdms rules.itso.ral.ibm.com rules
```

7.2 Proxy Server configuration

The following steps are the procedure for configuring the Proxy Server ProxNode4 to enable routing rules to split the SIP traffic to Shared List and the Presence Rules Servers. Figure 7-1 on page 269 shows the Proxy Server configuration.

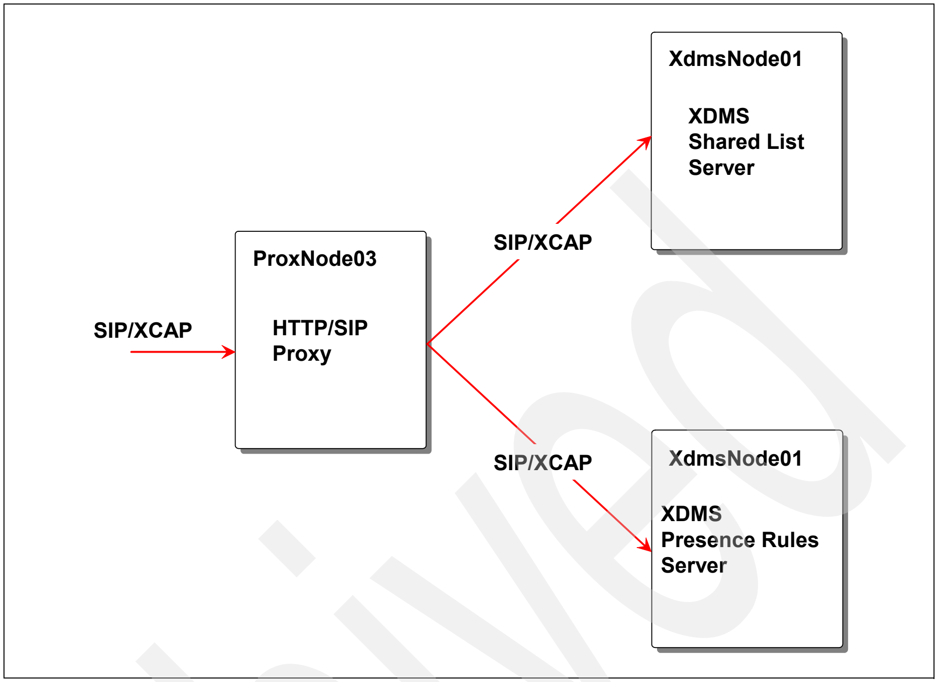


Figure 7-1 Proxy Server configuration

7.2.1 Setting up the default cluster for SIP traffic

To set up the default cluster for SIP traffic:

1. From the Integrated Solutions Console, click **Servers** → **Proxy Servers**.
2. Click **SIPProxSrv04**, as shown in Figure 7-2 on page 270.

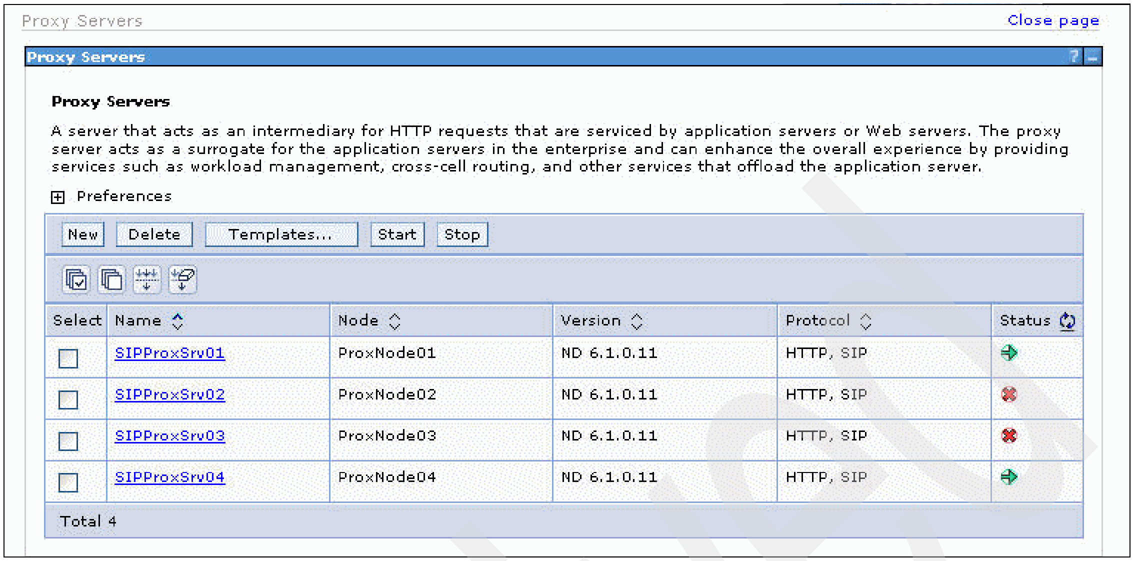


Figure 7-2 List of Proxy Servers

3. Under Proxy Settings, click **SIP Proxy Server Settings**.
4. From the Default Cluster pull-down list, select **SharedListCluster**, and click **Apply**, as shown in Figure 7-2.

Figure 7-3 XDMS SIP Proxy configuration

7.2.2 Configuring routing rules

To configure routing rules:

1. Under Additional properties, click **Routing rules**, and click **New**.
2. From the Cluster pull-down menu, select **PresenceRulesCluster**, and click **Apply**, as shown in Figure 7-4 on page 272.

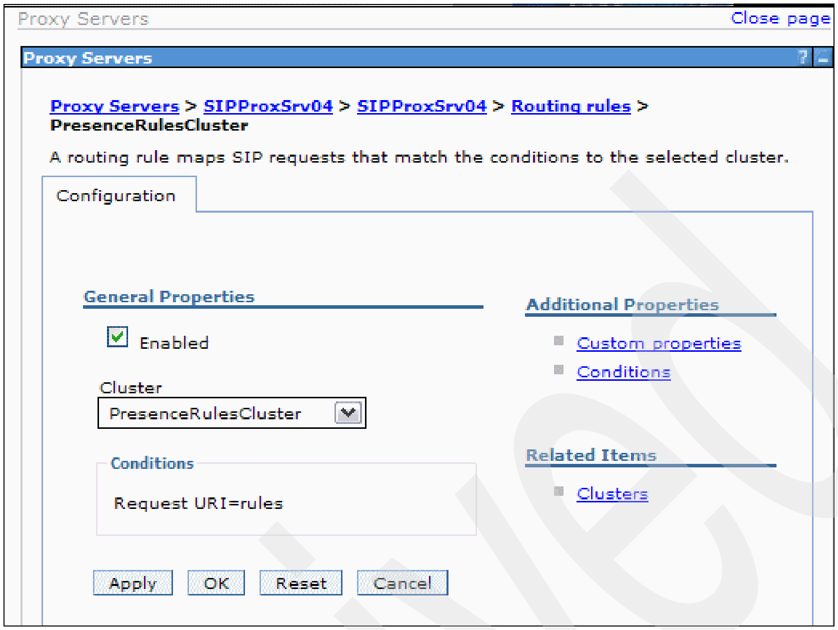


Figure 7-4 XDMS SIP Proxy configuration - Routing rules

3. Under Additional Properties, click **Conditions**, and click **New**.
4. On the new Condition page:
   a. Select **Condition type: Other**.
   b. From the Type menu, select **Request URI**.
   c. Enter **rules** for Condition value.
5. Click **OK**, as shown in Figure 7-5 on page 273.

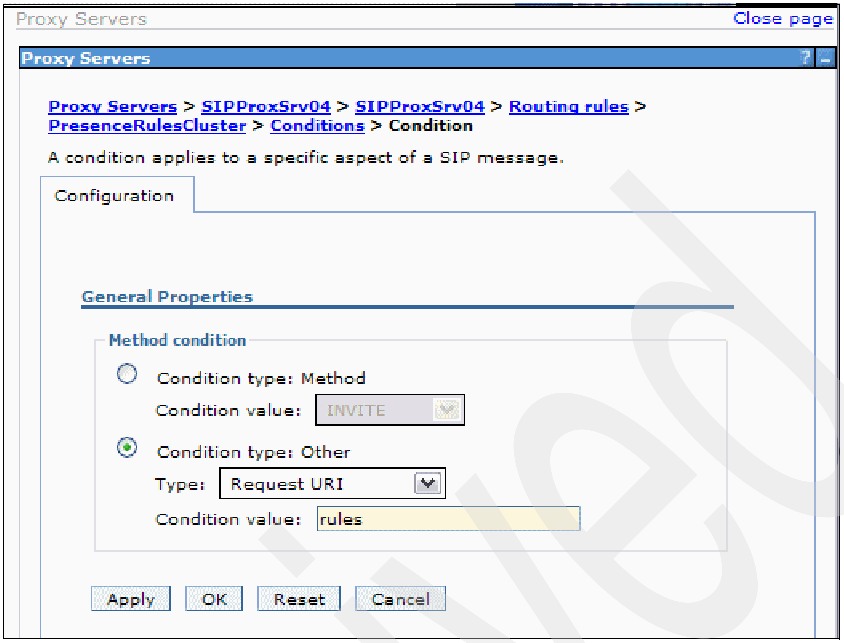


Figure 7-5 XDMS SIP Proxy configuration - Condition

7.3 Configuring the Presence Server

Available as additional materials for this book are scripts that are used to configure integration between Presence and XDMS Servers. The scripts are in the directory `.../scripts/PresenceServerConfiguration`. These scripts are:

- **SystemConfiguration_base.xml**

  Used for basic configuration without any integration.

- **SystemConfiguration_xdms_1.xml**

  Used for integration with Group List (XDMS Shared List SUBSCRIBE).

- **SystemConfiguration_xdms_2.xml**

  Used for integration with Shared List in scope of WhiteList and BlackList authorization.

- **SystemConfiguration_xdms.xml**

  Used for final configuration and integration with both XDMS Shared List and XDMS Presence Rules servers.

Note: The scripts are incrementally created and each script contains every configuration from the previous script. Run only the last script to perform the full integration.

7.3.1 Setting up connection parameters

To set up connection parameters:

1. Login as a root to the server MngNode - ldap.itso.ral.ibm.com.
2. Go to the following directory: /opt/IBM/WebSphere/AppServer/installableApps/presence/scripts/config
3. Open the ConfigurationParams.txt file for editing.
4. Set up parameters as shown in Example 7-2.

Example 7-2 Presence Server configuration - example of parameters file

```
cfg.system=/opt/IBM/WebSphere/AppServer/installableApps/presence/scripts/config/Syste
mConfiguration.xml
username=db2inst1
password=<db_passwprd>
dbConnectionString=jdbc:db2://db2.itso.ral.ibm.com:50000/PSDB
dbDriver=com.ibm.db2.jcc.DB2Driver
```

5. Save and close the file.

7.3.2 Setting up configuration parameters

You can either change the original configuration file (SystemConfiguration.xml) or use the one that we provide in the Additional Materials section (SystemConfiguration_xdms.xml). For full integration, use SystemConfiguration_xdms.xml file.

1. Open one of the following configuration files, either SystemConfiguration.xml or SystemConfiguration_xdms.xml for editing.
2. Set up the parameters using the commands in Example 7-3.

Example 7-3 Final configuration of Presence Server and XDMS Server integration

```
<groupListServer
                enable="true"
                sipAddress="sip:xdms.itso.ral.ibm.com:5060"
                fromURI="sip:superadmin@itso.ral.ibm.com"
                assertedIdentity="sip:superadmin@itso.ral.ibm.com"
```

```
                user=""
                password=""
                subscribeExpiration="63"
                retryInterval="5"
                xcapRoot=""
/>

<authorizationLists
                enable="true"
                sipAddress="sip:xdms.itso.ral.ibm.com:5060"
                fromURI="sip:superadmin@itso.ral.ibm.com"
                assertedIdentity="sip:superadmin@itso.ral.ibm.com"
                user=""
                password=""
                subscribeExpiration="63"
                retryInterval="1"
                xcapRoot=""
whiteListPath="xdms.itso.ral.ibm.com:9082/services/resource-lists/users/superadmin@it
so.ral.ibm.com/whiteList.xml"
blackListPath="xdms.itso.ral.ibm.com:9082/services/resource-lists/users/superadmin@it
so.ral.ibm.com/blackList.xml"
/>

<presenceRules
                enable="true"
                XDMSSipAddress="sip:rules.itso.ral.ibm.com:5060"
                fromURI="sip:superadmin@itso.ral.ibm.com"
                assertedIdentity="sip:superadmin@itso.ral.ibm.com"
                user=""
                password=""
                subscribeExpiration="63"
                retryInterval="5"
                xcapRoot=""
                enableMultipleIDMapping="true"
/>

<jmxListener user="wasadmin" password="<password>"/>
```

3. Save and close the file.

7.3.3 Saving the configuration to the Presence Server database

To save the configuration to the Presence Server database:

1. Run the command in Example 7-4 to update the Presence Server configuration in the database.

Example 7-4 Update Presence Server configuration in database

```
[root@ldap config]# ./configure.sh SystemConfiguration_xdms.xml
File loader created.
Property file loaded.
Property file parsed.
There are 1 elements to be configured.
XML file name:
/opt/IBM/WebSphere/AppServer/installableApps/presence/scripts/config/SystemConfigurat
ion.xml
XML source loaded from
/opt/IBM/WebSphere/AppServer/installableApps/presence/scripts/config/SystemConfigurat
ion.xml
DB manager created
Connecting with user: db2inst1
Connecting to: jdbc:db2://db2.itso.ral.ibm.com:50000/PSDB
DB connection created.
DB ResultSet created
Storing com.ibm.presence.configuration.SystemConfiguration in progress..
DB table is not empty. Updating the entry with key =
com.ibm.presence.configuration.SystemConfiguration .
SQL executed.
DB connection closed
XML stored in DB.
Done.
--------------------------------------------------------------------------------
```

7.3.4 Restarting the Presence Server Cluster

After you update the configuration in the Presence Server database, restart the Presence Server Application Server to make the changes visible:

1. From the Integrated Solutions Console, click **Servers** → **Cluster**, and select **PresenceCluster**.
2. Click **Stop**.
3. In the SystemOut.log file, check each server in the Presence Cluster to make sure they are stopped.

4. Select **PresenceCluster**, and click **Start**.
5. In the SystemOut.log file, verify that the Presence Servers start correctly, as shown in Example 7-5.

Example 7-5 Example starting log from PresNode01

```
[10/9/07 11:31:01:594 EDT] 0000000a WsServerImpl  A   WSVR0001I: Server PresSrv01
open for e-business
[10/9/07 11:31:02:026 EDT] 00000037 DiscoveryMBea I   ADMD0023I: The system
discovered process (name: nodeagent, type: NodeAgent, pid: 27772)
[10/9/07 11:31:02:181 EDT] 00000032 WorkSpaceMana A   WKSP0500I: Workspace
configuration consistency check is false.
[10/9/07 11:31:05:415 EDT] 00000038 WSChannelFram A   CHFW0019I: The Transport
Channel Service has started chain chain_1.
[10/9/07 11:31:16:741 EDT] 00000038 DomainStorage I DomainStorage
SIP_RFC3263_nameserver is missing. NAPTR resolve disabled
[10/9/07 11:31:16:753 EDT] 00000038 WSChannelFram A   CHFW0019I: The Transport
Channel Service has started chain UdpOutboundChain.
[10/9/07 11:31:16:804 EDT] 00000038 AggregatorCon C   CWSPS0204C: The default
configuration values were loaded  for
com.ibm.presence.configuration.AggregatorConfiguration
[10/9/07 11:31:16:994 EDT] 0000003e ServletWrappe I   SRVE0242I: [Presence Server]
[/siplets] [NotifySiplet]: Initialization successful.
[10/9/07 11:31:17:602 EDT] 00000038 PresenceTimer I   Initialization completed
successfully
[10/9/07 11:31:17:609 EDT] 00000038 PresenceTimer I
#--------------------- Presence Server Report ---------------------

# 1. Build Version
#Presence Server Build Number is : 'presence.62 20070930.1 [6.2.0.18]'
#
# 2. Starting System Initialization
# com.ibm.presence.init.StartupManager took 12251 millisecond
# com.ibm.presence.configuration.jmx.PresenceJMXManager took 0 millisecond
# Default init took 234 millisecond
# com.ibm.presence.pmi.PresencePMIManager took 10 millisecond
# com.ibm.presence.monitor.JMXManager took 11 millisecond
# com.ibm.presence.util.ServiceLocator took 0 millisecond
# com.ibm.presence.management.ServerIdGenerator took 0 millisecond
# com.ibm.presence.configuration.NaturalDecompositionConfiguration took 9 millisecond
# com.ibm.presence.configuration.PartialConfiguration took 0 millisecond
# com.ibm.presence.authorization.AuthorizationListsManager took 119 millisecond
# com.ibm.presence.aggregators.AggregatorManager took 32 millisecond
# com.ibm.presence.externalsource.SipExternalSourceInstaller took 2 millisecond
# com.ibm.presence.winfo.WatcherInfoManager took 8 millisecond
```

```
# com.ibm.presence.charging.ChargingManager took 17 millisecond
# com.ibm.presence.rls.RLSManager took 53 millisecond
# com.ibm.presence.authorization.AuthorizationManager took 3 millisecond
# com.ibm.presence.authorization.PresenceRulesManager took 131 millisecond
#
# 3. Presence Server Configuration Check
#Test Name: Configuration - Aggregator Configuration
#                                               Test Result:pass
#Test Name: Configuration - Presence Aggreator configuration
#                                               Test Result:pass
#Test Name: DB 1 - JDBC jndi lookup
#                                               Test Result:pass
#Test Name: DB 2 - Datasource connection
#                                               Test Result:pass
#Test Name: DB 3 - Table structure
#                                               Test Result:pass
#Test Name: DB 4 - Table structure (Managing)
#                                               Test Result:pass
#Test Name: DB 5 - Table structure (Authorization)
#                                               Test Result:pass
#Test Name: JMS 1 - TopicConnectionFactory lookup
#                                               Test Result:pass
#Test Name: JMS 2 - Topic lookup
#                                               Test Result:pass
#Test Name: JMS 3 - Topic Session Creation
#                                               Test Result:pass
#Test Name: ServiceLocator - External Source Listener Bean
#                                               Test Result:pass
#Test Name: ServiceLocator - JMS Publisher
#                                               Test Result:pass
#Test Name: ServiceLocator - Presence Services Bean
#                                               Test Result:pass
#Test Name: ServiceLocator - Presence Storage Bean
#                                               Test Result:pass
#Test Name: ServiceLocator - Register External Source Bean
#                                               Test Result:pass
#Test Name: ServiceLocator - Register Services Bean
#                                               Test Result:pass
[10/9/07 11:31:17:613 EDT] 00000038 PresenceTimer I   CWSPS0202I: Presence Server
Build Number : presence.62 20070930.1 [6.2.0.18]
```

Integration between the Presence Server and the XDMS Server is ready.

8

Testing the base scenario

In this chapter, we describe the test case scenarios and tests that validate the proper functioning of the base environment as configured.

This chapter contains:

8.1 Overview

The basis for the testing of the base environment is to ensure the accurate routing of messages and communication within the model of active servers. Table 8-1 lists the servers that we tested.

Table 8-1 List of active servers for the base environment

Host ID	Host Name
9.42.170.229	tester.itso.ral.ibm.com
9.42.170.212	prox1.itso.ral.ibm.com
9.42.170.216	agg1.itso.ral.ibm.com
9.42.170.220	ldap.itso.ral.ibm.com
9.42.170.223	prox4.itso.ral.ibm.com
9.42.170.228	xdms1.itso.ral.ibm.com

You can create Virtual Host IP and Virtual Host Names for the data that is populated.

As part of the tests, we created a resource-list and the management of the given list for presentity. The resource-list activities are tracked for creation, queries, and deletion. After the resource-list is generated in the XDMS database, we applied further presentity on the same database with SUBSCRIBE and PUBLISH activities.

8.1.1 Test case scenario 1

In this scenario, we verify the PUT for creating the resource-list, GET for query capability from the created resource-list, and to DELETE for deletion of this resource-list. The scripts run in the following sequence:

1. cr_itsoreslist.ksh for creating (PUT) the list.
2. get_itsoreslist.ksh for querying (GET) the list made through the previous PUT.
3. del_itsoreslist.ksh for deleting (DELETE) the created list.

These items are executed in conjunction with the supporting XML files of PutItsoResList.xml, GetItsoResList.xml, and DelItsoResList.xml respectively.

The three step approach of XCAP testing fulfills the need to create and manage the resource-list from the client request to the database entries.

Example 8-1 shows the script for cr_itsoreslist.ksh.

Example 8-1 Script cr_itsoreslish.ksh executes the following.

```
seagull -conf $SEAGULL_CONFIG/config/conf.client.xml -dico
$SEAGULL_CONFIG/config/xcap-dictionnary.xml -scen ./PutItsoResList.xml
-log ../logs/client.log -llevel MET
```

Figure 8-1 shows that seagull loads the respective files with the command line options from the environment variable declared for the availability location.

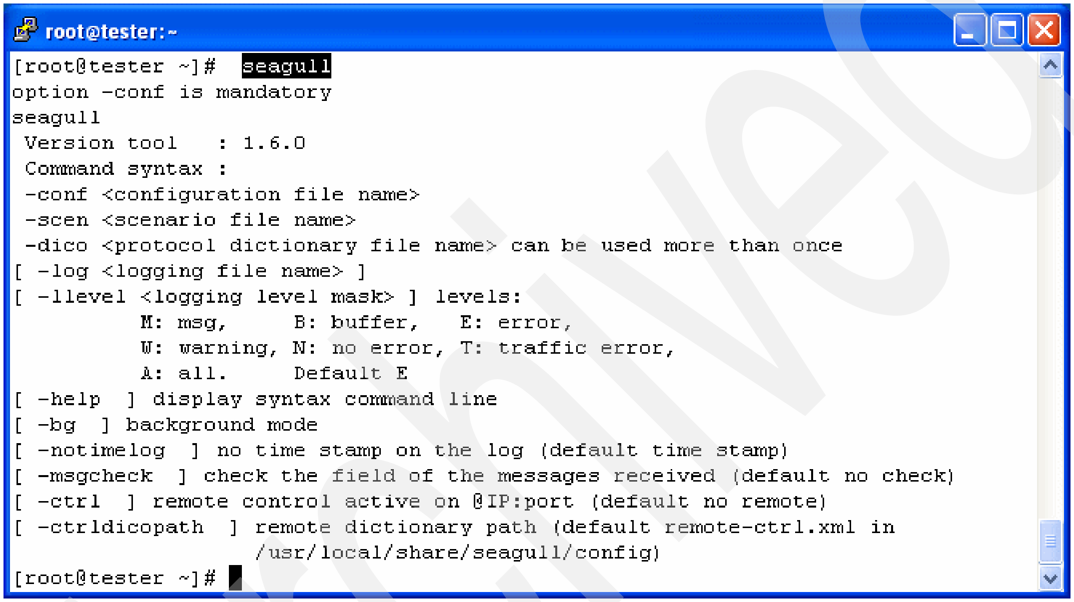


Figure 8-1 Seagull help options

Example 8-2 shows the command and XML document body in PutItsoResList.xml.

Example 8-2 Command and XML document body example in the PutItsoResList.xml

```
<message>
<!-- header -->
<![CDATA[PUT
/services/resource-lists/users/john@itso.ral.ibm.com/itsoteam.xml
HTTP/1.1
      Content-Type: application/resource-lists+xml
      X-XCAP-Asserted-Identity: "sip:john@itso.ral.ibm.com"
      Host: xdms.itso.ral.ibm.com:9082
      Connection: keep-alive
```

```
      User-Agent: seagull]]>
      <!-- body -->
      <![CDATA[<?xml version="1.0" encoding="UTF-8"?>
      <resource-lists xmlns="urn:ietf:params:xml:ns:resource-lists">
        <list name="imsitsoteam">
          <entry uri="sip:alice@itso.ral.ibm.com">
            <display-name>Alice</display-name>
          </entry>
          <entry uri="sip:bob@itso.ral.ibm.com">
            <display-name>Bob</display-name>
          </entry>
          <entry uri="sip:john@itso.ral.ibm.com">
            <display-name>John</display-name>
          </entry>
          <entry uri="sip:joe@itso.ral.ibm.com">
            <display-name>Joe</display-name>
          </entry>
          <entry uri="sip:bill@itso.ral.ibm.com">
            <display-name>Bill</display-name>
          </entry>
          <entry uri="sip:jane@itso.ral.ibm.com">
            <display-name>Jane</display-name>
          </entry>
        </list>
      </resource-lists>]]>
</message>
```

8.1.2 Test case scenario 2

In this scenario, we verify the SIP functionality for subscribing (SUBSCRIBE) to some presentity, getting notifications (NOTIFY) when the presentity is changed, and publishing to the requester’s presentity.

The subscription is through the same requester ID that we used to create the resource-list. The flow of events involves SIP proxies, Presence Server, and XDMS. The requester generates the subscription for the buddy list status, updates the status as defined by the respective person or identity in the list, and sends the publish to update the state to the list. While keeping the same subscribe request active, the requester also sends the publish with the availability condition of ‘0’ minutes, which publishes the list again to make these inactive in the list.

8.2 Managing the resource-list

This is a specification-driven activity, where the XCAP protocol that the client device uses and the target system allow requests and their corresponding responses to be generated. After the document is created, the contents are retrieved, edited, and deleted using the same mapping methodology of XML to URI.

> **Tip:** XCAP specifications are defined by IETF in RFC4825, and the manipulating presence document contents are defined in RFC4827.

The purpose of this test is to check the XCAP functionality to create (PUT), query (GET), and delete (DELETE) the resource-list using the 'RESOURCELISTS' table in the XDMS database. We used the following tools to test XCAP:

- Seagull

  It is an open source tool that generates traffic for testing and analysis. It can generate the XCAP requests from a client to the destination host. The use of Seagull was limited to testing the traffic capability of XCAP requests between the client and XMDS.

  The Seagull client creates the following logs in a defined location on the file system:

  – client.****-**-**.**-**-**.***.log
  – client-protocol-stat.xcap-protocol.****-**-**.**-**-**.***.csv
  – client-stat.****-**-**.**-**-**.***.csv

- fprint

  RPM for the RedHat Enterprise Linux 4 version is loaded to get the group systems status at a given point-of-time. It uses the Internet Control Message Protocol (ICMP) to echo the status of the host machine with the given IP address. When given a list of the hosts to be verified for the status in a single search, it uses the round-robin method to check the reachability of the server.

- Control Center

  The IBM DB2 Client v9.1.0.356 control center is used for database verifications.

8.2.1 Creating the resource-list

This test creates the resource-list, which contains a list of defined users for a requesting user. The list of the users is parsed to the resource-list as the XML document by the ascertained identity of the user who is making the request through the client application. In this case, we used the MIME type 'resource-list+xml', and the resource list server returned the resource-list document. The list structure of the users is not parsed by the PUT request because the XCAP behaves as the transport of blocked data to the target XDMS location. XDMS parses the block data for storage in the database in defined tables and sends the success or failure confirmation back to the client.

A sample list of users is added to the XML data block with five user names and their domain information. This list in turn becomes the list of users that belong to the requester. Following are the details of PUT payload used for this test and the text of 'resource-lists/users/john@itso.ral.ibm.com/itsoteam.xml' becomes the ID for these records after they are created.

Example 8-3 shows the MIME type in the XML payload.

Example 8-3 MIME type in the XML payload

```
/services/resource-lists/users/john@itso.ral.ibm.com/itsoteam.xml
HTTP/1.1 Content-Type: application/resource-lists+xml
```

> **Note:** In the base environment, SIP Interceptor is configured not to use the sip prefix in URI. We set the parameter enableMultipleIDMapping to **true**. It enables the stripping of the protocol scheme from the SIP URI while obtaining an ID. Because the request of the User ID is without schema, the same approach should apply to documents in the URI path.

The body of the text consists of the list of entries for the requester that are available in the XDMS database as the XML document in the RESOURCELIST table. Example 8-4 shows an example of the XML document.

Example 8-4 Body of user details that would be the document's content

```
<![CDATA[<?xml version="1.0" encoding="UTF-8"?>
  <resource-lists xmlns="urn:ietf:params:xml:ns:resource-lists">
      <list name="imsitsoteam">
         <entry uri="sip:alice@itso.ral.ibm.com">
            <display-name>Alice</display-name>
         </entry>
         <entry uri="sip:bob@itso.ral.ibm.com">
            <display-name>Bob</display-name>
         </entry>
         <entry uri="sip:john@itso.ral.ibm.com">
            <display-name>John</display-name>
         </entry>
         <entry uri="sip:joe@itso.ral.ibm.com">
            <display-name>Joe</display-name>
```

```
      </entry>
      <entry uri="sip:bill@itso.ral.ibm.com">
         <display-name>Bill</display-name>
      </entry>
      <entry uri="sip:jane@itso.ral.ibm.com">
         <display-name>Jane</display-name>
      </entry>
   </list>
</resource-lists>]]>
```

The XCAP request is sent from the client using the XML file as the payload to update the database. Figure 8-2 shows the sequence of messaging for XCAP PUT.

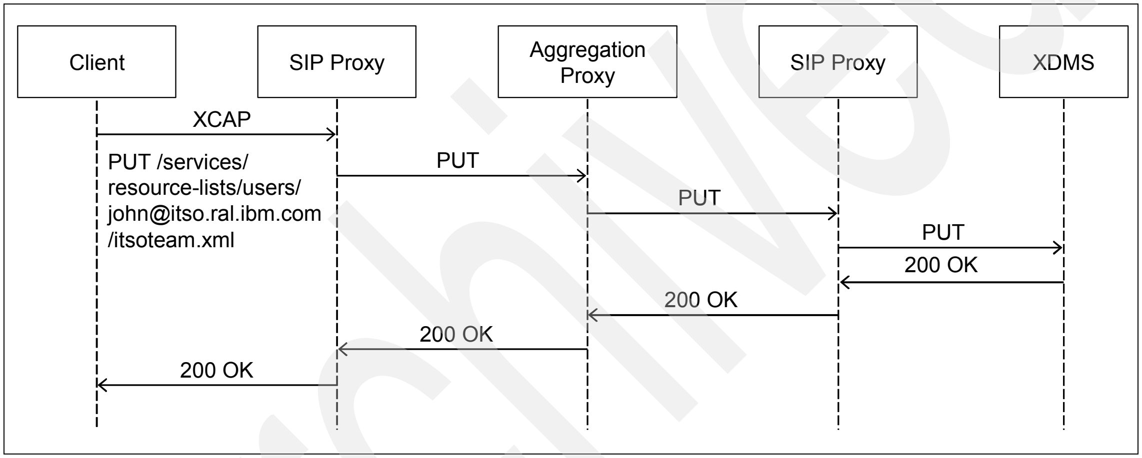


Figure 8-2 Sequence of messaging for XCAP PUT

The call flow is performed through the SIP Proxy, Aggregation proxy, SIP Proxy for XDMS, and the XDMS application. XDMS makes use of 'com.ibm.xdms.xcap.filter.impl.DomXmlProcessorXcapFilter' and 'com.ibm.xdms.function.impl.Db2Datastore' respectively.

> **Tip:** The details of the function are in the log. To see the details, modify the trace level configuration for the XDMS components using the Deployment Manager: Troubleshooting → Logs and Trace → Components.

The status of the defined servers is established to be alive before the PUT resource list, which ensures that the described servers are intact for test continuance.

Example 8-5 shows the status of the host servers.

Example 8-5 Status of the host servers

```
Thu Oct  4 17:02:02 EDT 2007

tester.itso.ral.ibm.com (9.42.170.229) is alive
lb1.itso.ral.ibm.com (9.42.170.210) is alive
prox1.itso.ral.ibm.com (9.42.170.212) is alive
agg1.itso.ral.ibm.com (9.42.170.216) is alive
lb2.itso.ral.ibm.com (9.42.170.224) is alive
ldap.itso.ral.ibm.com (9.42.170.220) is alive
prox4.itso.ral.ibm.com (9.42.170.223) is alive
xdms1.itso.ral.ibm.com (9.42.170.228) is alive

       8 targets
       8 alive
       0 unreachable
       0 unknown addresses

       0 timeouts (waiting for response)
       8 ICMP Echos sent
       8 ICMP Echo Replies received
       0 other ICMP received

0.15 ms (min round trip time)
0.92 ms (avg round trip time)
1.47 ms (max round trip time)
       0.298 sec (elapsed real time)
```

Before you perform the XCAP PUT, check the database XDMS to verify the number of entries in the RESOURCELISTS table. The information captured is the:

- Number of existing records
- Record IDs to determine that the requester's ID is not replicated. If it already exist in the table, the request would fail from the client.

Verify the number of records in RESOURCELIST table before sending the PUT request. Figure 8-3 on page 287 shows an example.

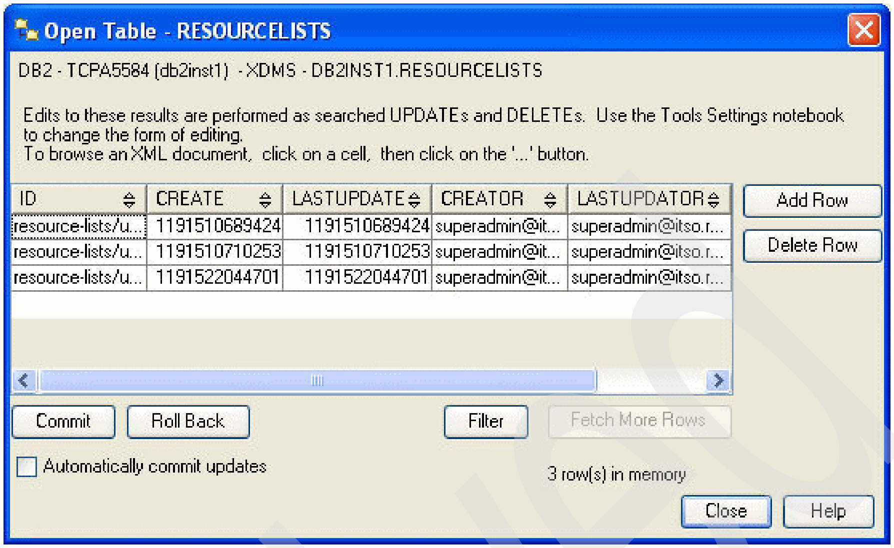


Figure 8-3 Current entries in the RESOURCELIST table

Running the PUT test scripts

Run the script cr_itsoresourcelist.ksh and observe the results:

- The window-based output to check the status of the request whether success or failure.
- The logs created by Seagull

The sequence of events are as follows:

1. The script generated the request for creation of the resource-list for the requester ID.
2. The corresponding XDMS instance responded with 200 OKs after the successful record insertion for the given document contents.

Figure 8-4 on page 288 shows the successful list insertion.

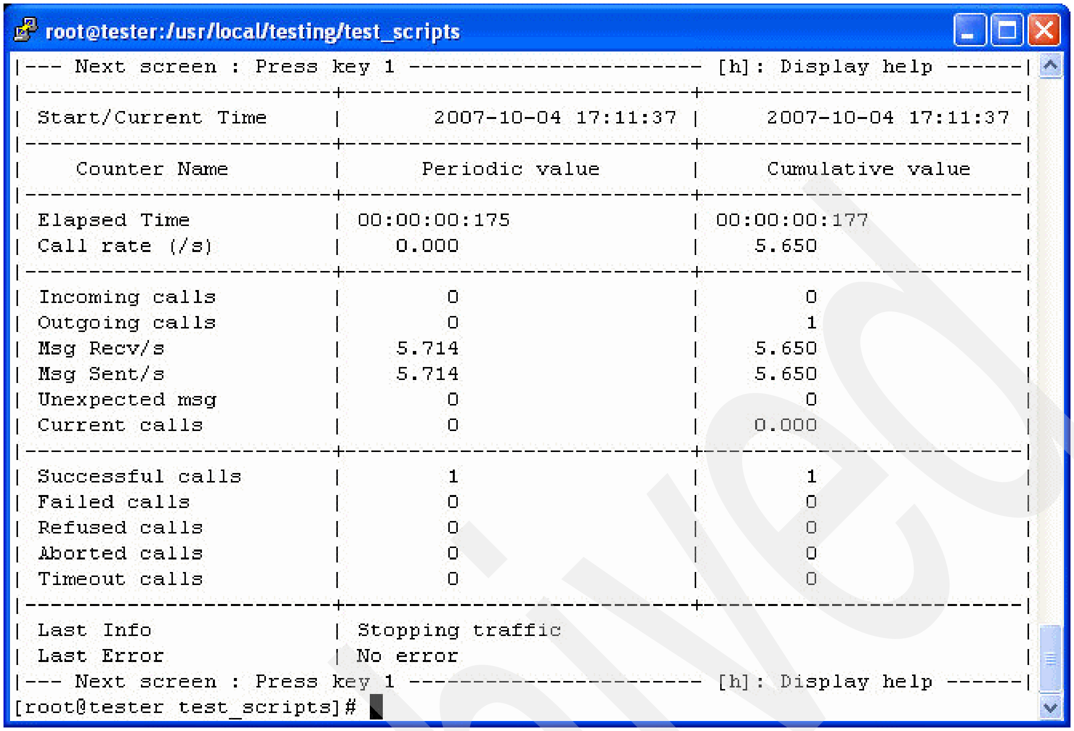


Figure 8-4 Successful list insertion

Verifying the PUT test script's run result

Subsequently, use the DB2 client to search the RESOURCELIST table to verify the results. The database updates are done with the addition of records with the ID '/services/resource-lists/users/john@itso.ral.ibm.com' who is the requester for the test list creation.

Figure 8-5 on page 289 shows an additional record that is added in the RESOURCELIST table.

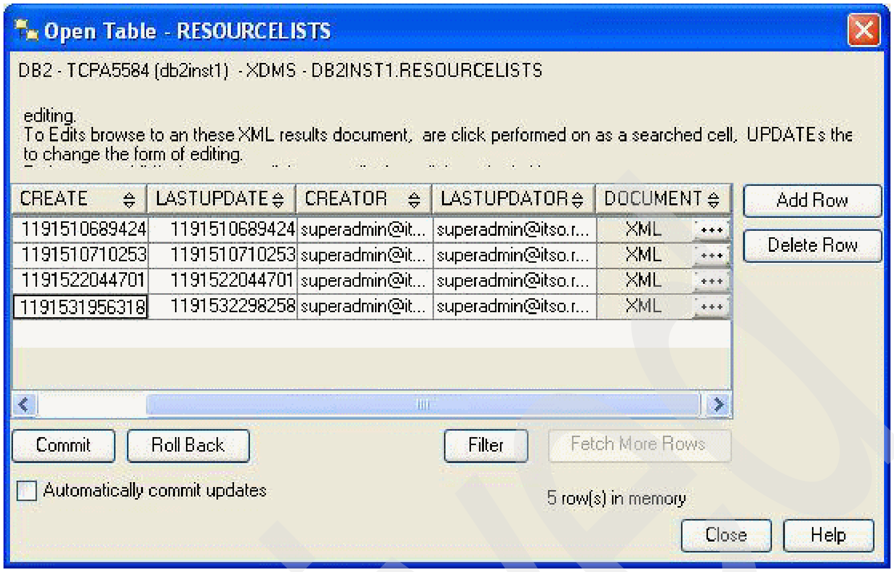


Figure 8-5 Additional record added in RESOURCELIST table

Check the XML column of the inserted record to verify that the contents match the list that was submitted in the XML body of the payload, which the Seagull client application (XCAP client request) sent.

Figure 8-6 on page 290 shows the content of the added list.

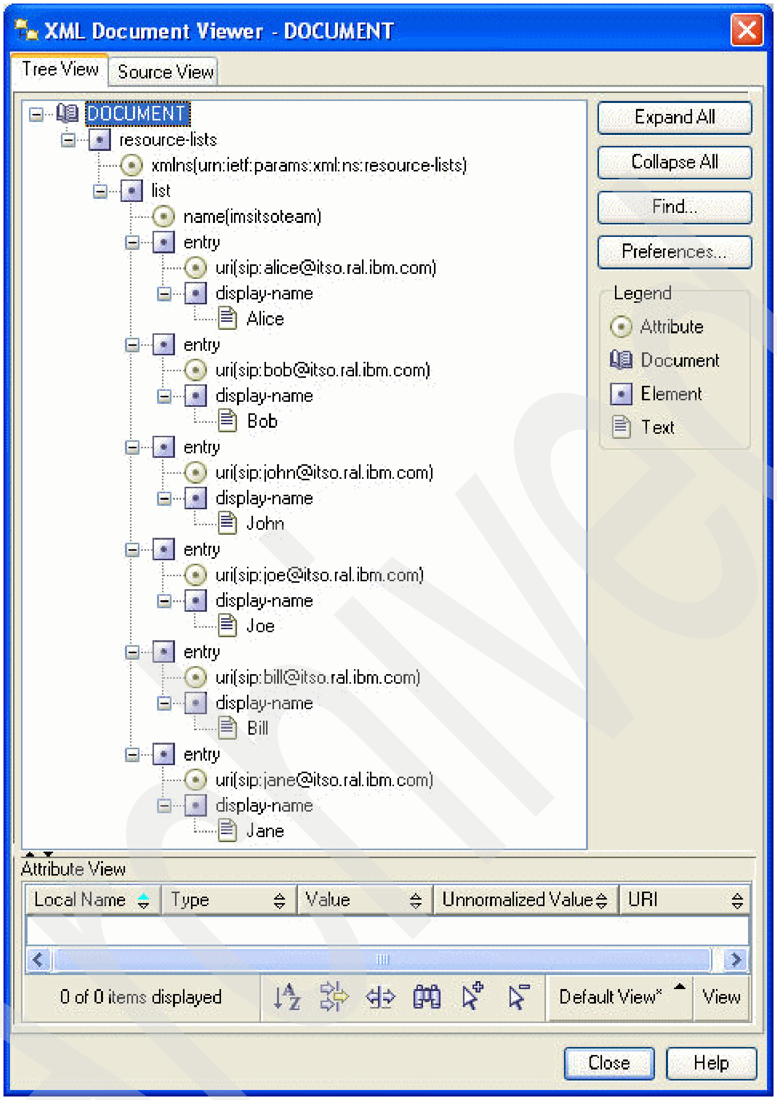


Figure 8-6 Content of added list

8.2.2 Query resource-list

This test queries the resource-list of defined users for a requesting user. This is the list that was created earlier by the PUT request using the same requester ID.

Figure 8-7 on page 291 shows the sequence of messaging for XCAP GET.

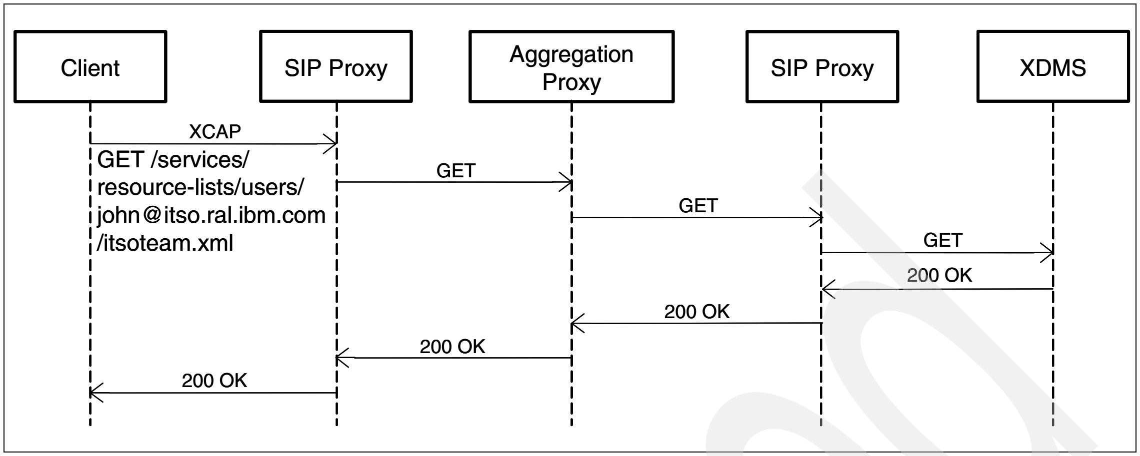


Figure 8-7 Sequence of messaging for XCAP GET

Verify that the defined servers are up and running before the GET resource list to make sure that the servers are intact for test continuance, as shown in Example 8-6.

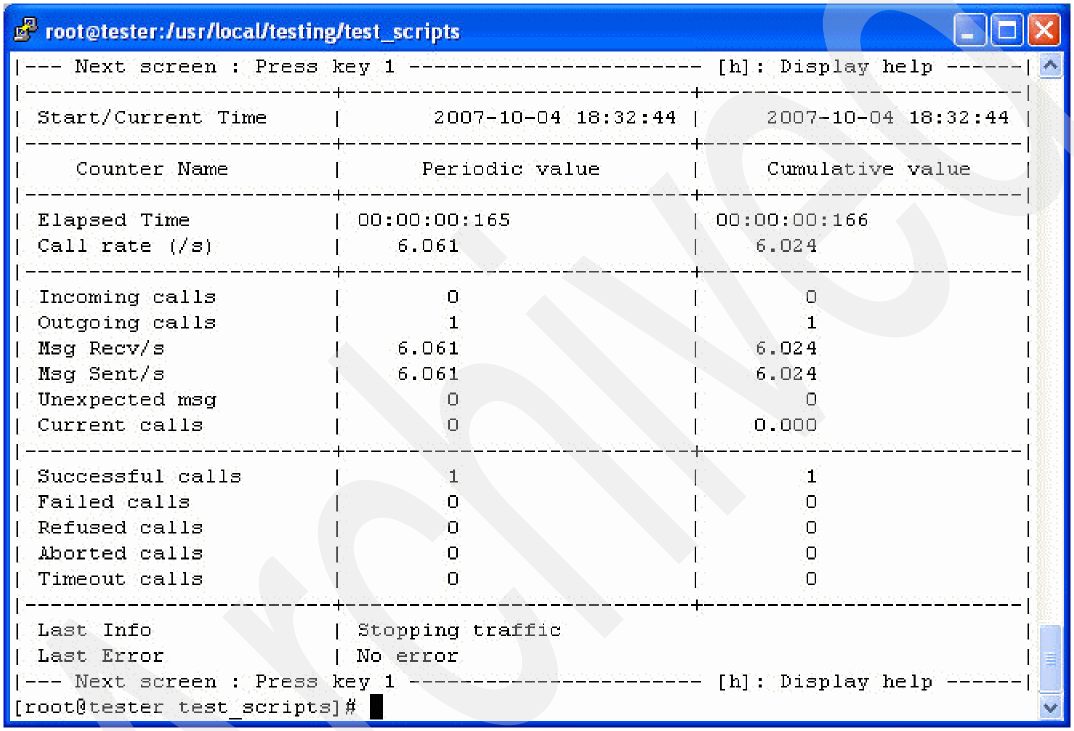
Example 8-6 Status of the host servers

```
Thu Oct  4 18:26:35 EDT 2007

tester.itso.ral.ibm.com (9.42.170.229) is alive
lb1.itso.ral.ibm.com (9.42.170.210) is alive
prox1.itso.ral.ibm.com (9.42.170.212) is alive
agg1.itso.ral.ibm.com (9.42.170.216) is alive
lb2.itso.ral.ibm.com (9.42.170.224) is alive
ldap.itso.ral.ibm.com (9.42.170.220) is alive
prox4.itso.ral.ibm.com (9.42.170.223) is alive
xdms1.itso.ral.ibm.com (9.42.170.228) is alive

       8 targets
       8 alive
       0 unreachable
       0 unknown addresses

       0 timeouts (waiting for response)
       8 ICMP Echos sent
       8 ICMP Echo Replies received
       0 other ICMP received

0.17 ms (min round trip time)
```

```
0.78 ms (avg round trip time)
1.48 ms (max round trip time)
       0.294 sec (elapsed real time)
```

Running the GET test scripts

Run the script get_itsoresourcelist.ksh, and check the status of the request by the client response. The success or failure is displayed on the window with the other correlated information from the Seagull logs, as shown in Figure 8-8.




```
|--- Next screen : Press key 1 ----------------------- [h]: Display help ------|
|------------------------+---------------------------+-------------------------|
| Start/Current Time     |       2007-10-04 18:32:44 |     2007-10-04 18:32:44 |
|------------------------+---------------------------+-------------------------|
|    Counter Name        |      Periodic value       |     Cumulative value    |
|------------------------+---------------------------+-------------------------|
| Elapsed Time           | 00:00:00:165              | 00:00:00:166            |
| Call rate (/s)         |    6.061                  |    6.024                |
|------------------------+---------------------------+-------------------------|
| Incoming calls         |         0                 |         0               |
| Outgoing calls         |         1                 |         1               |
| Msg Recv/s             |    6.061                  |    6.024                |
| Msg Sent/s             |    6.061                  |    6.024                |
| Unexpected msg         |         0                 |         0               |
| Current calls          |         0                 |    0.000                |
|------------------------+---------------------------+-------------------------|
| Successful calls       |         1                 |         1               |
| Failed calls           |         0                 |         0               |
| Refused calls          |         0                 |         0               |
| Aborted calls          |         0                 |         0               |
| Timeout calls          |         0                 |         0               |
|------------------------+---------------------------+-------------------------|
| Last Info              | Stopping traffic                                    |
| Last Error             | No error                                            |
|--- Next screen : Press key 1 ----------------------- [h]: Display help ------|
[root@tester test_scripts]#
```

Figure 8-8   Status of the servers

Verifying the GET test script's run result

The Seagull log files are the best means for verifying the output of this request. It is therefore essential to understand the log files that Seagull creates. The contents are captured as the GET and 200 OKs in the response. Analyze it for correctness against the list that the same requester previously submitted.

Here is the sequence of events:

1. The script submitted the request for query of the resource-list for the previously created requester ID.

2. The XDMS instance responded with the 200 OKs after the successful query for the given document contents.

Example 8-7 shows the log that was generated for the resource-list query.

Example 8-7 Log generated for resource-list query

```
2007-10-04.18:32:44.103|M|Send GET

/services/resource-lists/users/john@itso.ral.ibm.com/itsoteam.xml
HTTP/1.1..Content-Type: application/resource-lists+xml..X-XCAP-Asserted-Identity:
"sip:superadmin@itso.ral.ibm.com"..Host: xdms.itso.ral.ibm.com:9082..Connection:
keep-alive..User-Agent: seagull..content-length: 0..

2007-10-04.18:32:44.215|M|Received

HTTP/1.1 200 OK. .Last-Modified:  Thu, 04 Oct 2007  21:11:38 GMT..C ontent-Type: app
lication/resourc e-lists+xml; cha rset=utf-8..Cont ent-Length: 627..ETag: 119153229
8258..Content-La nguage: en-US..S et-Cookie:
LtpaToken2=bz4GQDOzlvORs2CZWXJ+JEyFaseCKv3eM31O66yRcDnghm5nq5iNEXUSJfNWygGN7Z5TDmpVbG
x6G5x3qqBSOltAfaSPMQ5F3BnyCaYhjHyziD2dVtNuJSfZD3lorgiWLHEItRvMn1LSvwwCm8rNuhAqbdDX2rS
IhnIGjZ4SI7H/8AIqCNJQOKaQkr3GAyRoWkeTNm4Tx76UalsL6iATE3okk1sW65ZPKB8coRNaoOFHlSnlnjCC
7Y6OMtaUgFeHbv98OQ/5kqjbeMdHEyOg/zITaAG7wqbfLGJsdFl7AwH9b/9naUPylr6oWoqj4SeQ1Ig6CVrz9
G39b4s3AbG/M/9MTcbHP7tRqqLhfL1UxZwsJi6GPSxswxaJuAagzWhJgCiNwlnwdj6JMK+5jNPS2r3+67lONy
ebtQtvXfD9DpOl5WpnOiHI3pGUpglSvOAMtiWxmjN/z9ccALz/xdxglh7RlrvVmb1QiJoQfvswnuKzbv/s7iZ
JSTHKHlNoyYX7STXheO/jcteEYjZFNFKUH6yEIzoHSbkQnCyHFaONJpR36d/8P7uIf15pV/UHebxTI8weqIQC
LwmTDZiH8VQPwV6iTEOY7WuOSNtC1fi5lKI9MjYGVtigIcivdiYLsicpzDEjsoBcfRO2DI1djq+mtw==;
Path=/; Domain=.itso.ral.ibm.com..Set-Cookie:
LtpaToken=fb2w8u28nLHKyhtpGhriij6jcegwQloUArMqgDCALtm4FoPkXwtTWeh26p4y5pKV9rD/NMLUPIv
PzCmyguQJ+HopoHqTxy+ONVQ/SOZ/C7riEal5YOa2KhXhmln1MwO3PjRFILTStWxYV9DhvZCGVe/pTbkOJ4EM
mXMbY4z3t1SRECSg6Ia4rDx6SynKwgd4iaZDOi5MsC9YtKwTNmHCez/oI7bluGBYOciI7WlZHI+il6uNKIdB+
cfe8cRRPO7dOdqPB5Ossn+F6qqq/5WOQPDslhRCouSX/HMqdZ5cdKI=; Path=/;
Domain=.itso.ral.ibm.com..Date: Thu, 04 Oct 2007 2 2:27:00 GMT..Server: Proxy/1.0..
Expires: Thu, 01  Dec 1994 16:00: 00 GMT..Cache-Control: no-cache= "set-cookie,set
-cookie2"..<?xml version="1 .0" encoding="UT F-8"?><resource-lists xmlns="urn
:ietf:params:xml :ns:resource-lis ts"><list name=" imsitsoteam"><entry uri="sip:ali
ce@itso.ral.ibm. com"><display-name>Alice</display-name></entry><entry
uri="sip:bob@itso.ral.ibm.com"><display-name>Bob</display-name></entry><entry
uri="sip:john@itso.ral.ibm.com"><display-name>John</display-name></entry><entry
uri="sip:joe@itso.ral.ibm.com"><display-name >Joe</display-name></entry><entry
uri="sip:bill@itso.ral.ibm.com "><display-name>Bill</display-name></entry><entry
uri="sip:jane@itso.ral.ibm.com
"><display-name>Jane</display-name></entry></list></resource-lists>
2007-10-04.18:32:49.104 STOP
```

8.2.3 Removing the resource-list

This test deletes the previously created entries by the same user in the resource-list. Figure 8-9 shows the sequence of messaging for XCAP DELETE.

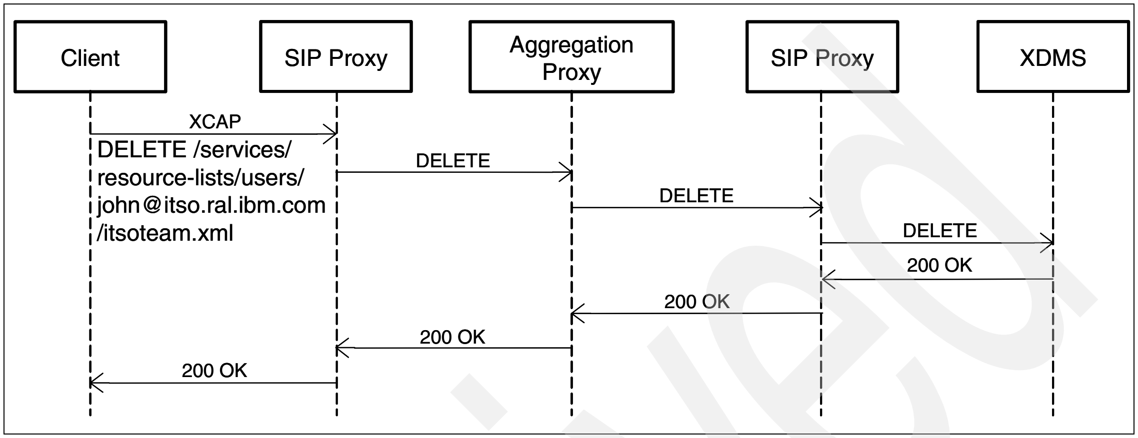


Figure 8-9 Sequence of messaging for XCAP DELETE

The status of the defined servers is established to be alive before the DELETE resource list. Example 8-8 shows the status of the host servers.

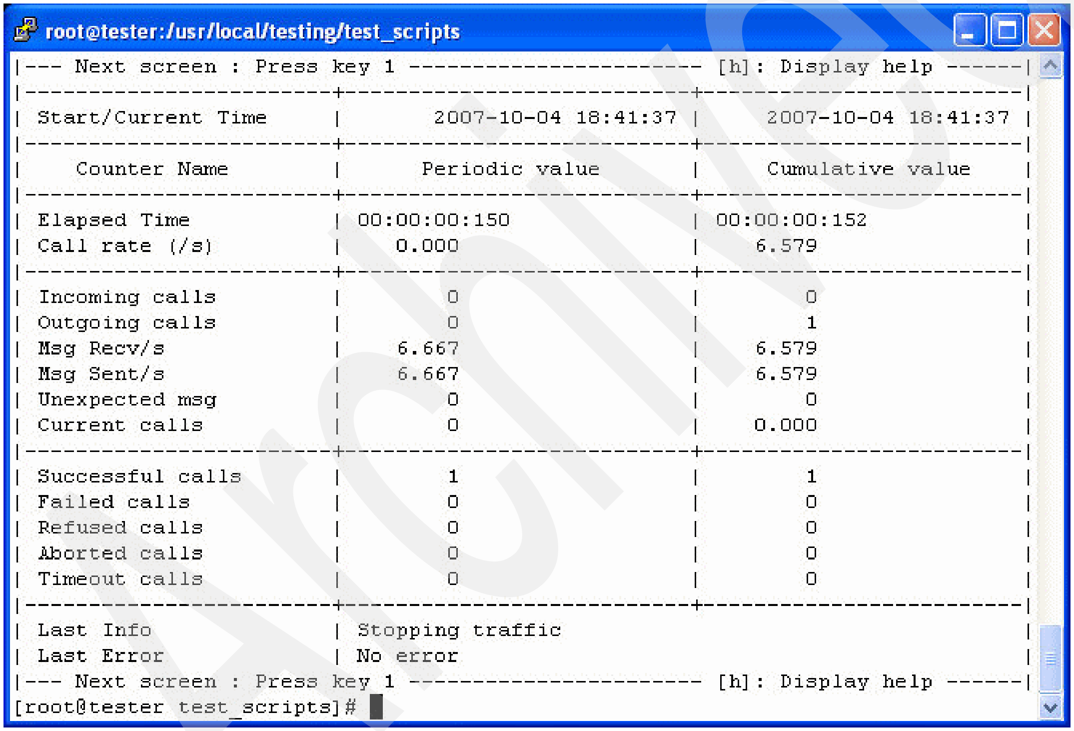
Example 8-8 Status of the host servers

```
Thu Oct  4 18:34:45 EDT 2007

tester.itso.ral.ibm.com (9.42.170.229) is alive
lb1.itso.ral.ibm.com (9.42.170.210) is alive
prox1.itso.ral.ibm.com (9.42.170.212) is alive
agg1.itso.ral.ibm.com (9.42.170.216) is alive
lb2.itso.ral.ibm.com (9.42.170.224) is alive
ldap.itso.ral.ibm.com (9.42.170.220) is alive
prox4.itso.ral.ibm.com (9.42.170.223) is alive
xdms1.itso.ral.ibm.com (9.42.170.228) is alive

       8 targets
       8 alive
       0 unreachable
       0 unknown addresses

       0 timeouts (waiting for response)
       8 ICMP Echos sent
       8 ICMP Echo Replies received
```

```
       0 other ICMP received

0.14 ms (min round trip time)
0.17 ms (avg round trip time)
0.20 ms (max round trip time)
       0.289 sec (elapsed real time)
```

Running the DELETE test scripts

Run the script del_itsoresourcelist.ksh, and observe the success or failure from the window output from the Seagull client request. Verification is also done with the logs that are created for this request. Figure 8-10 shows the status of the host servers.




```
|--- Next screen : Press key 1 ---------------------- [h]: Display help ------|
|------------------------+---------------------------+------------------------|
| Start/Current Time     |       2007-10-04 18:41:37 |    2007-10-04 18:41:37 |
|------------------------+---------------------------+------------------------|
|    Counter Name        |      Periodic value       |    Cumulative value    |
|------------------------+---------------------------+------------------------|
| Elapsed Time           | 00:00:00:150              | 00:00:00:152           |
| Call rate (/s)         |    0.000                  |    6.579               |
|------------------------+---------------------------+------------------------|
| Incoming calls         |        0                  |        0               |
| Outgoing calls         |        0                  |        1               |
| Msg Recv/s             |    6.667                  |    6.579               |
| Msg Sent/s             |    6.667                  |    6.579               |
| Unexpected msg         |        0                  |        0               |
| Current calls          |        0                  |    0.000               |
|------------------------+---------------------------+------------------------|
| Successful calls       |        1                  |        1               |
| Failed calls           |        0                  |        0               |
| Refused calls          |        0                  |        0               |
| Aborted calls          |        0                  |        0               |
| Timeout calls          |        0                  |        0               |
|------------------------+---------------------------+------------------------|
| Last Info              | Stopping traffic                                   |
| Last Error             | No error                                           |
|--- Next screen : Press key 1 ---------------------- [h]: Display help ------|
[root@tester test_scripts]#
```

Figure 8-10 Status of the host servers

Verifying the DELETE test script's run result

Use the DB2 client application to inspect the number of entries in the RESOURCELISTS table after running the delete list request. Verify that the additional record created by the PUT for ID '/services/resource-lists/users/john@itso.ral.ibm.com' is successfully deleted.

The delete request is generated by the same requester ID for the previously created entries.

Here is the sequence of events:

1. The request is submitted for deletion of the resource-list for the previously created requester ID.
2. The XDMS instance responds with 200 OKs after successfully deleting the given document contents.

Figure 8-11 shows the entries that are verified.

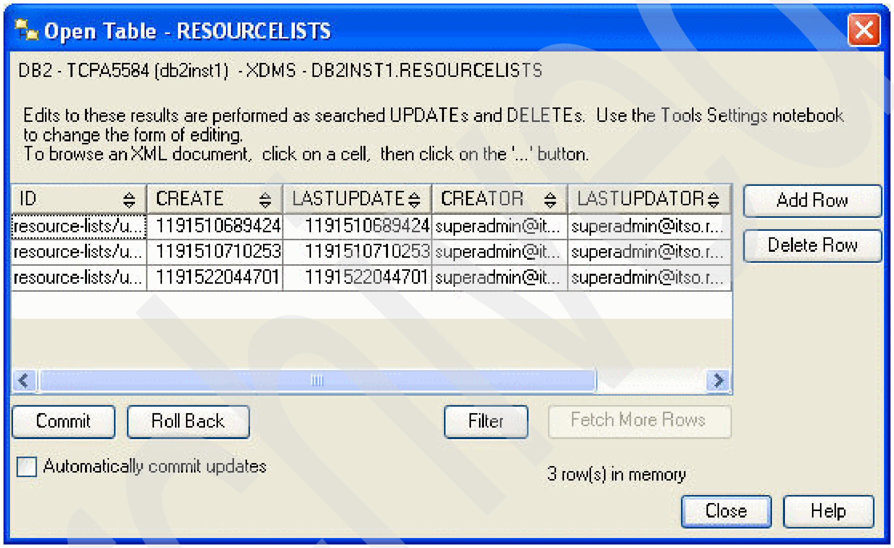


Figure 8-11 Current entries verified

8.3 Managing presentity

SIPp generates the requests. SIPp is an open source tool that can generate the SIP requests from a client to the destination host. It was used to test the traffic capability of the SIP request from the client to Presence Server and subsequently the expected behavior between Presence Server and XMDS.

The SIPp client creates the following log files in the specified location on the file system:

- scen1_sub1_****_message.log
- scen1_pub1_****.log

Before you run the SIP test scripts, verify the status of the Application Servers, as follows:

1. From the Integrated Solutions Console, click **Servers** → **Application Servers**. See Figure 8-12.

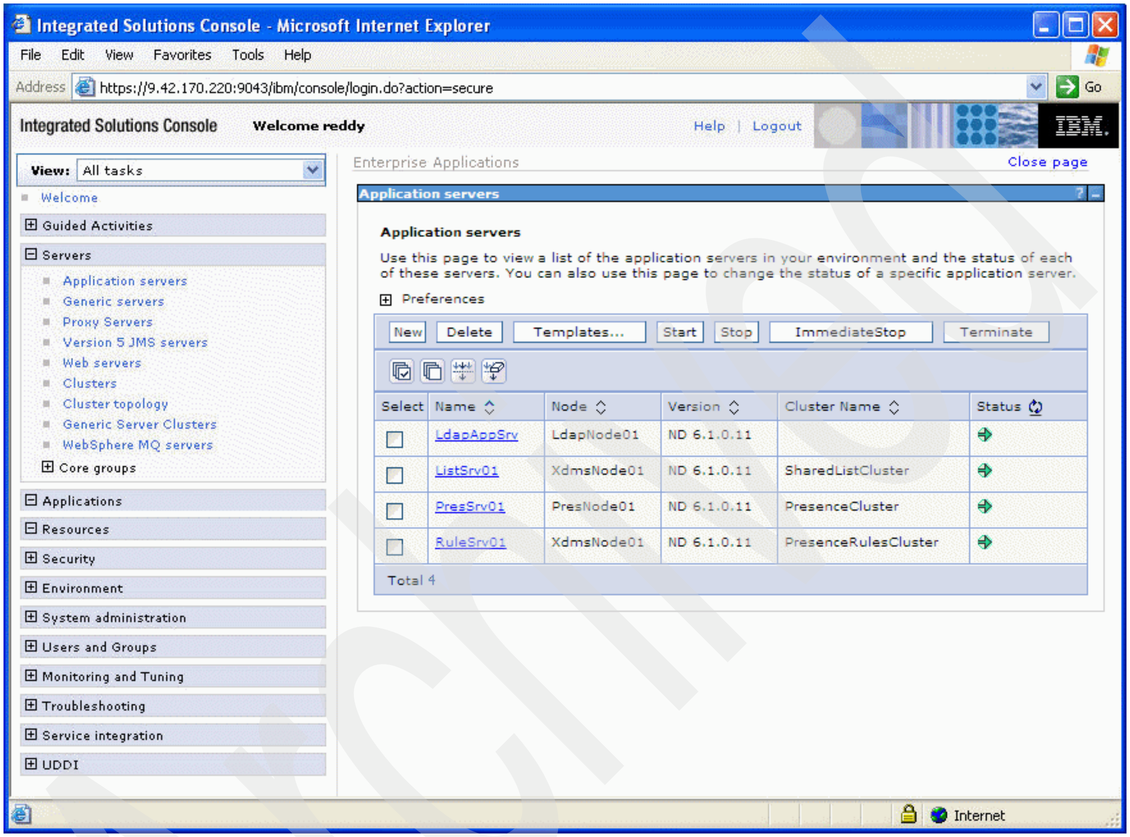


Figure 8-12 Application Servers status for the base environment (01 XCAP_Test_ApplicationStatus.gif)

Running the SIP SUBSCRIBE test scripts

The script executes the request by sending the SIP SUBSCRIBE request for the given resource-list from the requester as the identity. Here is the sequence of events:

1. The Client sends the request for SUBSCRIBE.
2. The Presence Server receives the request, sends the 200 OKs as a response, and then recreates the subscribe to XDMS.
3. The XDMS resource-list server responds to the Presence Server with 200 OKs.

4. After the notification process completes, Presence Server creates an XCAP request to the XDMS resource-list to query the available resource-list server.

This covers the first stage of communication between the Presence Server and the XDMS resource-list server. Figure 8-13 shows the sequence of messaging for SIP SUBSCRIBE for the resource-list in XDMS.

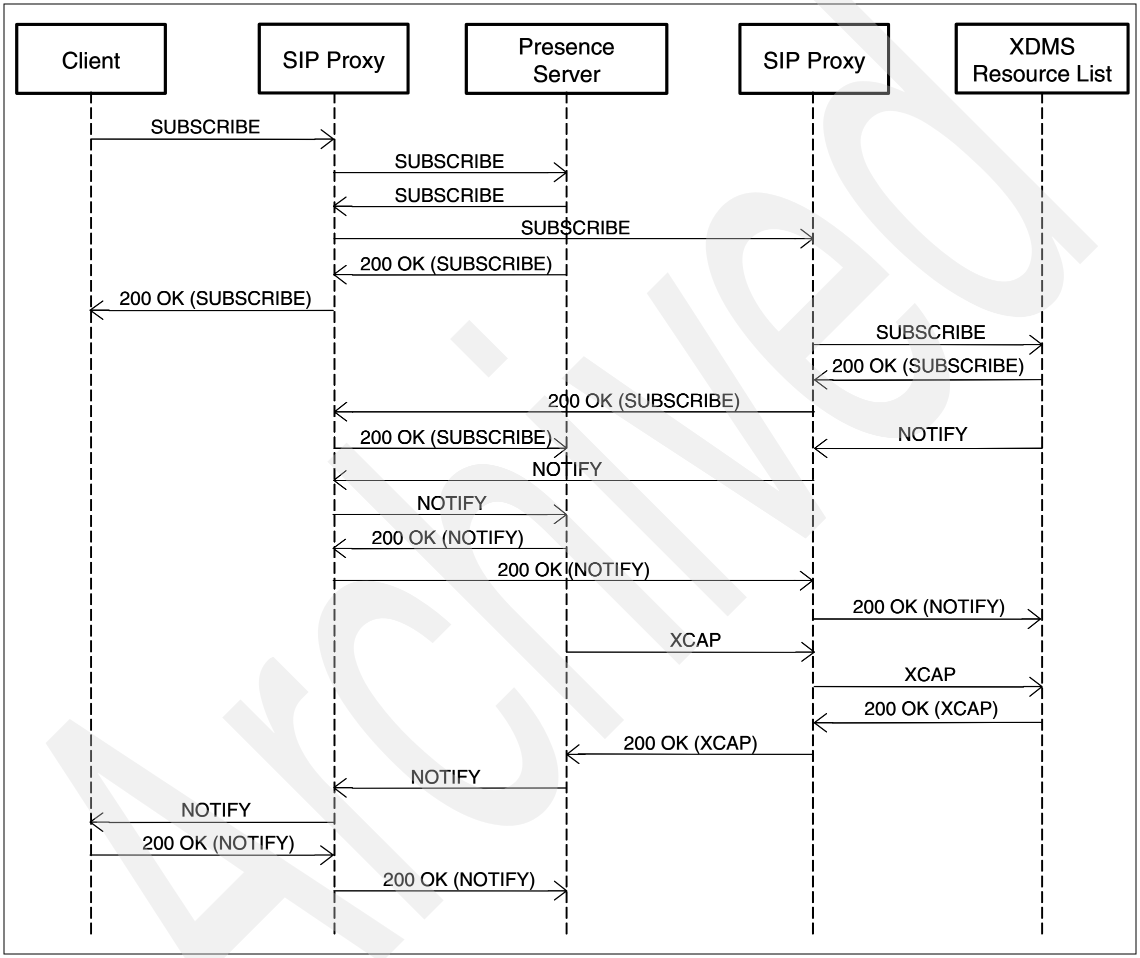


Figure 8-13 Sequence of messaging for SIP SUBSCRIBE for Resource List in XDMS

When the notification between the XDMS resource-list server and the Presence Server are complete, PUBLISH needs to occur for the users in the resource-list. Here is the sequence of events:

1. The Presence Server sends a SUBSCRIBE to the XDMS Presence Rules to ensure that the status changed for the specific user in the list

2. After the notification process is completed, the XCAP request is sent to the XDMS Presence Rules from the Presence Server.
3. This request is sent for the identity of the user that requested the change in availability.
4. The status is updated with the value that was submitted, for example, “available” or “busy”.
5. The above request and response process is repeated for the number of users that are listed in the resource-list document for the requester.

Figure 8-14 shows the sequence of messaging for SIP SUBSCRIBE for the Presence Rules in XDMS.

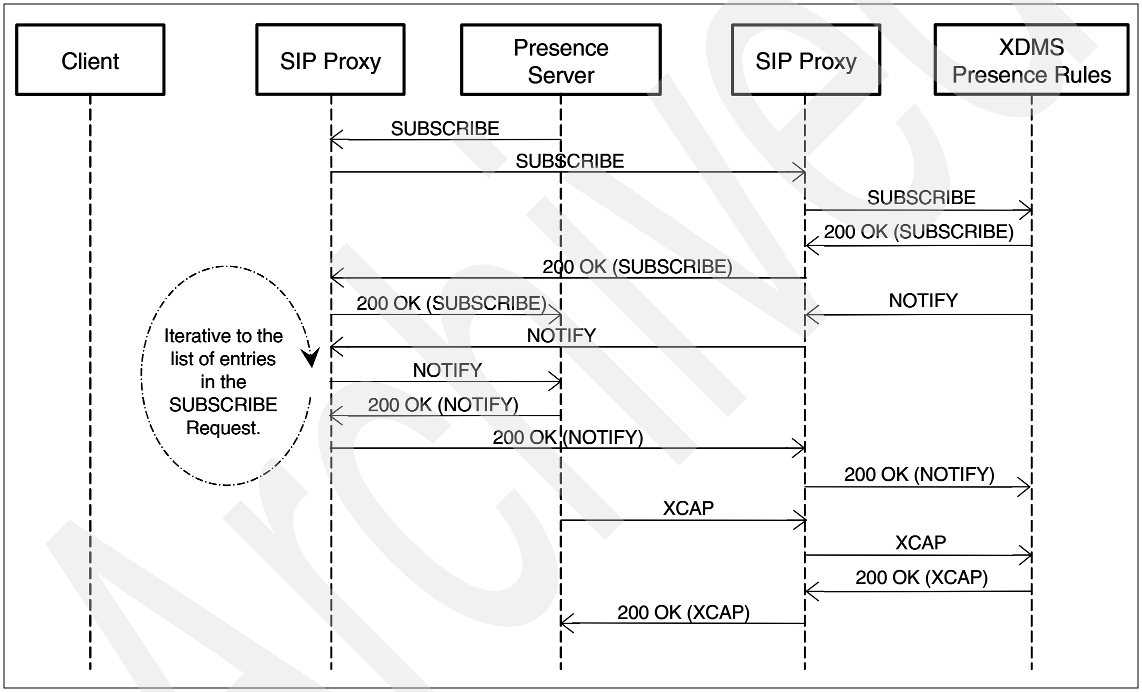


Figure 8-14 Sequence of messaging for SIP SUBSCRIBE for Presence Rules in XDMS

Example 8-9 shows a sample test that is supplied through SIPp for the SUBSCRIBE for the requester trying to establish the availability of the group in the list.

Example 8-9 Payload file body for sending the SUBSCRIBE command

```
<send crlf="true">
    <![CDATA[
```

```
      SUBSCRIBE sip:john@itso.ral.ibm.com SIP/2.0
      From: <sip:superadmin@itso.ral.ibm.com>;tag=[call_number]
      To: <sip:john@itso.ral.ibm.com>
      Via: SIP/2.0/[transport]
[local_ip]:[local_port];rport;branch=[branch]
      Call-ID: [call_id]
      Cseq: [cseq] SUBSCRIBE
      Contact:
<sip:superadmin@[local_ip]:[local_port];transport=[transport]>
      Content-Length: 0
      Event: presence
      Expires: 60
      Max-Forwards: 70
      P-Asserted-Identity: <sip:alice@itso.ral.ibm.com>
    ]]>
</send>
```

Example 8-10 is the body text that is supplied through SIPp to start publishing the presence information for each of the group members in the list.

Example 8-10 Payload file body for sending the publish command

```
<send>
    <![CDATA[
      PUBLISH sip:john@itso.ral.ibm.com SIP/2.0
      From: <sip:john@itso.ral.ibm.com>;tag=[call_number]
      To: <sip:john@itso.ral.ibm.com>
      Max-Forwards: 70
      Content-Type: application/pidf+xml
      Via: SIP/2.0/[transport] [local_ip]:[local_port];
      rport;branch=[branch]
      CSeq: [cseq] PUBLISH
      Call-ID: [call_id]
      Expires: 1000
      Event: presence
      Require: Simple
      Content-Length: [len]
      P-Asserted-Identity: <sip:john@itso.ral.ibm.com>
      <?xml version="1.0" encoding="UTF-8"?>
      <presence xmlns="urn:ietf:params:xml:ns:pidf"
      xmlns:rpid="urn:ietf:params:xml:ns:pidf:rpid"
      mlns:dm="urn:ietf:params:xml:ns:pidf:data-model"
      entity="sip:john@itso.ral.ibm.com">
      <tuple id="449889">
          <status>
```

```
            <basic>open</basic>
          </status>
          <dm:deviceID>urn:device:0003ba4811e3</dm:deviceID>
          <rpid:class>email</rpid:class>
          <dm:person id="1234">
            <rpid:sphere>office</rpid:sphere>
          </dm:person>
          <dm:note>In a meeting</dm:note>
          <contact>mailto:john@in.ibm.com</contact>
      </tuple>
      </presence>
    ]]>
</send>
```

Verifying the SIP SUBSCRIBE test script's run result

Execute the script to perform the SUBSCRIBE and the PUBLISH activity to the requested ID, and the logs verify the complete flow of interfacing between the client and SIP Proxy, and the Presence and XDMS Servers.

The results are logged in the following files:

- scen1_sub1_****_message.log
- scen1_pub1_****.log

The following events are logged to scen1_sub1_****_message.log file:

1. The client sends a SUBSCRIBE request and receives a 202 OK response.
2. The client receives a NOTIFY request, which indicates that 'Subscription-Stat' is active.
3. The client receives a NOTIFY request, which includes the status information of the subscribed user.
4. The client sends PUBLISH requests, which includes the status of the user in the resource-list in sequence of each other and receives a 200 OK response.

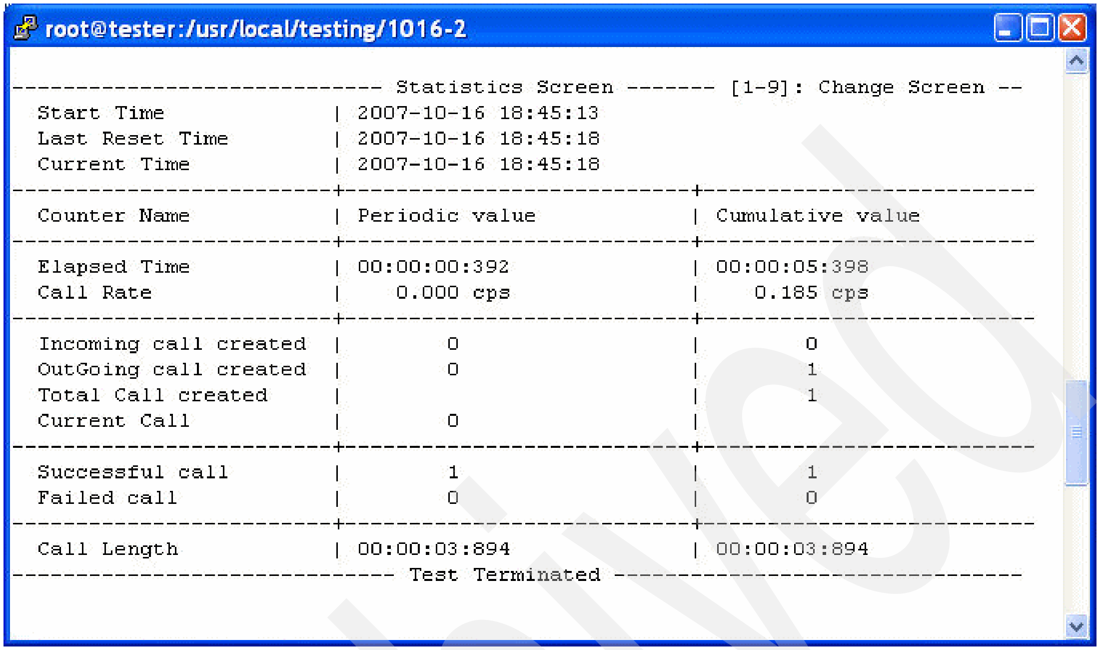
Figure 8-15 on page 302 displays the call flow that SIPp captures.

```
root@tester:/usr/local/testing/1016-2
How many sipp calls per execution [1]:
How many iterations of the test [1]:
Removing old log files... done
Executing
6051
5010
Background mode - PID=[24743]
Resolving remote host 'ims'... Done.
------------------------------ Scenario Screen -------- [1-9]: Change Screen --
  Call-rate(length)     Port   Total-time  Total-calls  Remote-host
   1.0(0 ms)/1.500s    5010       5.39 s            1  9.42.170.211:5060(TCP)

  Call limit reached (-m 1), 0.392 s period  1 ms scheduler resolution
  0 calls (limit 2)                       Peak was 1 calls, after 1 s
  0 Running, 1 Paused, 0 Woken up
  0 out-of-call msg (discarded)
  2 open sockets

                                 Messages  Retrans   Timeout   Unexpected-Msg
   SUBSCRIBE ---------->         1         0
         200 <----------         1         0                   0
     PUBLISH ---------->         1         0
         200 <----------         1         0                   0
     PUBLISH ---------->         1         0
         200 <----------         1         0                   0
     PUBLISH ---------->         1         0
         200 <----------         1         0                   0
     PUBLISH ---------->         1         0
         200 <----------         1         0                   0
     PUBLISH ---------->         1         0
         200 <----------         1         0                   0
   SUBSCRIBE ---------->         1         0
         200 <----------         1         0                   0
     PUBLISH ---------->         1         0
         200 <----------         1         0                   0
     PUBLISH ---------->         1         0
         200 <----------         1         0                   0
     PUBLISH ---------->         1         0
         200 <----------         1         0                   0
     PUBLISH ---------->         1         0
         200 <----------         1         0                   0
     PUBLISH ---------->         1         0
         200 <----------         1         0                   0
------------------------------ Test Terminated --------------------------------
```

Figure 8-15 Call flow captured by SIPp

The detailed trace is enabled to capture the high level of details in the log for verification purposes. Output, as in Figure 8-16 on page 303, can also be correlated with the SIP trace that the Presence and XDMS Servers generate.

Figure 8-16 displays the client test call status.



```
root@tester:/usr/local/testing/1016-2

----------------------------- Statistics Screen ------- [1-9]: Change Screen --
  Start Time             | 2007-10-16 18:45:13
  Last Reset Time        | 2007-10-16 18:45:18
  Current Time           | 2007-10-16 18:45:18
-------------------------+---------------------------+--------------------------
  Counter Name           | Periodic value            | Cumulative value
-------------------------+---------------------------+--------------------------
  Elapsed Time           | 00:00:00:392              | 00:00:05:398
  Call Rate              |    0.000 cps              |    0.185 cps
-------------------------+---------------------------+--------------------------
  Incoming call created  |         0                 |         0
  OutGoing call created  |         0                 |         1
  Total Call created     |                           |         1
  Current Call           |         0                 |
-------------------------+---------------------------+--------------------------
  Successful call        |         1                 |         1
  Failed call            |         0                 |         0
-------------------------+---------------------------+--------------------------
  Call Length            | 00:00:03:894              | 00:00:03:894
------------------------------ Test Terminated --------------------------------
```

Figure 8-16 Client test call status

> **Tip:** You can also use the command line utility ‘tcpdump’ to cross verify the time and packet delivery directions between the servers.

Part 3

The high-availability (HA) scenario

In this part, we describe the enhancements that you make to the base environment to make it highly available. We demonstrate failover processing following the failure of the servers in the environment.

9

The high-availability environment setup and configuration

In this chapter, we provide an overview of the high-availability (HA) scenario environment and the methodologies to create and configure a high-availability environment.

This chapter contains:

9.1 High-availability overview

High availability is the robustness or resiliency of a system. It is the measure of the system to always be able to process transactions or respond to requests in the event of any single point-of-failure.

Vertical scaling (multiple instances of a service running on the same machine) can provide a level of availability; however, the physical machine becomes a single point-of-failure. We revisit vertical scaling in Part 4, "The scalable scenario" on page 383. In this section, we focus on using horizontal scaling to build a high-availability deployment.

9.1.1 Considerations for availability

There are many factors to consider when you plan for high availability. As we see later in this chapter, there are various levels of high availability that you can design and achieve. The deciding factor tends to be the cost to implement and maintain the level of availability that is desired or required by the non-functional requirement for HA. Typically, as the level of availability increases the cost to implement and maintain also increases. While it is desirable to have a service available 99.999% of the time (also known as five-nines), the cost of implementing that level of availability might not be justified. For the purpose of this book, we describe a systematic approach to design an HA solution. However, we do not evaluate the cost that is associated with each decision or the cost of the total solution.

Similar to the base scenario section (Part 2, "The base scenario" on page 57), we focus on achieving a level of high availability for the core components of the Presence/XDMs environment, namely, the SIP/HTTP proxy service, the Presence service, the XDM Aggregation service, the Shared List service, and the Presence Rules service. There are many aspects of availability that must be considered in a production deployment, which includes items, such as redundant networks and network components to provide redundant routing and hosting of services, redundant power sources, and the availability of non-core components such as, the load balancer, directory server, and database servers. You can use the process that we describe in this section to identify these elements and aid you in designing a complete HA environment.

9.1.2 Architectures and topologies for availability

We use the following approach to design HA aspects of our IMS Presence and XDMs infrastructure:

1. Identify actors and external systems.
2. List services that the infrastructure provides.
3. Map services to IT systems that provide the infrastructure services.
4. Identify IT systems that you must make highly available.
5. Map IT system to products.
6. Choose the level of HA that you are going to provide.
7. Choose an HA mechanism for each identified system.
8. Design the Operational Model.
9. Map components of IT systems to deployment units.

We completed the first three steps in earlier sections of this book through the System Context diagram, Architecture Overview diagram, and Operational Model diagrams and discussions in Part 2. By reviewing these, you can identify components that have HA considerations.

As we mentioned earlier, for this book we focus on the availability of the core components of the solution. So, for Step 4, the systems we are concerned with are the Presence system, Shared List system, Presence system, Aggregation Proxy system, and the SIP/HTTP Proxy systems. We assume that all of the other components are already highly available.

We completed step 5 in the process, Map IT system to products, in Table 2-1 on page 54.

9.1.3 Defining the level of availability

The next step in the process is to decide on the level of availability to provide. There are many definitions that we can use to describe the different levels of availability. For the purpose of this book, we use the following definitions:

- Basic system
- Redundant data
- Component fail over
- System fail over
- Disaster recovery

The goal of this book is to create a System fail-over level of availability of the core IMS components that we described earlier. In a production environment, you must completely understand the customer's non-functional requirements in order to choose the appropriate level of availability. In Part 2, “The base scenario” on page 57 and Part 3, “The high-availability (HA) scenario” on page 305, we described a deployment of the base scenario with a basic system level of availability. There exist many single points-of-failures in our current deployment.

At this time, for the purpose of this book, we skip the component fail-over level of availability. As we mentioned earlier, by employing vertical scaling of the core services of our solution we could achieve component fail over because we do this as part of our scaling discussion in Part 4, “The scalable scenario” on page 383 of this book.

System fail over is typically achieved in one of two configurations (or a combination of the 2): Active-Active mutual takeover or Active-Passive takeover. As part of the next step, we choose how to implement HA for each of the core components in our solution.

9.1.4 Choosing the HA mechanism(s)

For the core components in our solution, we implement an Active-Active fail-over scheme. The benefits of the Active-Active fail over are that it provides:

- Better hardware utilization
- Minimized time needed to take over failed services
- Easy scalability of infrastructure
- Minimized need for additional HA software

The major reasons we chose an Active-Active solution is that this capability is built into the WebSphere Application Server Network Deployment product, which the IBM IMS components are built on. Because all of the core components of our solution are built on WebSphere Application Server ND, we only discuss the Active-Active fail-over scheme. For more information, refer to the IBM Redbook: *WebSphere Application Server Network Deployment V6: High Availability Solutions*, SG24-66888. This book provides the details of the HA mechanisms of the WebSphere Application Server ND product that we utilize, which includes information about the use of proxies, load balancers, and replication domains.

9.1.5 Updating the Operational Model

As we discussed and previewed in Chapter 2, “About the scenario” on page 35, the Operational Model defines and documents the distribution of the IMS Presence Server and XDM Server infrastructure components onto physical nodes, as depicted in the topology diagram in Figure 9-1 on page 311. In the base scenario, we deployed a non-HA version of this topology (one of each node type). Now we expand it to include a second node for each core system.

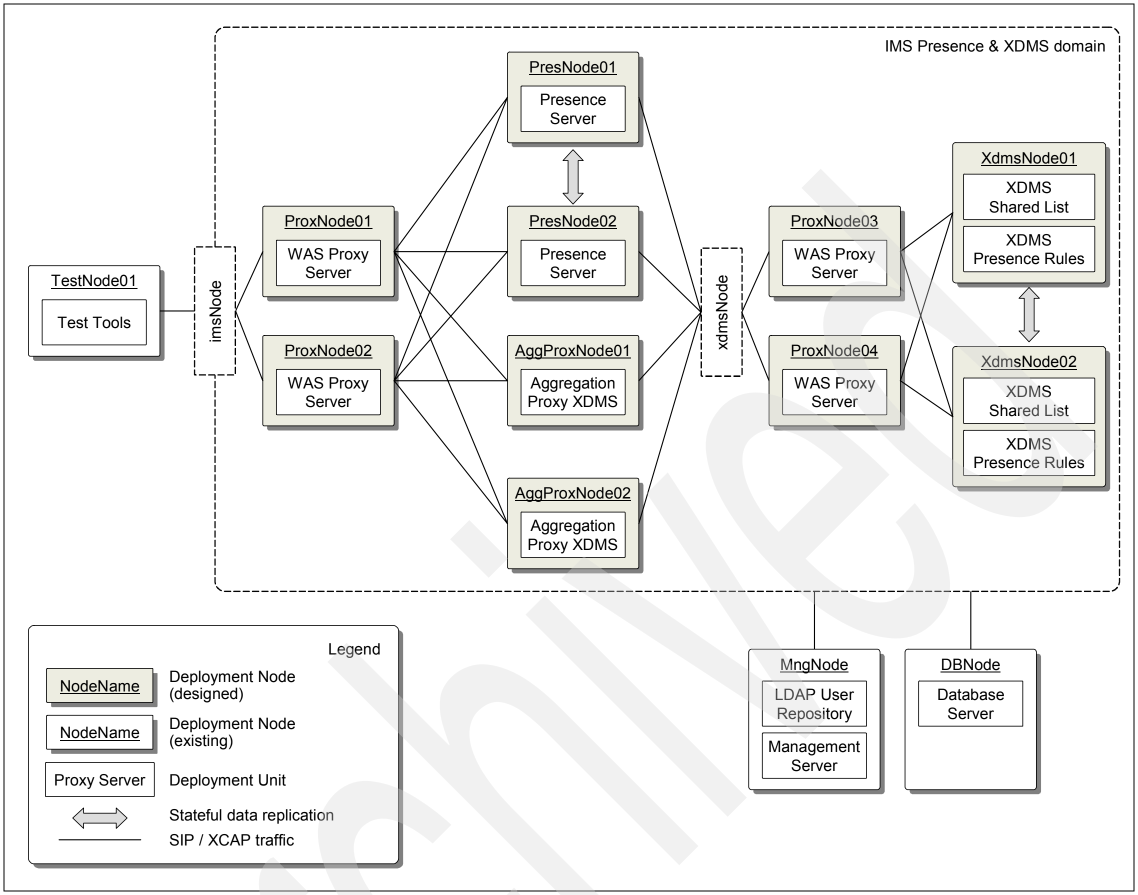


Figure 9-1 Topology

Utilizing the System Topology diagram, we can build Table 9-1, which describes each node and the technology that was used to provide high availability. Non-core components are listed in this table to illustrate what HA technology might be used.

Table 9-1 Node and technology for high availability

Deployment node	Deployment unit	High availability technology
PresNode01 PresNode02	Presence Server	WebSphere Application Server Cluster WebSphere Application Server Proxy Physical Machine Redundancy
XdmsNode01 XdmsNode02	XDMS Shared List Server	WebSphere Application Server Cluster WebSphere Application Server Proxy Physical Machine Redundancy

Deployment node	Deployment unit	High availability technology
XdmsNode01 XdmsNode02	XDMS Presence Rules Server	WebSphere Application Server Cluster WebSphere Application Server Proxy Physical Machine Redundancy
AggProxNode01 AggProxNode02	XDMS Aggregation Proxy	WebSphere Application Server Cluster WebSphere Application Server Proxy Physical Machine Redundancy
ProxNode01 ProxNode02 ProxNode03 ProxNode04	WebSphere Application Server Proxy Server	Physical machine redundancy behind Load Balancer that employs IP spraying. Proxy is working as a stateless component, and session replication data is not needed.
imsNode xdmsNode	Load Balancer	Typically implemented as Active-Standby using IP takeover software upon fault detection.
MngNode	LDAP User Repository	Typically implemented as Active-Active using repository synchronization capabilities of the Directory infrastructure or other synchronization software.

Building from Table 9-1 on page 311, we can build the deployment units and components' mapping table (Table 9-2), which we use in the deployment model. In Table 9-2, we map all of the components that make up the core deployment units of our solution.

Table 9-2 Deployment units and components mapping table

Deployment unit	Components
Presence Server	Presence Server EAR DHAIMSConnectorTAI WebSphere Application Server ND - managed profile
XDM Shared List Server	IBMSharedListXdms Server EAR DHAIMSConnectorTAI AggProxyTAI WebSphere Application Server ND - managed profile
XDM Presence Rules Server	IBMPresenceRulesXdms Server EAR DHAIMSConnectorTAI AggProxyTAI WebSphere Application Server ND - managed profile

Deployment unit	Components
XDM Aggregation Proxy	IBMXdmsAggregatinoProxy EAR DHAIMSConnectorTAI AggProxyTAI WebSphere Application Server ND - managed profile
WAS SIP/HTTP Proxy	WebSphere Application Server ND - Proxy Server WebSphere Application Server ND - managed profile

9.1.6 Finalizing the deployment model

Given the HA topology diagram and the Deployment Unit/Components matrix in Table 9-2 on page 312, we can build a deployment model for our solution. We represent the deployment model in three diagrams for readability:

- Presence Server Domain
- XDM Server Domain
- Aggregation Proxy Domain

To simplify the diagrams, we do not show all communication channels (for example, the Deployment Manager's communication to the node agents). For these details, you can review the information in the Redbooks publication, *WebSphere Application Server Network Deployment V6: High Availability Solutions*, SG24-6688.

Figure 9-2 on page 314 illustrates the Presence Server Domain.

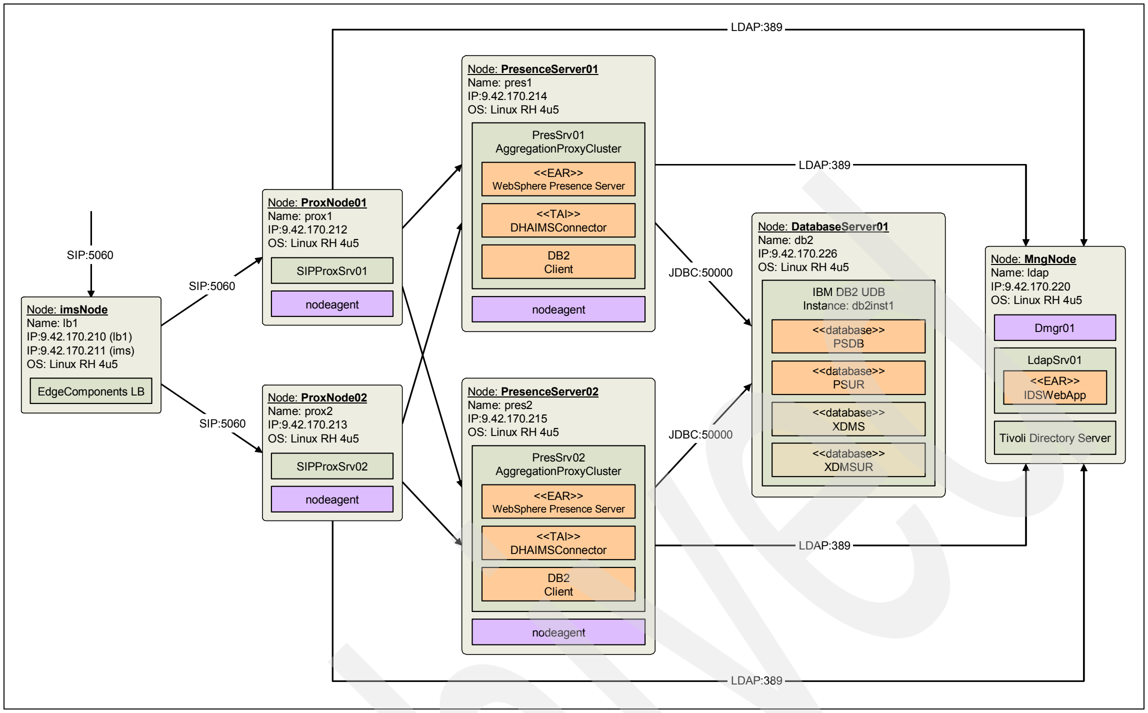


Figure 9-2 Presence Server Deployment diagram

Figure 9-3 on page 315 shows the components that make up the XDM Server Domain.

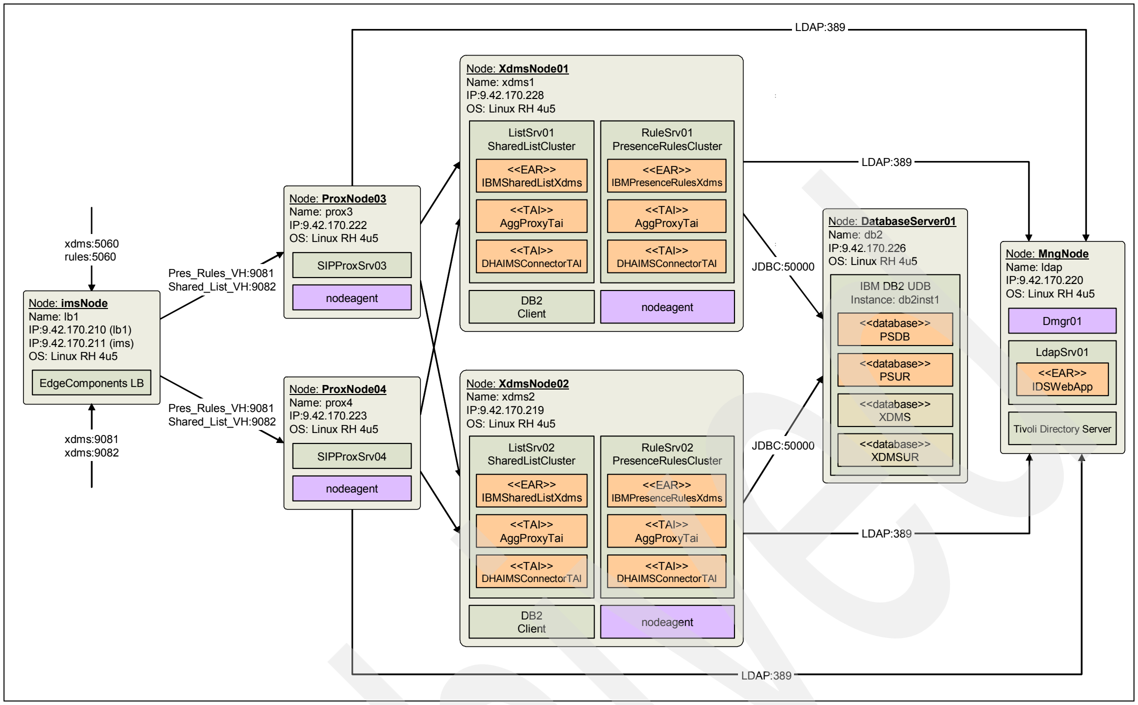


Figure 9-3 XDMS Server Deployment diagram

Figure 9-4 on page 316 shows the Aggregation Proxy domain.

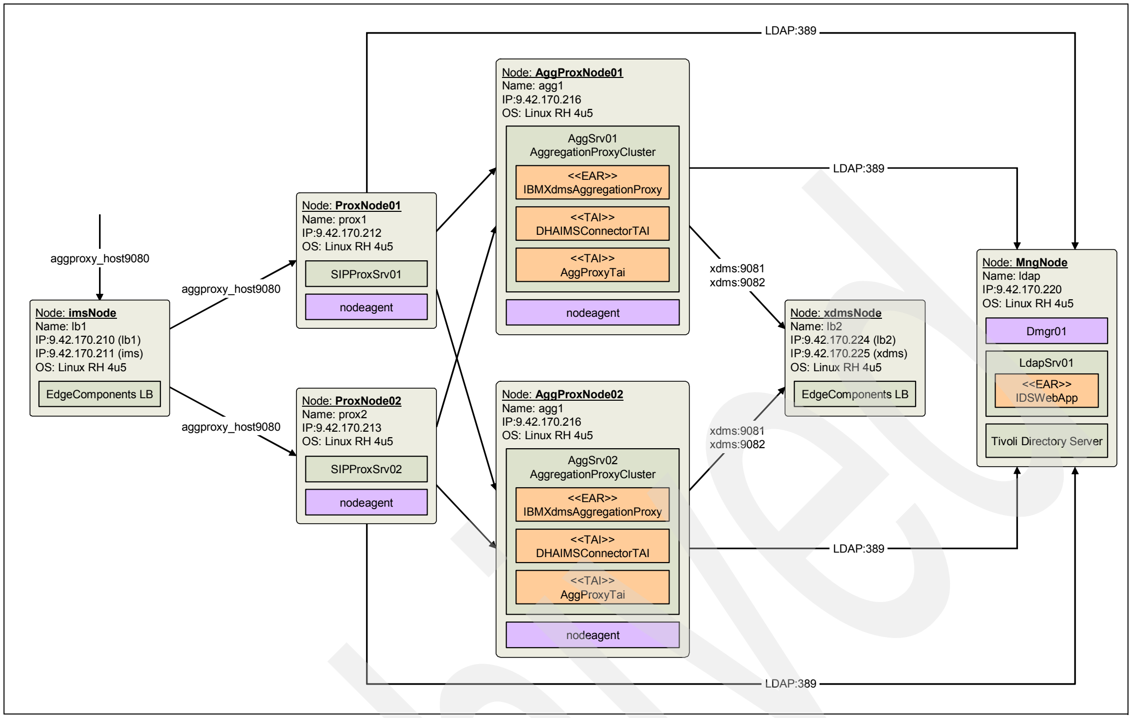


Figure 9-4 Aggregation Proxy Deployment diagram

In the next sections, we set up and validate our high-availability environment.

9.2 Setting up the high-availability environment

A prerequisite to continue is to configure the high-availability (HA) environment for the base scenario environment.

Table 9-3 shows the set of steps for enhancing the base scenario environment to make it highly available.

Table 9-3 Check list of setting up HA scenario environment

Step	Activity	Remarks
1.0	Environment preparation	--
1.1	Install OSes	Linux RHEL 4u5
1.2	Configure network	DNS

Step	Activity	Remarks
1.3	Install WebSphere Application Server Application Servers	Without profiles
1.4	Install DHAIMSConnectorTAI module	--
1.5	Install AggProxyTAI module	--
2.0	WebSphere Application Server Infrastructure	--
2.1	Create WebSphere Application Server profiles	PresNode02
2.2	Add members to the Presence Cluster	PresSrv02
2.3	Create WebSphere Application Server profile for SIP proxy	ProxNode02
2.4	Create Proxy Servers	ProxSrv02
2.5	Associate proxies to Presence Cluster	--

The WebSphere Presence Server and XDMS Servers are enterprise applications that run on the WebSphere Application Server; therefore, they take advantage of the WebSphere Application Server Network Deployment distributed architecture. The HA set up and configuration require that you create WebSphere Application Server profiles and servers and connect them to the existing base environment infrastructure. The rest of the tasks for setting up the HA environment are performed automatically in WebSphere Application Server. These remaining tasks are: distribution of JDBC resources, distribution of JMS resources, and distribution of the Presence Server and XDMS applications.

9.2.1 Preparing the environment

You must install the binaries of the Trust Association Interceptors (TAI) on all of the new servers as the basis for preparing the environment. This process is the same as the process for the base environment installation, which are:

1. Unpack the IMSConnectorInstallPackage_6.2.tar to a temporary directory.
2. Distribute the DHAIMSConnectorTAI.jar file to the <was_root>/lib/ext location on all of the servers that you added to the infrastructure, as shown in Example 9-1 on page 318:
   - PresNode02 (pres2.itso.ral.ibm.com)

– XdmsNode02(xdms2.itso.ral.ibm.com)
– AggProxNode02 (agg2.itso.ral.ibm.com)

Example 9-1 Copy the IMSConnector TAI library

```
[root@pres1 ext]# pwd
/opt/ibm/WebSphere/AppServer/lib/ext
[root@pres1 ext]# cp /mnt/IMS_62/DHAIMSConnectorTAI.jar ./
```

9.2.2 WebSphere Presence Server

In this section, we provide installation instructions for making the new servers part of the Presence Server infrastructure. The three major tasks are:

1. Configure the WebSphere Application Server.
2. Configure the WebSphere Proxy Server.
3. Distribute the WebSphere Presence Server binary.

WebSphere Application Server configuration

To configure the WebSphere Application Server for the new servers in the Presence Server Cluster, you must:

1. Create WebSphere Application Server profiles.
2. Add WebSphere Application Server to the Presence Cluster.
3. Add the new message engine to the Service Integration Bus (SIB).

Creating WebSphere Application Server profiles

You must create and federate WebSphere Application Server profiles to imsCell to use the newly created server environment.

> **Note:** To create and federate WebSphere Application Server nodes use the standard script from standard WebSphere Application Server distribution like: mangeprofiles.sh, addNode.sh.
>
> After you create node it is good practice to synchronize this node using syncNode.sh command.

To create WebSphere Application Server profiles:

1. Login to the machine PresNode02, and run the command in Example 9-2.

Example 9-2 Creating a WebSphere Application Server profile for PresNode02

```
/opt/IBM/WebSphere/AppServer/bin/manageprofiles.sh -create -profileName
PresNode02 -profilePath
/opt/IBM/WebSphere/AppServer/profiles/PresNode02 -templatePath
```

```
/opt/IBM/WebSphere/AppServer/profileTemplates/managed -nodeName
PresNode02 -hostName pres2.itso.ral.ibm.com
```

2. Federate the created node to the imsCell using the command in Example 9-3.

Example 9-3 Federate PresNode02 to the imsCell

```
/opt/IBM/WebSphere/AppServer/profiles/PresNode02/bin/addNode.sh
ldap.itso.ral.ibm.com -username wasadmin -password <password>
-profileName
PresNode02
```

3. Synchronize the created profiles using the **syncNode** command, as shown in Example 9-4.

Example 9-4 Synchronize PresNode02 with the imsCell

```
/opt/IBM/WebSphere/AppServer/profiles/PresNode02/bin/syncNode.sh
ldap.itso.ral.ibm.com -username wasadmin -password <password>
```

Adding the WebSphere Application Server to the Presence Cluster

When the profiles are ready, you can create a new Application Server as a member of *PresenceCluster* using the WebSphere Integrated Console:

1. Login to the Integrated Solutions Console as a user with an administrator role, for example `wasadmin`.
2. Click **Servers** → **Clusters**.
3. From the list of clusters, shown in Figure 9-5 on page 320, click **PresenceCluster**.

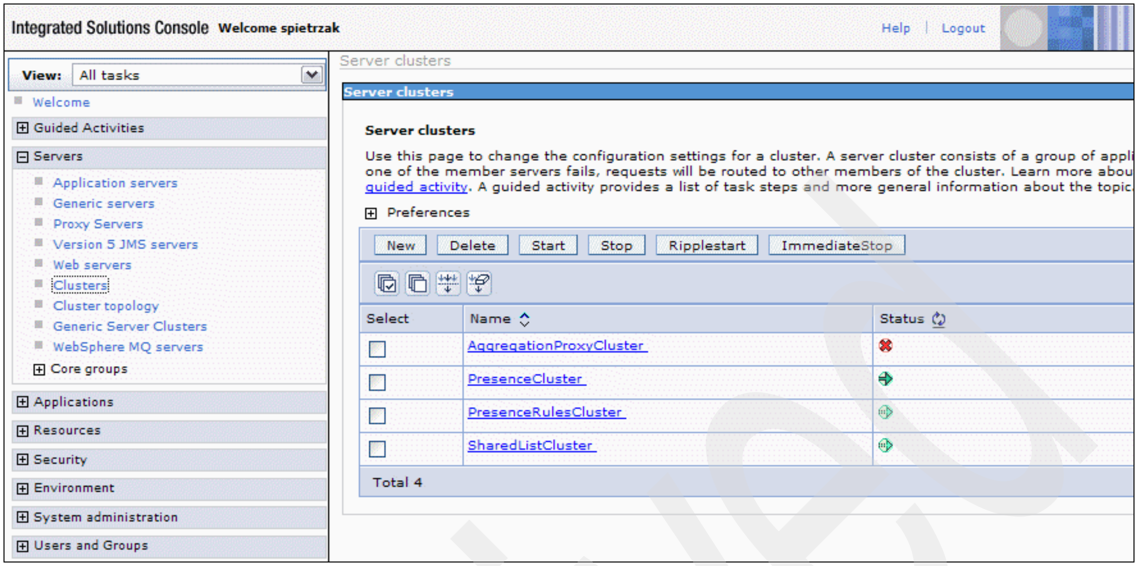


Figure 9-5 Presence Cluster - list

4. Under Additional Properties, click **Cluster members**, as shown in Figure 9-6.

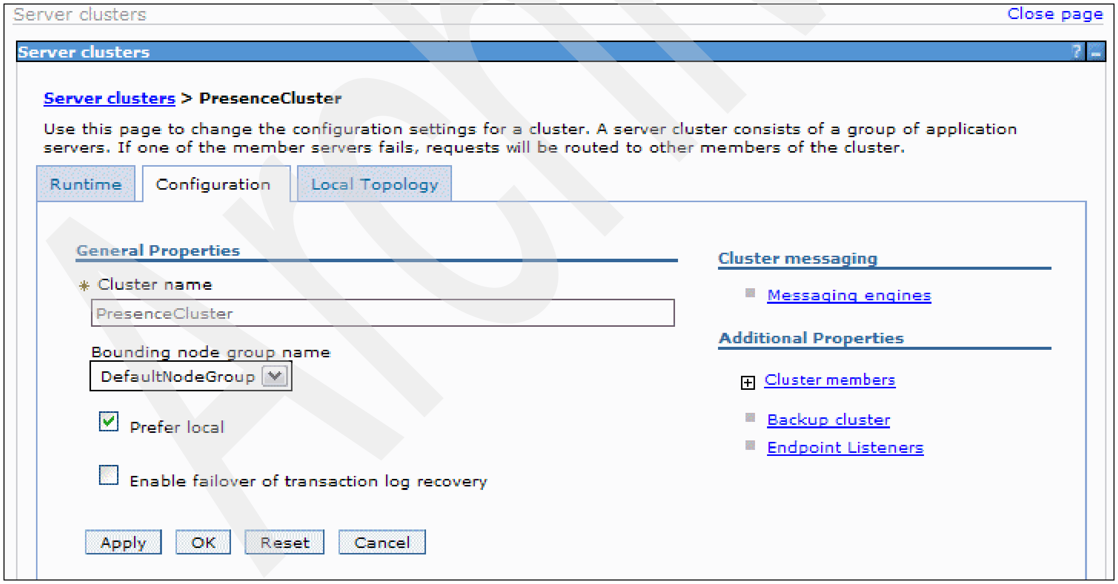


Figure 9-6 Presence Cluster - details

5. Click **New**.

6. On the new cluster member page (Figure 9-7):
   a. In the Member name field, type `PresSrv02`.
   b. From the Select node pull-down menu, select **PresNode02 (ND 6.1.0.11)**.

> **Note:** The weight option has the same value of 2 for both servers. It means that the workload will be evenly spread for both cluster members.

7. Click **Add Member**, and then click **Next**, as shown in Example 9-6 on page 328.

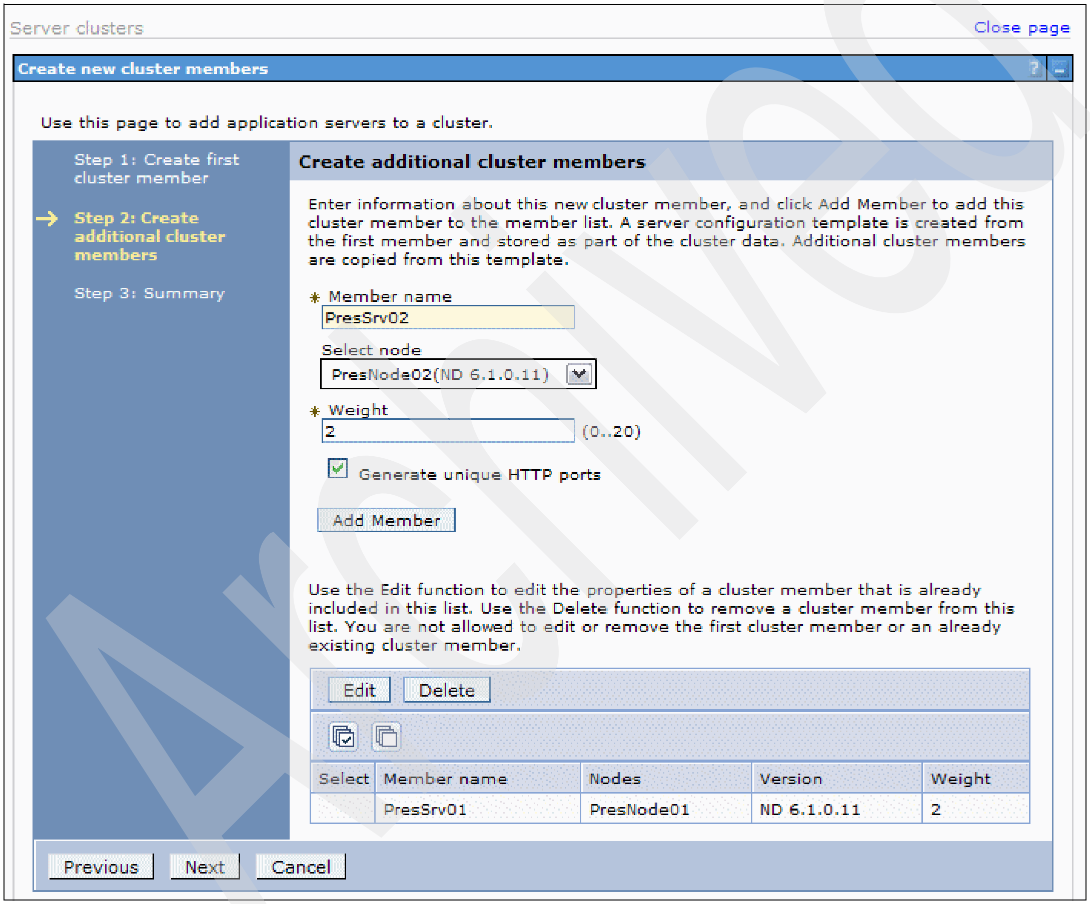


Figure 9-7 Adding PresSrv02 server to PresenceCluster

8. On the Summary page, click **Finish.**

9. From the Integrated Solutions Console, shown in Figure 9-8 on page 322, click **Servers** → **Application Server** to see the created cluster's members.

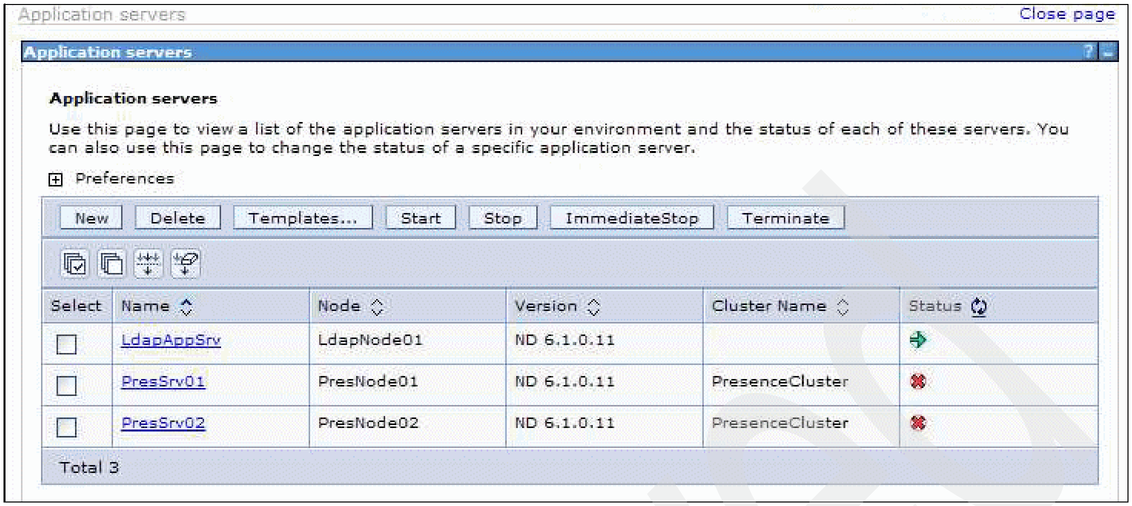


Figure 9-8 PresenceCluster's members

10. From the Integrated Solutions Console, click **Servers** → **Clusters**, and select **PresenceCluster** from the list, as shown in Figure 9-9.
11. Click **Start** to start the Cluster.

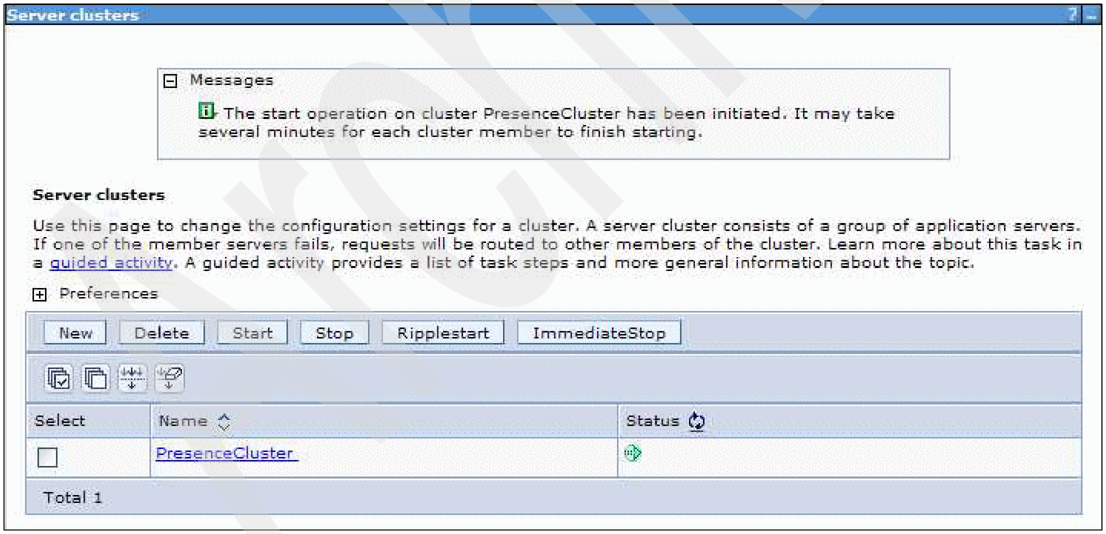


Figure 9-9 Starting PresenceCluster

12. To verify the correct configuration, inspect the following log files for any errors, for both Application Servers:

   – PresSrv01
   – PresSrv02

   Figure 9-10 shows an example of a log file for PresSrv02.

```
[root@pres2 PresSrv02]# pwd
/opt/ibm/WebSphere/AppServer/profiles/PresNode02/logs/PresSrv02
[root@pres2 PresSrv02]# ls -la
total 64
drwxr-xr-x  2 root root  4096 Sep 21 14:48 .
drwxr-xr-x  5 root root  4096 Sep 21 14:59 ..
-rw-r--r--  1 root root     0 Sep 21 14:48 native_stderr.log
-rw-r--r--  1 root root     0 Sep 21 14:48 native_stdout.log
-rw-r--r--  1 root root     5 Sep 21 14:49 PresSrv02.pid
-rw-r--r--  1 root root     0 Sep 21 14:48 SystemErr.log
-rw-r--r--  1 root root 23889 Sep 21 14:49 SystemOut.log
```

Figure 9-10 Starting the PresenceCluster logs list

13. Additionally, verify that the SystemOut.log does not contain any application errors.

14. Ensure that SystemOut.log finished with the following sentence:

```
Server PresSrv01 open for e-business.
```

Adding a new message engine to Service Integration Bus

After you create the new server, you must add it as a messaging engine to the Service Integration Bus that is dedicated for presence, the PresenceBUS. In the given configuration of environments, the messaging engines are used at the server level instead of clusters. Each time the new server is added to the Presence Server Cluster, you must add the server as messaging engine.

To add a new message engine to Service Integration Bus:

1. From the Integration Solutions Console, select **Service Integration** → **Buses**, as shown in Figure 9-10.

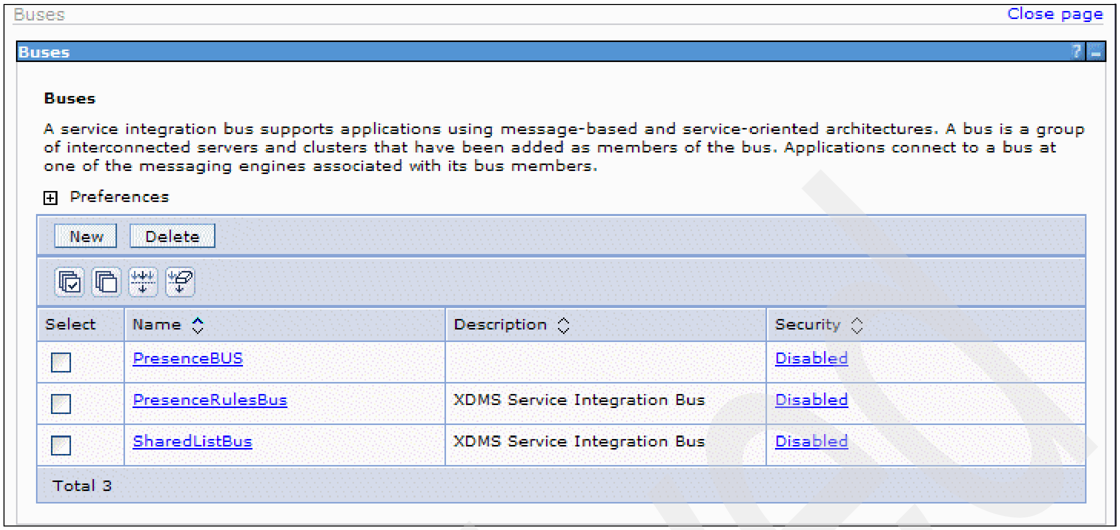


Figure 9-11 SIB - buses

2. Click **PresenceBUS**, as shown in Figure 9-11.
3. Under Topology, click **Bus Members**, as shown in Figure 9-12 on page 325.

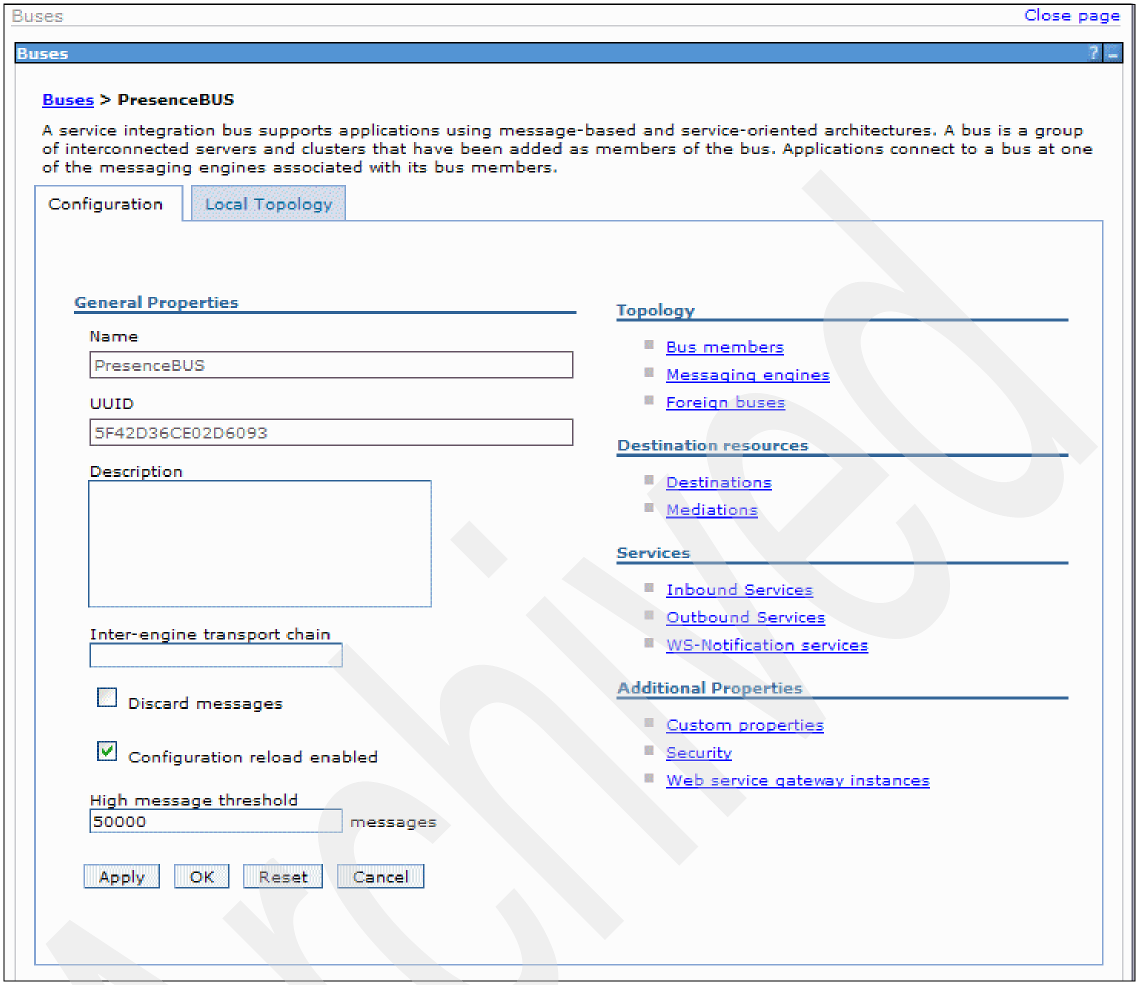


Figure 9-12 Presence Bus - details

4. Click **Add**, and on the Add a new bus member page (Figure 9-13 on page 326), from the Server pull-down menu, select **PresNode02:PresSrv02**, and click **Next**.

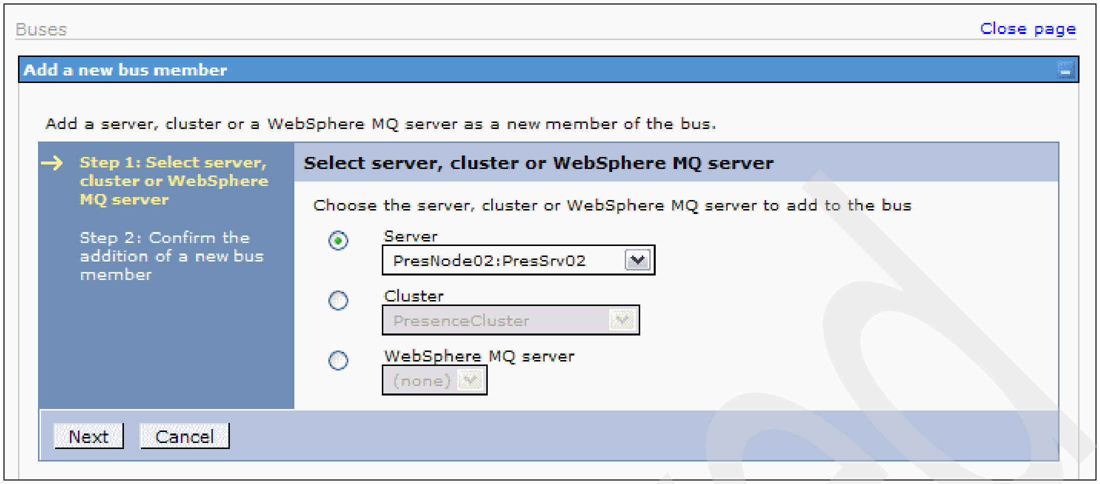


Figure 9-13 BUS engines - server PresSrv02

5. Select **File store**, and click **Next**, as shown in Figure 9-14.

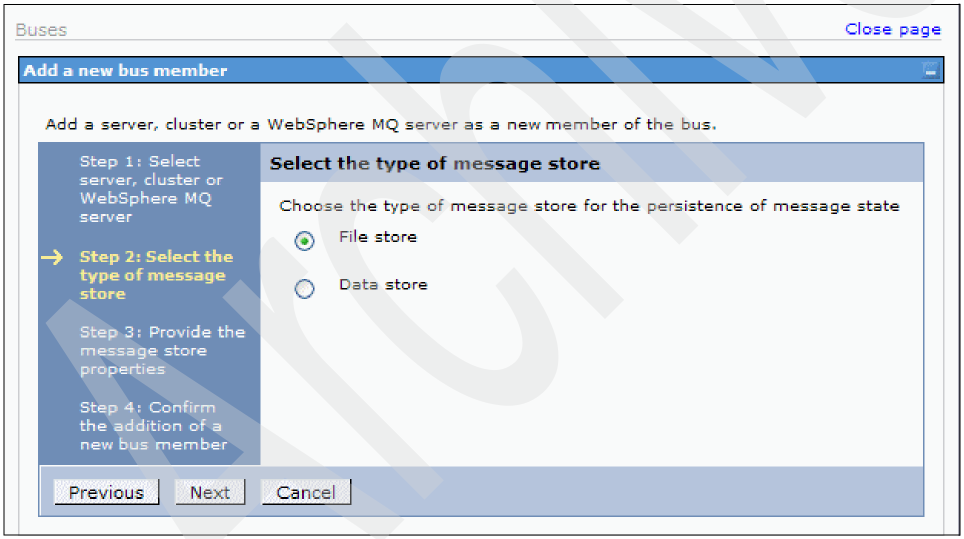


Figure 9-14 BUS engines - store type

6. Accept the default parameters on the page, and click **Next**.
7. Click **Finish**. A window similar to Figure 9-15 on page 327 is displayed.

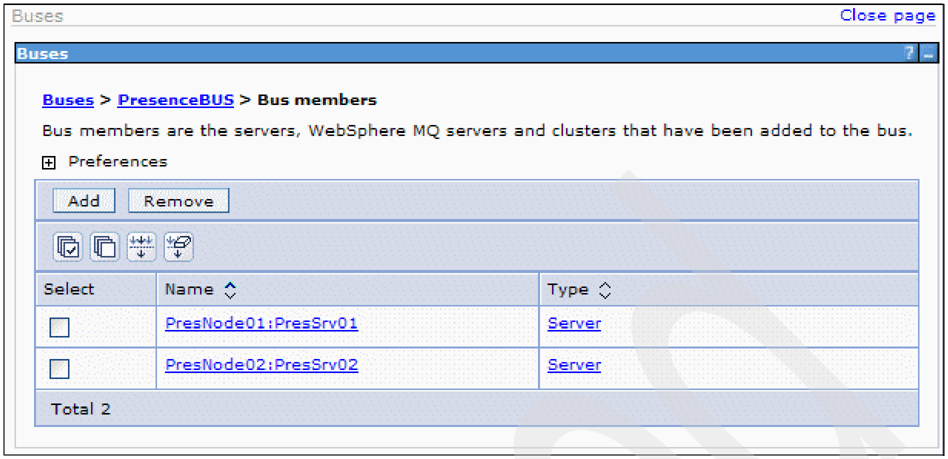


Figure 9-15 SIB PresenceBUS - Bus members

WebSphere Proxy Server configuration

HA is provided based on the redundancy of resources. In support of such a scenario, a second Proxy Server is added in front of the Presence Servers. Both ProxNode01, which is created as part of the base scenario environment, and ProxNode02 are ready to balance the workload between the Presence Servers. In the next section, we provide the steps for extending the environment with ProxNode02. The configured environment can help distribute the traffic to the available Presence Servers in case one of the Proxy Servers goes down.

Creating WebSphere Application Server profiles for SIP Proxy Server 2

1. Login to the ProxNode02 machine, and run the command in Example 9-5.

Example 9-5 Create WebSphere Application Server profile for ProxNode02

```
/opt/IBM/WebSphere/AppServer/bin/manageprofiles.sh -create -profileName
ProxNode02 -profilePath
/opt/IBM/WebSphere/AppServer/profiles/ProxNode02 -templatePath
/opt/IBM/WebSphere/AppServer/profileTemplates/managed -nodeName
ProxNode02 -hostName prox2.itso.ral.ibm.com
```

2. Federate the created node to the imsCell using the command in Example 9-6 on page 328.

Example 9-6 Federate ProxNode02 to the imsCell

```
/opt/IBM/WebSphere/AppServer/profiles/ProxNode02/bin/addNode.sh
ldap.itso.ral.ibm.com -username wasadmin -password <password>
-profileName ProxNode02
```

3. Synchronize the created profiles using the syncNode command, as shown in Example 9-7.

Example 9-7 Synchronize ProxNode02 with the imsCell

```
/opt/IBM/WebSphere/AppServer/profiles/ProxNode01/bin/syncNode.sh
ldap.itso.ral.ibm.com -username wasadmin -password <password>
```

Creating SIP Proxy Servers

1. Login to WebSphere Application Server Console as a user with administration privileges, for example `wsadmin`.
2. From the Integrated Solutions Console, click **Server** → **Proxy Servers**, and click **New**.
3. On the Create a new Proxy Server page (Figure 9-16), complete the following tasks, and click **Next**:
   a. Select **ProxNode02** from the Select node pull-down menu.
   b. Enter **SIPProxSrv02** for Server Name.

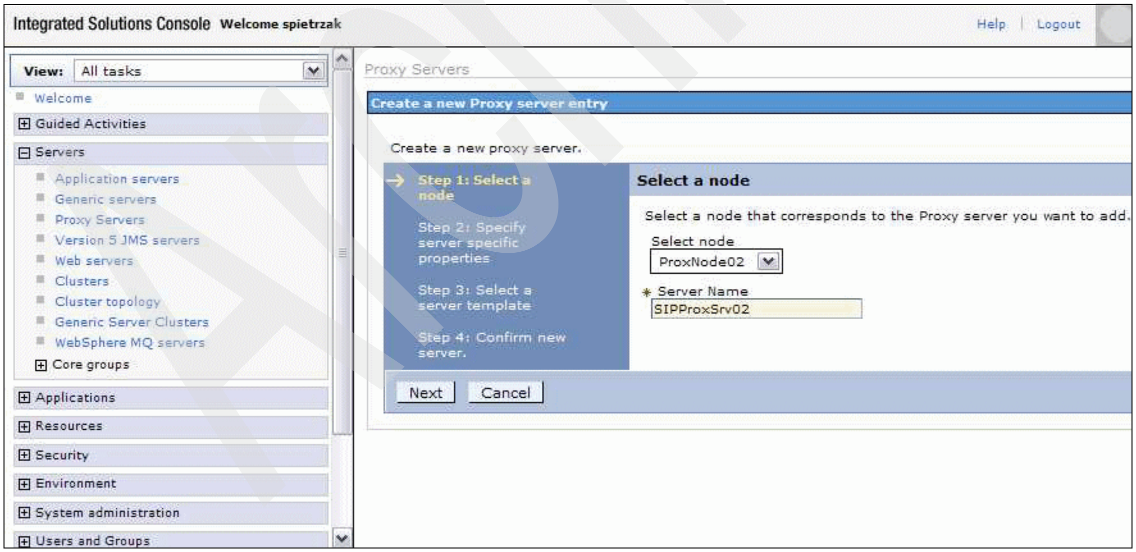


Figure 9-16 Creating a new Proxy Server - SIPProxSrv02

4. On the Specify server specific properties page (Figure 9-17), perform the following tasks, and then click **Next**:
   a. Verify that both HTTP and SIP protocols are selected.
   b. Select **Generate unique ports**.

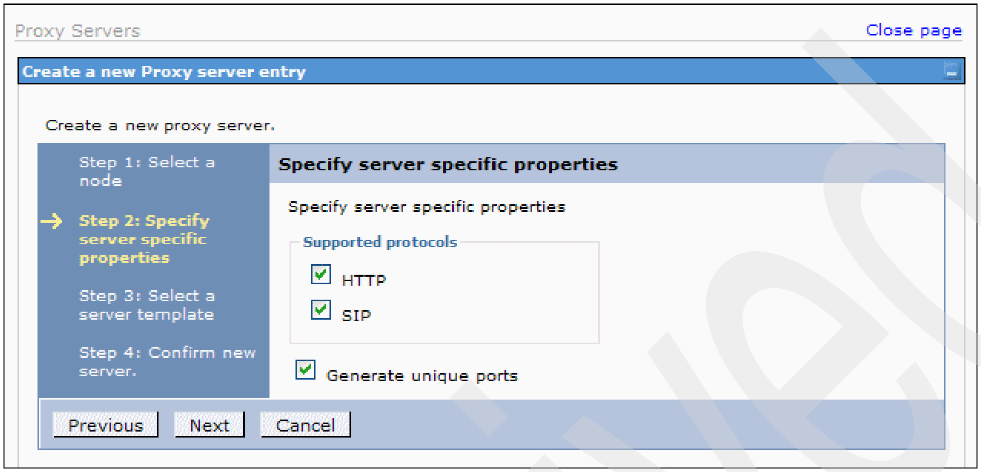


Figure 9-17 Creating a new Proxy Server - SIPProxSrv02 - Step 2

5. On the next page, click **Next**.
6. On the summary page, click **Finish**.
7. Save changes to master configuration.

Associating SIP Proxy with Application Servers

We configured ProxNode01 and ProxNode02 to include the load balancer. Although it is not necessary to add a load balancer to the base scenario with only ProxNode01 present, the availability of two servers makes it important to configure load balancing requirements to set the load balancer in front of the Proxy Servers.

To associate SIP Proxy with Application Servers:

1. From the Integrated Solutions Console, click **Servers** → **Proxy Servers**.
2. Click the recently created Proxy Server: **SIPProxSrv02**.
3. Under Proxy Settings, click **SIP Proxy Server Settings** → **SIP Proxy Settings**, as shown in Figure 9-18 on page 330.

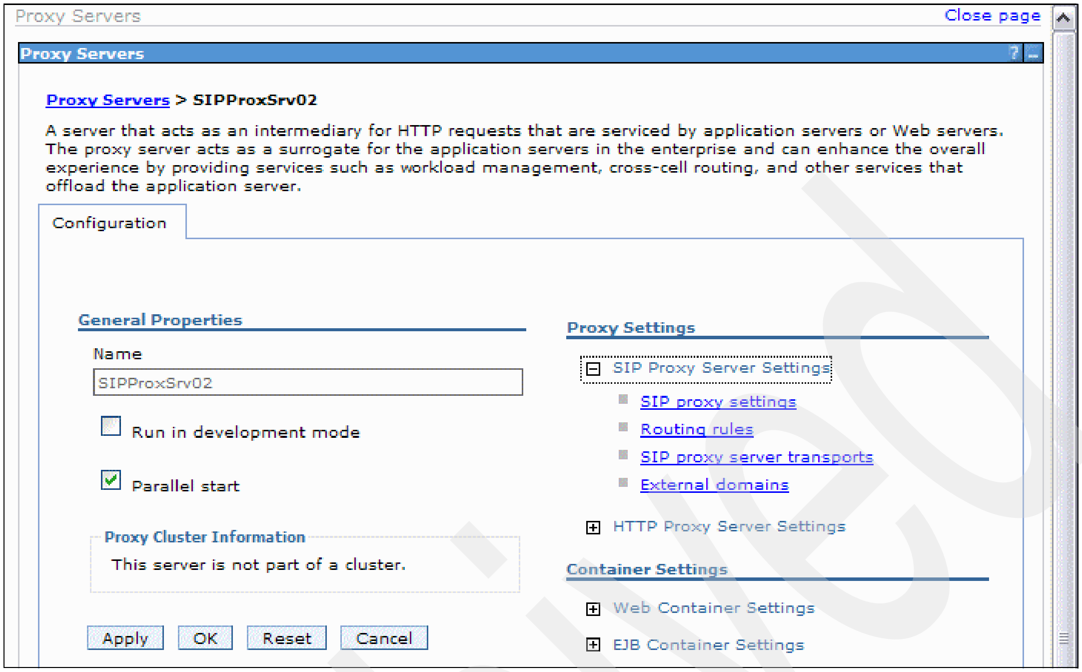


Figure 9-18 Proxy configuration - Step 1

4. On the proxy configuration page (Figure 9-19 on page 331) perform the following:

   a. From the Default cluster pull-down menu, select **PresenceCluster**.
   b. Select the **Enable TCP spayer** option.
   c. In the TCP host (IP Address of Load Balancer) field, enter `9.42.170.211`.
   d. In the TCP Port field, enter `5060`.
   e. Select the **Enable SSL spayer** option.
   f. In the SSLhost field, enter `9.42.170.211` (IP Address of Load Balancer).
   g. In the SSL Port field, enter `5061`.
   h. Select the **Enable UDP spayer** option.
   i. In the UDPhost field, enter `9.42.170.211` (IP Address of Load Balancer).
   j. In the UDPPort field, enter 5060.
   k. To enable logging for SIP traffic, select **Enable access logging**.
   l. Click **OK**.

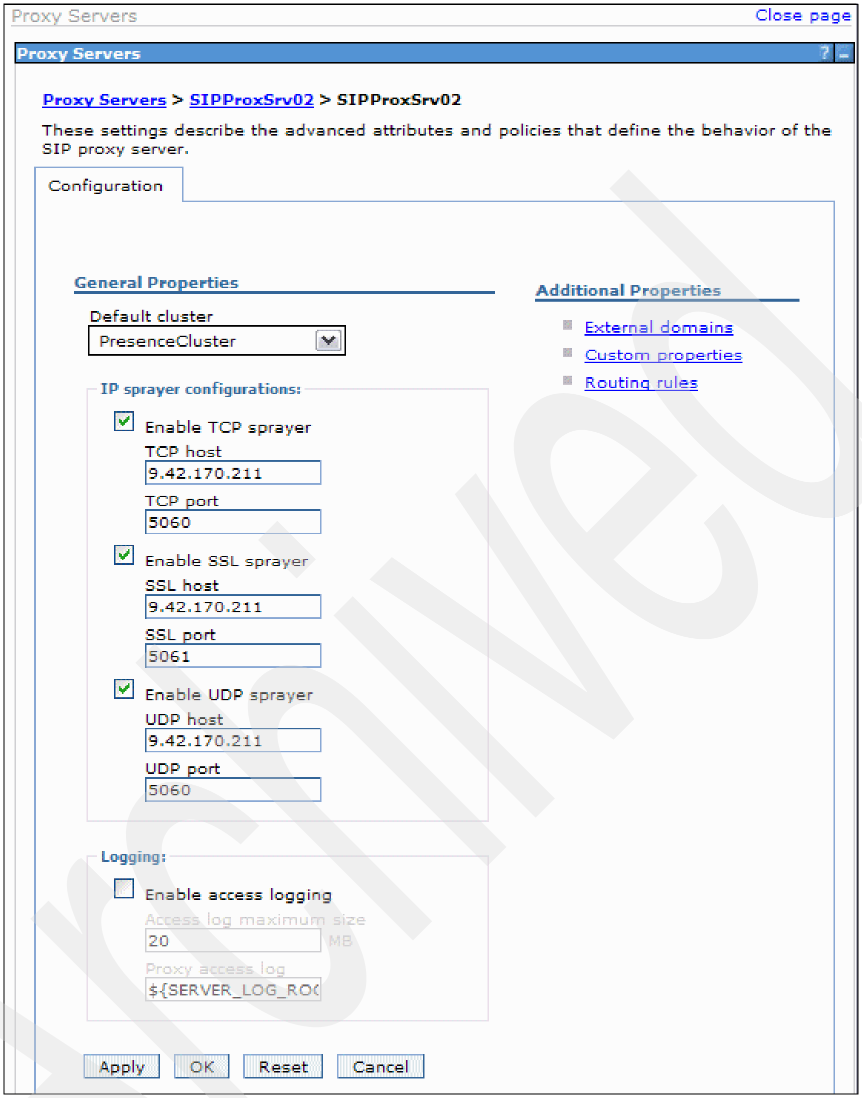


Figure 9-19 Configure proxy 2 to work with load balancer

5. From the navigation pane, click **Servers** → **Proxy Servers**, and select **SIPProxSrv02**.
6. Click **Start**.

9.2.3 Installing WebSphere XML Document Management Servers (XDMS)

In this section, we provide installation steps to expand the XML Document Management Servers (XDMS) installation in the base environment (explained in Chapter 3, “Overview of the base environment” on page 59) to support a high-available environment and cater for fail over and replication. Making the XDMS a highly available environment involves the following process:

1. Configuring WebSphere Application Server.
2. Configuring WebSphere Proxy Server.

Configuring WebSphere Application Server

Configuring WebSphere Application Server involves adding an additional XDM server for the Shared List and Presence Rules enterprise applications to handle failover of the single XDMS as depicted in the base environment. The additional server is added in a horizontal clustered configuration with one additional node and one server in the node. XdmsNode02, as depicted in Figure 9-3 on page 315, is the new node that we added to enable the high-availability support for XDMS. The WebSphere Application Server configuration involves three steps:

1. Creating and federating the HA node for XDMS.
2. Creating an Application Server for the Shared List Cluster.
3. Creating an Application Server for the Presence Rules Cluster.

Creating and federating the HA node for XDMS

Creating a managed node is a two step process. The first step is to install the WebSphere Application Server image on the commissioned machine that is named *xdms2.itso.ral.ibm.com*. To create the node involves creating a profile. The newly created profile is then federated to the *Cell* profile named imsCell for our environment.

To create and federate the HA node for XDMS:

1. Login to the machine *xdms2.itso.ral.ibm.com*, and run the command in Example 9-8.

Example 9-8 Create WebSphere Application Server profile for XdmsNode02

```
/opt/IBM/WebSphere/AppServer/bin/manageprofiles.sh -create -profileName
XdmsNode02 -profilePath
/opt/IBM/WebSphere/AppServer/profiles/XdmsNode02 -templatePath
/opt/IBM/WebSphere/AppServer/profileTemplates/managed -nodeName
XdmsNode02 -hostName xdms2.itso.ral.ibm.com
```

2. Federate the created node to the imsCell using the command in Example 9-9 on page 333.

Example 9-9 Federate XdmsNode02 to the imsCell

```
/opt/IBM/WebSphere/AppServer/profiles/XdmsNode02/bin/addNode.sh
ldap.itso.ral.ibm.com -username wasadmin -password <password>
-profileName XdmsNode03
```

3. Synchronize the created profile using the **syncNode** command, as shown in Example 9-10.

Example 9-10 Synchronize XdmsNode02

```
/opt/IBM/WebSphere/AppServer/profiles/XdmsNode02/bin/syncNode.sh
ldap.itso.ral.ibm.com -username wasadmin -password <password>
```

4. Start the recently created node XdmsNode02, as shown in Example 9-11.

Example 9-11 Start XDMS Node XdmsNode02

```
/opt/IBM/WebSphere/AppServer/profiles/XdmsNode02/bin/startNode.sh
```

For details about installation and profile management, refer to:

http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.wsfep.multiplatform.doc/info/ae/ae/tpro_instancesmngdnode.html

> **Note:** In the node *xdms2.itso.ral.ibm.com* we create two Application Servers, one for the Shared List application and the other for the Presence Rules application.

Creating an Application Server and adding it to the Shared List Cluster

After you create and federate a new node on the xdms2.itso.ral.ibm.com machine, log into the Application Server's Integration Solution Console.

To create an Application Server and add it to the Shared List Cluster:

1. On the left panel, click the **Servers** → **Application Servers** hyperlinks.

   In the right panel, the list of Application Servers is displayed along with their respective status.

2. Locate and click the **New** button to create a new Application Server.

3. In the **Select a node** panel, from the list of available nodes, select the newly commissioned node. In the Server name field, specify a meaningful name. In the current scenario field, enter the name `ListSrv02`. Click **Next**.
4. In the Select server template panel, select **default template**. Click **Next**.
5. In the Specify server specific properties panel, select the **Generate Unique Ports** option. Click **Next**.
6. Click **Finish** to create the new Application Server in the new node.

Creating an Application Server and adding it to the Presence Rules Cluster

After you create and federate a new node on the xdms2.itso.ral.ibm.com machine, log into the Integration Solutions Console of the Deployment Manager.

To create an Application Server and add it to the Presence Rules Cluster:

1. On the left panel, click the **Servers** → **Application Servers** hyperlinks.

   In the right panel, the list of Application Servers is displayed along with their respective status.
2. Locate and click the **New** button to create a new Application Server.
3. In the **Select a node** panel, from the list of available nodes, select the newly commissioned node. In the Server name field, specify a meaningful name. In the current scenario field, enter the name `RuleSrv02`. Click **Next**.
4. In the Select server template panel, select **default template**. Click **Next**.
5. In the Specify server specific properties panel, select the **Generate Unique Ports** option. Click **Next**.
6. Click **Finish** to create the new Application Server in the new node.

Configuring the WebSphere Proxy

Configuring the WebSphere Proxy involves adding an additional Converged (HTTP and SIP) Proxy Server to handle fail over of the Proxy Server that fronts the XDMS Servers. You add the Proxy Server on a new WebSphere Application Server node. ProxNode04, as shown in Figure 9-6 on page 320, is the new Proxy Server that is added to enable the high availability support for the Proxy Server that is fronting XDMS.

Creating an HA Proxy Server for the XDMS Cluster

After you create and federate a new node on the Prox4.itso.ral.ibm.com machine for the high available Proxy Server for the XDMS Server follow these steps.

To create an HA Proxy Server for the XDMS Cluster:

1. Login into the Integration Solutions Console of the Deployment Manager.
2. Click **Proxy Servers**, and click **New** to create a new Proxy Server.
3. In the Select a node panel, select the newly commissioned node from the list of available nodes for the new Proxy Server. In the Server name text box, specify a meaningful name. In the current scenario, the name is *ProxNode04*. Click **Next**.
4. In the Server Specific Properties panel, accept the defaults. The HTTP and SIP supported protocols are selected, which means a converged proxy is being created.
5. Click **Next** to accept the defaults for the Server Template, and click **Next**.
6. On the Confirm new server panel, click **Finish** to complete the HA Proxy Server.

Configuring the Proxy Server for Presence Rules

Perform the following steps for the Proxy Server ProxNode04. The goal of the configuration is to enable routing rules for SIP traffic and split it for two clusters: Presence Rules and Shared List. Proxy Server gets one cluster as default; however, in our case we set up Shared List Cluster as the default. The traffic to be forwarded to the Presence Rules Cluster is coming from rules.itso.ral.ibm.com, so based on this name we can create routing rules and forward these request to cluster different than the default.

Configuring a default cluster

To configure a default cluster:

1. From the Integrated Solutions Console, click **Servers** → **Proxy Servers.**
2. Click **SIPProxSrv03** to select it for configuration.
3. Under Proxy Settings (Figure 9-20 on page 336), click **SIP Proxy Server Settings** → **Sip Proxy Settings**.
4. From the Default Cluster pull-down menu, select **SharedListCluster**.
5. Click **Apply**.

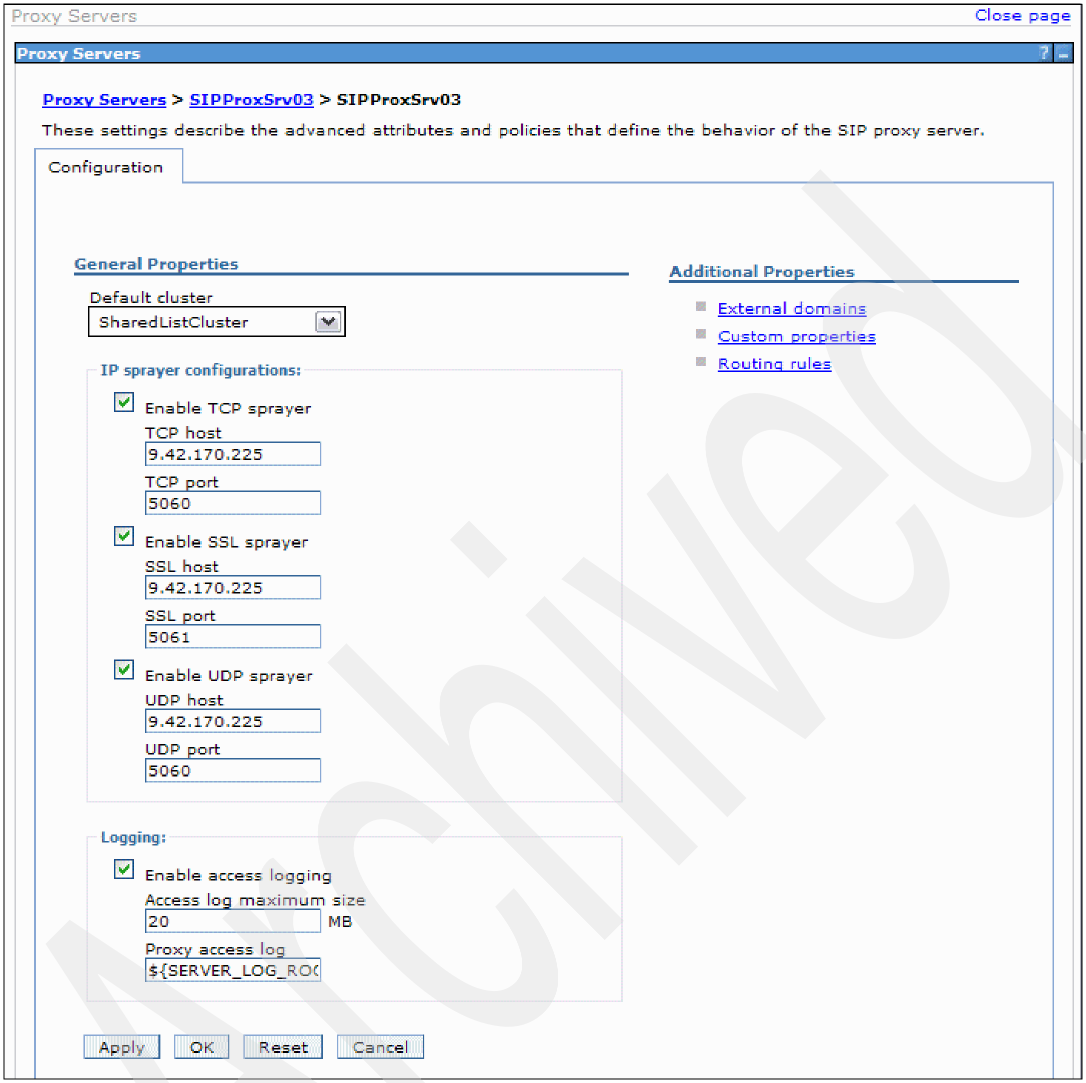


Figure 9-20 XDMS SIP Proxy 3 configuration - default cluster

Configure routing rules

To configure routing rules:

1. Under Additional properties of SIPProxSrv03, click **Routing rules**, and then click **New**.
2. From the Cluster pull-down menu shown in Figure 9-21 on page 337, select **PresenceRulesCluster**, and click **Apply**.

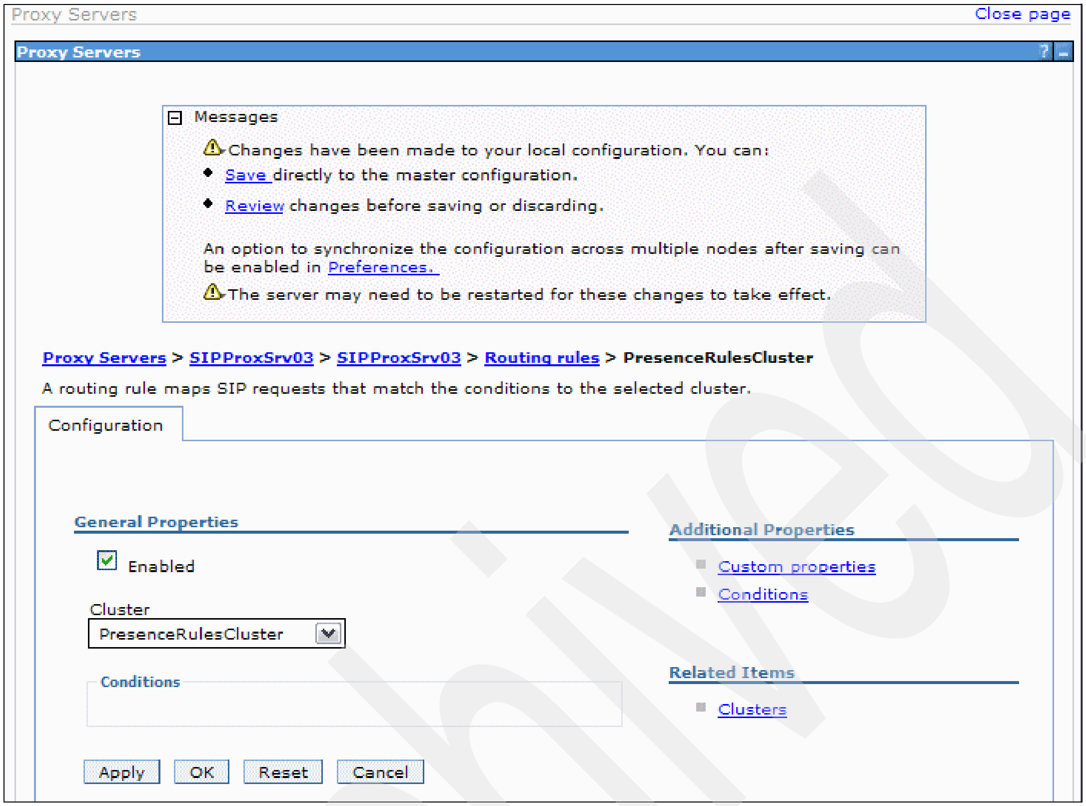


Figure 9-21 XDMS SIP Proxy 3 configuration - routing rules

3. Under Additional Properties, click **Conditions**, and then click **New**.
4. Select **Condition type: Other**.
5. From the Type pull-down menu, shown in Figure 9-22 on page 338, select **Request URI**.
6. For the Condition value, enter **rules**, and click **OK**.

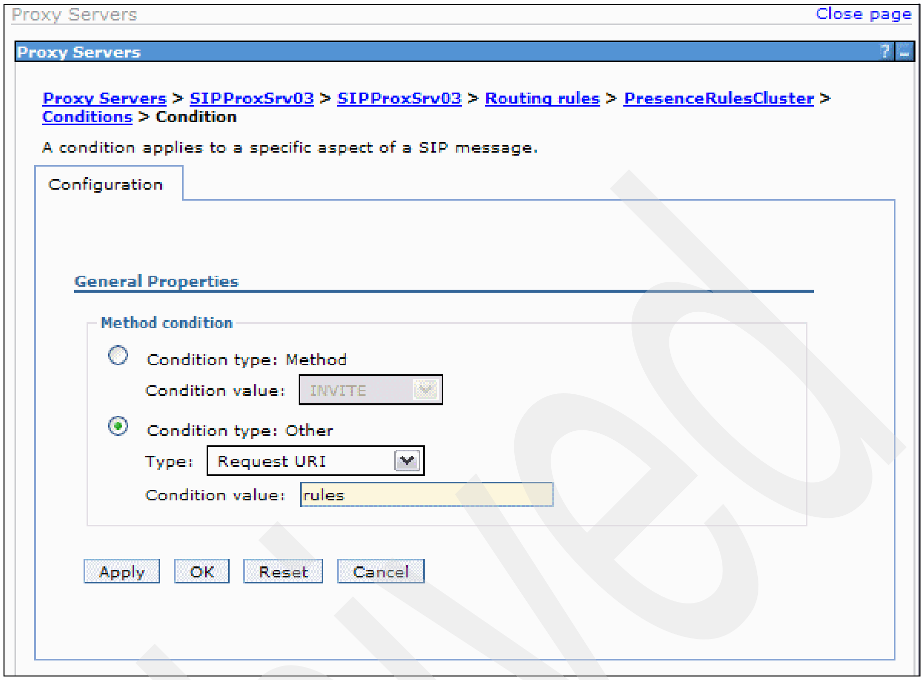


Figure 9-22 XDMS SIP Proxy 3 configuration - condition configuration

7. Save the changes to the master repository.

9.2.4 Installing the Aggregation Proxy

A high-availability (HA) environment is created with additional nodes for all major components. The Aggregation Proxy is also extended for this capability to take care of the fail-over conditions. AGGProxySrv01 is already available with the base environment; however, you must add AGGproxySrv02.

To install the Aggregation Proxy:

1. From the Integrated Solutions Console, click **Servers** → **Clusters**.
2. Click **AggregationProxyCluster** to select it for configuration, as shown in Figure 9-23 on page 339.

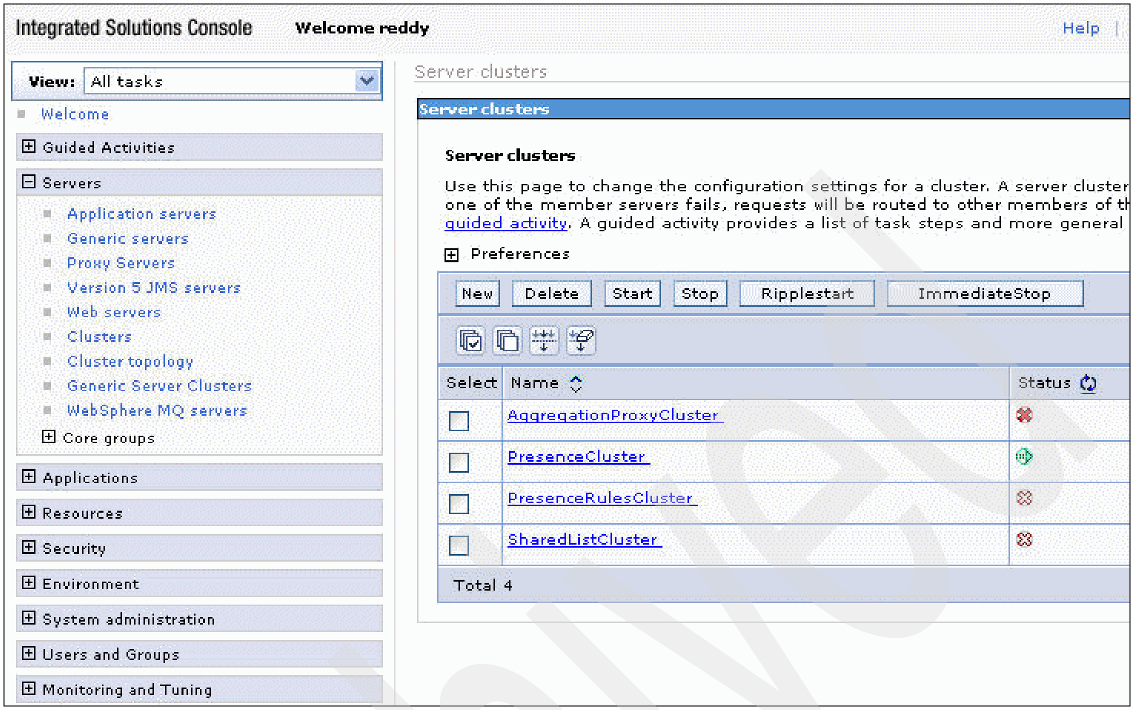


Figure 9-23 List of available clusters

3. Under Additional Properties, shown in Figure 9-24 on page 340, click **Cluster members**.

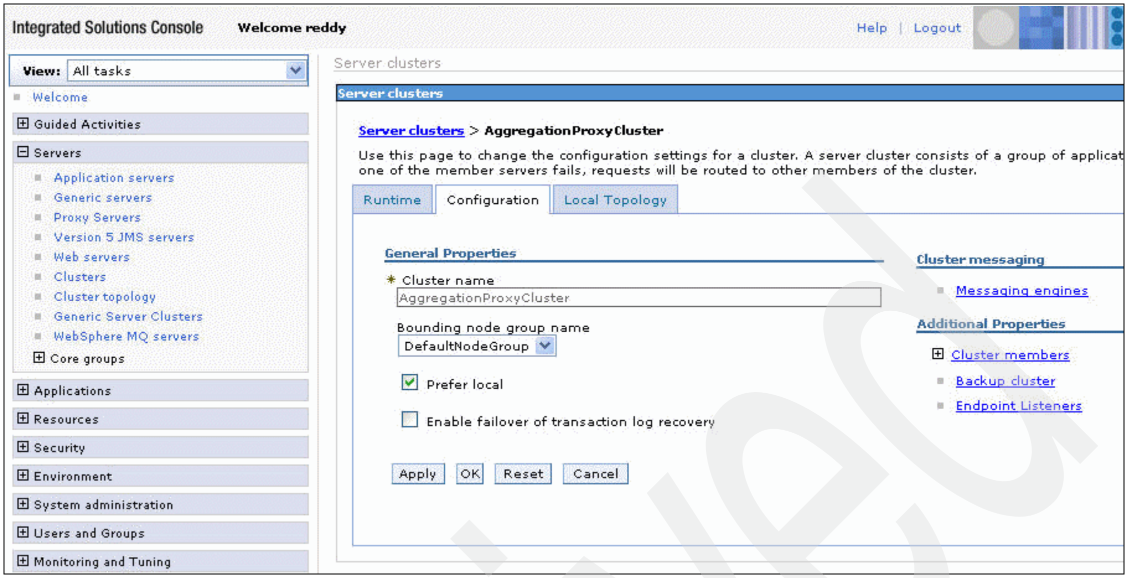


Figure 9-24 Basic cluster information

4. Click **New** to create a new member, as shown in Figure 9-25.

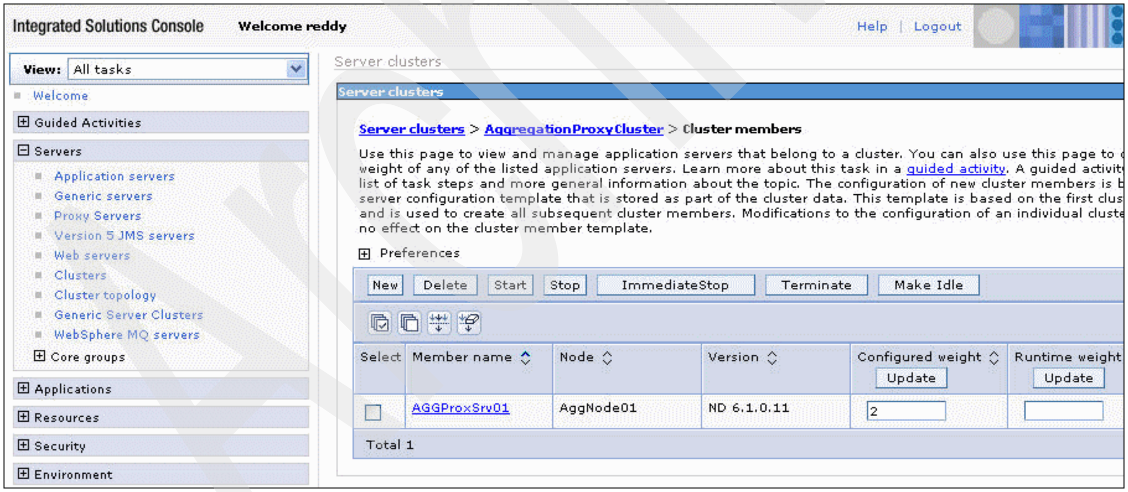


Figure 9-25 Add a new member

5. On the Create new cluster members page (Figure 9-26 on page 341), perform the following, and then click **Next**:

   a. In the Member name field, enter **AGGProxySvr02**.

   b. From the **Select node** pull-down menu, select **AggNode02(ND 6.1.0.11)**.

c. Set the Weight value to 2.
d. Select the **Generate unique HTTP ports** option.

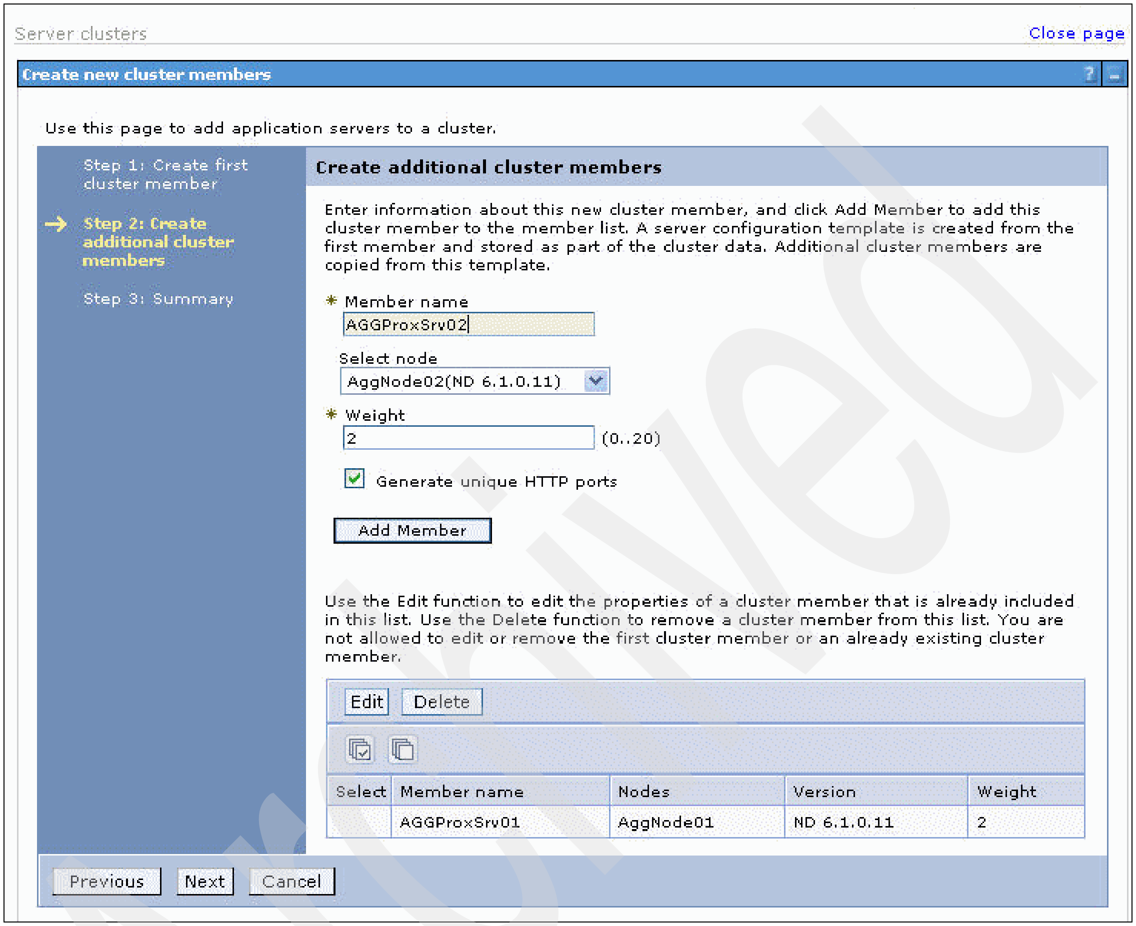


Figure 9-26 Added the member to this cluster

6. On the Summary page (Figure 9-27 on page 342), review the summary, and click **Finish** to accept the changes.

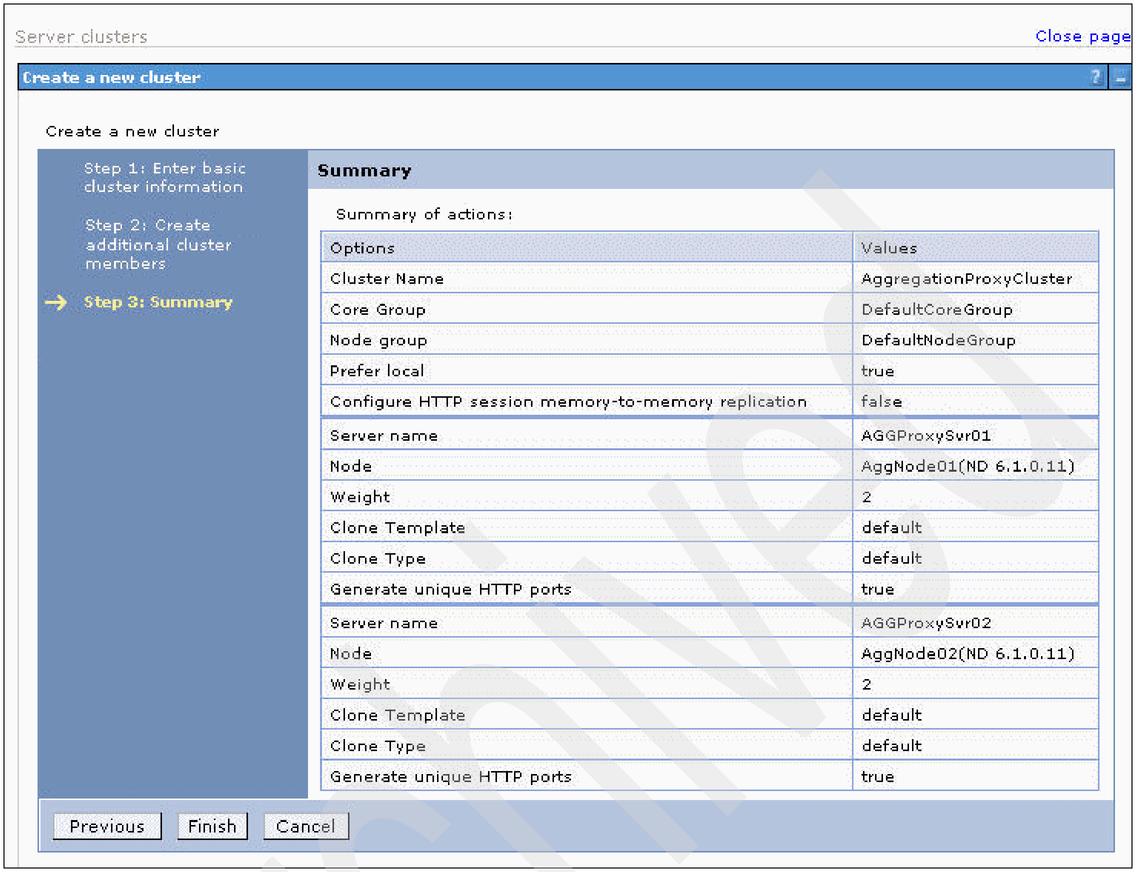


Figure 9-27 New cluster member summary page

7. In the Server clusters messages box (Figure 9-28 on page 343), click **Save** to apply the changes to the master configuration.

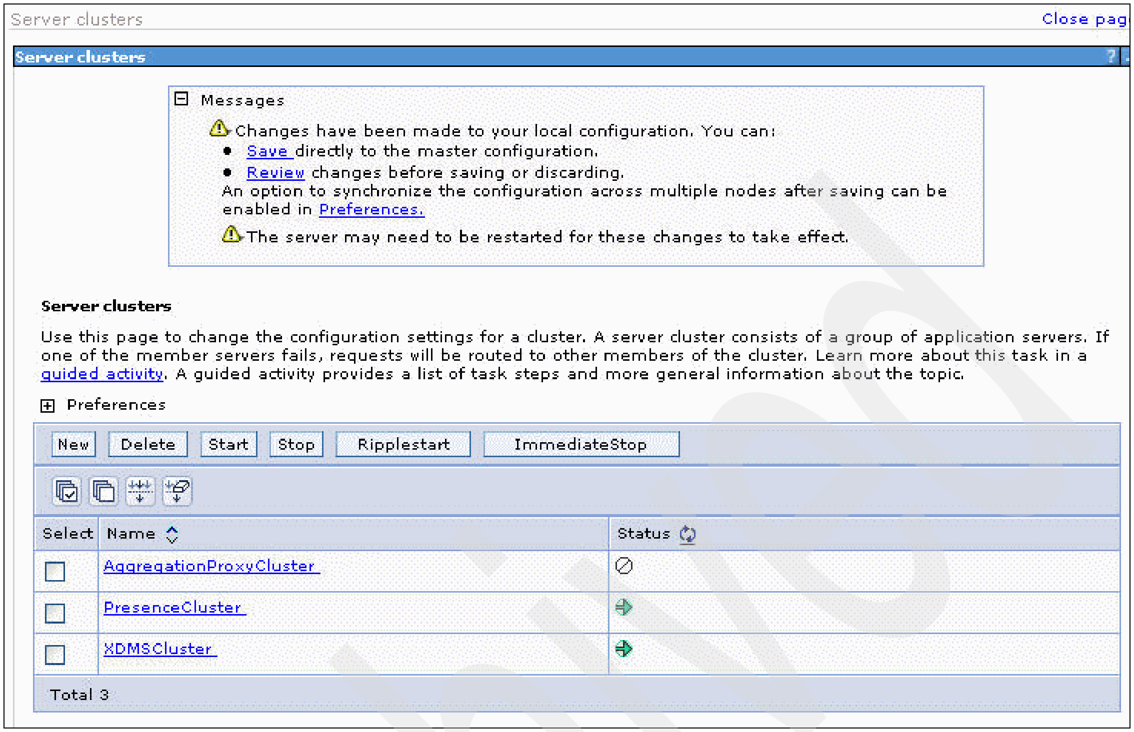


Figure 9-28 Session completed status

9.3 Validating the high-availability (HA) environment

In this section, we discuss the following tests and reports:

- Basic validation test

  For a discussion report, look for file report.20071006.zip in Appendix B, “Additional material” on page 479.

- HA validation tests

  For a discussion report, look for file report.20071008.zip in Appendix B, “Additional material” on page 479.

9.3.1 Basic validation tests

For basic validation of the HA environment, we used the following three scenarios:

Availability Test XCAP GET requests sent to the XDMS Shared List and XDMS Presence Rules servers.

Write Test Sequence of XCAP PUT, GET, DELETE requests that are sent to the XDMS Shared List and XDMS Presence Rules servers with database verification after PUT and DELETE operations.

Subscribe Test SIP SUBSCRIBE request that is sent to the Presence Server.

We prepared the tests as Unit Test scenarios based on JUnit framework and packaged for particular components:

- XDMS Shared List: com.ibm.itso.tests.basic.sharedlist
- XDMS Presence Rules: com.ibm.itso.tests.basic.presrules
- Presence Server: com.ibm.itso.tests.basic.presence

To run tests, we used an ant script, as shown in Example 9-12, from the LDAP machine.

Example 9-12 Test run

```
[root@ldap test_env]# ant test-html
Buildfile: build.xml

test-html:
    [junit] Running com.ibm.itso.tests.basic.presence.PresenceSubscribeTest
    [junit] Tests run: 1, Failures: 0, Errors: 0, Time elapsed: 4.135 sec
  [junit] Running com.ibm.itso.tests.basic.presrules.XDMSPresenceRulesAvailabi
lityTest
    [junit] Tests run: 5, Failures: 0, Errors: 0, Time elapsed: 1.056 sec
  [junit] Running com.ibm.itso.tests.basic.presrules.XDMSPresenceRulesWriteTes
t
    [junit] Tests run: 5, Failures: 0, Errors: 0, Time elapsed: 1.757 sec
    [junit] Running
com.ibm.itso.tests.basic.sharedlist.XDMSSharedListAvailabilityTest
    [junit] Tests run: 10, Failures: 0, Errors: 0, Time elapsed: 1.37 sec
    [junit] Running com.ibm.itso.tests.basic.sharedlist.XDMSSharedListWriteTest
    [junit] Tests run: 5, Failures: 0, Errors: 0, Time elapsed: 1.964 sec
    [junit] Running com.ibm.xcap.client.ITSOSimpleTest
    [junit] Tests run: 2, Failures: 0, Errors: 0, Time elapsed: 0.256 sec
```

```
[junitreport] Processing /root/test_env/reports/TESTS-TestSuites.xml to
/tmp/null184085093
[junitreport] Loading stylesheet /root/test_env/resources/junit-frames.xsl
[junitreport] Transform time: 3974ms
[junitreport] Deleting: /tmp/null184085093
     [copy] Copying 34 files to /var/www/html

BUILD SUCCESSFUL
Total time: 21 seconds
```

Note: The report that is generated from tests are (report.20071006.zip) in Appendix B, "Additional material" on page 479.

XDMS Shared List tests

The first scenario for this component is XDMSSharedListAvailabilityTest. This scenario contains several tests with different input points in the infrastructure. The Test run scenario calls the following nodes: *XdmsNode01, XdmsNode02, ProxNode03, ProxNode04, Agg.Prox01, AggProx02, ProxNode01, ProxNode02, imsNode*. For information about the nodes in the infrastructure and the role of particular nodes, see 2.4, "Implementation overview" on page 50.

Figure 9-29 on page 346 shows the sequence diagram of xdmsTest of the XDMSSharedListAvailabilityTest scenario.

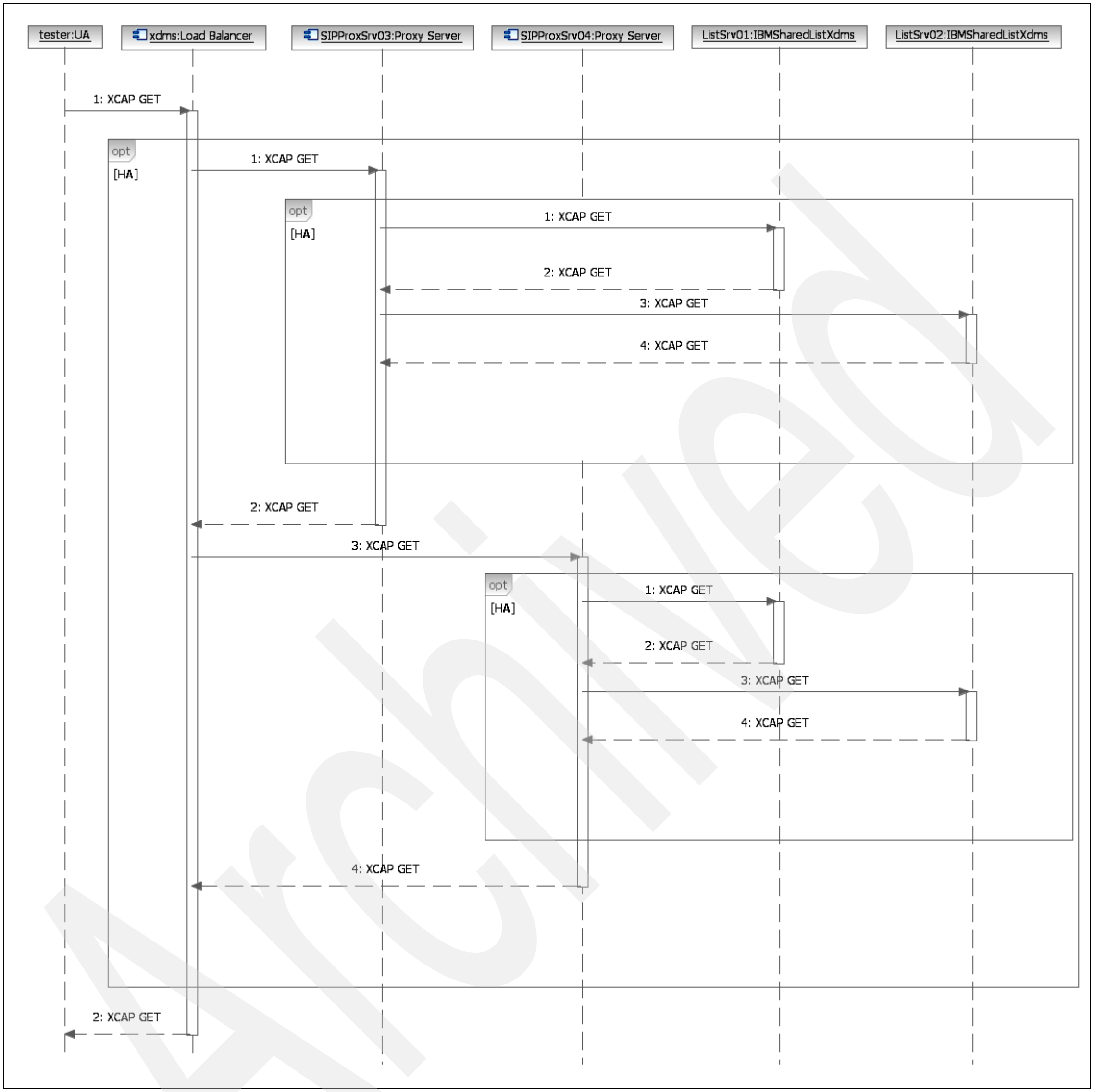


Figure 9-29 XDMSSharedListAvailabilityTest scenario

When executing the scenario, the XCAP GET request is sent serially to each server, and if HTTP 200 OK response is received, the test finished successfully. The called address is:

http://<hostname>:9082/services/xcap-caps/global/index

If all of the components are up and working correctly, your results are similar to Figure 9-30.

ITSO Infrastructure - Unit Test Results

Class com.ibm.itso.tests.basic.sharedlist.XDMSSharedListAvailabilityTest

Name	Tests	Errors	Failures	Time(s)	Time Stamp	Host
XDMSSharedListAvailabilityTest	10	0	0	1.372	2007-10-07T07:25:14	ldap.itso.ral.ibm.com

Tests

Name	Status	Type	Time(s)
testXdmsNode01	Success		0.627
testXdmsNode02	Success		0.052
testProxNode03	Success		0.058
testProxNode04	Success		0.056
testXDMSNode	Success		0.073
testAggProxyNode01	Success		0.046
testAggProxyNode02	Success		0.055
testProxyNode01	Success		0.042
testProxyNode02	Success		0.077
testIMSNode	Success		0.079

Figure 9-30 XDMSSharedListAvailabilityTest results

The second scenario for the XDMS Shared List component is XDMSSharedListWriteTest. This scenario contains five tests for following nodes: xdms1, xdms2, prox3 prox4, and xdms. Figure 9-31 on page 348 presents the general sequence diagram of this scenario. On Figure 9-31 on page 348, the component SharedList represents a particular node that is used in a particular test. XDMS is the database name of the Shared List Server.

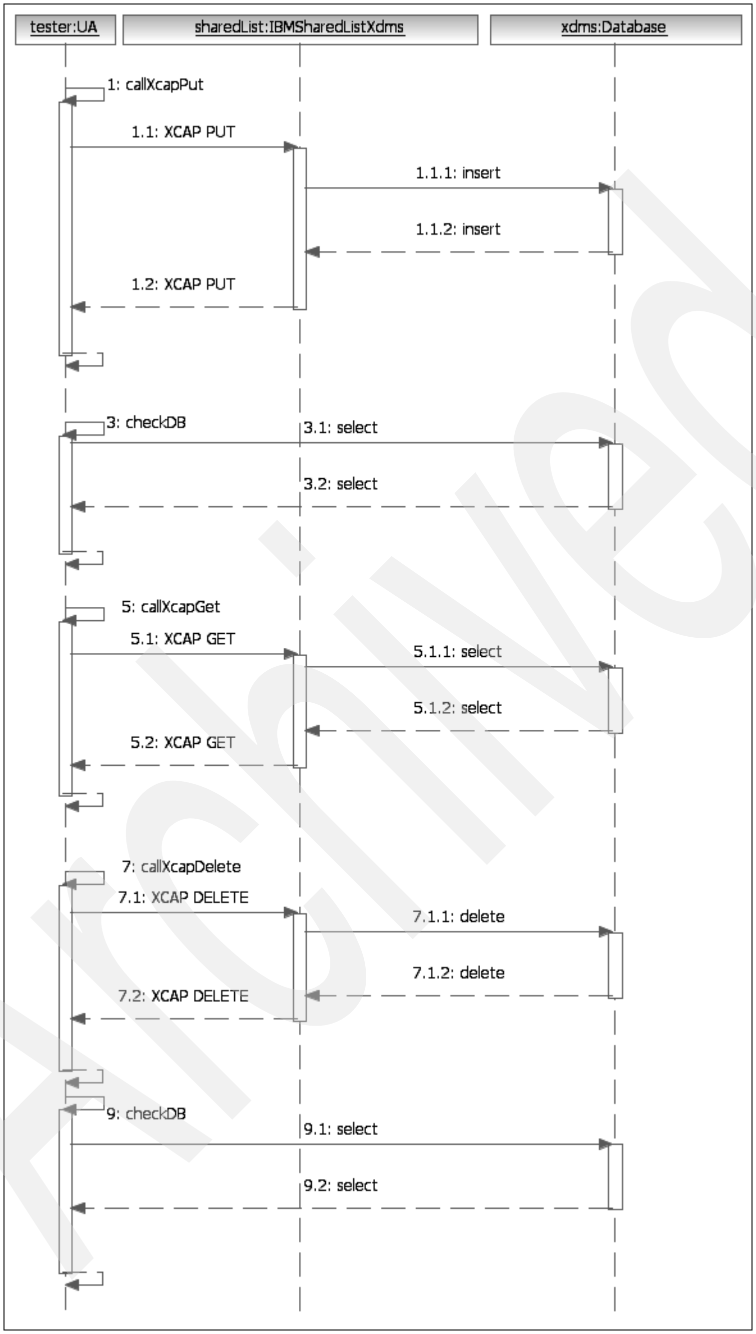
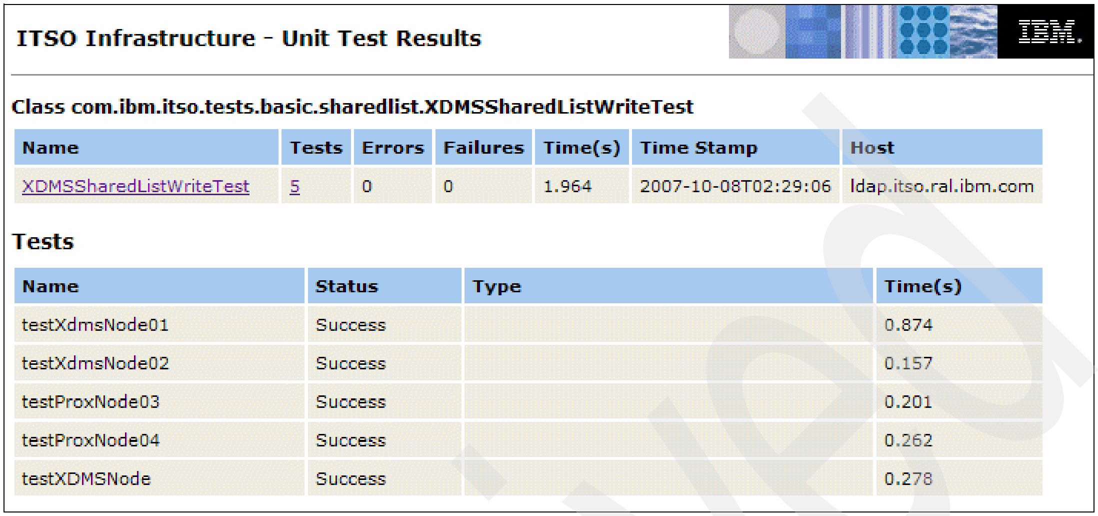


Figure 9-31 XDMSSharedListWriteTest scenario

Figure 9-32 shows the results of the XDMSSharedListWriteTest from our ITSO lab environment.



ITSO Infrastructure - Unit Test Results

Class com.ibm.itso.tests.basic.sharedlist.XDMSSharedListWriteTest

Name	Tests	Errors	Failures	Time(s)	Time Stamp	Host
XDMSSharedListWriteTest	5	0	0	1.964	2007-10-08T02:29:06	ldap.itso.ral.ibm.com

Tests

Name	Status	Type	Time(s)
testXdmsNode01	Success		0.874
testXdmsNode02	Success		0.157
testProxNode03	Success		0.201
testProxNode04	Success		0.262
testXDMSNode	Success		0.278

Figure 9-32 XDMSSharedListWriteTest results

XDMS Presence Rules tests

Similar to tests for the XDMS Shared List component, we used two scenarios the for XDMS Presence Rules testing. Figure 9-33 on page 350 presents the first scenario, XDMSPresenceRulesAvailabilityTest.

Class com.ibm.itso.tests.basic.presrules.XDMSPresenceRulesAvailabilityTest

Name	Tests	Errors	Failures	Time(s)	Time Stamp	Host
XDMSPresenceRulesAvailabilityTest	5	0	0	1.056	2007-10-08T02:28:59	ldap.itso.ral.ibm.com

Tests

Name	Status	Type	Time(s)
testXdmsNode01	Success		0.596
testXdmsNode02	Success		0.062
testProxNode03	Success		0.081
testProxNode04	Success		0.060
testXDMSNode	Success		0.064

Scenario

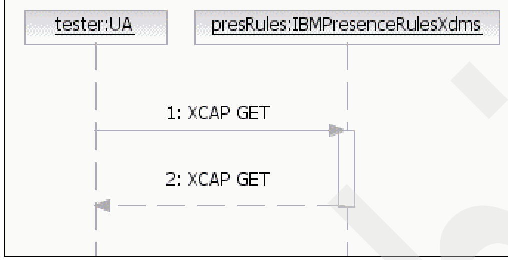


Figure 9-33 XDMSPresenceRulesAvailabilityTest

The second scenario, XDMSPresenceRulesWriteTest, is based on the sequence that Figure 9-34 on page 351 presents.

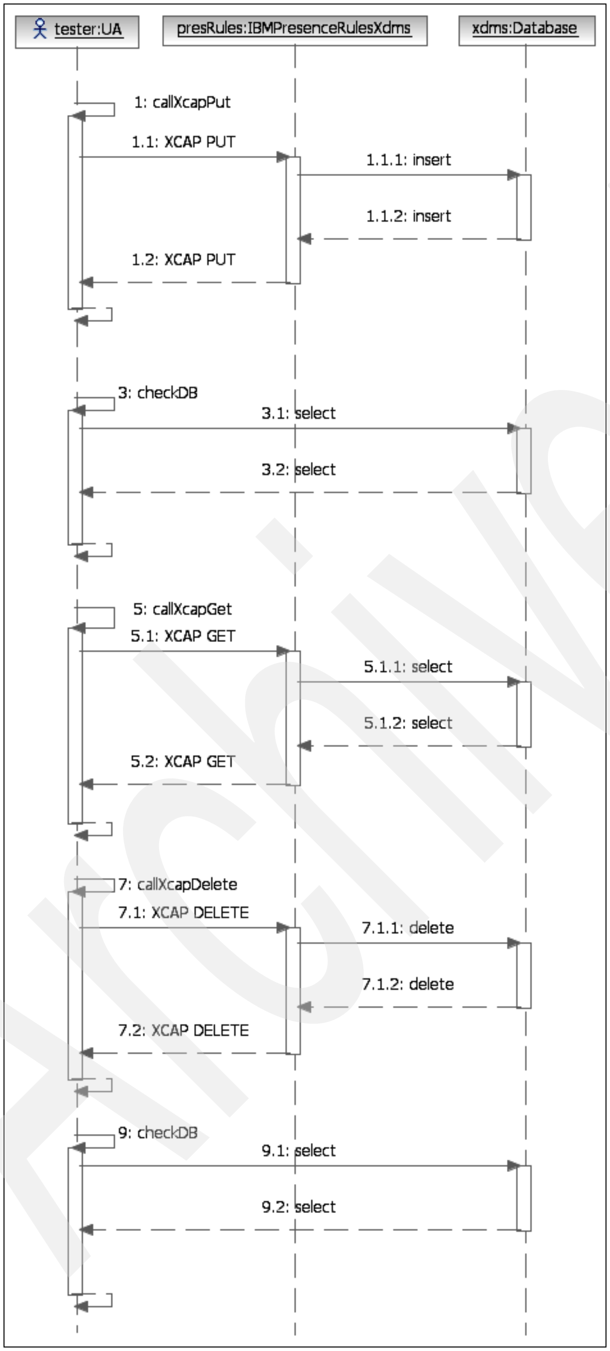


Figure 9-34 XDMSPresenceRulesWriteTest scenario

Presence Server tests

We tested the Presence Server based on the scenario in Figure 9-35. This test scenario performs the following steps:

1. The UA sends a SUBSCRIBE message (1) to the Presence Server.
2. The Presence Server sends a NOTIFY message (1.1) with pending status to the UA.
3. The Presence Server sends a SUBSCRIBE message (1.2) to the XDMS Presence Rules to subscribe to a document with Presence Rules for a user who initialized a subscription dialog.
4. A notification (1.2.1) with a Presence Rules document address is returned to the Presence Server.
5. Based on retrieved information, the Presence Server gets a document (1.2.1.1 / 1.2.1.2) with Presence Rules for a user using the XCAP GET interface.
6. The Presence Rules are verified on the Presence Server, and in case of a successful verification, a second NOTIFY message is sent to UA with status Active (1.2.1.3).

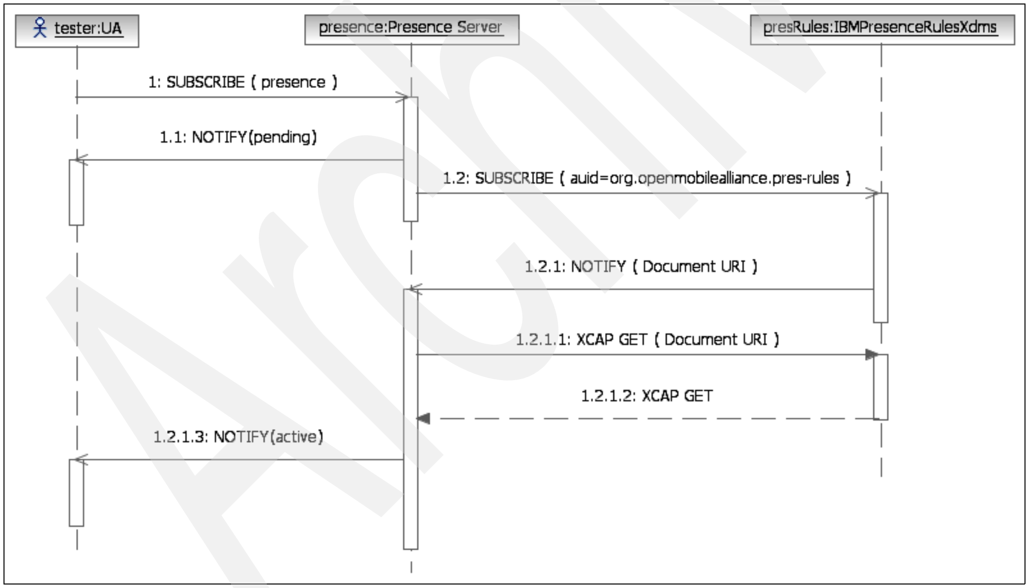


Figure 9-35 PresenceSubscribeTest scenario

Figure 9-36 on page 353 shows the results of the PresenceSubscribeTest scenario.

ITSO Infrastructure - Unit Test Results

Class com.ibm.itso.tests.basic.presence.PresenceSubscribeTest

Name	Tests	Errors	Failures	Time(s)	Time Stamp	Host
PresenceSubscribeTest	1	0	0	4.135	2007-10-08T02:28:54	ldap.itso.ral.ibm.com

Tests

Name	Status	Type	Time(s)
testSubscribeToPresent	Success		3.947

Figure 9-36 PresenceSubscribeTest results

9.3.2 HA validation tests

Basic HA validation is based on the Unit Test scenarios that we presented in the previous sections but with the assumption that one component in each cluster is not available. Figure 9-37 on page 354 shows the environment that we tested in this case.

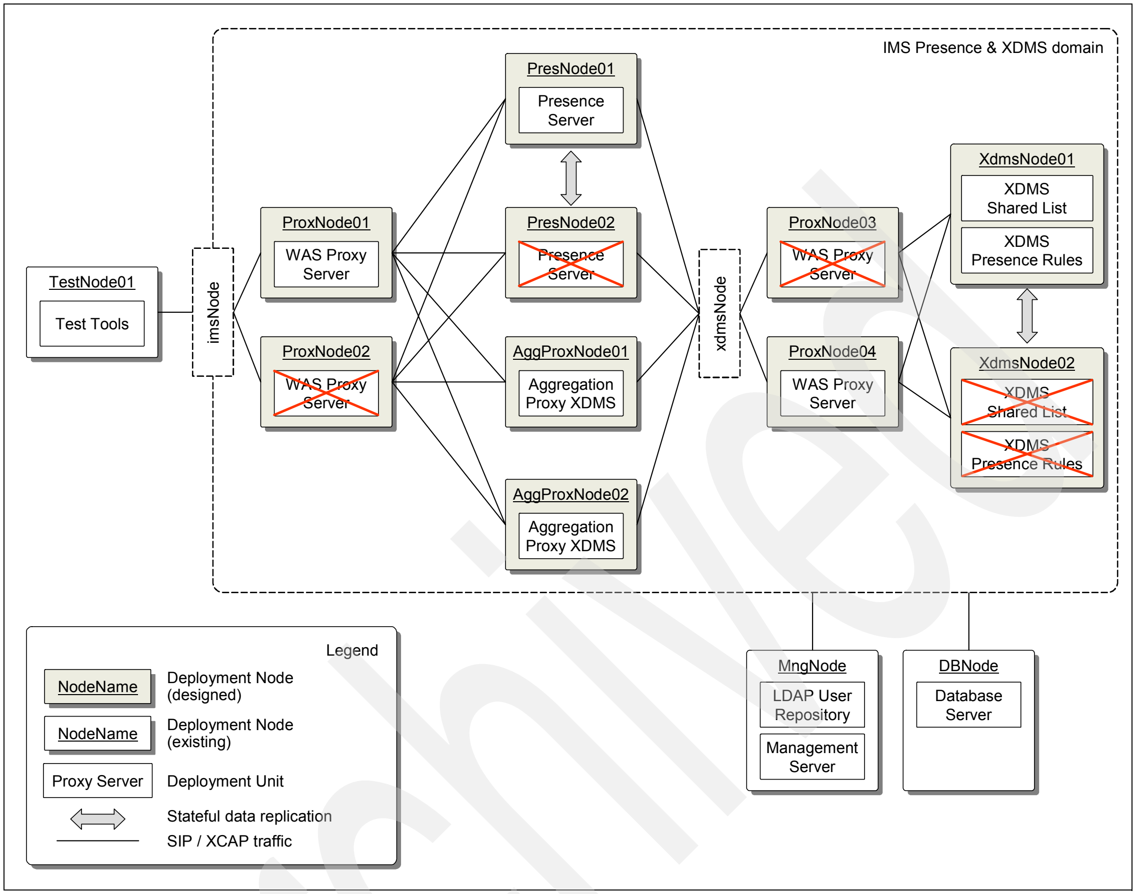
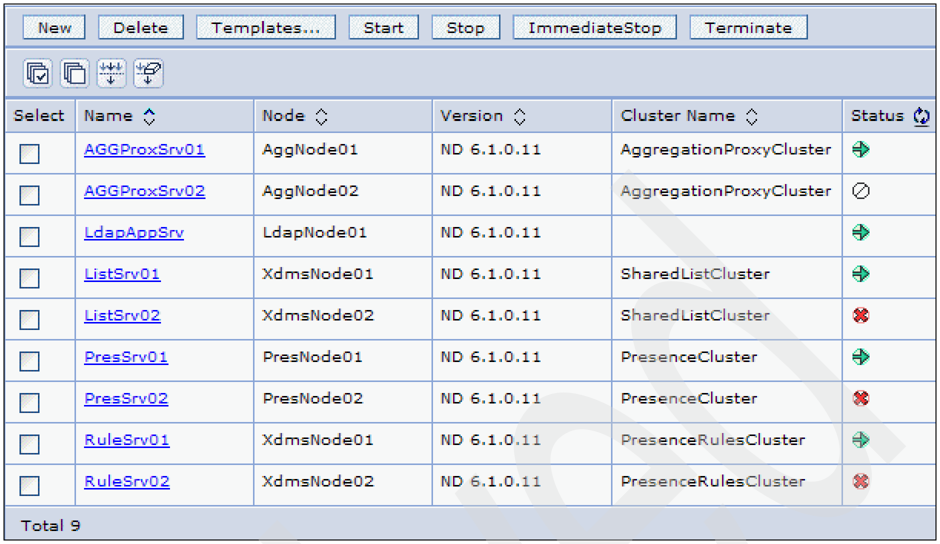


Figure 9-37 HA validation test

Figure 9-38 on page 355 presents the status of each Application Server, and Figure 9-39 on page 355 shows the status of the Proxy Server in the tested environment during this HA basic test.

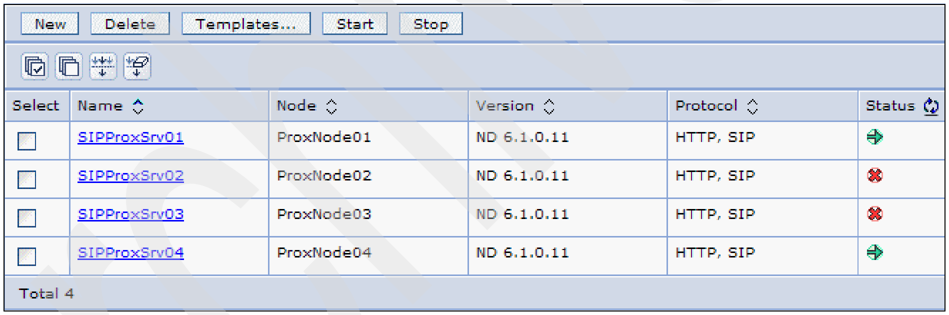


New | Delete | Templates... | Start | Stop | ImmediateStop | Terminate

Select	Name	Node	Version	Cluster Name	Status
☐	AGGProxSrv01	AggNode01	ND 6.1.0.11	AggregationProxyCluster	
☐	AGGProxSrv02	AggNode02	ND 6.1.0.11	AggregationProxyCluster	
☐	LdapAppSrv	LdapNode01	ND 6.1.0.11		
☐	ListSrv01	XdmsNode01	ND 6.1.0.11	SharedListCluster	
☐	ListSrv02	XdmsNode02	ND 6.1.0.11	SharedListCluster	
☐	PresSrv01	PresNode01	ND 6.1.0.11	PresenceCluster	
☐	PresSrv02	PresNode02	ND 6.1.0.11	PresenceCluster	
☐	RuleSrv01	XdmsNode01	ND 6.1.0.11	PresenceRulesCluster	
☐	RuleSrv02	XdmsNode02	ND 6.1.0.11	PresenceRulesCluster	

Total 9

Figure 9-38 WebSphere Application Server Console - Application Server view



New | Delete | Templates... | Start | Stop

Select	Name	Node	Version	Protocol	Status
☐	SIPProxSrv01	ProxNode01	ND 6.1.0.11	HTTP, SIP	
☐	SIPProxSrv02	ProxNode02	ND 6.1.0.11	HTTP, SIP	
☐	SIPProxSrv03	ProxNode03	ND 6.1.0.11	HTTP, SIP	
☐	SIPProxSrv04	ProxNode04	ND 6.1.0.11	HTTP, SIP	

Total 4

Figure 9-39 WebSphere Application Server Console - Proxy Server view

From these tests, we can expect that even if one server in the cluster is not running, functionality from another server remains available for end users, in this case, Tester.

Figure 9-40 on page 356 shows the results for the PresenceSubscribeTest, XDMSPresenceRulesAvailabilityTest, and XDMSPresenceRulesWriteTest.

PresenceSubscribeTest	testSubscribeToPresent	Success	
XDMSPresenceRulesAvailabilityTest	testXdmsNode01	Success	
XDMSPresenceRulesAvailabilityTest	**testXdmsNode02**	**Failure**	**Should not catch Exception: Connection refused** junit.framework.AssertionFailedError: Should not catch Exception: Connection refused at com.ibm.itso.tests.ItsoXcapCase.callXcapGet(ItsoXcapCase.java:141) at com.ibm.itso.tests.basic.presrules.XDMSPresenceRulesAvailabilityTest.testXdmsNode02(XDMSPresenceRulesAvailabilityTest.java:23
XDMSPresenceRulesAvailabilityTest	**testProxNode03**	**Failure**	**Should not catch Exception: Connection refused** junit.framework.AssertionFailedError: Should not catch Exception: Connection refused at com.ibm.itso.tests.ItsoXcapCase.callXcapGet(ItsoXcapCase.java:141) at com.ibm.itso.tests.basic.presrules.XDMSPresenceRulesAvailabilityTest.testProxNode03(XDMSPresenceRulesAvailabilityTest.java:29
XDMSPresenceRulesAvailabilityTest	testProxNode04	Success	
XDMSPresenceRulesAvailabilityTest	testXDMSNode	Success	
XDMSPresenceRulesWriteTest	testXdmsNode01	Success	
XDMSPresenceRulesWriteTest	**testXdmsNode02**	**Failure**	**==[XCapPUT]== should not catch Exception: Connection refused** junit.framework.AssertionFailedError: ==[XCapPUT]== should not catch Exception: Connection refused at com.ibm.itso.tests.ItsoXcapCase.callXcapPut(ItsoXcapCase.java:170) at com.ibm.itso.tests.basic.presrules.XDMSPresenceRulesWriteTest.testXdmsNode02(XDMSPresenceRulesWriteTest.java:28)
XDMSPresenceRulesWriteTest	**testProxNode03**	**Failure**	**==[XCapPUT]== should not catch Exception: Connection refused** junit.framework.AssertionFailedError: ==[XCapPUT]== should not catch Exception: Connection refused at com.ibm.itso.tests.ItsoXcapCase.callXcapPut(ItsoXcapCase.java:170) at com.ibm.itso.tests.basic.presrules.XDMSPresenceRulesWriteTest.testProxNode03(XDMSPresenceRulesWriteTest.java:36)
XDMSPresenceRulesWriteTest	testProxNode04	Success	
XDMSPresenceRulesWriteTest	testXDMSNode	Success	
XDMSSharedListAvailabilityTest	testXdmsNode01	Success	

Figure 9-40 HA tests results from the Presence Server and XDMS Presence Rules

In Figure 9-40, the first row contains information about the SUBSCRIBE scenario for the Presence Server. The test finished successfully even when PresSrv02 is down. As you remember from the sequence diagram, for this scenario there is several internal communications between the Presence Server and the XDMS Server during this test. Because all tests finished successfully, we know that internal communications were successful as well.

The next five rows, in Figure 9-40, concern the XDMSPresenceRulesAvailabilityTest. We see that two of the servers are down, and connection to these servers was refused. Even that last test in this section, testXDMSNode, which uses XDMS as an access point for the Presence Rules interfaces, finished successfully. It means that from a functional point-of-view, the infrastructure provided the expected functionality.

The scenario, XDMSPresenceRulesWriteTest, that Figure 9-40 shows is very similar to the previous one. Functionality is available (testXDMSNode finished successfully) even when two nodes are down.

Figure 9-41 on page 357 shows the results for the XDMSSharedListAvailabilityTest and XDMSSharedListWriteTest scenarios. As you can see on Figure 9-41 on page 357: testXDMSNode and testIMSNode finished successfully, which means that functions of the XDMS components are available, even if some servers in the clusters are down.

XDMSPresenceRulesWriteTest	testProxNode04	Success	
XDMSPresenceRulesWriteTest	testXDMSNode	Success	
XDMSSharedListAvailabilityTest	testXdmsNode01	Success	
XDMSSharedListAvailabilityTest	**testXdmsNode02**	**Failure**	**Should not catch Exception: Connection refused** junit.framework.AssertionFailedError: Should not catch Exception: Connection refused at com.ibm.itso.tests.ItsoXcapCase.callXcapGet(ItsoXcapCase.java:141) at com.ibm.itso.tests.basic.sharedlist.XDMSSharedListAvailabilityTest.testXdmsNode02(XDMSSharedListAvailabilityTest.java:23)
XDMSSharedListAvailabilityTest	**testProxNode03**	**Failure**	**Should not catch Exception: Connection refused** junit.framework.AssertionFailedError: Should not catch Exception: Connection refused at com.ibm.itso.tests.ItsoXcapCase.callXcapGet(ItsoXcapCase.java:141) at com.ibm.itso.tests.basic.sharedlist.XDMSSharedListAvailabilityTest.testProxNode03(XDMSSharedListAvailabilityTest.java:29)
XDMSSharedListAvailabilityTest	testProxNode04	Success	
XDMSSharedListAvailabilityTest	testXDMSNode	Success	
XDMSSharedListAvailabilityTest	testAggProxyNode01	Success	
XDMSSharedListAvailabilityTest	**testAggProxyNode02**	**Failure**	**Should not catch Exception: Connection refused** junit.framework.AssertionFailedError: Should not catch Exception: Connection refused at com.ibm.itso.tests.ItsoXcapCase.callXcapGet(ItsoXcapCase.java:141) at com.ibm.itso.tests.basic.sharedlist.XDMSSharedListAvailabilityTest.testAggProxyNode02(XDMSSharedListAvailabilityTest.java:53)
XDMSSharedListAvailabilityTest	testProxyNode01	Success	
XDMSSharedListAvailabilityTest	**testProxyNode02**	**Failure**	**Should not catch Exception: Connection refused** junit.framework.AssertionFailedError: Should not catch Exception: Connection refused at com.ibm.itso.tests.ItsoXcapCase.callXcapGet(ItsoXcapCase.java:141) at com.ibm.itso.tests.basic.sharedlist.XDMSSharedListAvailabilityTest.testProxyNode02(XDMSSharedListAvailabilityTest.java:65)
XDMSSharedListAvailabilityTest	testIMSNode	Success	
XDMSSharedListWriteTest	testXdmsNode01	Success	
XDMSSharedListWriteTest	**testXdmsNode02**	**Failure**	**==[XCapPUT]== should not catch Exception: Connection refused** junit.framework.AssertionFailedError: ==[XCapPUT]== should not catch Exception: Connection refused at com.ibm.itso.tests.ItsoXcapCase.callXcapPut(ItsoXcapCase.java:170) at com.ibm.itso.tests.basic.sharedlist.XDMSSharedListWriteTest.testXdmsNode02(XDMSSharedListWriteTest.java:27)
XDMSSharedListWriteTest	**testProxNode03**	**Failure**	**==[XCapPUT]== should not catch Exception: Connection refused** junit.framework.AssertionFailedError: ==[XCapPUT]== should not catch Exception: Connection refused at com.ibm.itso.tests.ItsoXcapCase.callXcapPut(ItsoXcapCase.java:170) at com.ibm.itso.tests.basic.sharedlist.XDMSSharedListWriteTest.testProxNode03(XDMSSharedListWriteTest.java:37)
XDMSSharedListWriteTest	testProxNode04	Success	
XDMSSharedListWriteTest	testXDMSNode	Success	

Figure 9-41 HA tests results from the XDMS Shared List

10

Testing the high-availability (HA) scenario

In this chapter, we provide the details about the configured use of high availability for end-to-end message transactions for XDMS and Presence. The description includes the failure of different nodes in the message path and the rerouting for successful handling of event bursts. We focus the experiences in this chapter on controlled stoppage of different servers at different times during message transactions.

10.1 Overview of the high-availability (HA) scenario test cases

In this chapter, we test the architecture that we presented in Chapter 9, “The high-availability environment setup and configuration” on page 307. These tests are mostly focused on the fail-over functionality, which is one of the most important items from a service availability point-of-view. We described some of the fail-over capabilities of this solution in previous chapters. In the following sections, we illustrate failures of the key software elements of the solution, and how fail over and recovery are handled.

10.2 Scenarios that we discuss

We discuss high availability and fail over in details based on the following scenarios:

- TS.01.Presence Server unavailable
- TS.02.XDMS Server unavailable
- TS.03.Redudant nodes unavailable
- TS.04.Aggregation Proxy unavailable

10.2.1 TS.01.Presence Server unavailable

For a normal working scenario of all of the servers, Proxy Servers process the SIP requests to both instances of the Presence Servers. The SIP sessions are established between the client and the Presence Servers through Proxy Servers. Replication Domain function is an integral capability of WebSphere Application Server that helps to replicate these sessions within the Presence Cluster.

In a situation where one of the Presence Servers is force stopped or non-functional, the second available Presence Server takes over and starts processing the ongoing requests from the client. Proxy Servers are provided with the information about the rerouted messages to the available Presence Server after knowing that the required Presence Server is down.

To demonstrate both situations, we use the following test cases in this scenario:

- Planned stop / start of Presence Server
  `stopServer.sh` command is used.
- Unexpected stop of Presence Server
  java process is killed.

Figure 10-1 presents the Operational Model for this test scenario.

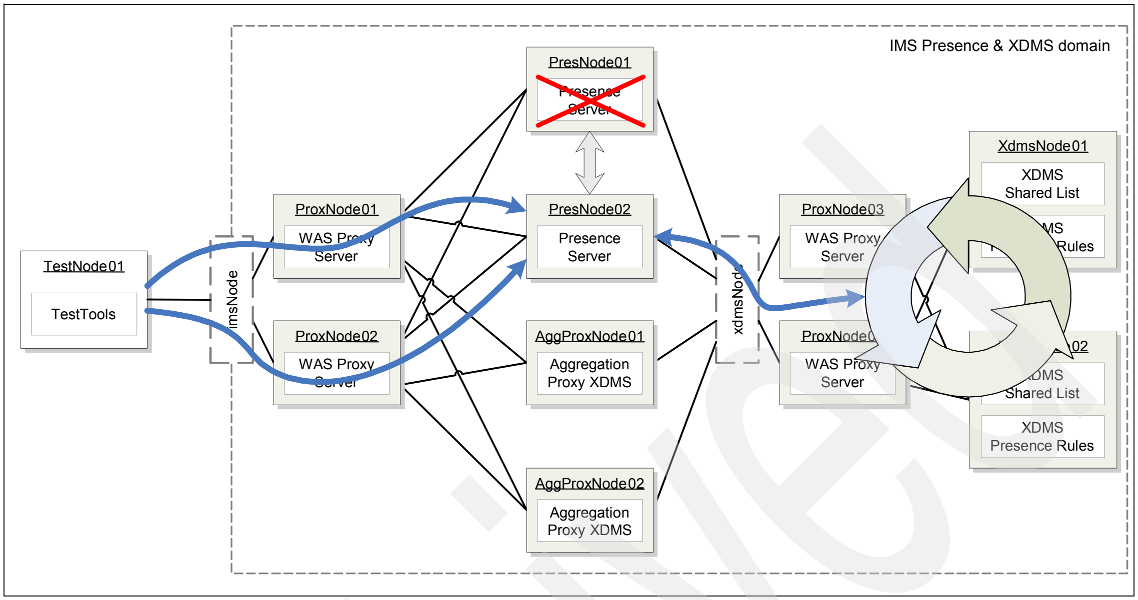


Figure 10-1 Test scenario 1 - Operational Diagram

As presented in Figure 10-1, we simulate the stop or crash of the Presence Server that is working on PresSrv01. Whereas, if there is a controlled stop of the server before the server stops, the fail-over event is sent to all servers and replication is completed. If there is a crash of the Java process, the HA Manager that is running in the cell is informed and fail-over procedures are taken. Presence Server 2, which is working on PresSrv02, is informed of the failure and activates its copy of the session to begin accepting requests and handling responses for that session.

In addition, the PresenceCluster is updated with the unavailability of PresSrv01 on PresNode01. This information is also distributed to the SIP proxies. From that time any requests for the established session are rerouted to PresSrv02, which is running on PresNode02 and no new requests are routed to PresSrv01.

If there is a crash of the Presence Server instance, the recovery of that instance is handled by the nodeagent that is running on the same node. In our case, this is the nodeagent on PresNode01. After it establishes that PresSrv01 is no longer running and has handled the fail-over procedures, it automatically attempts to restart the instance on the server, PresSrv01.

After the new instance of PresSrv01 is running, the Presence Cluster and the Proxy Servers are informed that the new instance is available to receive new requests.

> **Note:** The failed over sessions are not migrated back to PresSrv01 on Node01. Only new requests are sent to the instance PresSrv01. However, the Replication Domain process is reestablished, so the two instances or Presence Server can provide continued availability.

The described scenario was tested and presented in 10.3.2, “TS.01.Presence Server unavailable” on page 368.

10.2.2 TS.02. XMDS Server unavailable

The functioning of this scenario is similar to the Presence Server. The controlled stop is performed on one of the XDMS Servers to maintain the concurrency of messaging between client and available XDMS Server:

- Planned stop / start of XDMS Server (Shared List, Presence Rules)
- Unexpected stop of XDMS Server (Shared List, Presence Rules)

Figure 10-2 shows the Operational Diagram for test scenario 02.

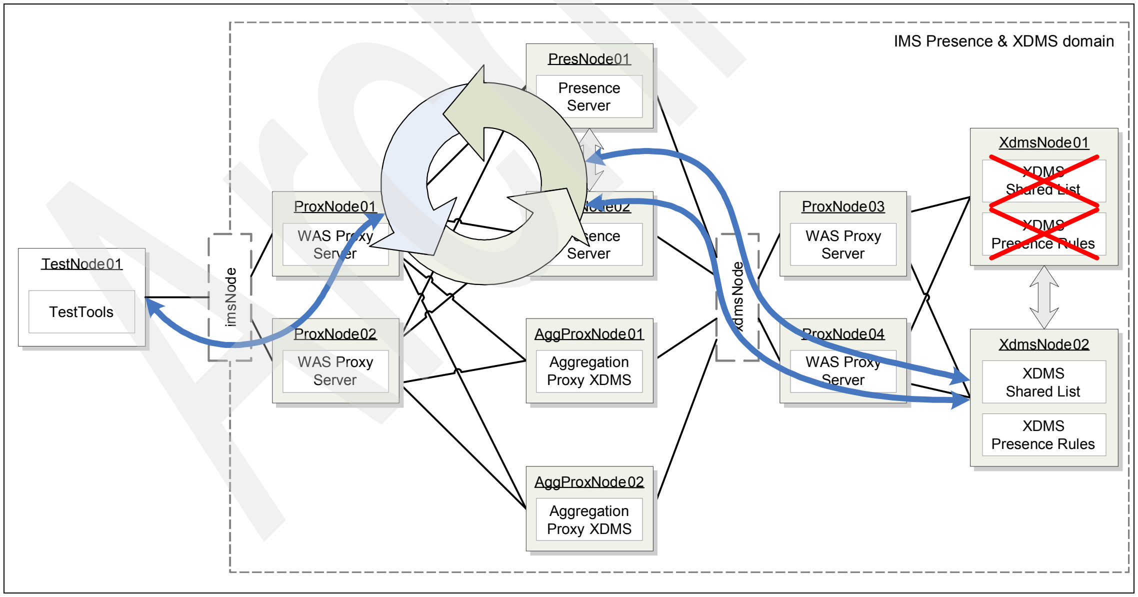


Figure 10-2 Test scenario 02 - Operational Diagram

Because the XDMS Server and the Presence Server based on the WebSphere Application Server infrastructure, the high availability is provided in the same manner for both components.

In a case where the XDMS Servers experience a system crash, a similar situation occurs as we described in 10.3.2, “TS.01.Presence Server unavailable” on page 368. The Replication Domain process is run and the session that was established on XDMS Server 1 is handled by XDMS Server 2. Either the Shared List Server or the Presence Rules Server on node XdmsNode1 is marked as unavailable in the HA Manager, and no new sessions are sent to that server by the Proxy Servers.

The test results that correspond to the scenario that is described for a controlled failure of an XDMS Server is described in 10.3.3, “TS.02.XMDS Server unavailable” on page 372.

Proxy Server failure

A failed Proxy Server must be handled by an out of scope element, the Load Balancer. The Load Balancer sprays requests to one of the Proxy Servers in the Proxy Cluster (typically done in a round-robin fashion). If a Proxy Server fails, the Load Balancer must detect its unavailability and stop routing requests to it until it recovers. Mostly, to identify if the Proxy Server is available, the Load Balancer sends a request to Proxy in periodical fashion:

- For HTTP Proxy, usually it is HTTP GET / request
- For SIP Proxy, usually it is SIP OPTION request

> **Note:** Refer to the WebSphere Application Server InfoCenter to see details about the SIP Proxy configuration with Load Balancer, in the scope of SIP OPTION Requests:
>
> http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.nd.multiplatform.doc/info/ae/ae/tjpx_lb_proxy.html

10.2.3 TS.03.Redundant nodes unavailable

This test is similar to the controlled stopping that is planned in case of the individual server that is responsible for respective tasks. But, this time it is the controlled stopping of more than one Application Servers.

Figure 10-3 on page 364 shows the possibility of the message transaction survival, even if more than one redundant server fails or forces closure.

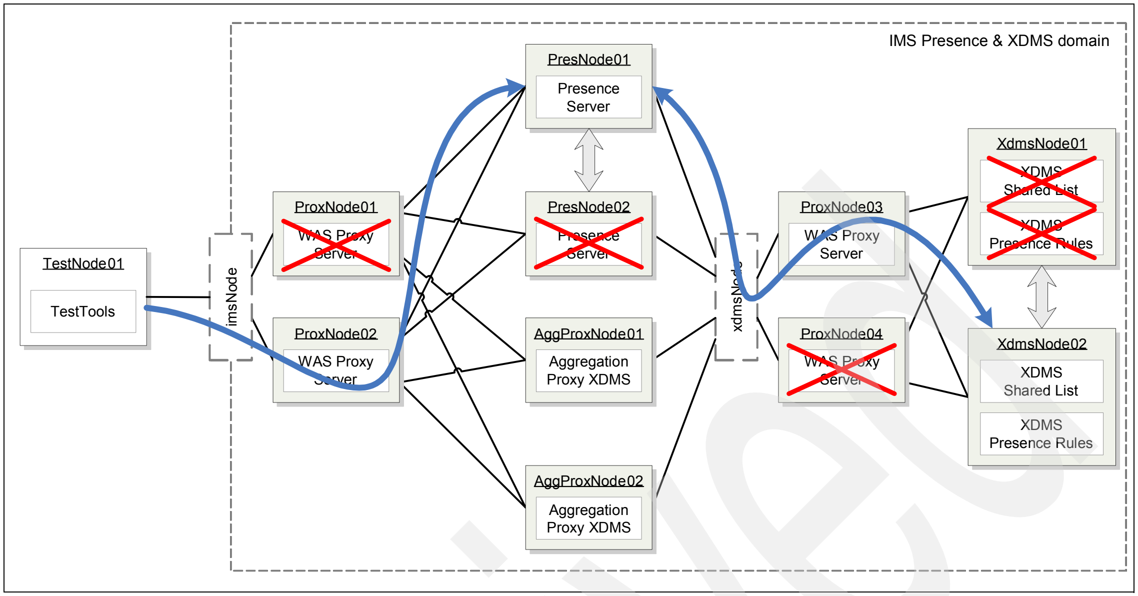


Figure 10-3 Test scenario 04 - Operational Diagram

In Figure 10-3, all of the elements are always active, which means that they are working in ACTIVE-ACTIVE mode. To maintain complete service at full utilization, each element should be sized to handle twice its normal peak throughput to provide the necessary availability in case of a peer element fails.

This test case demonstrate the possibility of stopping or failing the entire redundant path for each type of server in the transaction path. The results of this test are discussed in detail in 10.3.4, “TS.03.Redundant nodes unavailable” on page 381.

10.2.4 TS.04.Aggregation Proxy unavailable

Despite the commonality in the names, the Aggregation Proxy is implemented differently than the converged HTTP/SIP proxy that we just discussed. The major difference is that the Aggregation Proxy is implemented as a WAS application like the Presence and XDMS Server and not a special entity, such as the WAS HTTP/SIP proxy. Given that, the fail-over capabilities of the Aggregation Proxy are the same as that of the Presence and XDMS. The Aggregation Proxy utilizes the fail-over capabilities that the WebSphere Application Server ND infrastructure (for example, replication domains, HA Manager, and so on) provides. As seen in Figure 10-4 on page 365, the Aggregation Proxies are fronted by the converged HTTP/SIP proxies, which provided the initial fail-over capability to the Aggregation Proxy Cluster. In the event of a planned or unplanned failure, the HTTP/SIP proxies detect the unavailability of the downstream server and appropriately handle the routing of new requests. The replication domains that exist between the Aggregation Proxy instances handle the communications between the instances and the fail-over duties.

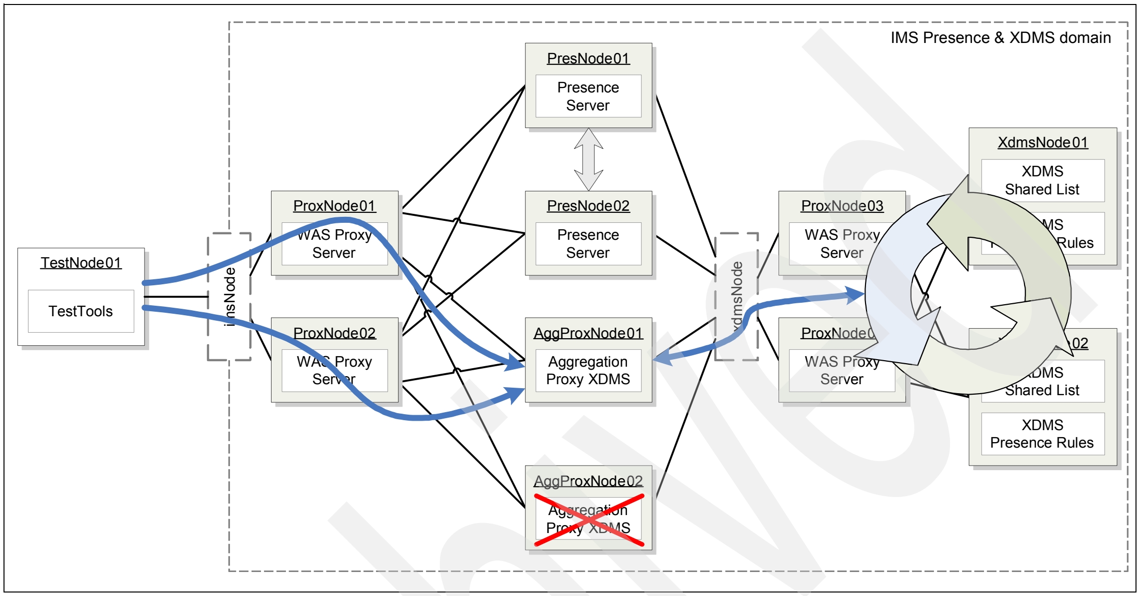


Figure 10-4 Test scenario 05 - Operational Diagram

Given that the fail-over mechanisms of the Aggregation Proxy are the same as that of the Presence and XDM Servers, we did not documented the execution of the failure scenarios for this component.

10.3 Executing the highly-available scenario test cases

The high-available scenario is defined to make sure that the messaging transaction across the entire path is intact, even after the failure of certain servers that are responsible for communication. The redundancy is planned with double the sizing capacity, anticipating that the messaging traffic gets diverted to the next available server for processing. The tests are conducted in multiples with Proxy Servers, Presence Servers, and XDMS Servers, which includes the controlled failure of the individual redundant server and the complete redundancy.

10.3.1 TS.00.Preliminary test

This preliminary test scenario validates the testing approach before real HA tests. In this scenario, we use test case TC.00.0.Preliminary Test. In this section, we present how the scenario was run and what results we obtained. Almost all HA test scenarios, except the last one TS.14 Aggregation Proxy, are based on the same test approach. The steps for the HA test scenarios are very similar to the preliminary test.

Preliminary test case

We used the sipp tool for our business scenario. The process that is responsible for sending messages from sipp is named Sender.

To generate traffic, we ran five sipp scenarios that use Sender, based on our business scenario. Each Sender sent a number of requests, as shown on Figure 10-5. For the preliminary test, 20 requests were sent per sender.

```
                              Messages  Retrans   Timeout   Unexpected-Msg
SUBSCRIBE ---------->         20        0
      200 <----------         20        0                   0
    Pause [   1000ms]         20                            0
  PUBLISH ---------->         20        0
      200 <----------         20        0                   0
  PUBLISH ---------->         20        0
      200 <----------         20        0                   0
  PUBLISH ---------->         20        0
      200 <----------         20        0                   0
  PUBLISH ---------->         20        0
      200 <----------         20        0                   0
  PUBLISH ---------->         20        0
      200 <----------         20        0                   0
    Pause [   1000ms]         20                            0
SUBSCRIBE ---------->         20        0
      200 <----------         20        0                   0
  PUBLISH ---------->         20        0
      200 <----------         20        0                   0
  PUBLISH ---------->         20        0
      200 <----------         20        0                   0
  PUBLISH ---------->         20        0
      200 <----------         20        0                   0
  PUBLISH ---------->         20        0
      200 <----------         20        0                   0
  PUBLISH ---------->         20        0
      200 <----------         20        0                   0
    Pause [   1000ms]         20                            0
```

Figure 10-5 TC.00.0 Sender panel

In the same time frame and for each sipp Sender, we ran the co-respondent sipp process in background mode (named as Listener) to collect all notification from our test. Figure 10-6 shows the presence view from one of our listener processes.

```
                                 Messages  Retrans   Timeout   Unexpected-Msg
  ----------> NOTIFY             159       0                   0

  <---------- 200                159       0
------------------------------ Sipp Server Mode ------------------------------
```

Figure 10-6   TC.00.0 Listener panel

The test result data is verified for the expected notifications of all active users from the buddy list. These are the users who sent PUBLISH to the Presence Server. The shell script, with regular expressions, is prepared to add multiple PUBLISH requests, which generated the active users with buddy list and the expected required responses to the client. Figure 10-7 describes the responses to the client.

```
============================================
====[ Testing file: ./get_notify_10178_screen.log ]====
Expected NOTIFY calls : 140
Recived NOTIFY calls  : 159
====[ Test 1 Successfull ]====
====[ Testing file: ./get_notify_10178_messages.log ]====
Test2 for <user>1: 1 : Pass
Test2 for <user>2: 1 : Pass
Test2 for <user>3: 1 : Pass
Test2 for <user>4: 1 : Pass
Test2 for <user>5: 1 : Pass
Test2 for <user>6: 1 : Pass
Test2 for <user>7: 2 : Pass
Test2 for <user>8: 1 : Pass
Test2 for <user>9: 1 : Pass
Test2 for <user>10: 4 : Pass
Test2 for <user>11: 1 : Pass
Test2 for <user>12: 2 : Pass
Test2 for <user>13: 2 : Pass
Test2 for <user>14: 3 : Pass
Test2 for <user>15: 2 : Pass
Test2 for <user>16: 1 : Pass
Test2 for <user>17: 1 : Pass
Test2 for <user>18: 1 : Pass
Test2 for <user>19: 1 : Pass
Test2 for <user>20: 4 : Pass
============================================
```

Figure 10-7   TC.00.0 Test results

Collected messages are tested under the condition to get active notification of users from BuddyList. Example 10-1 on page 368 shows regular expressions that we used in our test. In Example 10-1 on page 368, we used user alice1 and the messages file from the test described above.

Example 10-1 Regular expression used in results test

```
[root@tester ha.TC.00.0.client]# egrep "^<\?xml (.*(state=\"active\").*){5}" \
./get_notify_10178_messages.log | grep [^0-9]alice1friends@itso.ral.ibm.com | wc -l
1
[root@tester ha.TC.11.0.client]#
```

Based on test results, we generated statistics from the load distribution. Figure 10-8 presents request distribution to Presence Servers during the test case.

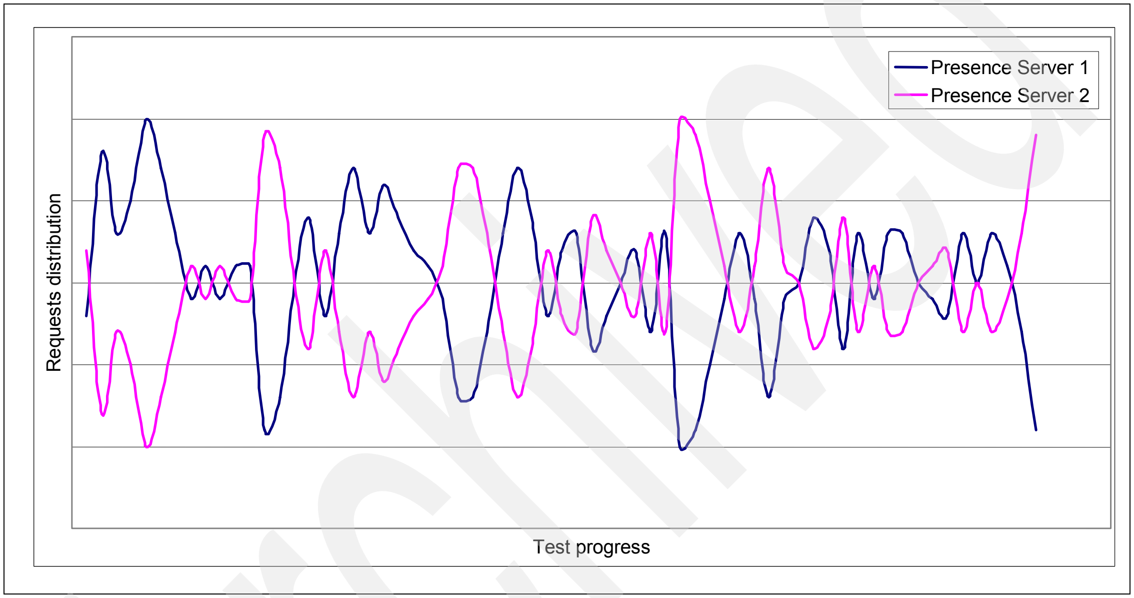


Figure 10-8 TC.00.0 Load distribution

As we can see in Figure 10-8, requests are quite evenly spread between both servers. It means that both nodes processed the request during this test, which we expected because while running this test case, we did not turn off any of servers.

10.3.2 TS.01.Presence Server unavailable

To test Presence Server availability, we are using Test Scenario TS.01.Presence Server unavailable described in 4.3, "Preparing the integration infrastructure" on page 119. The two test cases are run:

- TC.01.1.Planned stop and start of Presence Server
- TC.01.2.Unexpected stop of Presence Server

Planned stop and start of Presence Server

We performed this test case by stopping a Presence Server using the **stopServer.sh** command from the WebSphere Application Server package. The script for the requests is scheduled from the test server with the given business scenario of high availability, which we described in 2.1, “The scenario overview” on page 36. The execution of this test case is as similar as the preliminary test, but with the differences of controlled sopping of a Presence Server in between a client request session of a scheduled shell script on the test server.

Figure 10-9 presents the Sequence Diagram for this test.

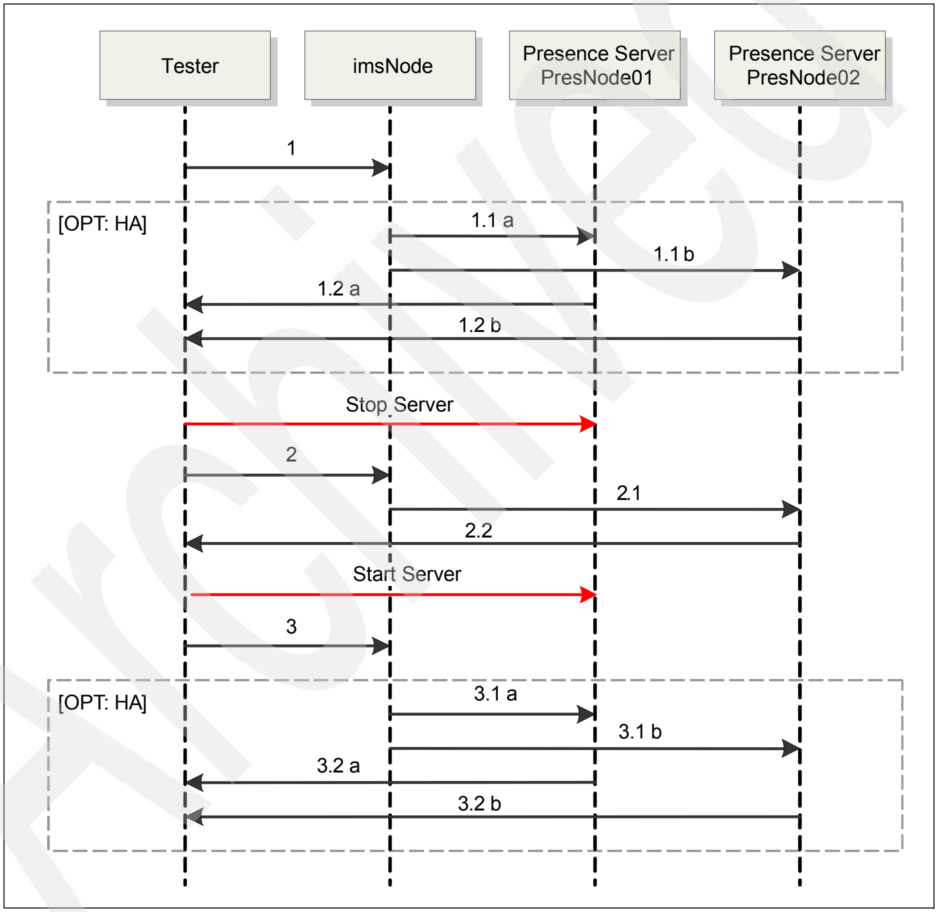


Figure 10-9 TC.01 Sequence Diagram

At the beginning, all traffic is distributed between two Presence Servers[1]. When the test are running, we stop Presence Server 1 [Stop Server] that is working on node PresNode01. After the server stops, we expect that all traffic gets directed only to server 2 on PresNode02. After that, we start Presence Server 1 [Start Server] and traffic should be again sprayed to both servers.

The command in Example 10-2 is an example of using **stopServer.sh**, which allows the controlled stopping of PresNode01.

Example 10-2 Stop Presence Server on node PresNode01

```
[root@pres1 bin]# pwd
/opt/ibm/WebSphere/AppServer/profiles/PresNode01/bin
[root@pres1 bin]# ./stopServer.sh PresSrv01 -user wasadmin -password <password>
```

Figure 10-10 presents how we spread the traffic during this test.

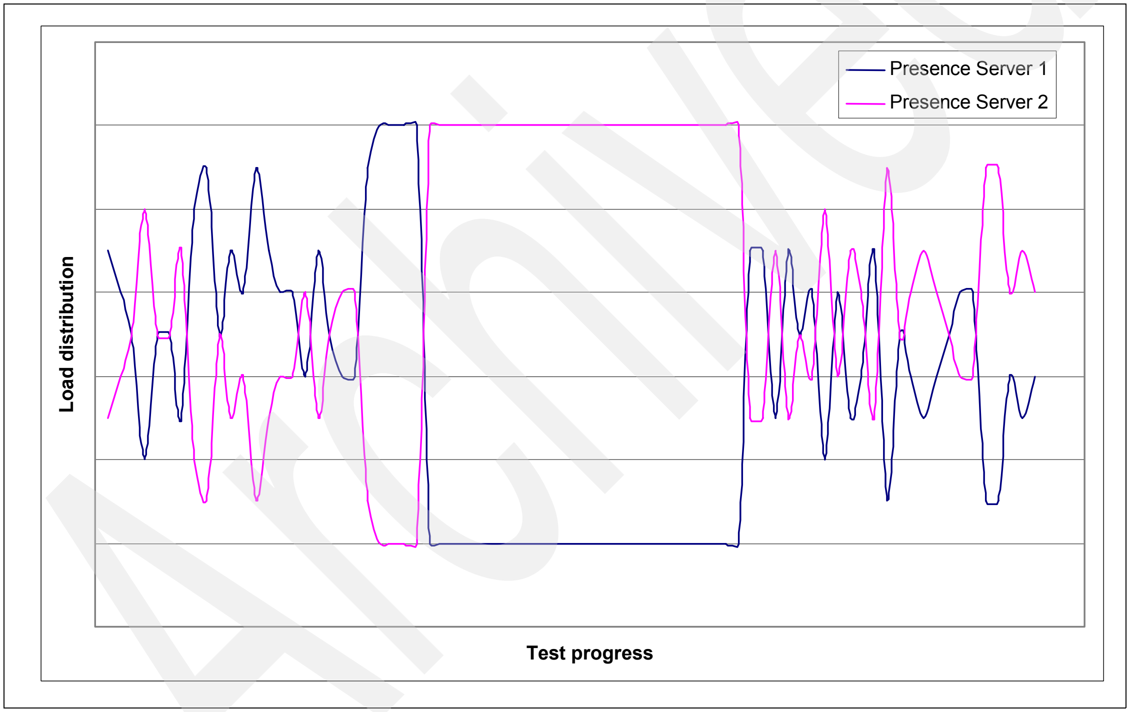


Figure 10-10 TC.11.1 Load distribution

As we expected, after we stopped Server Presence 1, Presence Server 2 processed all requests.

During this test, all tests finished successfully, which means that we were notified about the Presence status for all tested subscriptions.

Unexpected stop of Presence Server

During this test case, we checked the Presence Server availability in terms of fail-over capabilities, as in the previous test. So the scenario is the same. The main difference is that the server process is unexpectedly killed or a controlled stopped is performed.

To kill the process, we use the **kill** command on the server PresNode01. Before doing so, we obtained the java process number of Presence Server. To obtain the java process number of Presence Server, go to the logs directory on the server, and look into the file PresSrv01.pid. Example 10-3 demonstrates how to do this.

Example 10-3 Kill Presence Server on PresNode01

```
[root@pres1 PresSrv01]# pwd
/opt/ibm/WebSphere/AppServer/profiles/PresNode01/logs/PresSrv01
[root@pres1 PresSrv01]# cat PresSrv01.pid
15842[root@pres1 PresSrv01]# kill -9 15842
[root@pres1 PresSrv01]#
```

After the server is killed, the role of the Node Agent is to start the server immediately. Example 10-4 demonstrates how you look into the SystemOut.log file to see that the server is starting after it is killed.

Example 10-4 Verify that server is starting up

```
[root@pres1 PresSrv01]# pwd
/opt/ibm/WebSphere/AppServer/profiles/PresNode01/logs/PresSrv01
[root@pres1 PresSrv01]# tail -f SystemOut.log
************ Start Display Current Environment ************
WebSphere Platform 6.1 [ND 6.1.0.11 cf110734.37] [Presence 6.2.0.0
20071014.1 20071014.1322]  running with process name
imsCell\PresNode01\PresSrv01 and process id 15842
Host Operating System is Linux, version 2.6.9-55.ELsmp
Java version = J2RE 1.5.0 IBM J9 2.3 Linux x86-32 j9vmxi3223-20060504
(JIT enabled)
J9VM - 20060501_06428_lHdSMR
JIT  - 20060428_1800_r8
GC   - 20060501_AA, Java Compiler = j9jit23, Java VM name = IBM J9 VM
was.install.root = /opt/IBM/WebSphere/AppServer
user.install.root = /opt/IBM/WebSphere/AppServer/profiles/PresNode01
....
```

Figure 10-11 on page 372 shows the load distribution between Presence Server 1 and Presence Server 2.

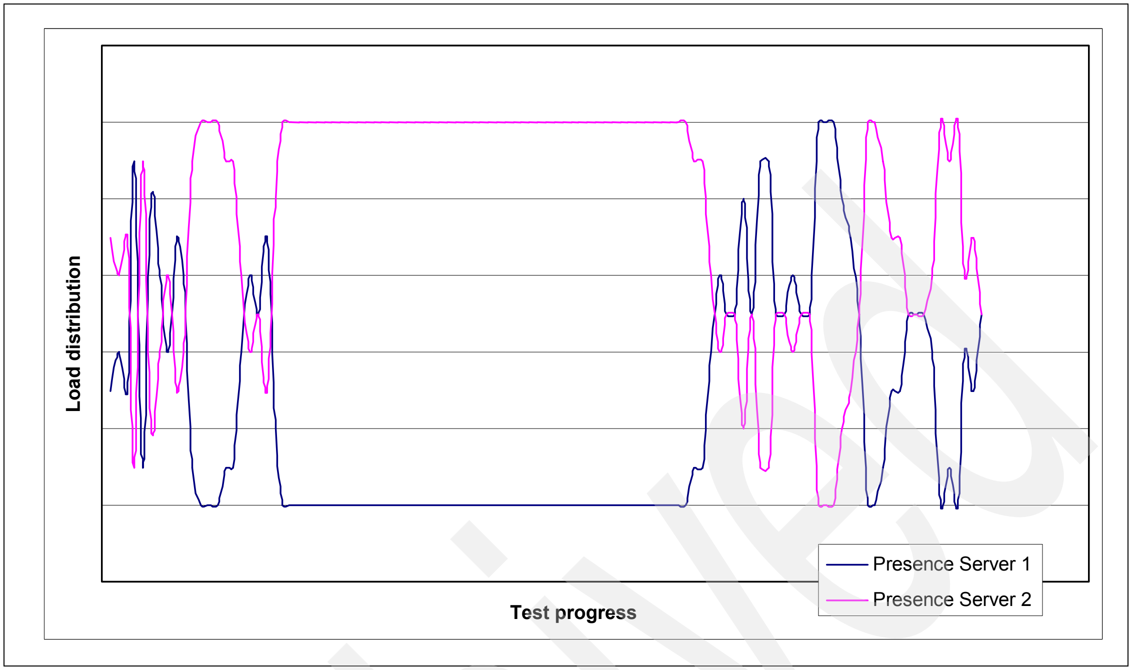


Figure 10-11 TC.11.2 Load distribution

You can see in Figure 10-11 that while the server was stopped, Presence Server 2 processed all traffic. After server 1 was started, the load was distributed to both servers again.

During the test, we recorded an interesting observation. One test failed because we killed the server in the moment when a SUBSCRIBE message was received but not fully processed by PresSrv01. Thus, information about the received subscription was not distributed across Presence Cluster.

10.3.3 TS.02.XMDS Server unavailable

To test XDMS Server services availability, we use test scenario TS.02.XDMS Server unavailable, which we described in 10.1, “Overview of the high-availability (HA) scenario test cases” on page 360. The following test cases are presented in this section:

- Planned stop and start of XDMS Servers
- Unexpected stop of XDMS Servers

Planned stop and start of the XDMS Servers

Planned server stop is achieved by using the **stopServer.sh** command from the WebSphere Application Server package on machine XdmsNode02. During the test, we stop and start the Shared List and Presence Rules servers.

Figure 10-12 presents the Sequence Diagram for this test.

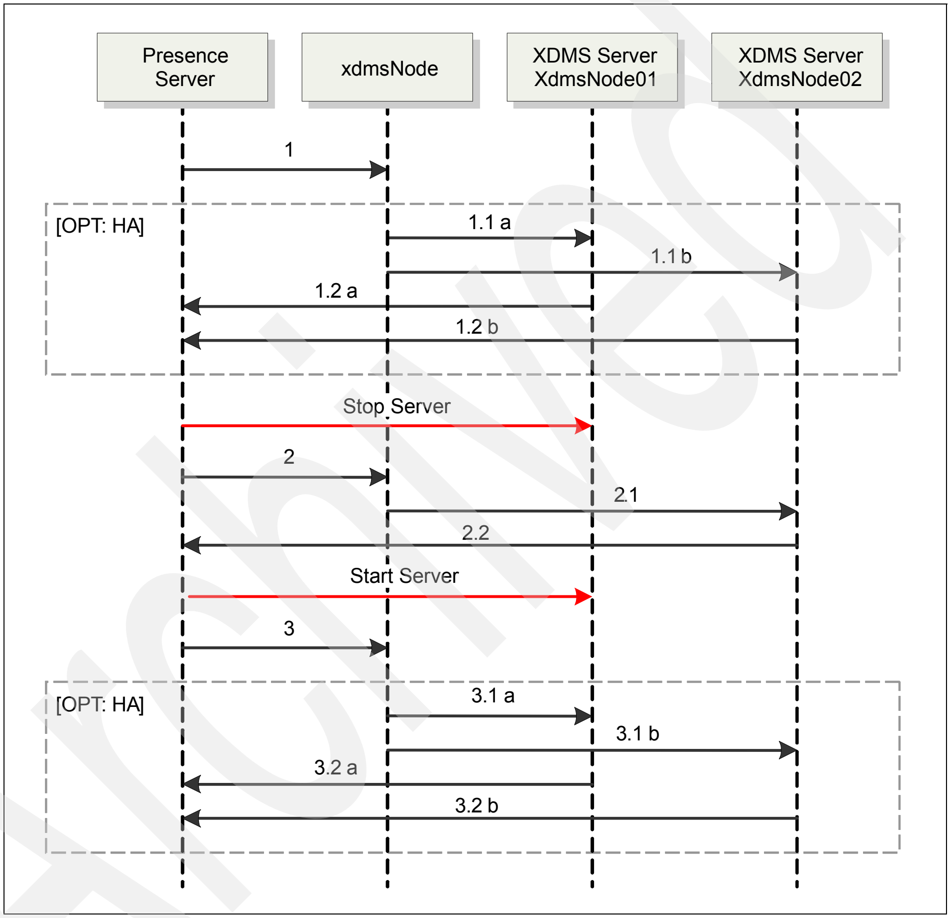


Figure 10-12 TS.02 Sequence Diagram

Initially, the entire traffic is distributed between both the XDMS Servers. In between the test execution of generating multiple PUBLISH requests, the controlled stopping is initiated either for Shared List or Presence Rules in XDMSNode02. After this action, the expectation is to observe the redirection of the existing transactions and the new requests to XDMSNode01. After the stabilization of the take over by XDMSNode01, the XDMSNode02 is restarted again to observe that the load of transaction is redistributed among both the XDMS Servers.

Example 10-5 shows how to use **stopServer.sh** to permit the controlled stopping of the Shared List on XDMSNode02:

Example 10-5 Stop Shared List Server on node XdmsNode02

```
[root@xdms2 bin]# pwd
/opt/ibm/WebSphere/AppServer/profiles/XdmsNode02/bin
[root@xdms2 bin]# ./stopServer.sh ListSrv02 -user wasadmin -password <password>
```

Example 10-6 shows how to use **stopServer.sh** to permit the controlled stopping of the Presence Rules on XDMSNode02:

Example 10-6 Stop Presence Rules Server on node XdmsNode02

```
[root@xdms2 bin]# pwd
/opt/ibm/WebSphere/AppServer/profiles/XdmsNode02/bin
[root@xdms2 bin]# ./stopServer.sh RuleSrv02 -user wasadmin -password <password>
```

After the server is stopped, we start it again using the **startServer** command from WebSphere Application Server.

Example 10-7 presents the start of the Shared List server.

Example 10-7 Start Shared List server

```
[root@xdms2 bin]# ./startServer.sh ListSrv02
```

Example 10-8 shows the start of the Presence Rules Server.

Example 10-8 Start Presence Rules Server

```
[root@xdms2 bin]# ./startServer.sh RuleSrv02
```

Figure 10-13 on page 375 shows the Shared List server’s behavior during the test. The graph demonstrates the percentage of request that the Shared List servers on XdmsNode01 and XdmsNode02 processed. Two Presence Servers worked as clients of the Shared List in this test, so on the graph you can see four lines, which represents two lines per client (Presence Server) on PresNode01 and PresNode02.

The Shared List server on XdmsNode01 was not working between numbers 6 and 8 in Figure 10-13 on page 375.

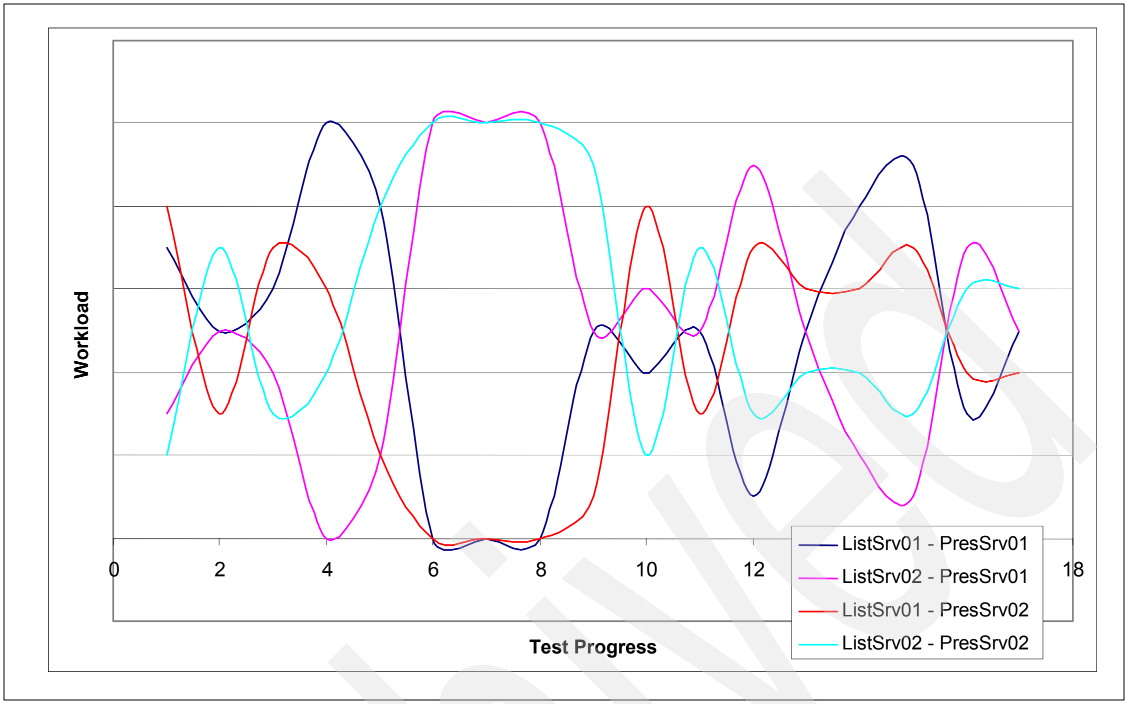


Figure 10-13 TC.02. Shared List behavior during the test

Similarly, for the Presence Rules server, the Figure 10-14 on page 376 shows the behavior of the servers that watch from the Presence Servers point-of-view as clients.

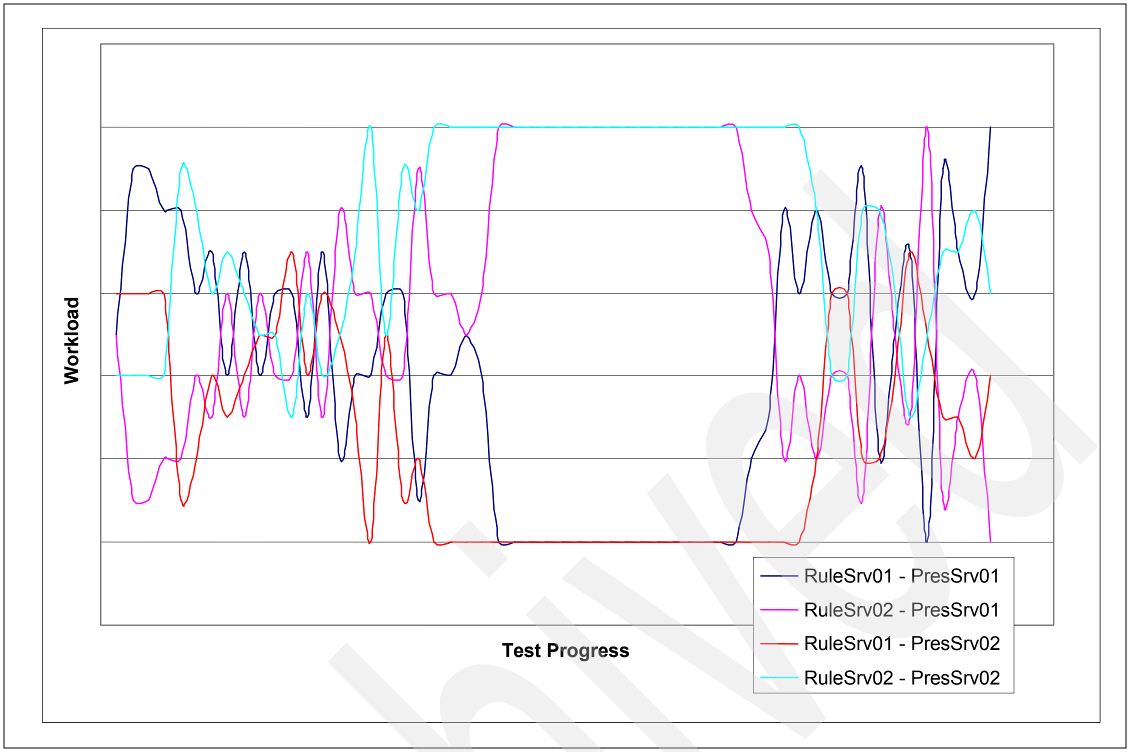


Figure 10-14 TC.02. Presence Rules behavior during the test

In the logs for ListSrv02 during the stopping process of ListSrv01, you can see the messages in Figure 10-15, which means that the replicating of data between servers is done.

```
[10/17/07 16:08:30:699 EDT] 0000001b FailoverMgrIm I FailoverMgrImpl  CWSCT0006I: Failover started. Number of received
objects 342 for logical name ListSrv01.1192651242807.0.
[10/17/07 16:08:30:713 EDT] 00000090 FailoverDataR I FailoverDataReactivator  CWSCT0008I: Failover ended for logical
name ListSrv01.1192651242807.0
```

Figure 10-15 TC.02.Stopping ListSrv01 - replicating on ListSrv02

Also, when we stopped the Presence Rule Server, the fail-over replication process synchronized objects between the Presence Rules Servers, within the PresenceCluster.

Additionally, the PresenceRulesBus worked at that time on server 1. In Figure 10-14, you can see that the Presence Rule Server on XdmsNode02 received a replicating event at the start and after replication. The message engine for the Presence Rules Cluster was started on RuleServ02.

```
[10/17/07 16:18:43:385 EDT] 0000003d SessionContex I   Received event REPLICATION_DOWN.C
[10/17/07 16:18:43:387 EDT] 00000021 RoleMember    I   DCSV8052I: DCS Stack DefaultCoreGroup.PresenceRulesCluster at Member imsCell\XdmsNode02
\RuleSrv02: Core group membership set changed. Removed: [imsCell\XdmsNode01\RuleSrv01].
[10/17/07 16:18:43:413 EDT] 0000003d FailoverMgrIm I FailoverMgrImpl  CWSCT0006I: Failover started. Number of received objects 11808 for logica
name RuleSrv01.1192633031886.0.
[10/17/07 16:18:43:434 EDT] 000000b0 SibMessage    I   [PresenceRulesBus:PresenceRulesCluster.000-PresenceRulesBus] CWSID0016I: Messaging engin
PresenceRulesCluster.000-PresenceRulesBus is in state Starting.
[10/17/07 16:18:43:761 EDT] 000000b0 SibMessage    I   [PresenceRulesBus:PresenceRulesCluster.000-PresenceRulesBus] CWSIU0001I: The runtime
property sib.msgstore.cachedDataBufferSize has been changed to value 640000.
[10/17/07 16:18:43:768 EDT] 000000b0 SibMessage    I   [PresenceRulesBus:PresenceRulesCluster.000-PresenceRulesBus] CWSIU0001I: The runtime
property sib.msgstore.discardableDataBufferSize has been changed to value 640000.
[10/17/07 16:18:43:873 EDT] 000000b0 InternalDB2Un I   DSRA8203I: Database product name : DB2/LINUX
[10/17/07 16:18:43:883 EDT] 000000b0 InternalDB2Un I   DSRA8204I: Database product version : SQL09012
[10/17/07 16:18:43:885 EDT] 000000b0 InternalDB2Un I   DSRA8205I: JDBC driver name  : IBM DB2 JDBC Universal Driver Architecture
[10/17/07 16:18:43:886 EDT] 000000b0 InternalDB2Un I   DSRA8206I: JDBC driver version  : 3.3.54
[10/17/07 16:18:43:887 EDT] 000000b0 InternalDB2Un I   DSRA8212I: DataStoreHelper name is:
com.ibm.websphere.rsadapter.DB2UniversalDataStoreHelper@78607860.
[10/17/07 16:18:43:888 EDT] 000000b0 WSRdbDataSour I   DSRA8208I: JDBC driver type  : 4
[10/17/07 16:18:43:954 EDT] 000000b3 SibMessage    I   [PresenceRulesBus:PresenceRulesCluster.000-PresenceRulesBus] CWSIS1538I: The messaging
engine, ME_UUID=612C993C3598851F, INC_UUID=19E819E8AF9E3C6F, is attempting to obtain an exclusive lock on the data store.
[10/17/07 16:18:43:966 EDT] 000000b5 SibMessage    I   [PresenceRulesBus:PresenceRulesCluster.000-PresenceRulesBus] CWSIS1545I: A single
previous owner was found in the messaging engine's data store, ME_UUID=612C993C3598851F, INC_UUID=7FFE7FFEAE779132
[10/17/07 16:18:43:977 EDT] 000000b3 SibMessage    I   [PresenceRulesBus:PresenceRulesCluster.000-PresenceRulesBus] CWSIS1537I: The messaging
engine, ME_UUID=612C993C3598851F, INC_UUID=19E819E8AF9E3C6F, has acquired an exclusive lock on the data store.
[10/17/07 16:18:44:140 EDT] 000000b8 FailoverDataR I FailoverDataReactivator  CWSCT0008I: Failover ended for logical name
RuleSrv01.1192633031886.0
[10/17/07 16:18:45:061 EDT] 000000b0 SibMessage    I   [PresenceRulesBus:PresenceRulesCluster.000-PresenceRulesBus] CWSIP0212I: Messaging engin
PresenceRulesCluster.000-PresenceRulesBus on bus PresenceRulesBus is starting to reconcile the WCCM destination and link configuration.
[10/17/07 16:18:45:093 EDT] 000000b0 SibMessage    I   [PresenceRulesBus:PresenceRulesCluster.000-PresenceRulesBus] CWSIP0213I: Messaging engin
PresenceRulesCluster.000-PresenceRulesBus on bus PresenceRulesBus has finished reconciling the WCCM destination and link configuration.
[10/17/07 16:18:45:242 EDT] 000000b0 SibMessage    I   [PresenceRulesBus:PresenceRulesCluster.000-PresenceRulesBus] CWSIV0764I: A consumer has
been created for a message-driven bean against destination PresenceRulesCluster on bus PresenceRulesBus following the activation of messaging
engine PresenceRulesCluster.000-PresenceRulesBus.
[10/17/07 16:18:45:244 EDT] 000000b0 SibMessage    I   [PresenceRulesBus:PresenceRulesCluster.000-PresenceRulesBus] CWSID0016I: Messaging engin
PresenceRulesCluster.000-PresenceRulesBus is in state Started.
```

Figure 10-16 TC.02.Stopping RuleSrv01 - replicating on RuleSrv02

During this test, we ran the Presence Server behavior analysis to check if there was any negative influence of XDMS service unavailability to Presence Server. Figure 10-17 displays the Presence Servers workload during the Shared List restarting process (left diagram) and the Presence Rules restarting process (right diagram).

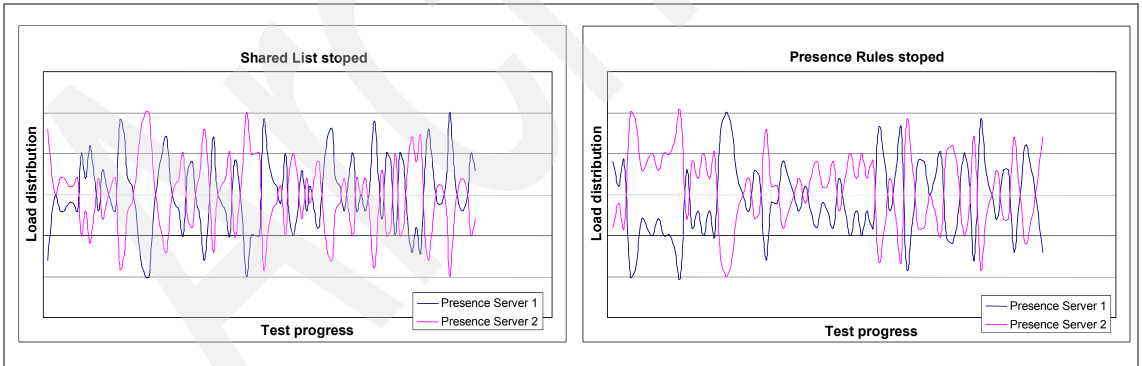


Figure 10-17 Presence Servers workload during Shared List restarting

Summarizing, the requests during this test were evenly processed by both Presence Servers in the Presence Cluster without any significant deviation in the Presence Server behavior.

From the business scenario point-of-view, all tests finished successfully during these test cases, which means that all changes in the presence status that were published to Presence Server were delivered to Subscribed as notification.

Unexpected stop of the XDMS Server

We stopped the XDMS Server, which included the Shared List and Presence Rules in unexpected manner, by killing the java process of the mentioned servers. The sequence of steps during this test are similar to the sequence of steps in “Planned stop and start of the XDMS Servers” on page 373. We used the following steps:

1. Login to the tester machine.
2. Start sipp test.
3. On a separated console, login to XdmsNode01.
4. Go to the directory of the server logs for the Shared List server, as shown in Example 10-9.

Example 10-9 Logs directory of Shared List server

```
[root@xdms1 ListSrv01]# pwd
/opt/IBM/WebSphere/AppServer/profiles/XdmsNode01/logs/ListSrv01
```

5. Check the Process ID number for the Shared List server, as shown in Example 10-10.

Example 10-10 PID number for Shared List

```
[root@xdms1 ListSrv01]# cat ListSrv01.pid; echo;
504
[root@xdms1 ListSrv01]#
```

6. Run the Kill server process, as shown in Example 10-11.

Example 10-11 Kill Shared List server process

```
[root@xdms1 ListSrv01]# kill 504
```

7. Immediately open the SystemOut.log file to see if the process is starting automatically by nodeagent, as shown in Example 10-12.

Example 10-12 Check SystemOut.log file

```
[root@xdms1 ListSrv01]# tail -f SystemOut.log
************ Start Display Current Environment ************
WebSphere Platform 6.1 [ND 6.1.0.11 cf110734.37] [XDMS 6.2.0.0
20071012_1648]  running with process name imsCell\XdmsNode01\ListSrv01
and process id 19566
```

```
Host Operating System is Linux, version 2.6.9-55.ELsmp
Java version = J2RE 1.5.0 IBM J9 2.3 Linux x86-32 j9vmxi3223-20060504
(JIT enabled)
J9VM - 20060501_06428_lHdSMR
JIT  - 20060428_1800_r8M
GC   - 20060501_AA, Java Compiler = j9jit23, Java VM name = IBM J9 VM
was.install.root = /opt/IBM/WebSphere/AppServer
...
```

8. After the server ListSrv01 is started, requests are again processed by both SharedList servers, which you can see in Figure 10-18 on page 380.

Figure 10-18 on page 380 shows a workload distribution on Shared List during the test.

On the legend, we used the following pattern:

[Server which processed request] - [Client which generate request]

As an example, the blue line on the diagram represents requests that ListSrv01 processes and that PresSrv01 generates. The server PresSrv01 was working as a client of List Server functionality in this case.

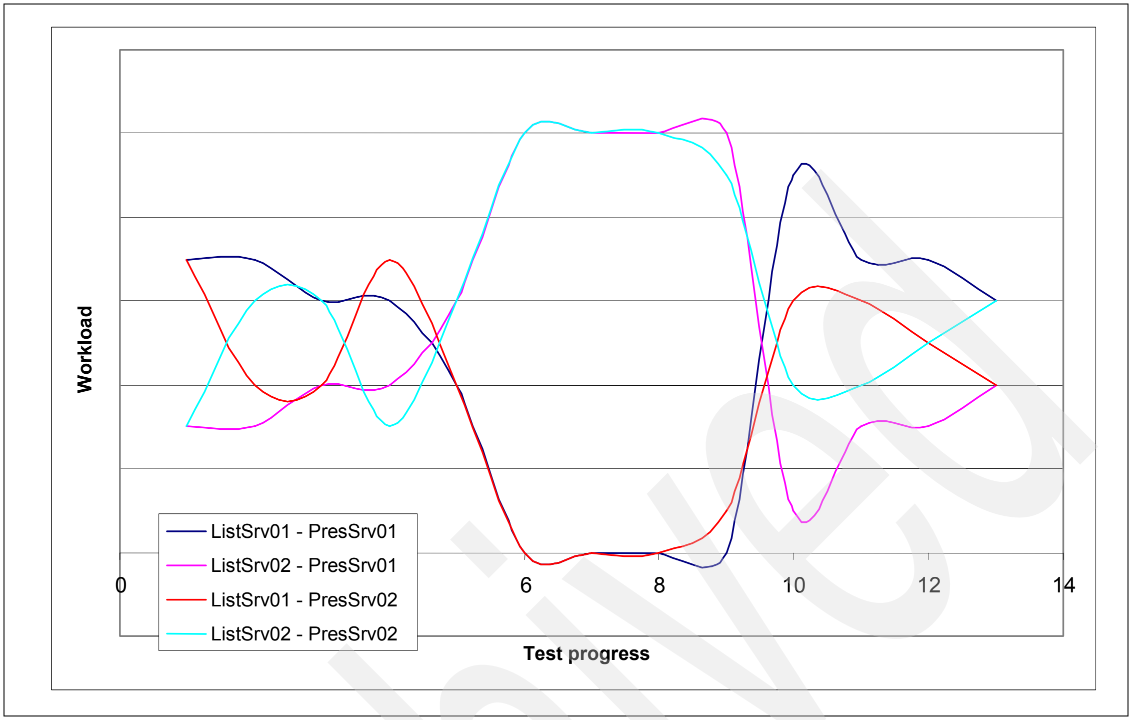


Figure 10-18 TC.02.3 Shared List server behavior during the test

During this test, we stopped the ListSrv01. On the graph, you can see that between 6 and 9 ListSrv02 processes all traffic. Based on the procedure described for Shared List, we performed the same steps for the Presence Rules server.

Figure 10-19 on page 381 illustrates the workload distribution between RuleSrv01 and RuleSrv02 during the test.

On Figure 10-19 on page 381, you can see two lines per server. On the legend, we used the following pattern:

[Server which processed request] - [Client which sent request]

As an example, the blue line on the diagram represents requests that RuleSrv01 processes and that PresSrv01 generates. The server PresSrv01 was working as a client of List Server functionality in this case.

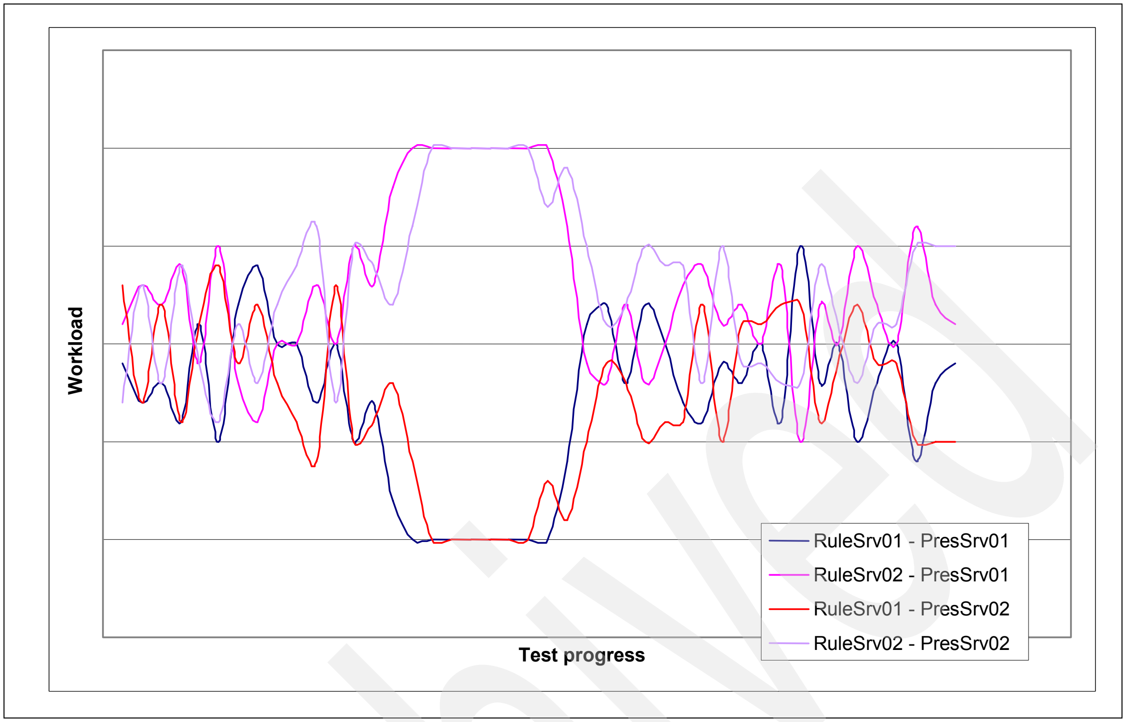


Figure 10-19 TC.02.4 Presence Rules behavior during the test

As we can see, the workload on the servers during normal work is quite evenly spread. But when we stopped RuleSrv01, RuleSrv02 processed all requests from both clients.

10.3.4 TS.03.Redundant nodes unavailable

This test was based on the test case that we describe in 10.3.4, “TS.03.Redundant nodes unavailable” on page 381. During this test, we killed redundant servers.

Figure 10-20 on page 382 shows how requests are processed by the Presence Servers during this test.

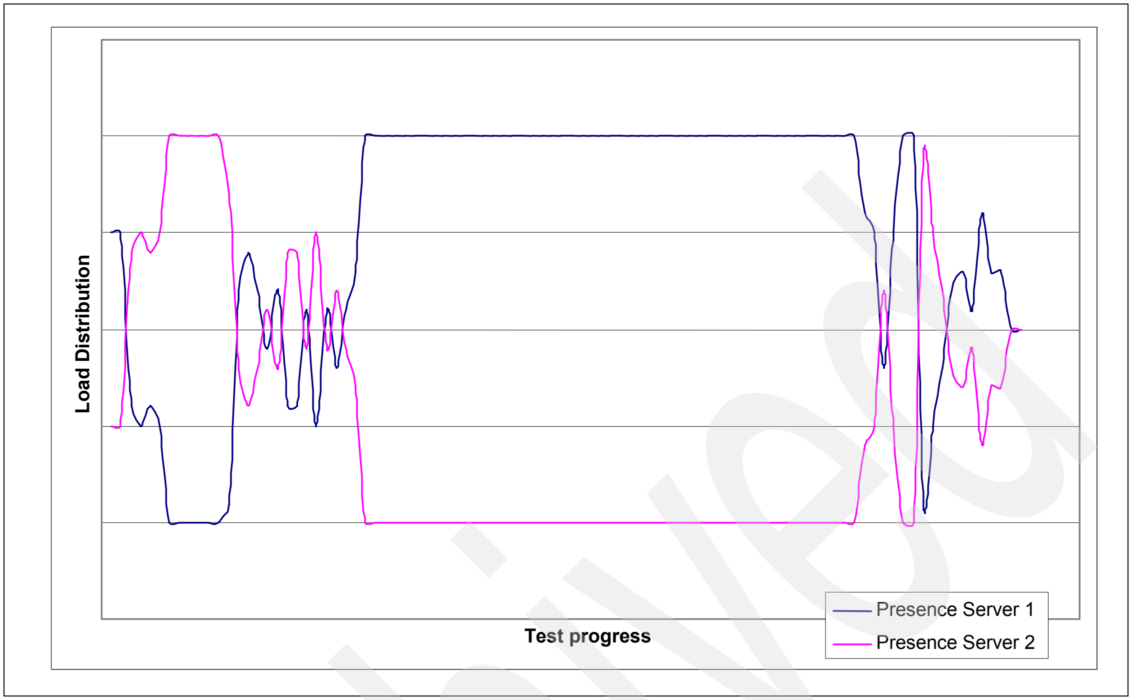


Figure 10-20

After we killed the PresSrv02 java process, Presence Server 1 processed all request. After the Node Agent started the Application Server on PresNode02, the requests were distributed to both nodes.

In this test, our business scenario failed after receiving only four of five expected notifications for the published presence information. This is because the PUBLISH message that was sent to the Presence Server 2 was not fully processed prior to the transaction server being killed. Thus, the response 200 was not sent back to our sipp client for the PUBLISH request. Also, this call was not distributed to Presence Server 1 within the Presence Cluster because of the same broke transaction.

Part 4

The scalable scenario

In this part, we describe how to configure additional servers to make the highly-available environment scalable. We demonstrate the distribution of messages to additional servers that we introduce to the environment.

11

The scalable environment setup and configuration

In this chapter, we provide an overview and step-by-step procedures for setting up and configuring a scalable environment.

11.1 Scalability overview

In the previous chapters, we discussed the scenario of setting up and testing a high-availability environment. We now discuss the scenario of setting up and testing the Scalable Presence Server environment. Our goal is to enhance the system to be more scalable in terms of increased throughput, which brings to focus the clustering features of WebSphere Application Server. In the WebSphere Application Server environment, scalability features are realized through these techniques:

- Vertical clustering
- Horizontal Clustering
- Combination of Horizontal and Vertical Clustering

For more information about scalability features of the WebSphere Application Server, refer to the Redpaper, *WebSphere Application Server V6.1: Technical Overview*, REDP-4191. In the following sections, we discuss, in detail, the deployed environment details that are relevant to the architecture of Presence Server and XDMS Server respectively. Adding on to the existing high-availability environment, we adopted a combination of horizontal and vertical clustering for the scalability scenario. Figure 11-1 on page 387 shows the schematic topology for the scalability scenario.

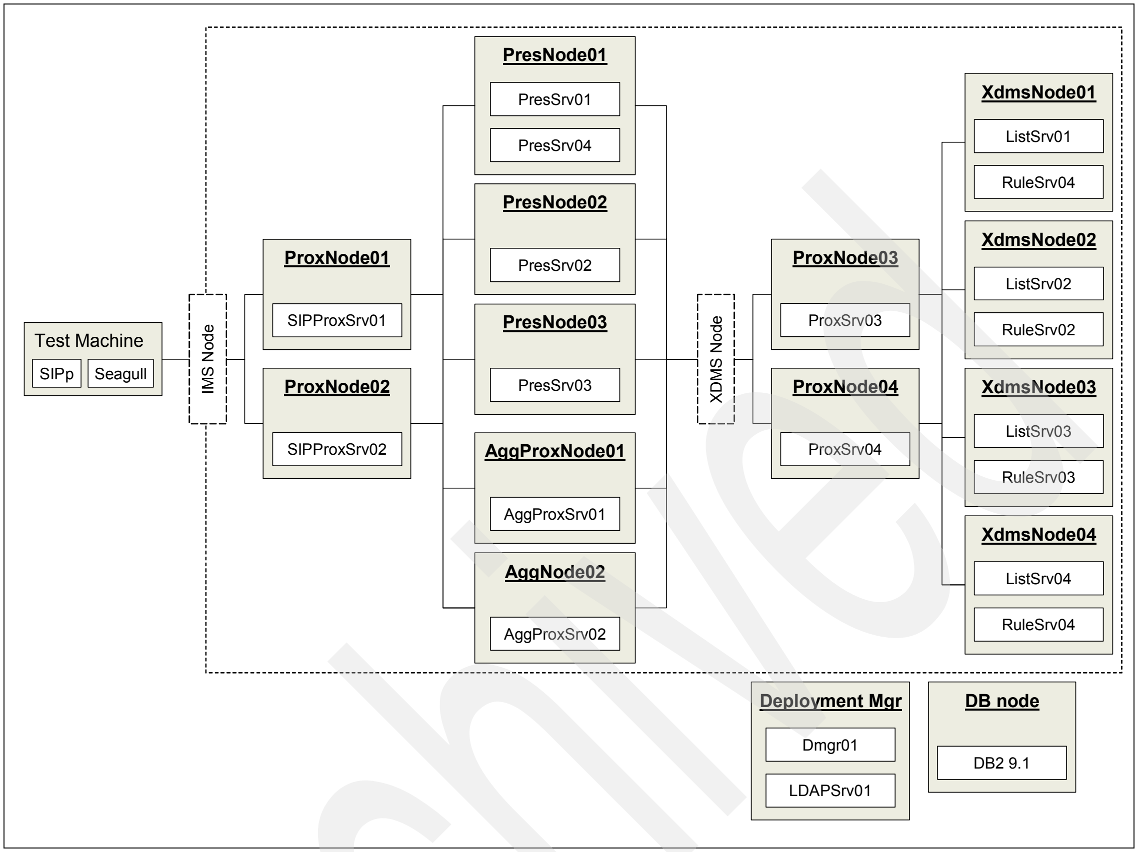


Figure 11-1 Scalability environment topology

For the scope of this book, the configuration settings that are relevant to the scalability scenario are based on the Presence Server architecture, which we discussed in 1.3.4, "The architecture" on page 18. Refer to The Layers of the Presence Server in Figure 1-7 on page 19, and notice that the following components significantly affect Presence Server processing capabilities:

- SIP Container: SIP Container component of WebSphere Application Server services SIP requests and responses. In the Presence Server architecture, the SIP container handles SIP SUBSCRIBE, SIP PUBLISH, and SIP NOTIFY requests. Apart from being a key contributor in achieving high-availability features of the SIP infrastructure, the SIP container provides several configurable parameters that allow for performance and scalability tuning.

For more information about SIP container settings, refer to the SIP container settings in the Infocenter:

http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.express.doc/info/exp/ae/usip_rcont.html

- JDBC Data Source: The Data Layer of the Presence Server architecture encapsulates several JDBC calls to the underlying data store. The operations that go behind servicing SIP SUBSCRIBE, SIP PUBLISH, and Partial Document updates require significant planning in terms of JDBC Connection pool settings. To avoid a bottleneck in the data access layer, consider tuning JDBC resources to achieve overall scalability. For more information about tuning and overload settings, refer to Appendix A, “Performance, monitoring, and tuning” on page 451.
- SI Bus: The business logic layer of the Presence Server architecture contains logic to handle presence documents as part of handling SUBSCRIPTION, PUBLICATION, and NOTIFICATION functions. The said functions leverage standards that are based on JMS functionality and the underlying SI Bus transport facilities. So, it becomes imperative to tune SI Bus and JMS resources to achieve the desired throughput. For more information about tuning and overload settings, refer to Appendix A, “Performance, monitoring, and tuning” on page 451.

11.2 Setting up the scalable environment

The scenario to demonstrate the Scalable Presence Server environment requires additional installation and configuration of various components. In the following sections, we describe each of the installation and configuration activities in detail.

11.2.1 Horizontal clustering configuration

The scalability scenario is realized by enhancing the existing Presence Server environment by provisioning additional server instances following the horizontal clustering technique. The list of individual server instances that needs to be added to the Presence Server environment are:

- Presence Server - PresSrv03
- Shared List server - ListSrv03
- Shared List server - ListSrv04
- Presence Rules Server - RuleSrv03
- Presence Rules Server - RuleSrv04

In the following sections, we discuss the installation and configuration settings for each of the newly added infrastructure components.

Creating and federating the new Presence Server node

Creating a managed node is a two-step process. The first step is to install a WebSphere Application Server image on the commissioned machine, by name *pres3.itso.ral.ibm.com*. Creating a node is essentially the creation of a profile. The newly created profile is then federated to the *imsCell*.

To create and federate the new Presence Server node:

1. Login to the machine pres3.itso.ral.ibm.com, and run the command in Example 11-1.

Example 11-1 Create WebSphere Application Server profile for PresNode03

```
/opt/IBM/WebSphere/AppServer/bin/manageprofiles.sh -create -profileName
PresNode03 -profilePath
/opt/IBM/WebSphere/AppServer/profiles/PresNode03 -templatePath
/opt/IBM/WebSphere/AppServer/profileTemplates/managed -nodeName
PresNode03 -hostName pres3.itso.ral.ibm.com
```

2. Federate the created node to the imsCell using the command in Example 11-2.

Example 11-2 Federate PresNode03 to the imsCell

```
/opt/IBM/WebSphere/AppServer/profiles/PresNode03/bin/addNode.sh
ldap.itso.ral.ibm.com -username wasadmin -password <password>
-profileName PresNode03
```

3. Synchronize the created profile using the **syncNode** command, as shown in Example 11-3.

Example 11-3 Synchronize PresNode03

```
/opt/IBM/WebSphere/AppServer/profiles/PresNode03/bin/syncNode.sh
ldap.itso.ral.ibm.com -username wasadmin -password <password>
```

4. Start the recently created node *PresNode03*, as shown in Example 11-4.

Example 11-4 Start Presence Node PresNode03

```
/opt/IBM/WebSphere/AppServer/profiles/PresNode03/bin/startNode.sh
```

Creating an Application Server, and adding it to the Presence Server Cluster

After we create and federate a new node on the pres3.itso.ral.ibm.com machine, we create a new server instance from the existing default server template. While creating the *PresSrv01* cluster member, the same is marked as a *default server* template. Login to the Deployment Manager Integration Server Console/administration console, and follow these steps:

1. On the left panel, click the **Servers** hyperlink.
2. Click the **Application Servers** hyperlink.
3. Under Server clusters, select **PresenceCluster** from the cluster list in Figure 11-2.

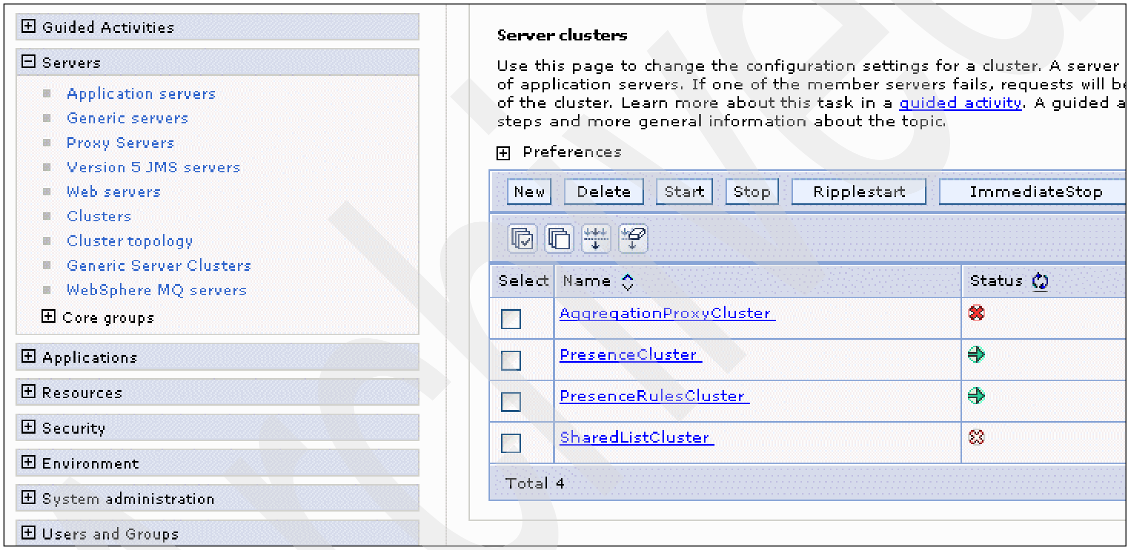


Figure 11-2 Select PresenceCluster

4. In the PresenceCluster detail page, shown in Figure 11-3 on page 391, under Additional Properties, click **Cluster members**.

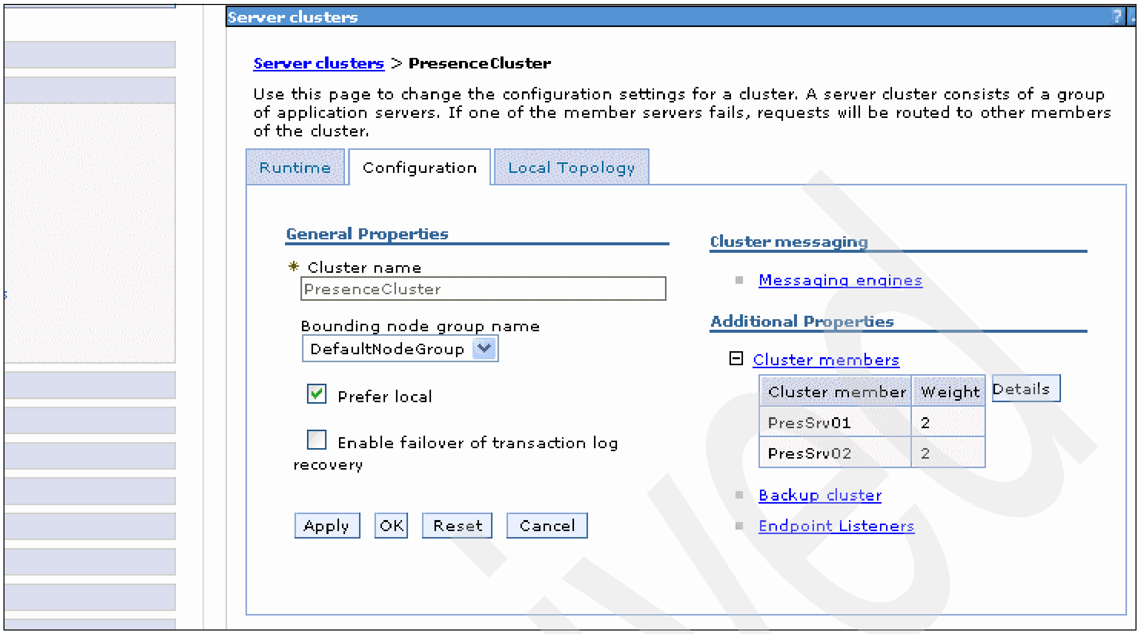


Figure 11-3 Select the Cluster members

5. In the Cluster members list page (Figure 11-4), click the **New** button.

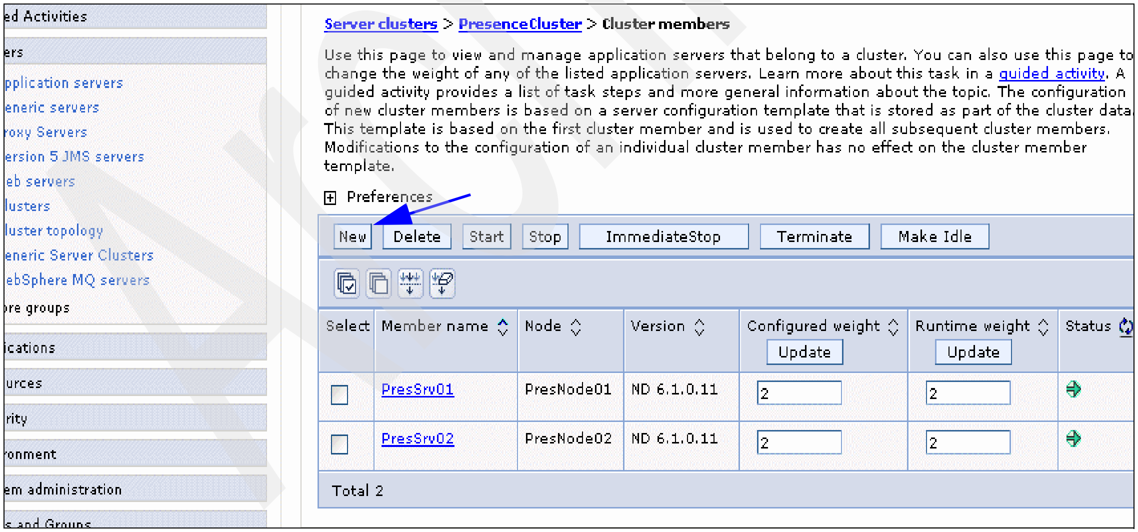


Figure 11-4 Create new cluster

6. On the Create additional cluster members page, Figure 11-5, perform the following tasks:

   a. In the Member name field, type the new Application Server name `PresSrv03`.

   b. In the Select node field, select **PresNode03 (ND 6.1.0.11)**.

   c. Click **Add Member**.

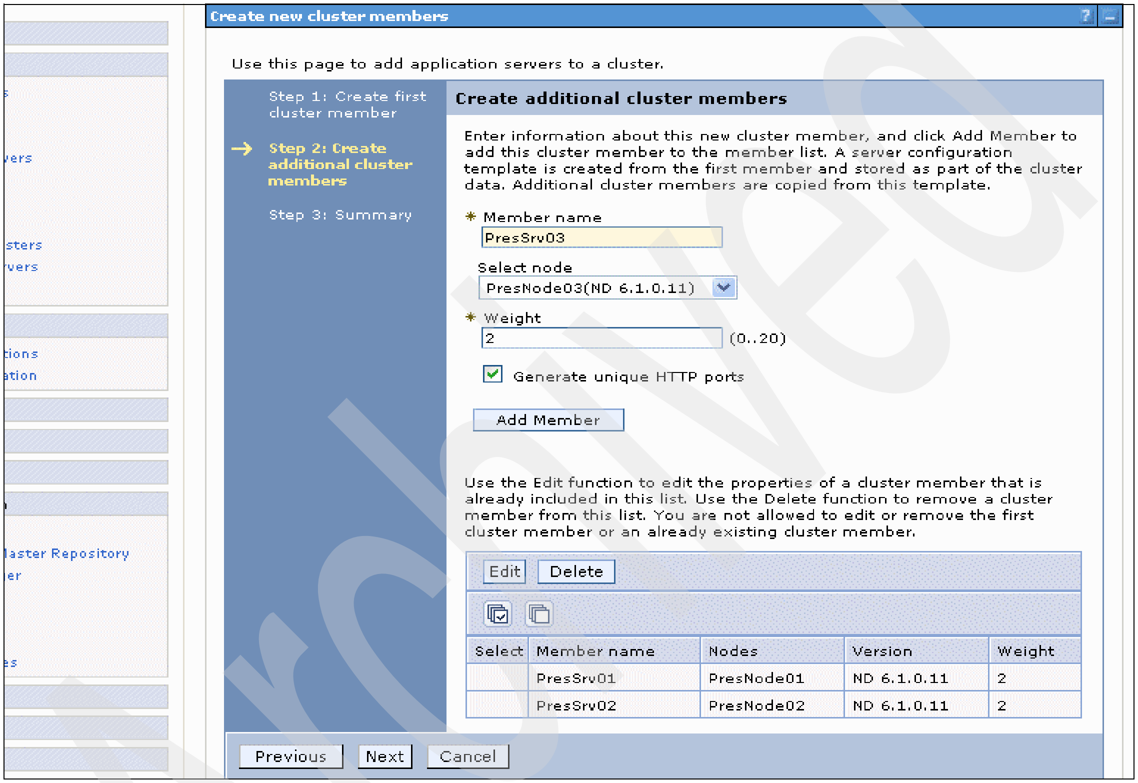


Figure 11-5 Create PresSrv03

7. Make sure that new member is added to the cluster member list, in Figure 11-6 on page 393, and click the **Add Member** button. Click the **Next** button.

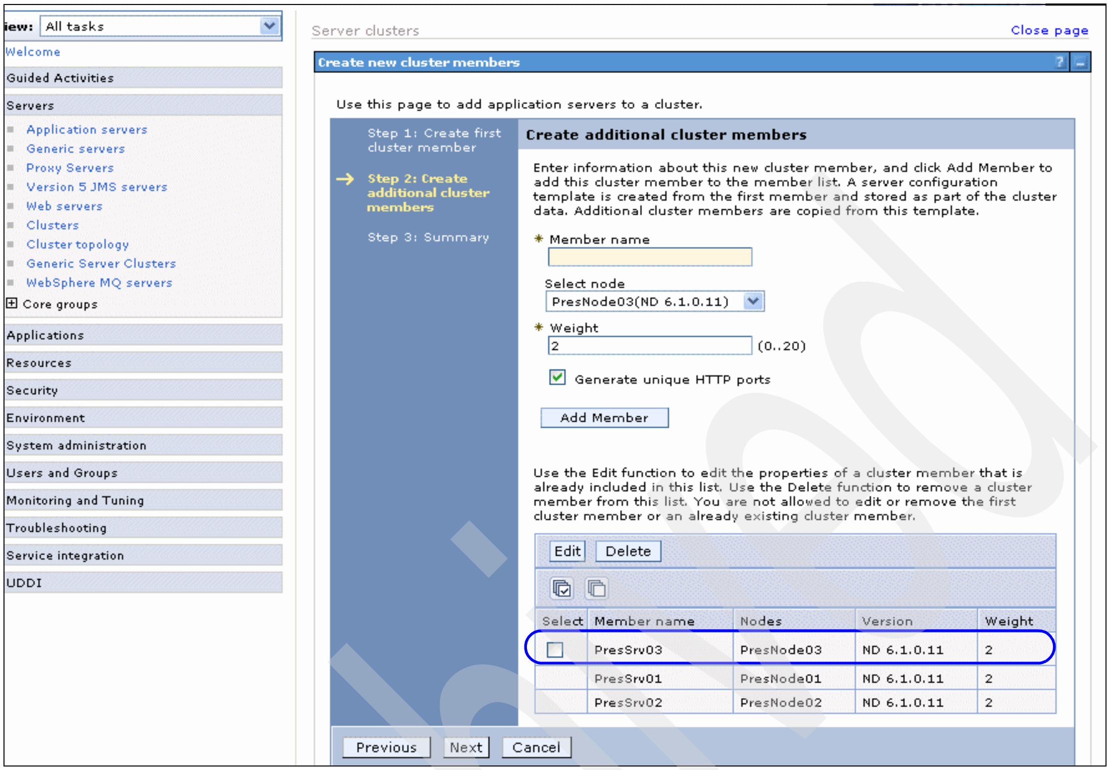


Figure 11-6 Check the existence of the newly created cluster member

8. Review the summary page, Figure 11-7 on page 394, and click **Finish**.

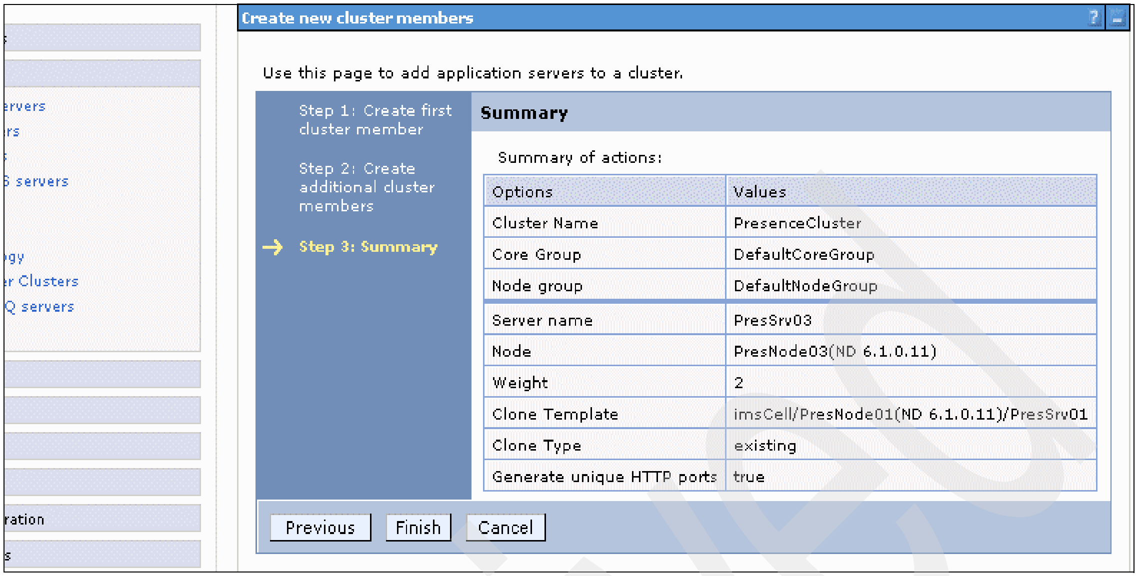


Figure 11-7 Add new cluster Summary page

9. Make sure there is a new Application Server in the cluster members list of PresenceCluster, as shown in Figure 11-8 on page 395. Click **Save** to save directly to the master configuration.

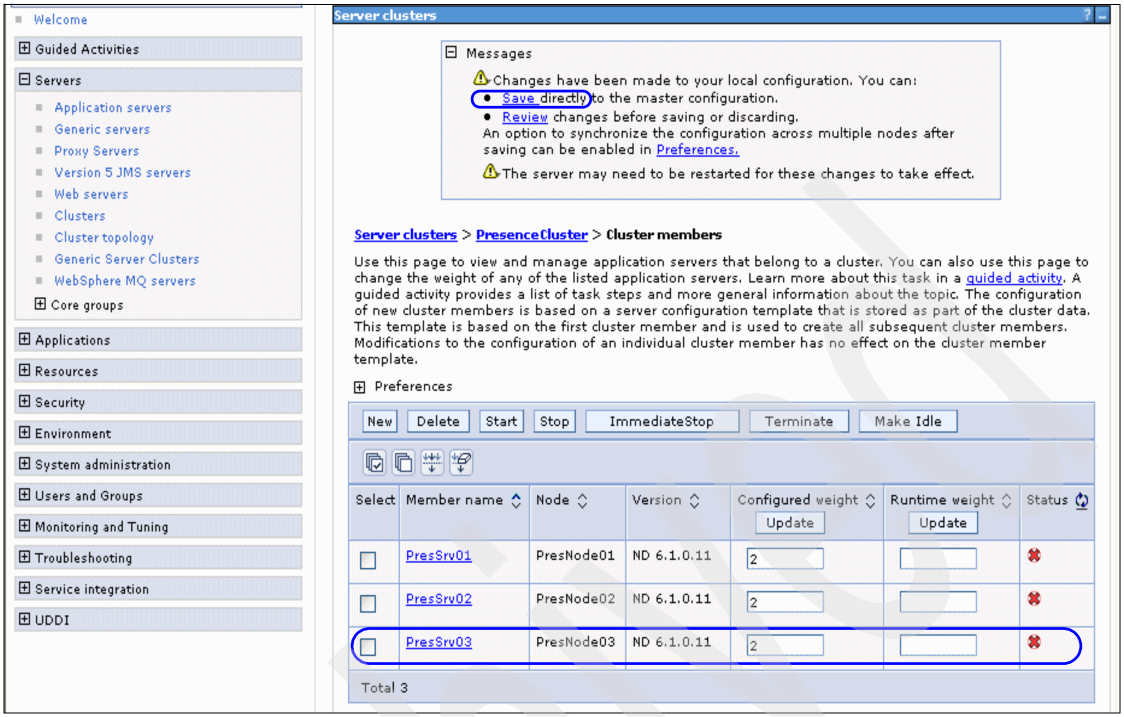


Figure 11-8   Save the new cluster

Adding the new Presence Server as a bus member to SI Bus

To add the new Application Server to Service Integration Bus (SIBus) for the Presence Servers as a bus member:

1. From the left panel, Figure 11-8, select **Service Integration**.
2. From the Buses list, select **PresenceBUS**, as shown in Figure 11-9 on page 396.

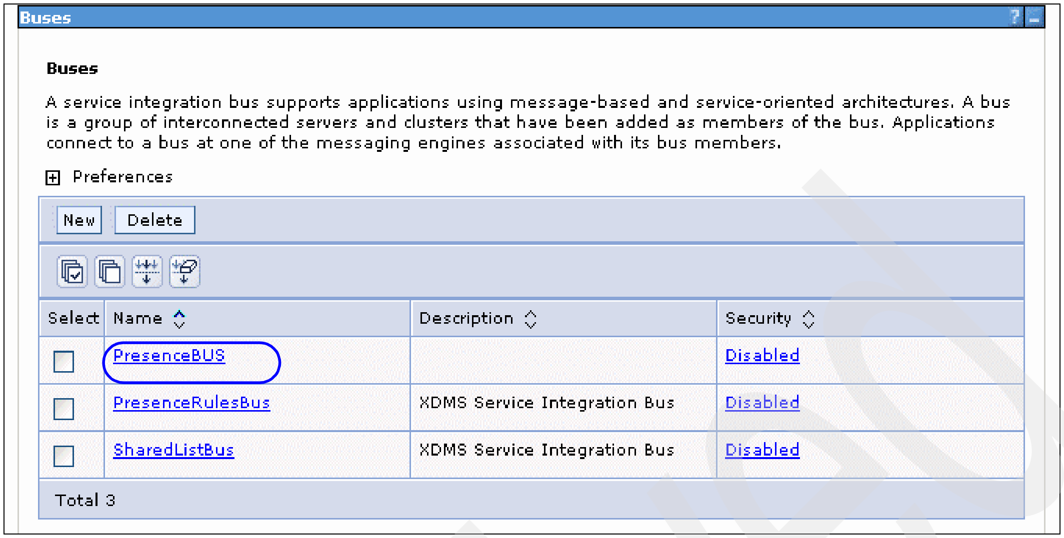


Figure 11-9 Select PresenceBUS

3. Detailed information for the bus is shown in Figure 11-10 on page 397. At the top, right side of the page, under the Topology section, click the **Bus members** link.

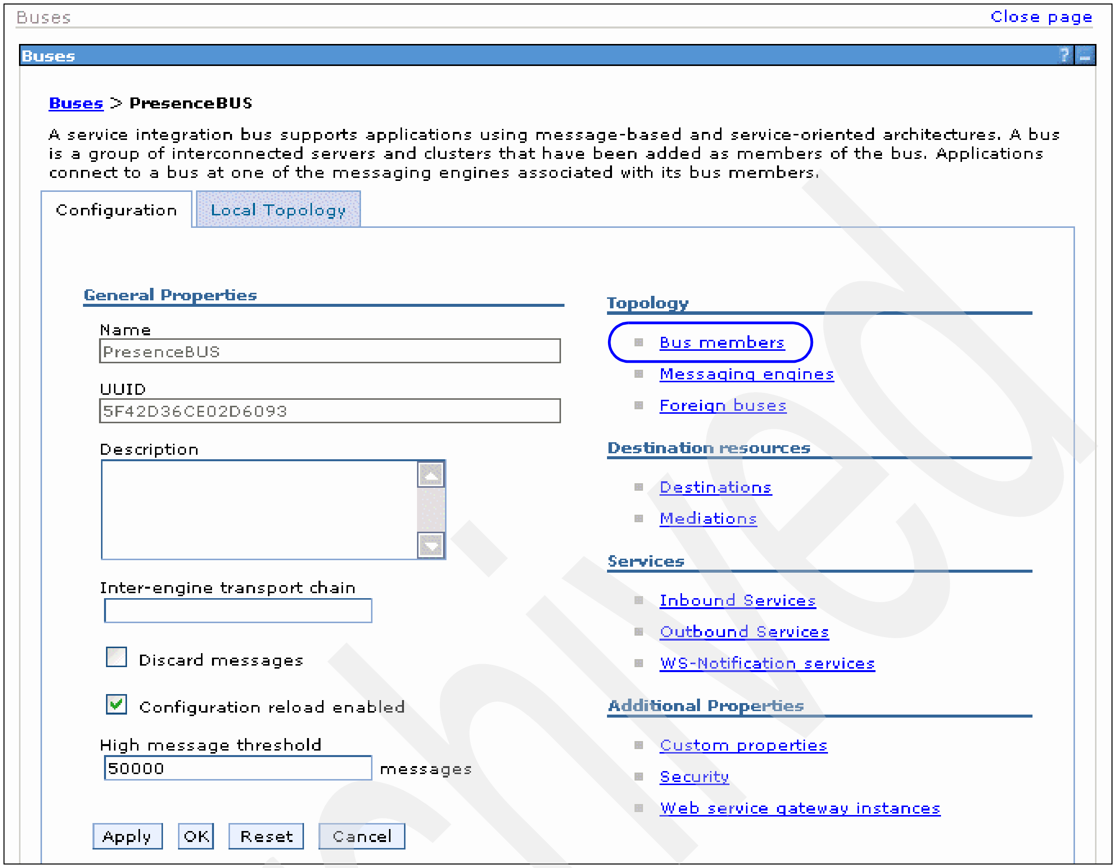


Figure 11-10 BUS detail information

4. Under the Bus member list, Figure 11-11 on page 398, click the **Add** button.

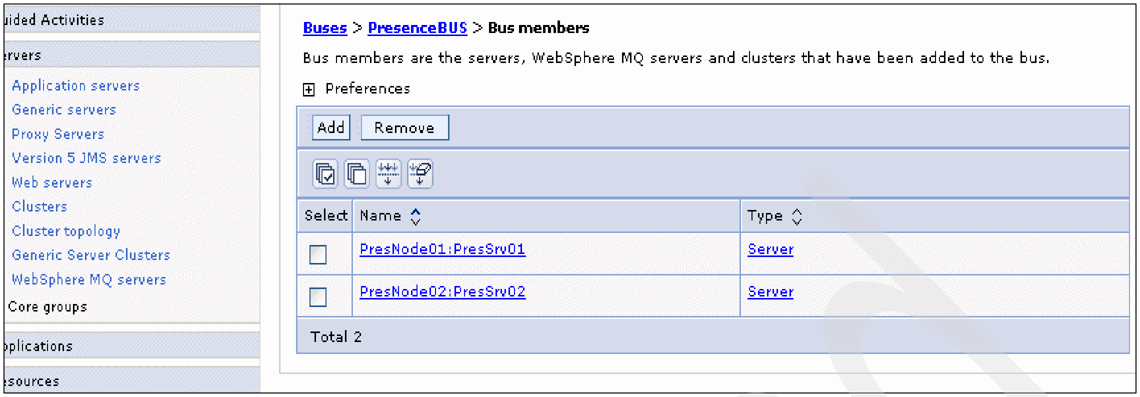


Figure 11-11 Add bus

5. Choose the server that you want to add to SIBus as a member, for example, it is PresNode03:PresServ03. See Figure 11-12.

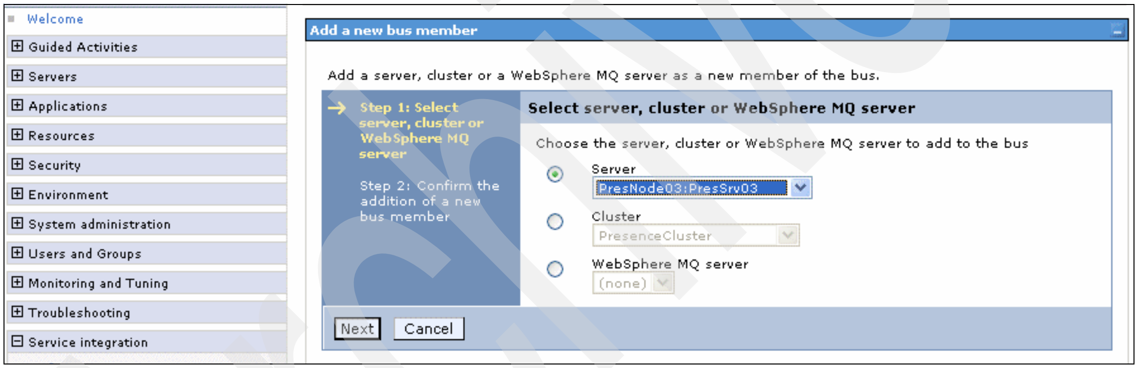


Figure 11-12 Select server to add to SIBus as a member

6. Select the **File Store** as the type of message store, as shown in see Figure 11-13 on page 399, and click the **Next** button.

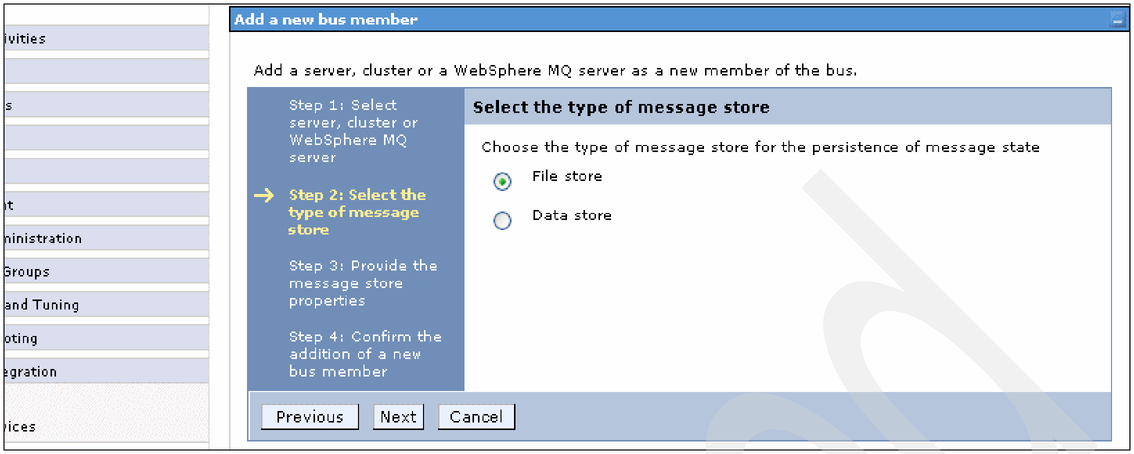


Figure 11-13 Choose the File store

7. In Figure 11-14, under Provide the message store properties, accept all of the defaults, and click the **Next** button.

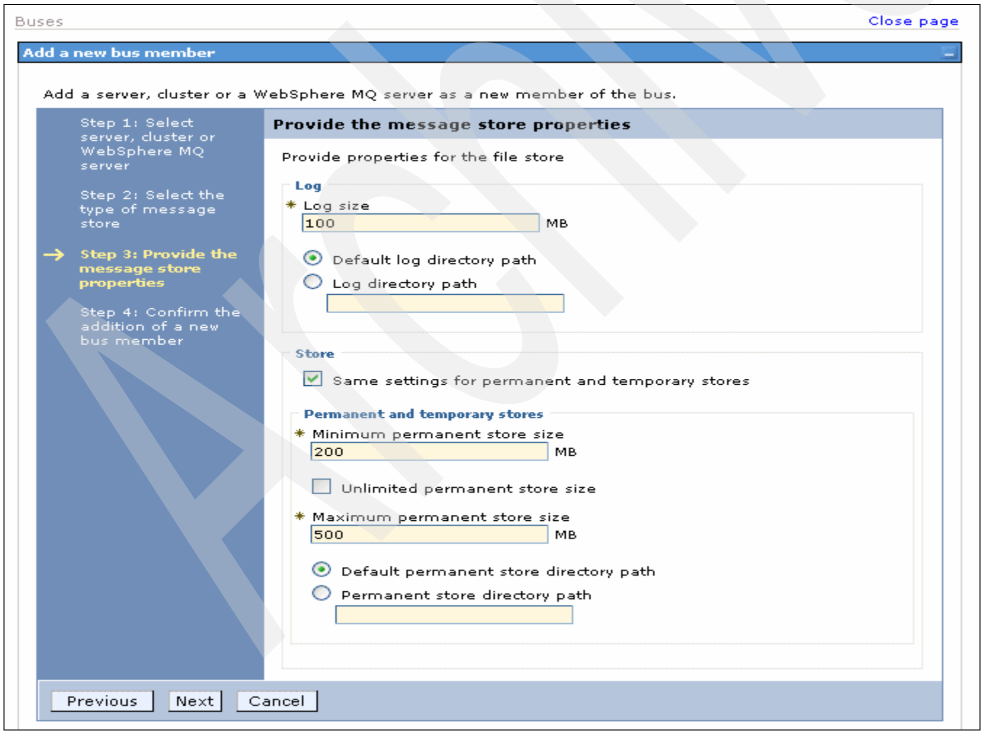


Figure 11-14 Accept the message store properties defaults

8. Confirm the addition of a new bus member, as shown in Figure 11-15, and click **Finish**.

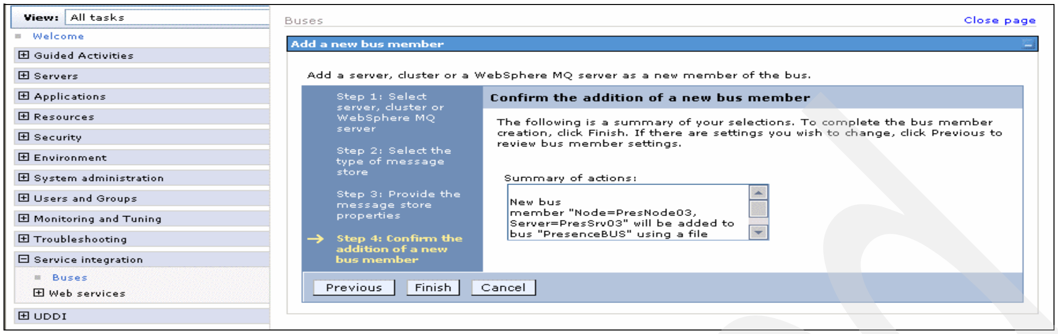


Figure 11-15 Add a new bus member window

9. Confirm that the new bus member is added to the bus member list. Then, click **Save** to save directly to the master configuration, as shown in Figure 11-16.

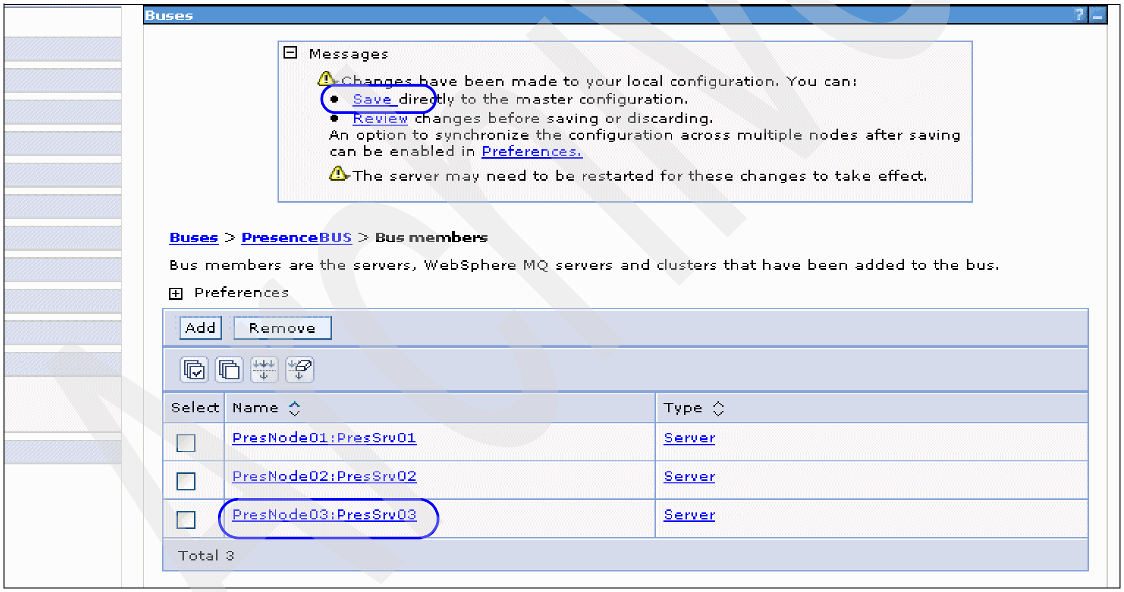


Figure 11-16 Save directly to the master configuration

XDMS Server Clusters

Two new nodes and four new Application Servers are created on the XDMS Server Clusters. Each node has two servers: one node is added to the Shared List Cluster and the other to the Presence Rules Cluster.

Before you start the configuration, make sure that the WebSphere Application Server and DB2Client are copied and properly installed on the new host machine. In this case, it is named pres3.itso.ibm.com. Also, copy the DHAIMSConnectorTAI.jar file onto it.

The configuration process is:

1. Create and Federate nodes for the Shared List and Presence Rules.
2. Create two Application Servers for each node (adding one server to the resource list cluster and the other server to the Presence Server Cluster).
3. Set up the Virtual Host.

Creating a Node and federating it to Deployment Manager

For new Application Servers, new nodes are created at two new hosts, which are named `xdms3.itso.ibm.com` and `xdms4.itso.ibm.com` in the environment. These new nodes are federated into the Deployment Manager. Refer to the setup of XdmsNode02 that we performed in Chapter 9, "The high-availability environment setup and configuration" on page 307. XdmsNode03 and XdmsNode04 will follow the same procedure.

Creating and Federating nodes for Shared List and Presence Rules

This activity involves creating two nodes in two newly commissioned machines with the names `xdms3.itso.ral.ibm.com` and `xdms4.itso.ral.ibm.com`.

Creating a managed node is a two-step process. The first step is to install the WebSphere Application Server image on the commissioned machine with the name xdms3.itso.ral.ibm.com. Create a managed node profile. The newly created profile is then federated to the imsCell.

To create and federate nodes on xdms3.itso.ral.ibm.com:

1. Login to the machine xdms3.itso.ral.ibm.com, and run command in Example 11-5.

Example 11-5 Create WebSphere Application Server profile for XdmsNode03

```
/opt/IBM/WebSphere/AppServer/bin/manageprofiles.sh -create -profileName
XdmsNode03 -profilePath
/opt/IBM/WebSphere/AppServer/profiles/XdmssNode03 -templatePath
/opt/IBM/WebSphere/AppServer/profileTemplates/managed -nodeName
XdmsNode03 -hostName xdms3.itso.ral.ibm.com
```

2. Federate the created node to the imsCell using the command in Example 11-6.

Example 11-6 Federate XdmsNode03 to the imsCell

```
/opt/IBM/WebSphere/AppServer/profiles/XdmsNode03/bin/addNode.sh
ldap.itso.ral.ibm.com -username wasadmin -password <password>
-profileName XdmsNode03
```

3. Synchronize the created profile using the **syncNode** command, as shown in Example 11-7.

Example 11-7 Synchronize XdmsNode03

```
/opt/IBM/WebSphere/AppServer/profiles/XdmsNode03/bin/syncNode.sh
ldap.itso.ral.ibm.com -username wasadmin -password <password>
```

4. Start the recently created node XdmsNode03, using the command in Example 11-8.

Example 11-8 Start Presence Node XdmsNode03

```
/opt/IBM/WebSphere/AppServer/profiles/XdmsNode03/bin/startNode.sh
```

Repeat steps 1 through 4 to create and federate a node on the machine xdms4.itso.ral.ibm.com. Create a managed node profile. The newly created profile is then federated to the imsCell.

To create and federate nodes on xdms4.itso.ral.ibm.com:

1. Login to the machine xdms4.itso.ral.ibm.com and run the command in Example 11-9.

Example 11-9 Create WebSphere Application Server profile for XdmsNode04

```
/opt/IBM/WebSphere/AppServer/bin/manageprofiles.sh -create -profileName
XdmsNode04 -profilePath
/opt/IBM/WebSphere/AppServer/profiles/XdmssNode04 -templatePath
/opt/IBM/WebSphere/AppServer/profileTemplates/managed -nodeName
XdmsNode04 -hostName xdms4.itso.ral.ibm.com
```

2. Federate the created node to the imsCell using the command in Example 11-10 on page 403.

Example 11-10 Federate XdmsNode04 to the imsCell

```
/opt/IBM/WebSphere/AppServer/profiles/XdmsNode04/bin/addNode.sh
ldap.itso.ral.ibm.com -username wasadmin -password <password>
-profileName XdmsNode04
```

3. Synchronize the created profile using the **syncNode** command, as shown in Example 11-11.

Example 11-11 Synchronize XdmsNode04

```
/opt/IBM/WebSphere/AppServer/profiles/XdmsNode04/bin/syncNode.sh
ldap.itso.ral.ibm.com -username wasadmin -password <password>
```

4. Start the recently created node XdmsNode04, using the command in Example 11-12.

Example 11-12 Start Presence Node XdmsNode04

```
/opt/IBM/WebSphere/AppServer/profiles/XdmsNode04/bin/startNode.sh
```

Important: In the nodes xdms3.itso.ral.ibm.com and xdms4.itso.ral.ibm.com, we created two Application Servers per node. One Application Server instance is for Shared List and the other is for Presence Rules.

Creating an Application Server and adding it to the Shared List Cluster

After we create and federate a new node on the xdms3.itso.ral.ibm.com machine, we create a new server instance from the existing default server template. Login to the Deployment Manager Integration Server Console/administration console and follow these steps:

For the Shared List Cluster:

1. From the left panel, select **Servers** → **Clusters**. In the Server clusters window (Figure 11-17 on page 404), from the cluster list, select **SharedListCluster**.

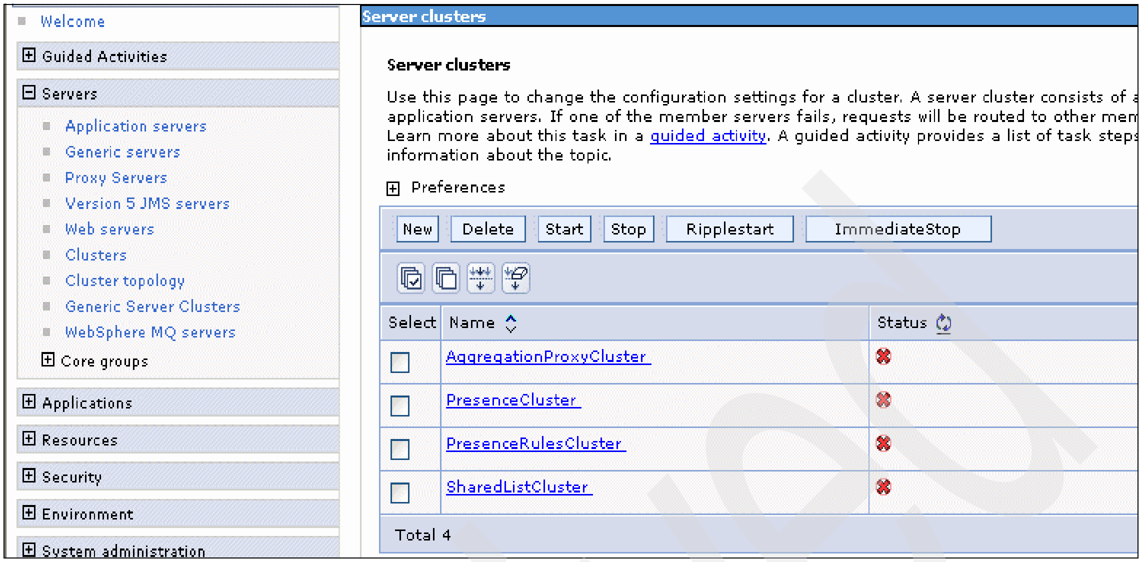


Figure 11-17 Select the SharedListCluster

2. Under the Configuration tab, shown in Figure 11-18, at the right side of the page, click the **Cluster members** link.

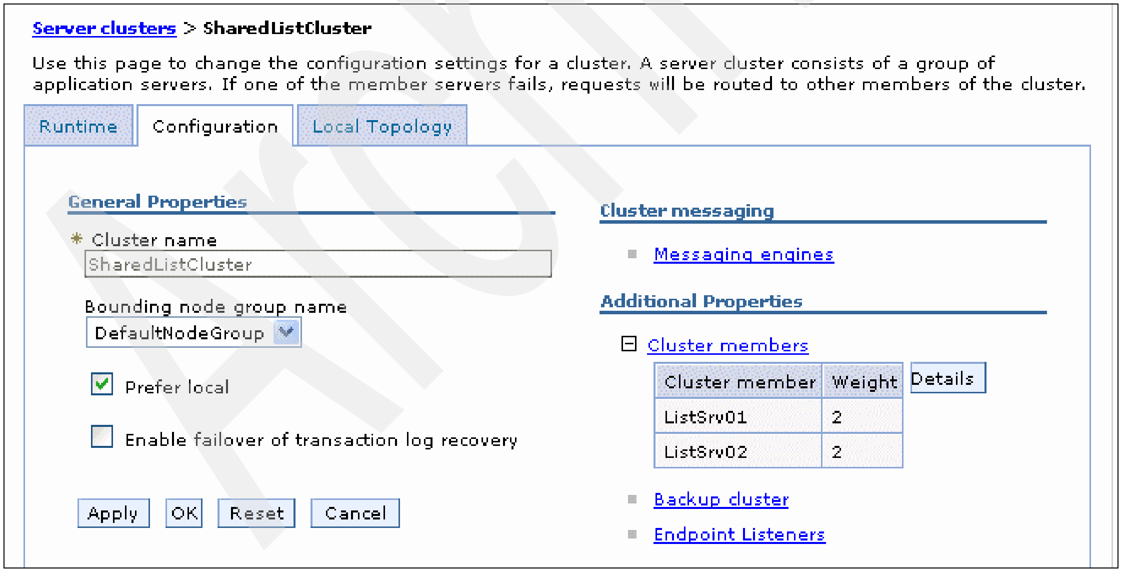


Figure 11-18 Select Cluster members

3. In the Cluster members list page, Figure 11-19, click the **New** button.

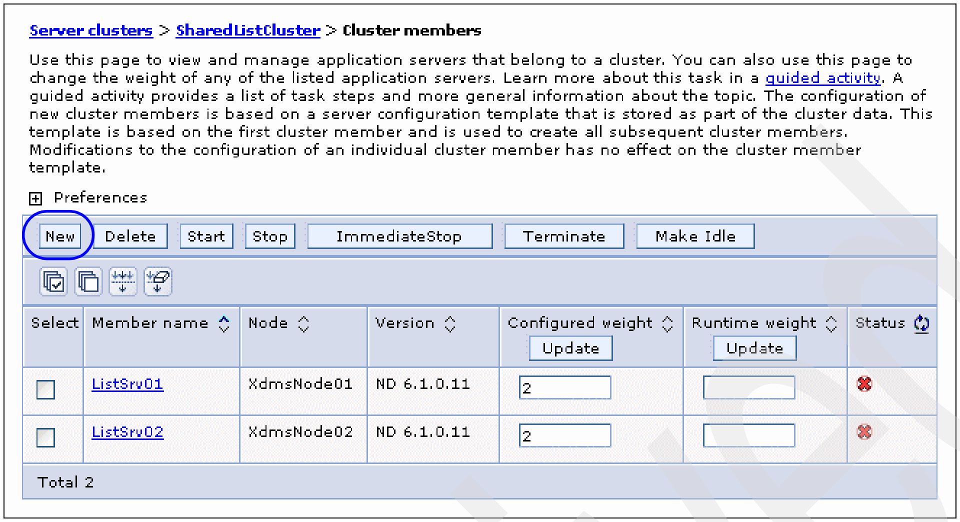


Figure 11-19 Click the New button

4. In the Member name field, type the new Application Server name (for our example, ListSrv03) as shown Figure 11-20 on page 406. In the Select node field, select the new Node name (for our example, XdmsNode03(ND 6.1.0.11). Click the **Add Member** button.

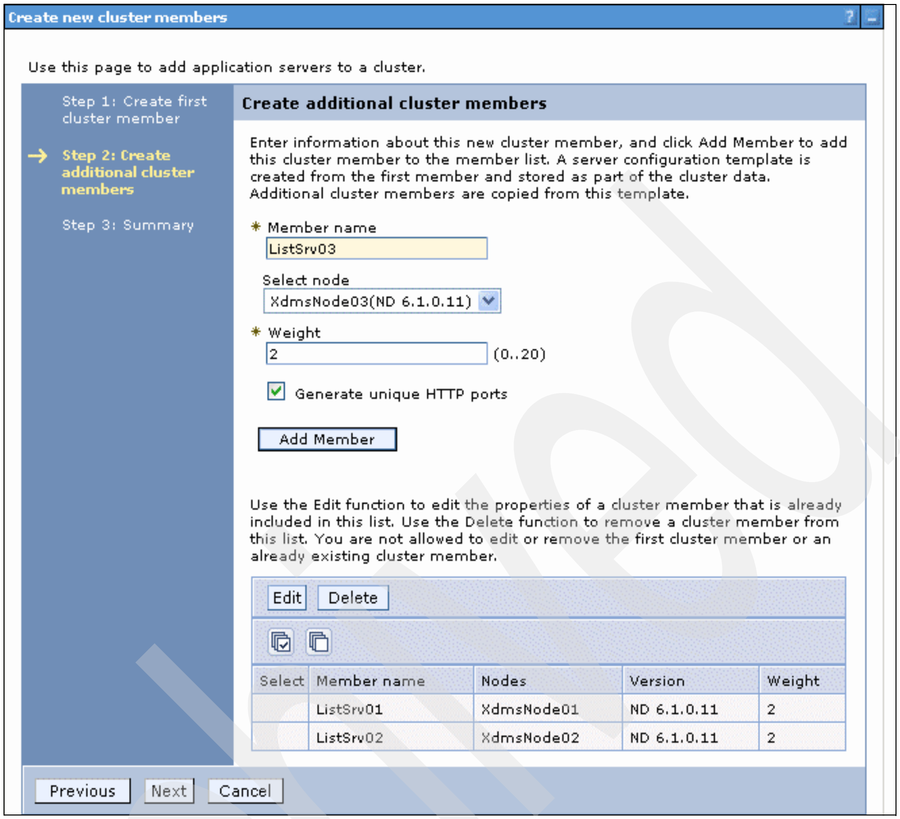


Figure 11-20 Create additional cluster members

5. In the Member name field, type the second new Application Server name (for our example, ListSrv04). In the Select node field, select the new Node name (for our example, XdmsNode04(ND 6.1.0.11), as shown in Figure 11-21 on page 407. Click the **Add Member** button.

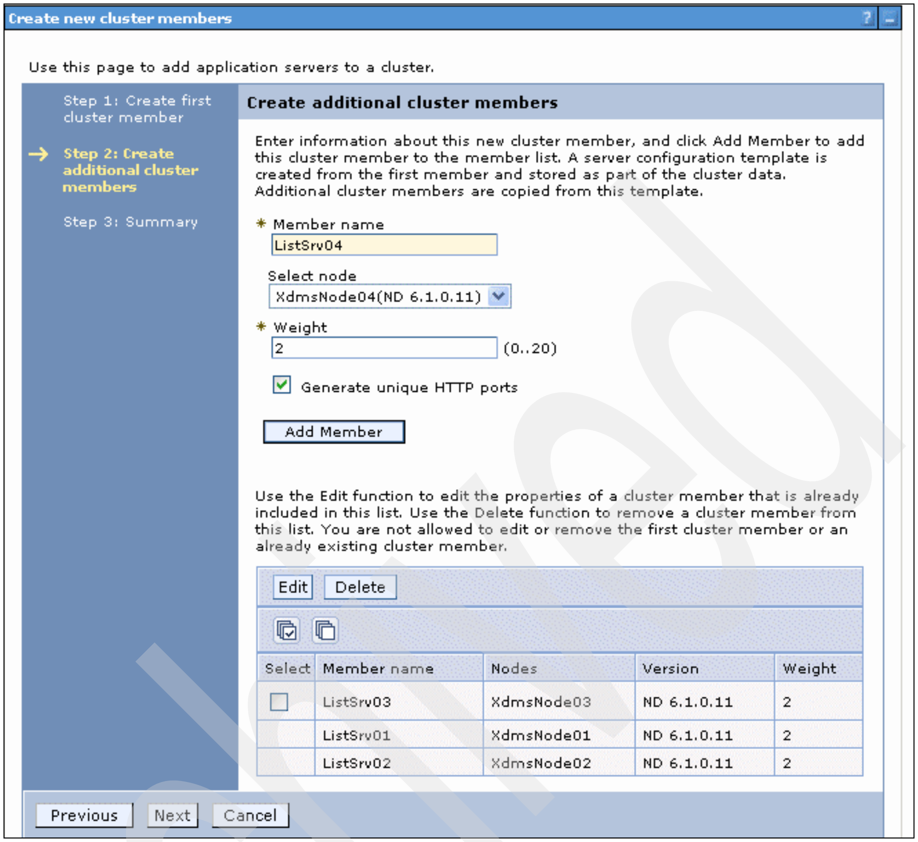


Figure 11-21   (0105Add_ClusterMember_ListSrv04)

6. Make sure the new member is added to the list in Figure 11-22 on page 408. Click the **Add Member** button, and then click the **Next** button.

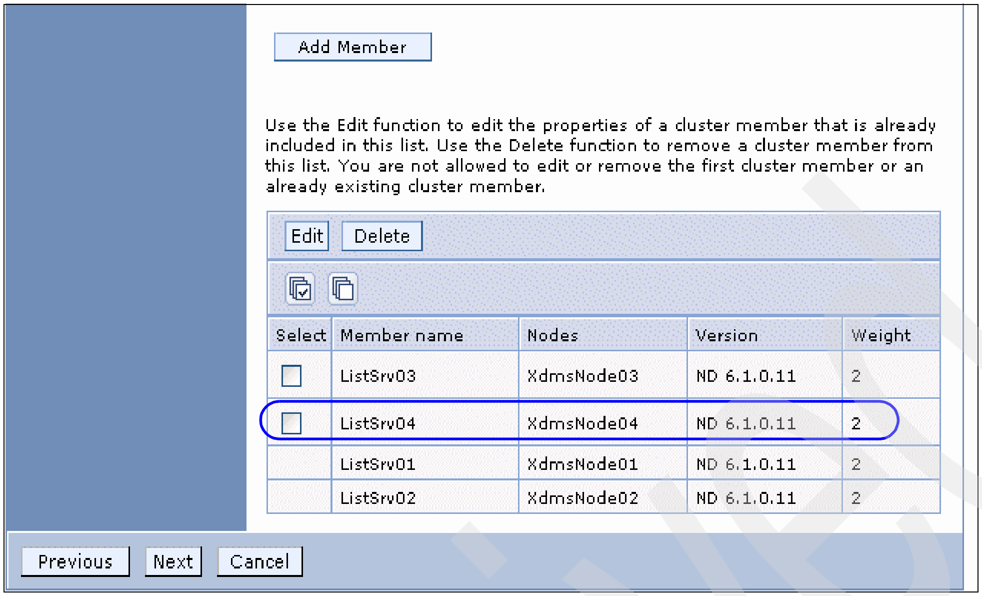


Figure 11-22 Add the member

7. In the Summary page (Figure 11-23), click **Finish**.

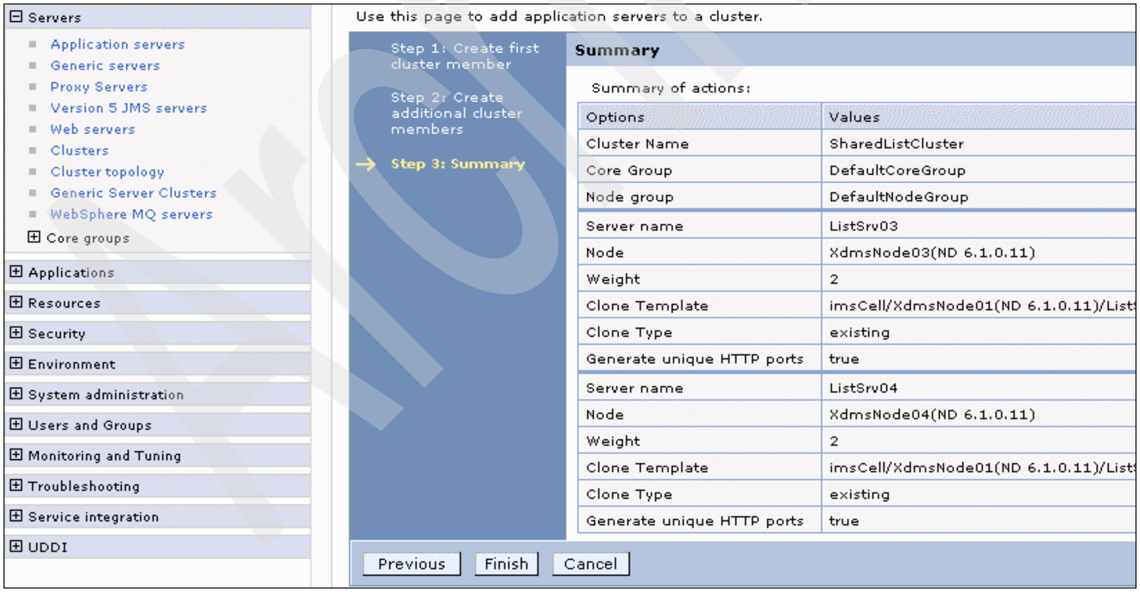


Figure 11-23 Summary page

8. In the cluster members name list of the Presence Cluster, make sure that there is a new Application Server. Click **Save** to save directly to the master configuration, as shown in Figure 11-24.

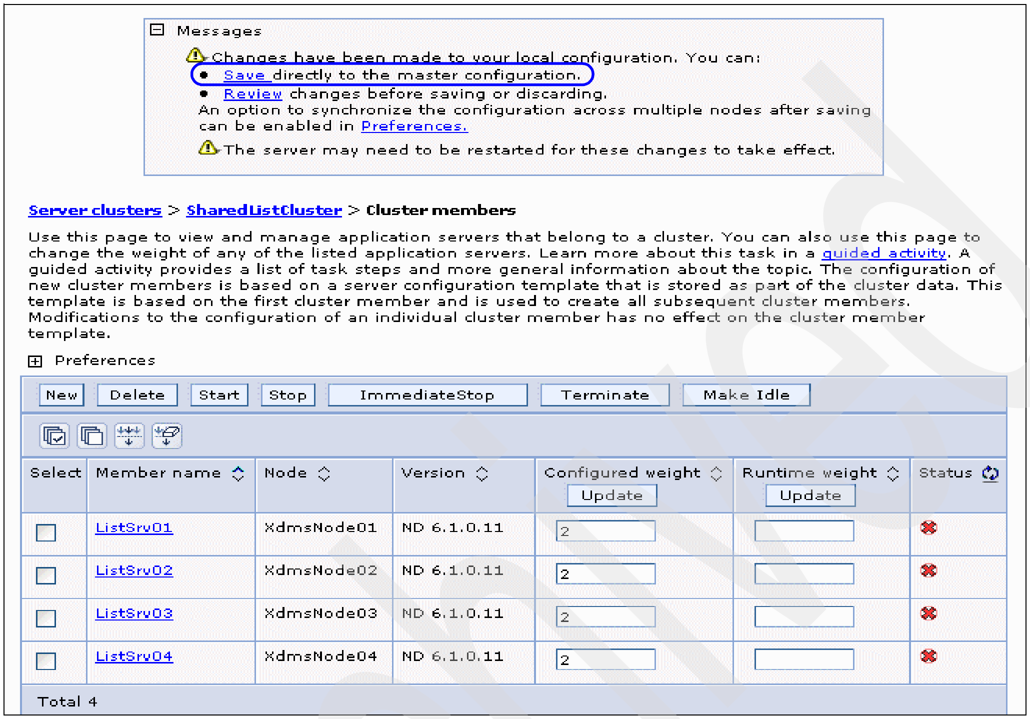


Figure 11-24 Save directly to the master configuration

Creating an Application Server, and adding it to the Presence Rules Cluster

After we create and federate a new node on the xdms3.itso.ral.ibm.com machine, we create a new server instance from the existing *default server* template. While we create the RulesSrv01 cluster member, the same is marked as a default server template. Login to the Deployment Manager Integration Server Console/administration console and follow these steps:

1. Under the Server clusters window, Figure 11-25 on page 410, from the cluster list, select **PresenceRulesCluster**.

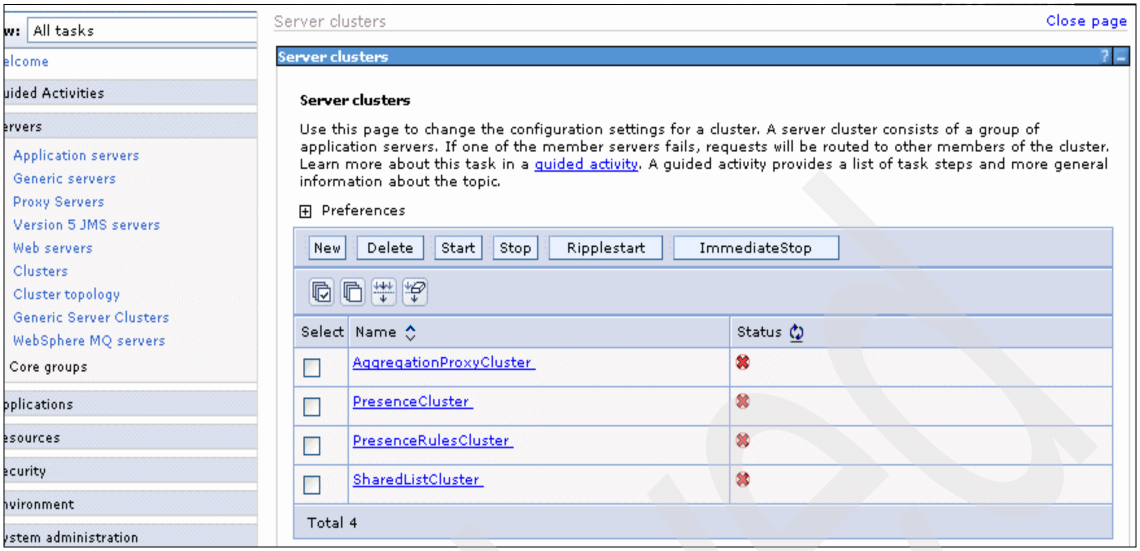


Figure 11-25 Select the PresenceRulesCluster

2. In the PresenceRuleCluster page, Figure 11-26 on page 411, on the right side of the page, under Additional Properties, click the **Cluster members** link.

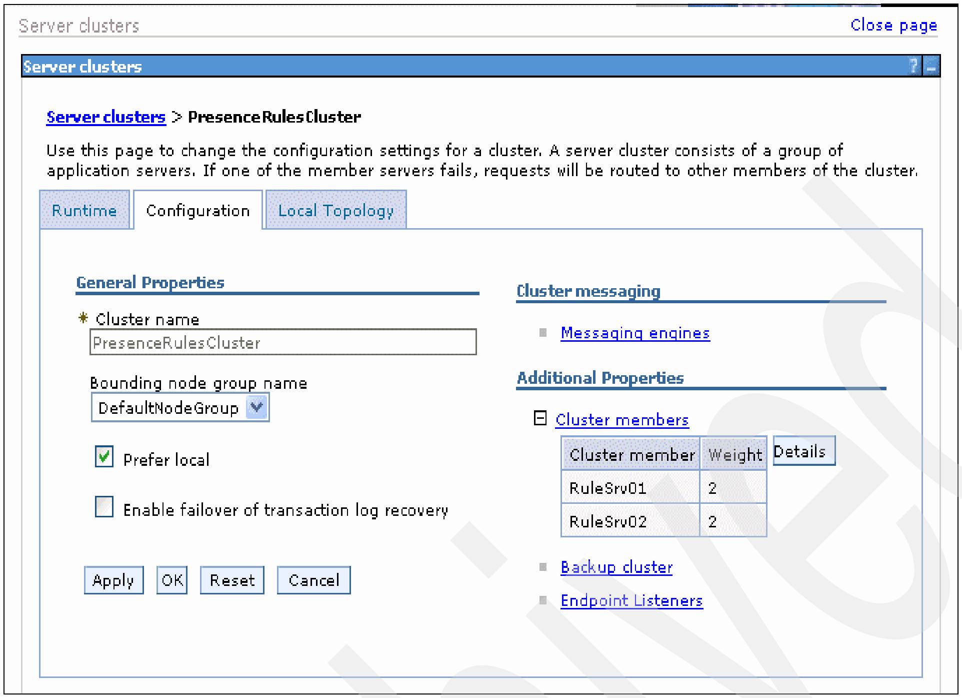


Figure 11-26 Click Cluster members

3. In Figure 11-27, click the **New** button.

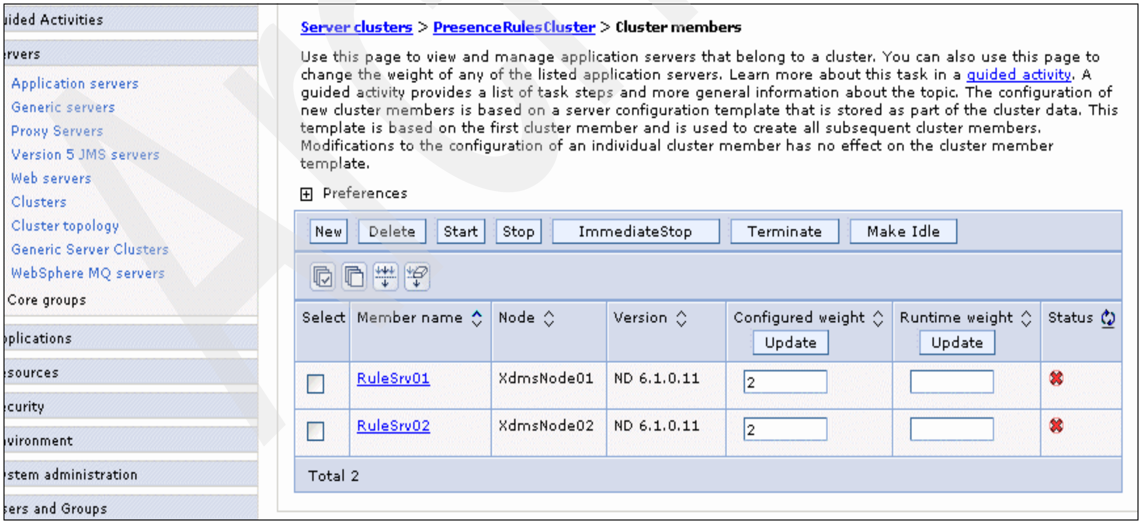


Figure 11-27 Click the New button

4. In Figure 11-28, in the Member name field, type the new Application Server name (for our example, RuleSrv03). In the Select node field, select the new Node name (for our example, XdmsNode03(ND 6.1.0.11). Click the **Add Member** button.

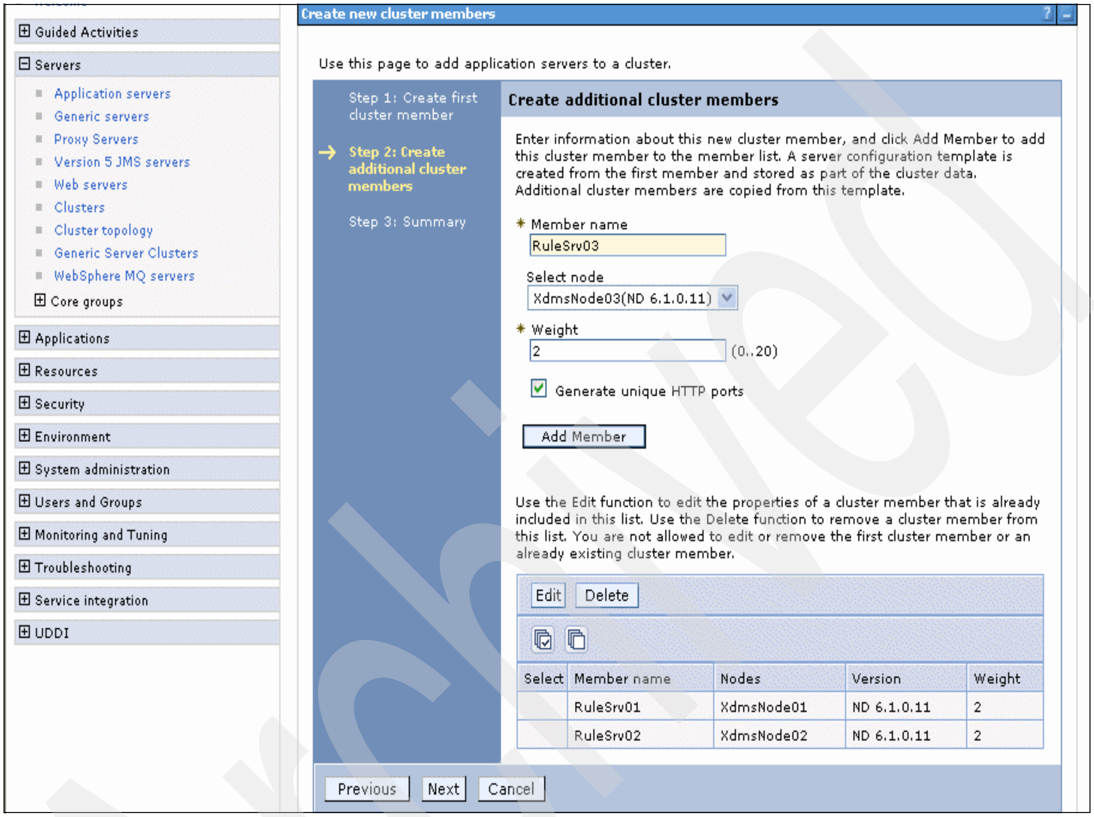


Figure 11-28 Add the new member (RuleSrv03) and select node

5. In Figure 11-28, in the Member name field, type the new Application Server name (for our example, RuleSrv04). In the Select node field, select the new Node name (for our example, XdmsNode04(ND 6.1.0.11). Click the **Add Member** button.

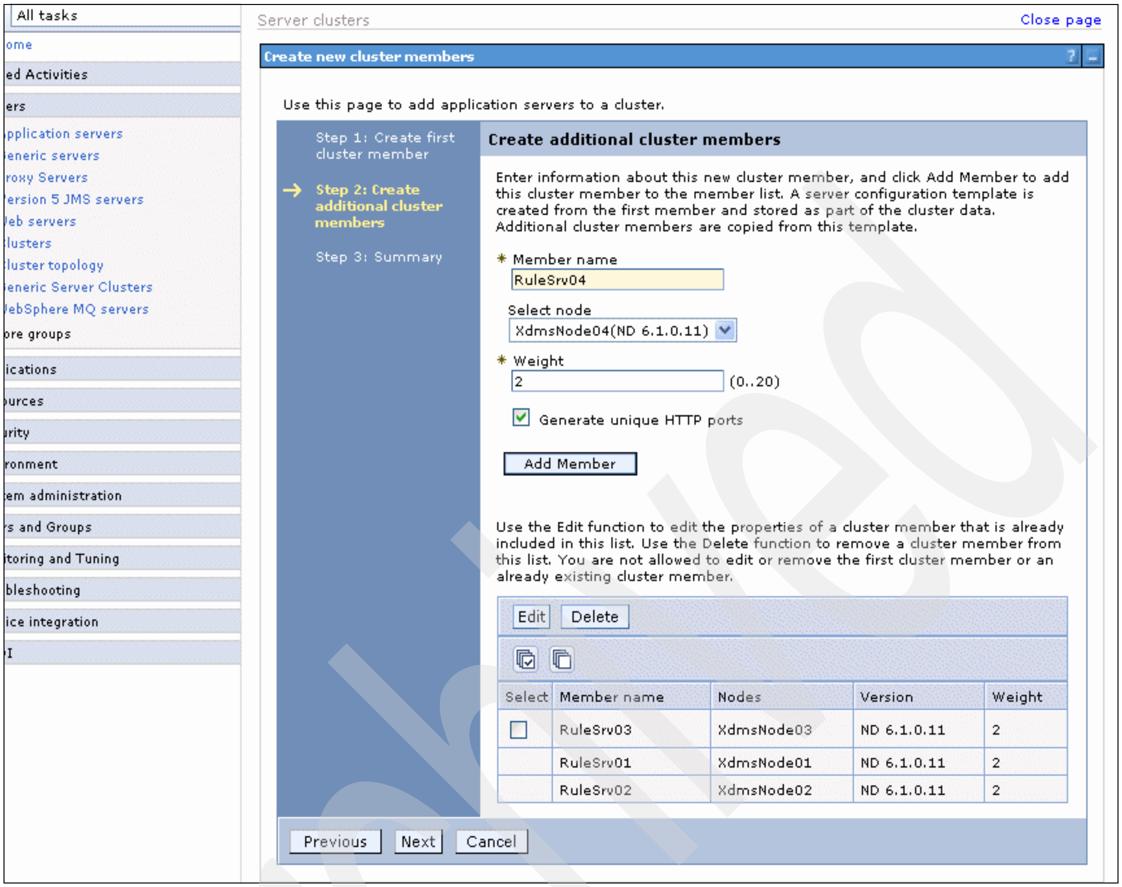


Figure 11-29 Add the next member (RuleSrv04) and select node

6. Make sure that the new member is added to the list, Figure 11-30 on page 414, click the **Add Member** button, and then, click the **Next** button.

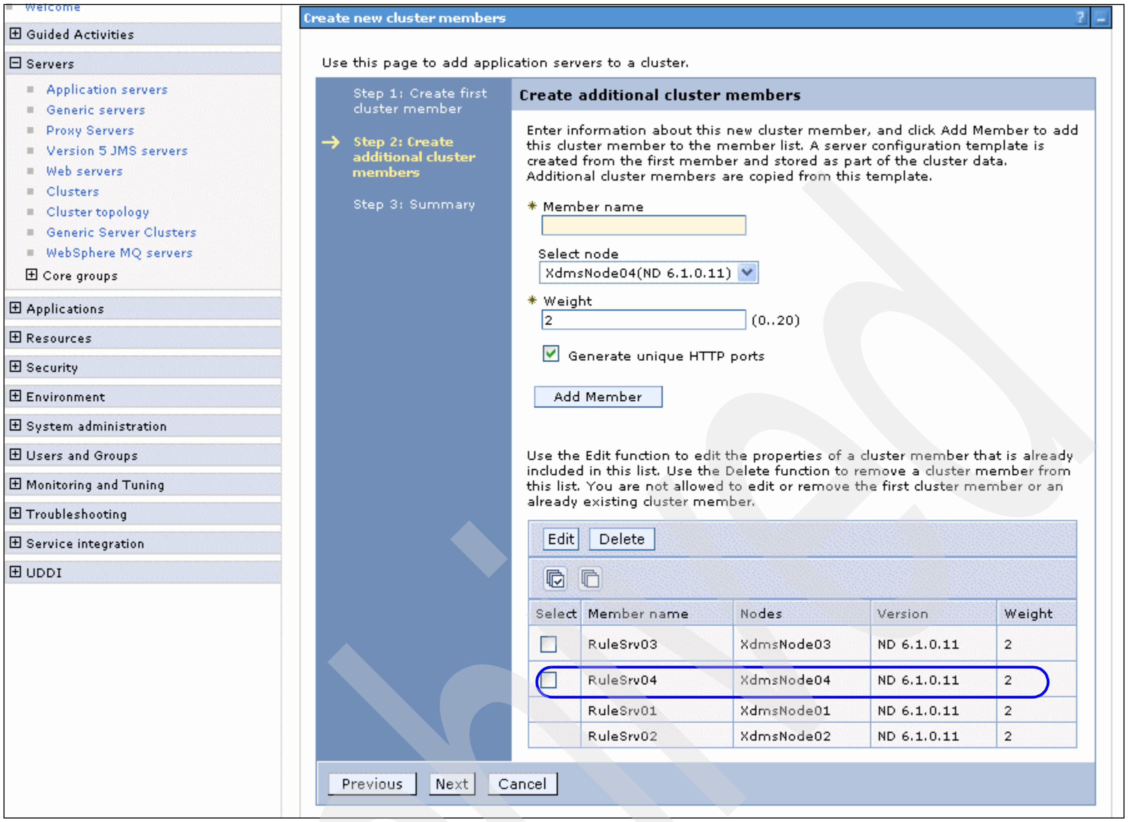


Figure 11-30 Look for the new members in the list

7. In the Summary page (Figure 11-31 on page 415), click **Finish**.

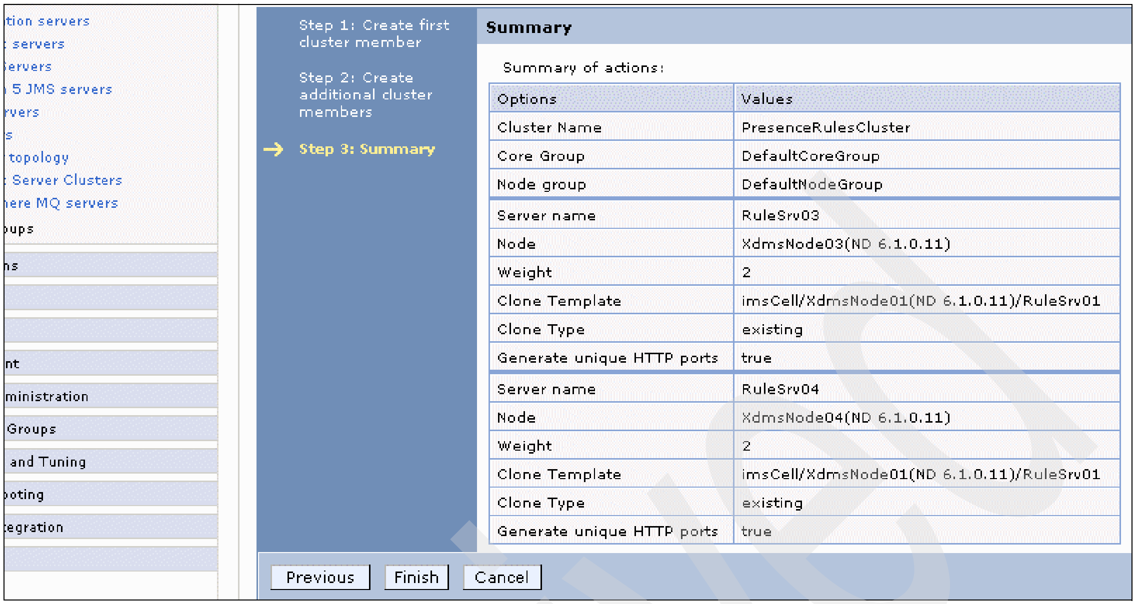


Figure 11-31 Summary page

8. Make sure that there is a new Application Server on the cluster members list of Presence Cluster, as shown in Figure 11-32 on page 416. Click **Save** to save directly to the master configuration.

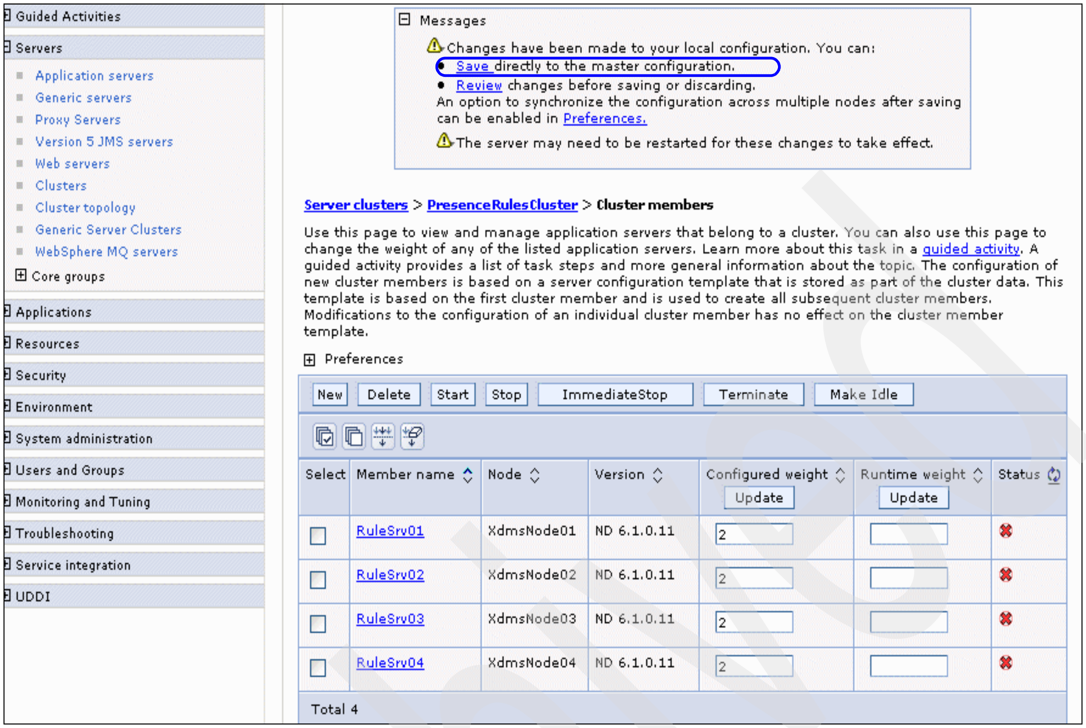


Figure 11-32 Save directly to the master configuration

Setting up the virtual host

For routing requests to new servers, host alias are added to the virtual host for the Shared List and Presence Rules Cluster.

Before you create the virtual host setting, make sure that:

- ► The wc_defaulthost port number of the two new Application Servers in the Shared List Cluster are 9082 and that the servers are listening on the port.
- ► The wc_defaulthost port number of the two new Application servers in the Presence Rules Cluster are 9081 and that the servers are listening on the port.

To set up the virtual host for a routing request to the Shared List Cluster:

1. From the left panel, click **Environment** → **Virtual Hosts**. Click **Shared_List_Host**, as shown in Figure 11-33 on page 417.

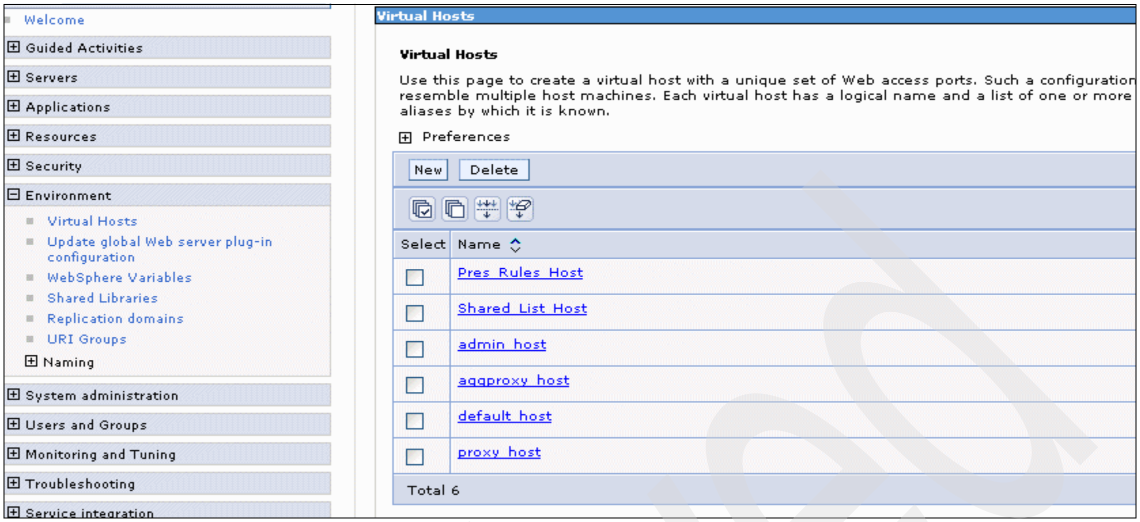


Figure 11-33 Virtual Hosts list

2. At the right side of the Virtual Hosts detail information page, Figure 11-34, click the **Host Aliases** link.

Figure 11-34 Select Host Aliases

3. At the top of Figure 11-35 on page 418, click the **New** button, which shows the Host Aliases list.

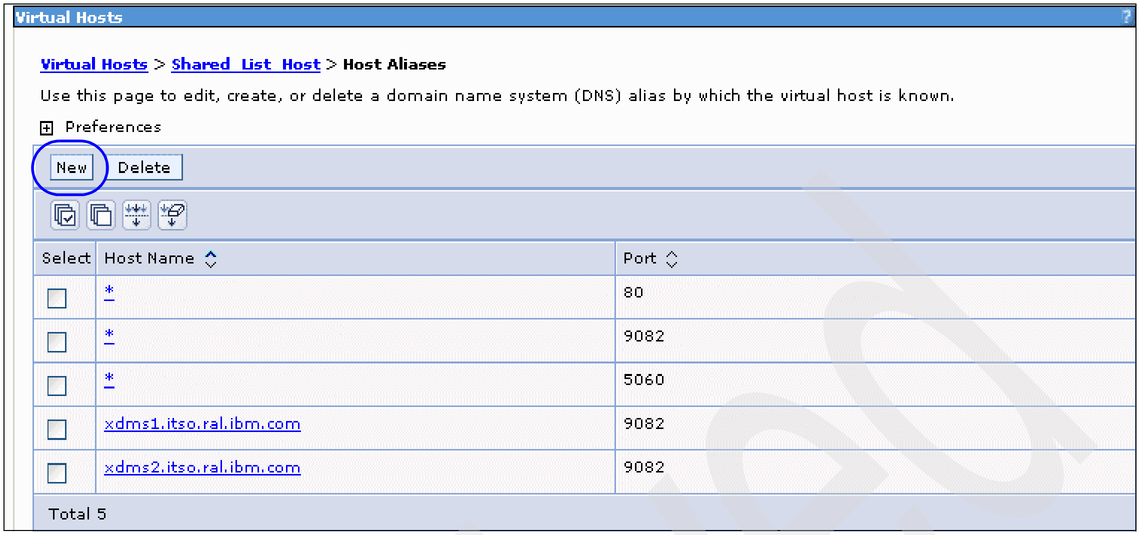


Figure 11-35   Click New to display the Host Aliases list

4. In the Host Name field, type the new host name, `xdms3.itso.ibm.com`. In the Port field, type `9082`, as shown in Figure 11-36. Click **Apply**, and then save.

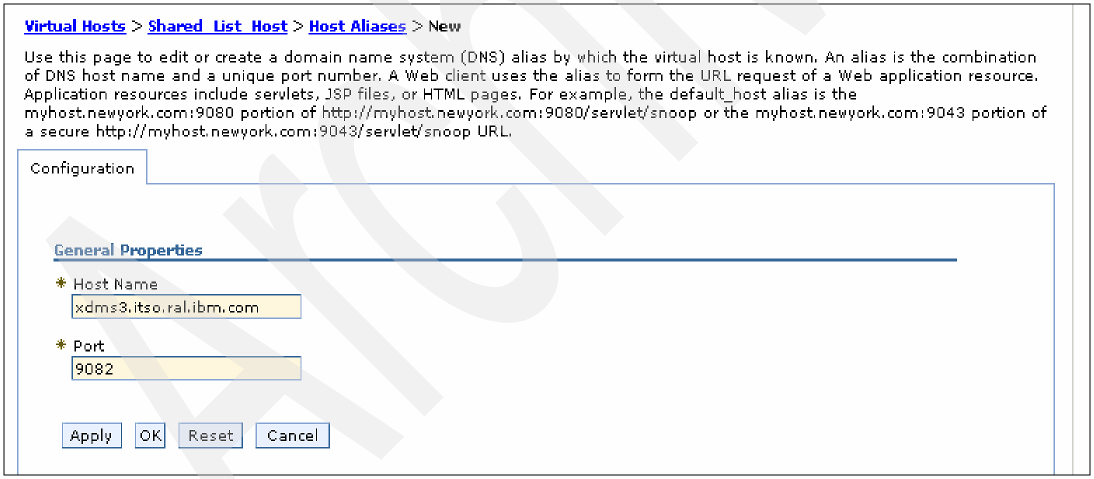


Figure 11-36   0204Add_SharedLists_HostAliases_xdms3

5. To create one more Host Alias, click the **New** button, as shown in Figure 11-37 on page 419.

6. In the Host Name field, type the new host name, `xdms4.itso.ibm.com`. In the Port field, type `9082`, as shown in Figure 11-36. Click **Apply**, and then save.

7. Make sure that the results are displayed, as in Figure 11-37.

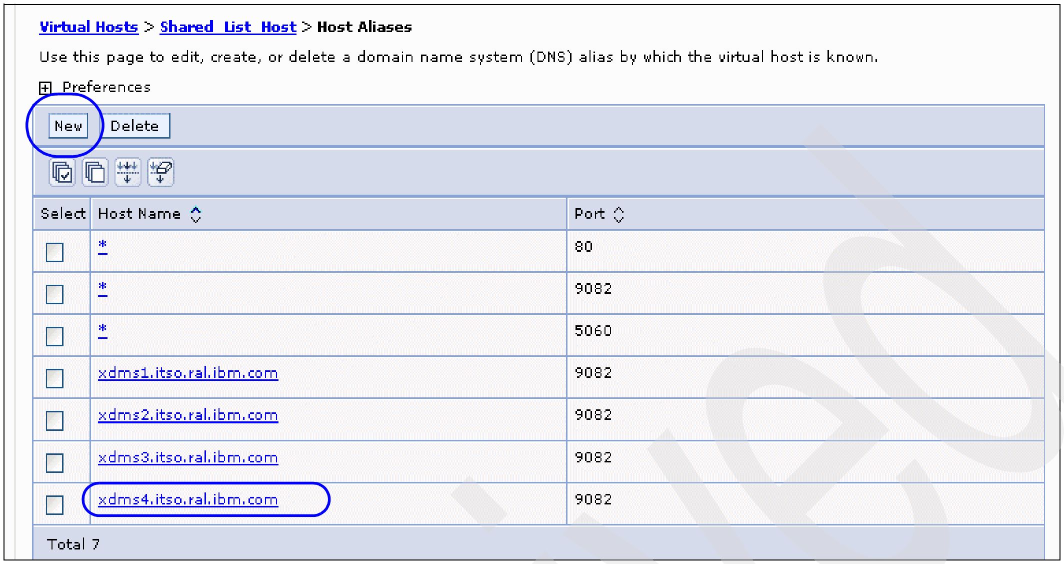


Figure 11-37 Added Virtual Hosts

To request routing to the Presence Rules Cluster:

1. From the Virtual Hosts window, Figure 11-38, select **Presence_Rules_Host**.

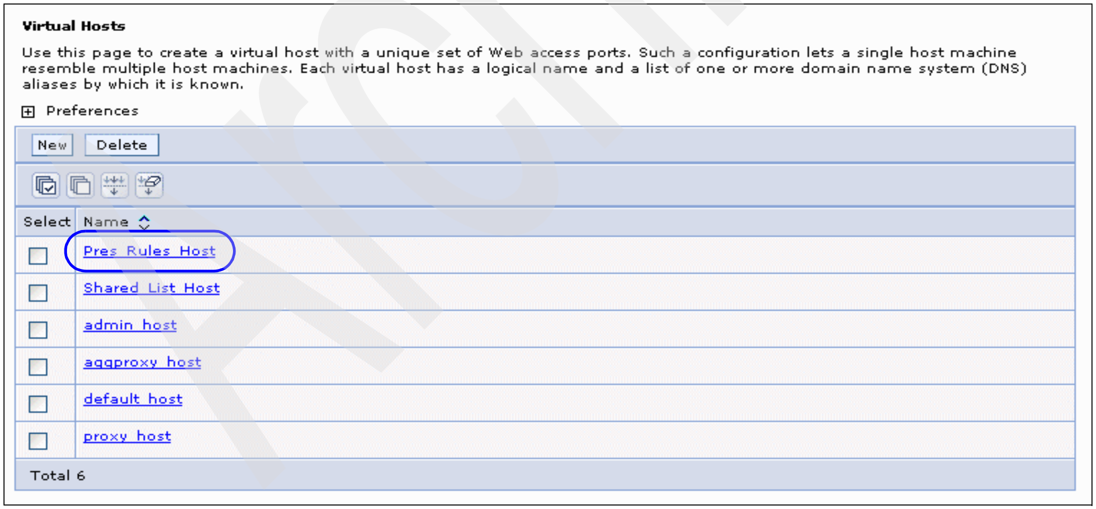


Figure 11-38 Select the Virtual Host

2. On the Virtual Host detail information page, at the right side, click the **Host Aliases** link.
3. At the top of the next page, click the **New** button, which shows the Host Aliases list.
4. In the Host Name field, type the new host name, `xdms3.itso.ibm.com`. In the Port field, type `9081`, as shown in Figure 11-39. Click **Apply**, and then save.

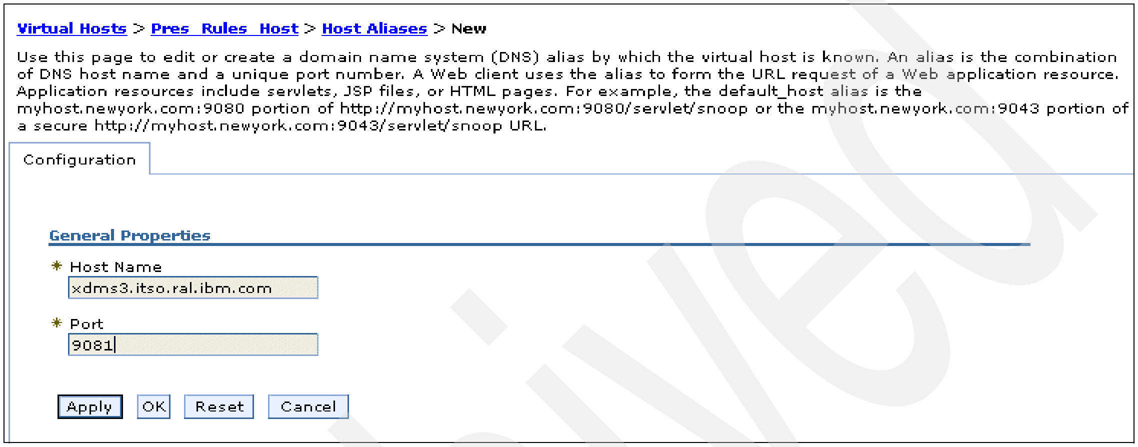


Figure 11-39 Specify Host Name and Port

5. To create one more Host Alias, on the next page click the **New** button.
6. In the Host Name field, type the new host name, `xdms4.itso.ibm.com`. In the Port field, type `9081`, as shown in Figure 11-40. Click **Apply**, and then save.

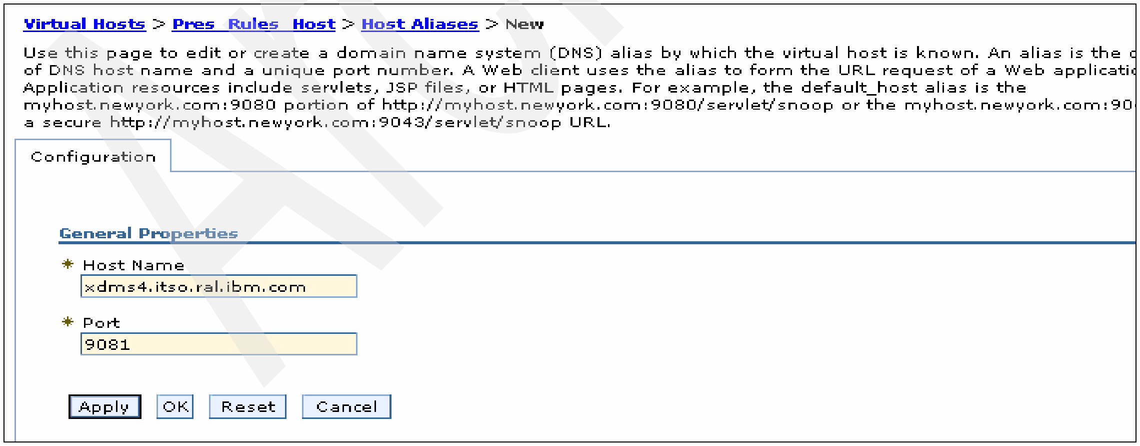


Figure 11-40 Specify Host Name and Port for additional Host Alias

7. Make sure the results are displayed, similar to Figure 11-41.

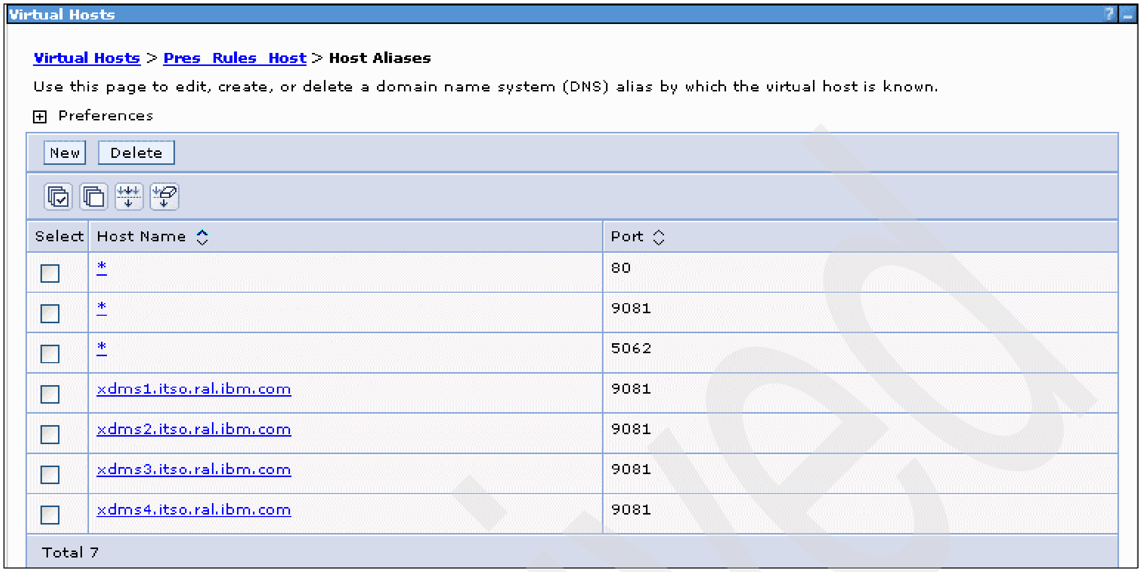


Figure 11-41 Host Aliases results

11.2.2 Vertical clustering configuration

To vertically scale the Presence Server Cluster, create one new Application Server on the node, which is created for the basic environment. Then the server is added to the Presence Server Cluster.

The configuration process is:

1. Create an Application Server, and add it to the Presence Server Cluster.
2. Add the server to SIBus as a bus member.
3. Set up the Virtual Host.

Creating an Application Server, and adding it to the Presence Server Cluster

To begin creating an Application Server and adding it to the Presence Server Cluster:

1. From the left panel, click **Servers** → **Clusters**. From the cluster list in Figure 11-42 on page 422, select **PresenceCluster**.

Figure 11-42 Select PresenceCluster

2. In the PresenceCluster detail page, on the right side, click the **Cluster members** link.
3. In the Cluster members list page, Figure 11-43, click the **New** button.

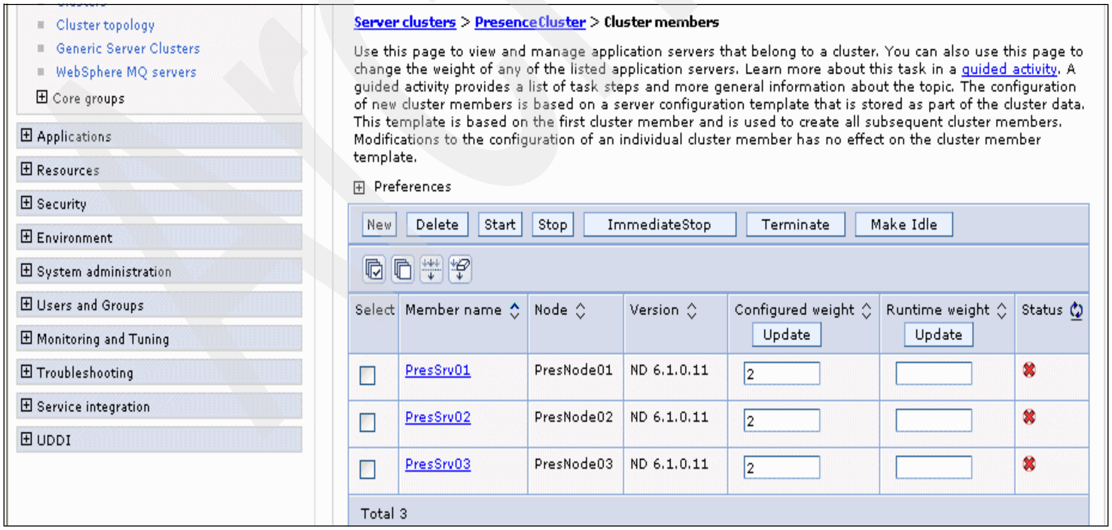


Figure 11-43 Click the New button

4. In the Member name field, type the new Application Server name (for our example, `PresSrv04`), as shown in Figure 11-44. In the Select node name field, select **PresNode01(ND 6.1.0.11)**. Click the **Add Member** button.

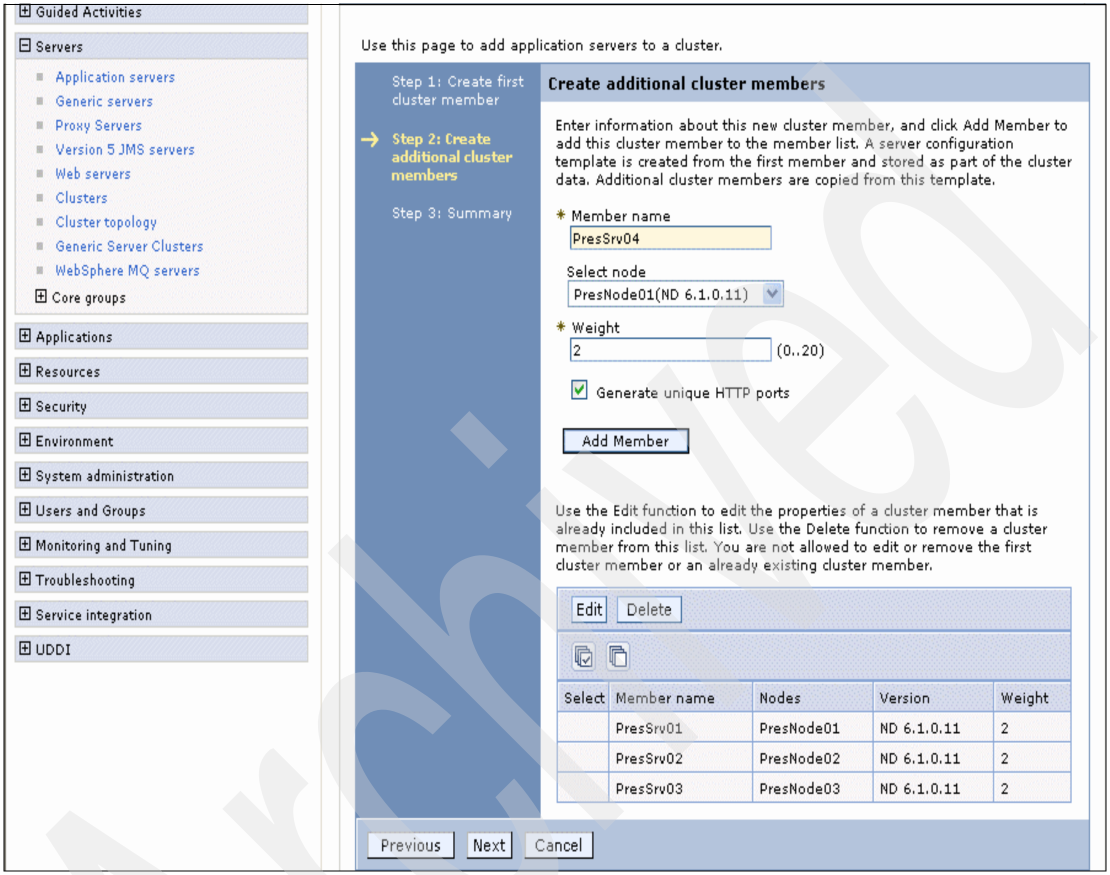


Figure 11-44 Specify the Member name and select the node

5. Make sure that the new member is added to the list in Figure 11-45 on page 424. Click the **Add Member** button, and then click the **Next** button.

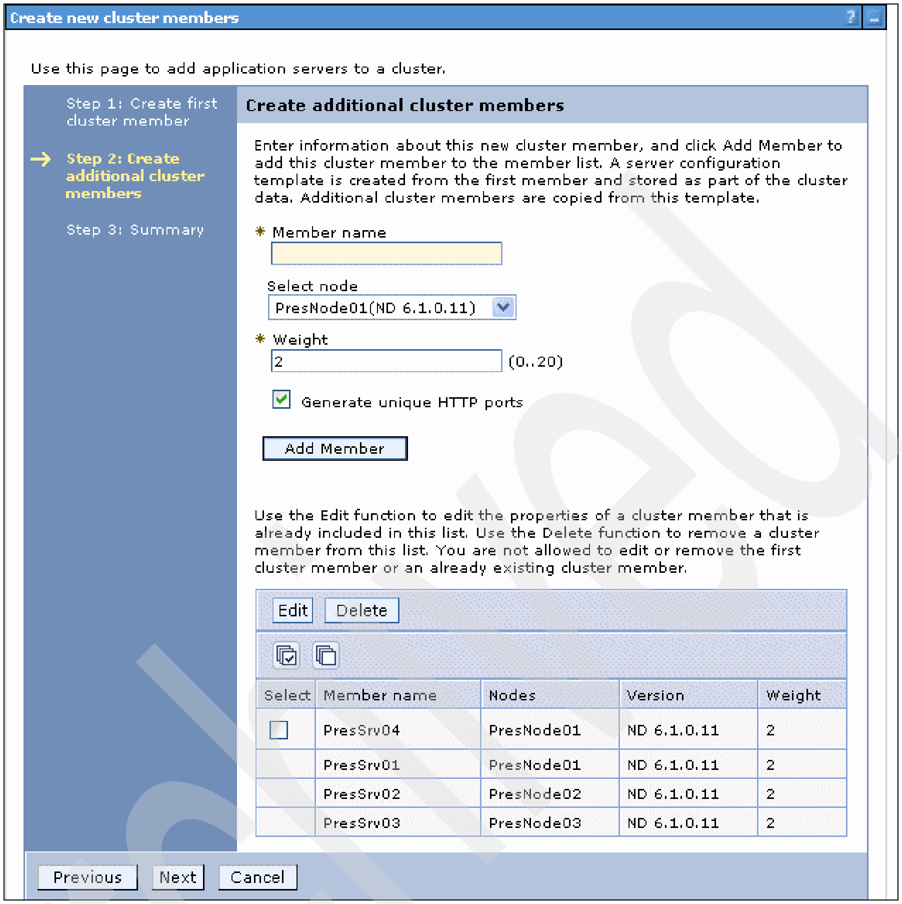


Figure 11-45 Type Member Name and select node

6. In the Summary page, Figure 11-46 on page 425, click **Finish**.

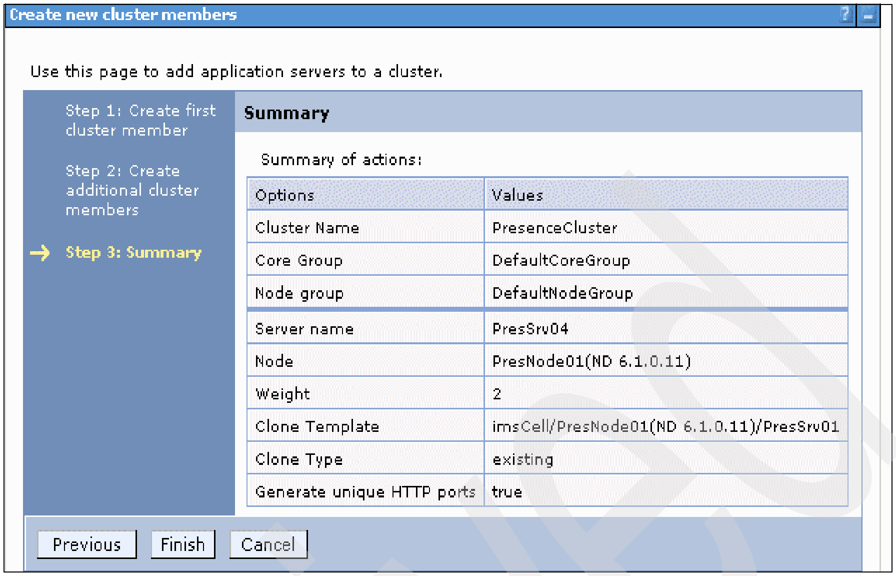


Figure 11-46 New cluster members' Summary page

7. On the cluster members list of the Presence Cluster, make sure that there is a new Application Server, as shown in Figure 11-47 on page 426. Click **Save** to save directly to the master configuration.

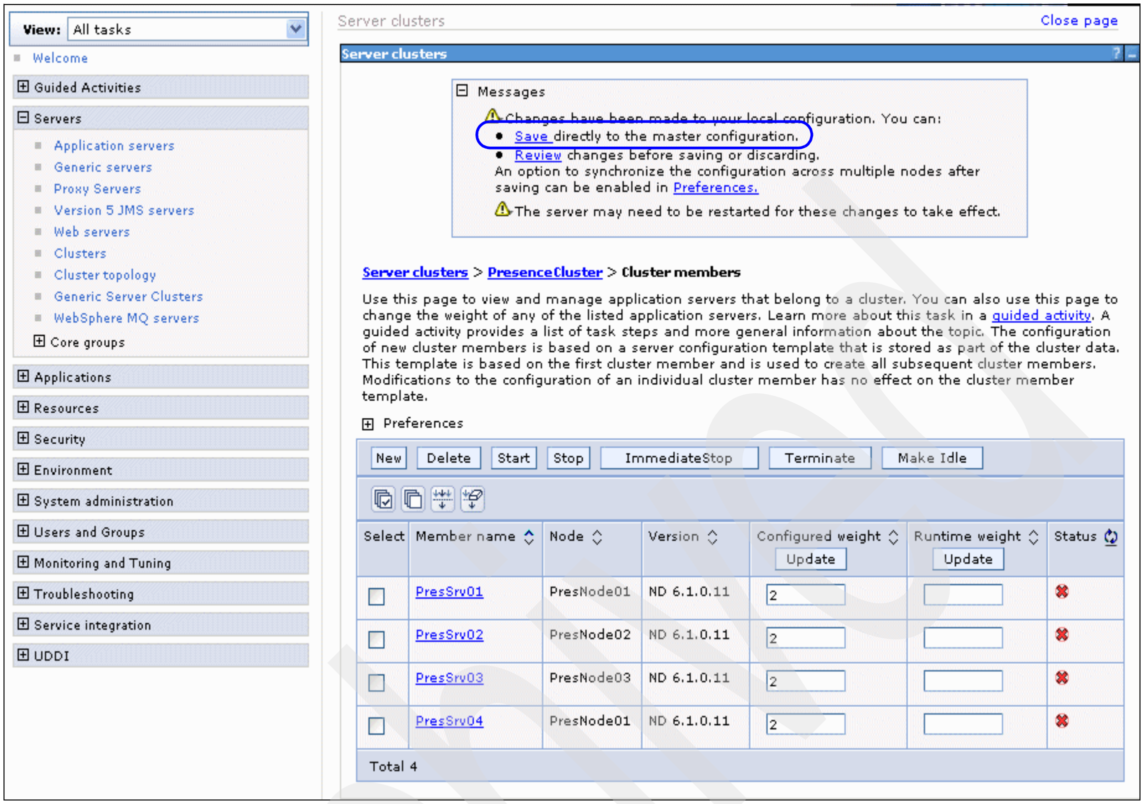


Figure 11-47 Save directly to the master configuration

Adding the server to SIBus as a bus member

In this section, we add the new Application Server `PresenceBUS(SIBus for Presence Servers)` as a bus member. Perform the following steps:

1. In Figure 11-48 on page 427, from the Bus list, select **PresenceBUS**(SIBus for Presence Servers).

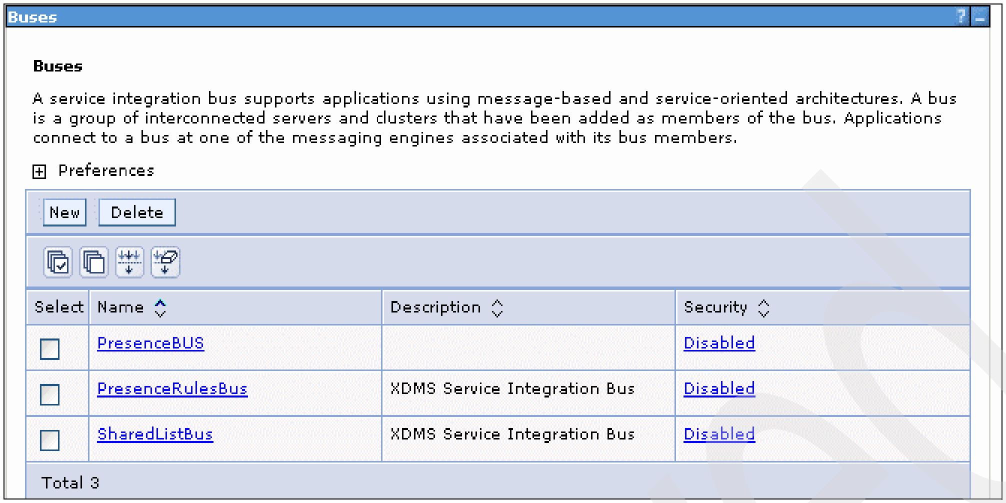


Figure 11-48 Select PresenceBUS

2. Figure 11-49 on page 428 shows the detailed information of the bus. At the top, right side of the page, click the **Bus members** link.

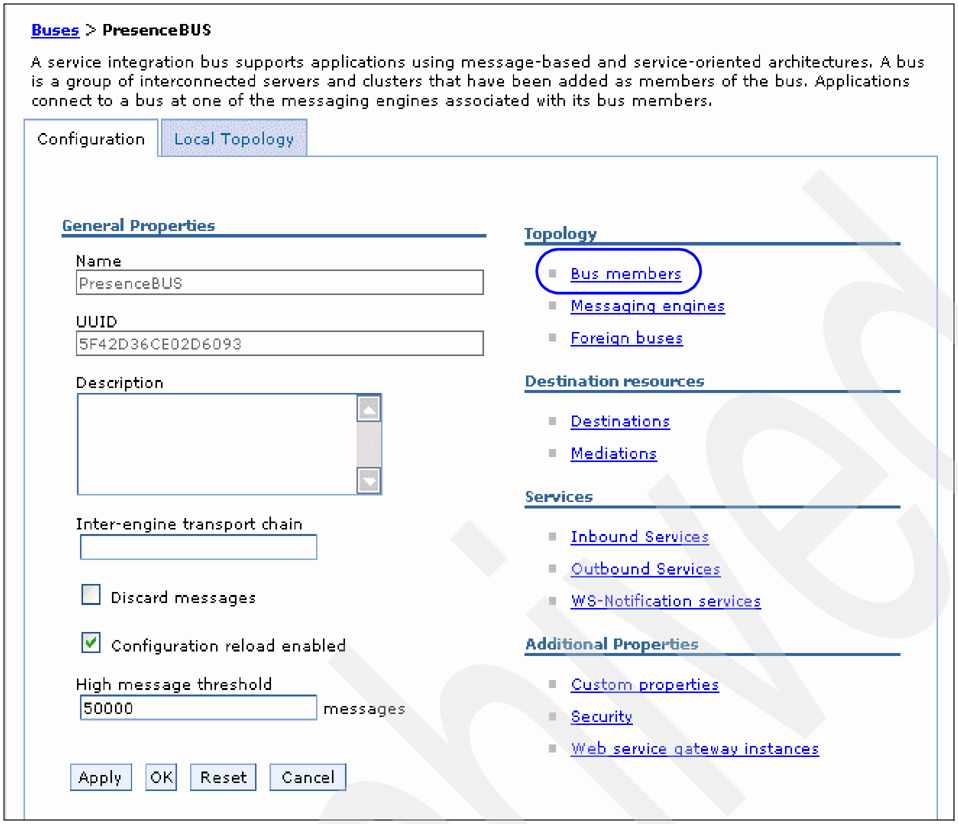


Figure 11-49 Click the Bus members link

3. On the Bus member list, shown in Figure 11-49, click the **Add** button.

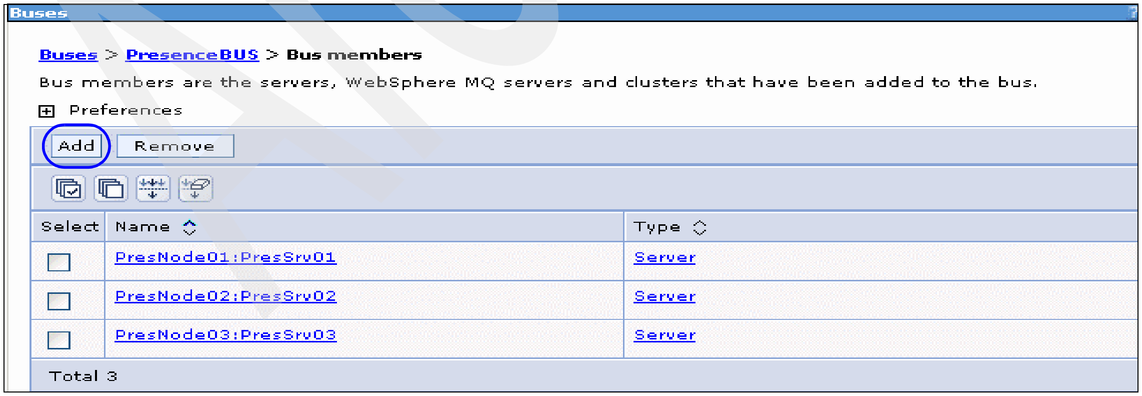


Figure 11-50 Click Add

4. On Figure 11-51, choose the server that you want added to SIBus as a member, for example, it is PresNode01:PresSrv04.

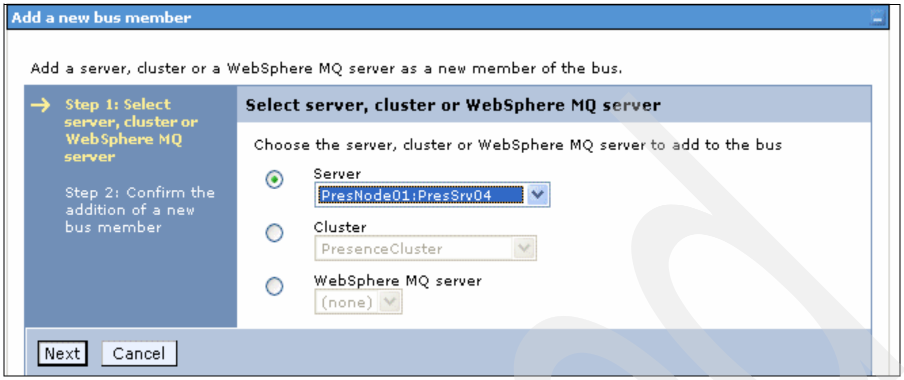


Figure 11-51 Select the server to add to SIBus as a member

5. As the type of message store, select **File Store**, as shown in Figure 11-52. Click the **Next** button.

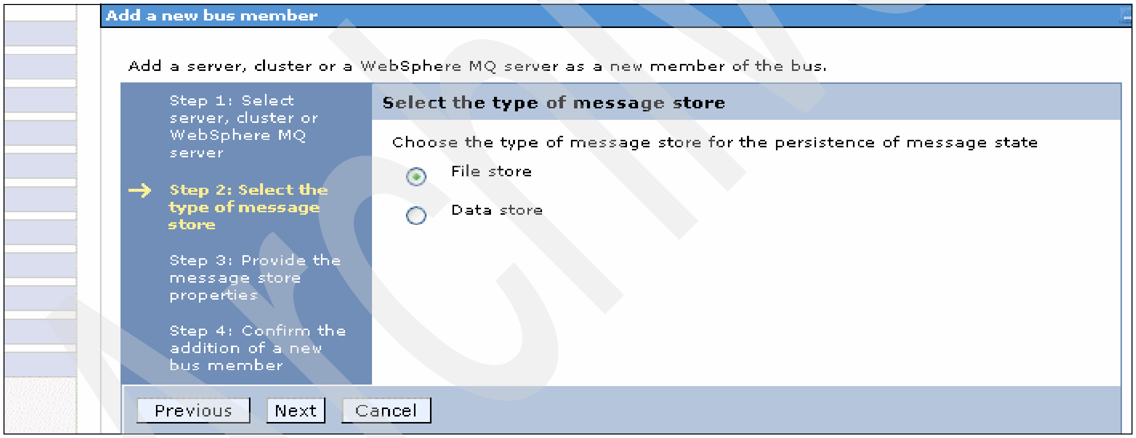


Figure 11-52 Select the type of message store

6. In Figure 11-53 on page 430, under the Provide the message store properties section, accept all of the defaults, and click the **Next** button.

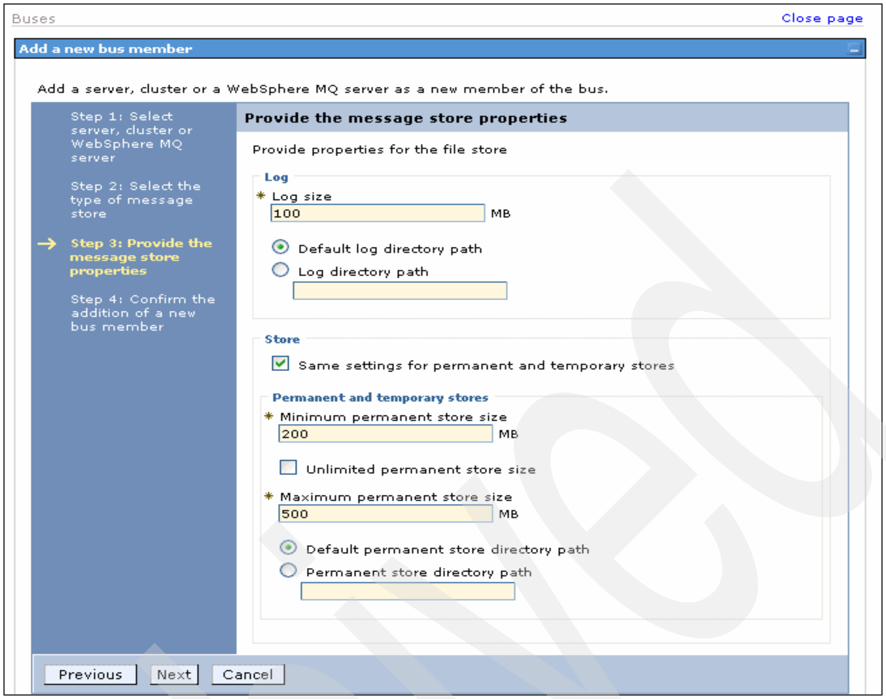


Figure 11-53 Accept these default message store properties

7. In Figure 11-54, confirm the addition of the new bus member, and then click **Finish**.

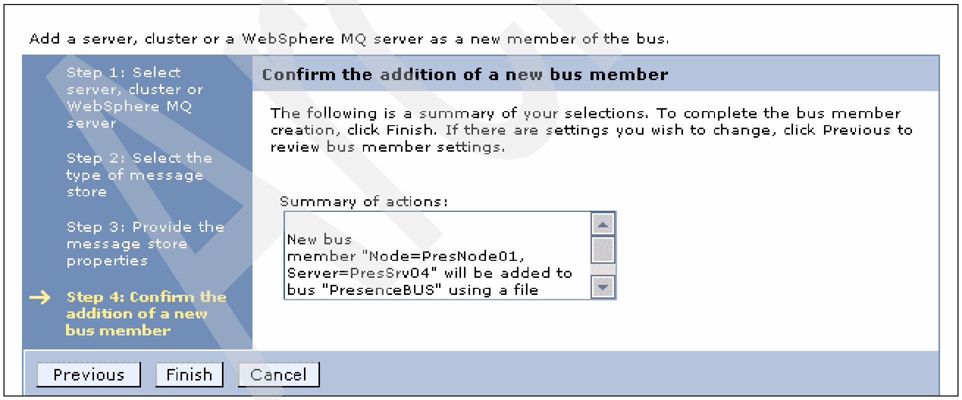


Figure 11-54 Confirm the addition of a new bus member

8. Confirm that the new bus member is added to the bus member list, as shown in Figure 11-55, and click **Save**.

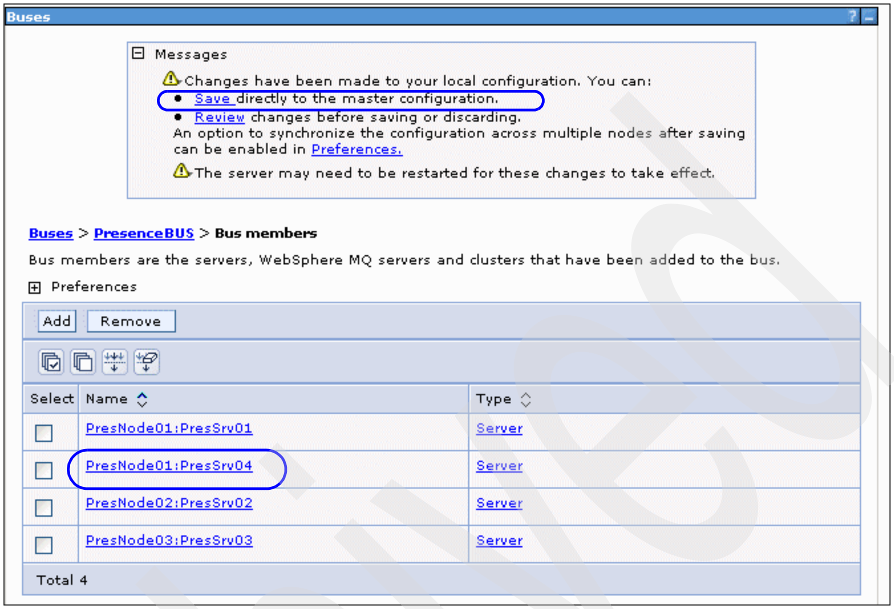


Figure 11-55 Save directly to the master configuration

Setting up the Virtual Host

When a new Application Server is created on the node on which an application is already installed, the new server's SIP_DEFAULTHOST port number is something other than 5060. Whereas, 5060 is usually the default SIP_DEFAULTHOST port number. In this case, this port number has to be on the list of Host Aliases from the Virtual Host, which is related to the Presence Server application (*default_host* in our environment); therefore, the Presence Server application can process SIP requests that come to this new server. Perform the following steps:

1. Check the new server's SIP_DEFAULTHOST port number, as shown in Figure 11-56 on page 432. In our environment, it is 5062.

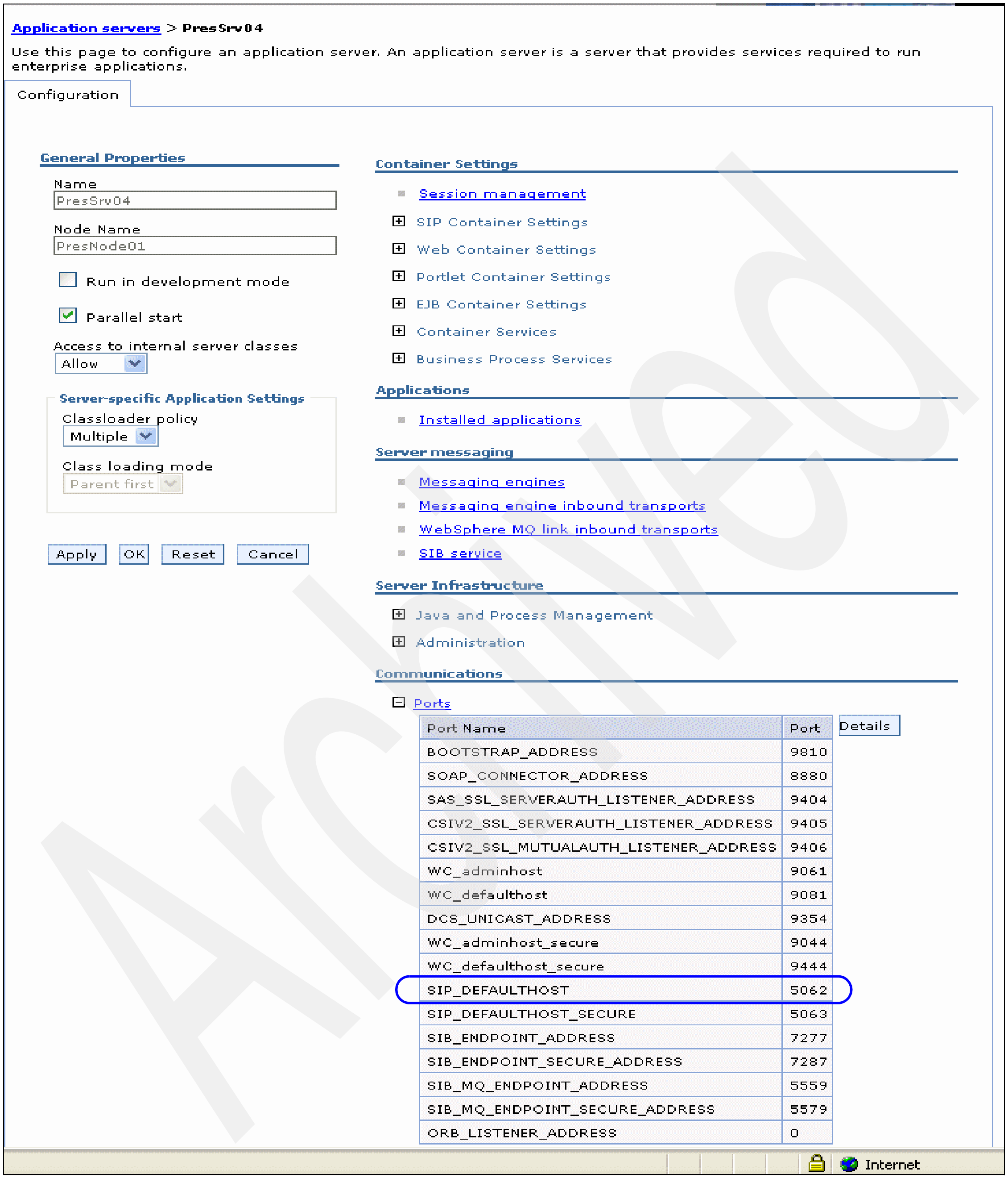


Figure 11-56 Check the new server's SIP_DEFAULTHOST port number

2. Check the virtual host for the presence application:

   a. From the left side of the menu, click **Applications** → **Enterprise Applications**.

   b. Select **Presence Server**, as shown in Figure 11-57 to view the detailed information.

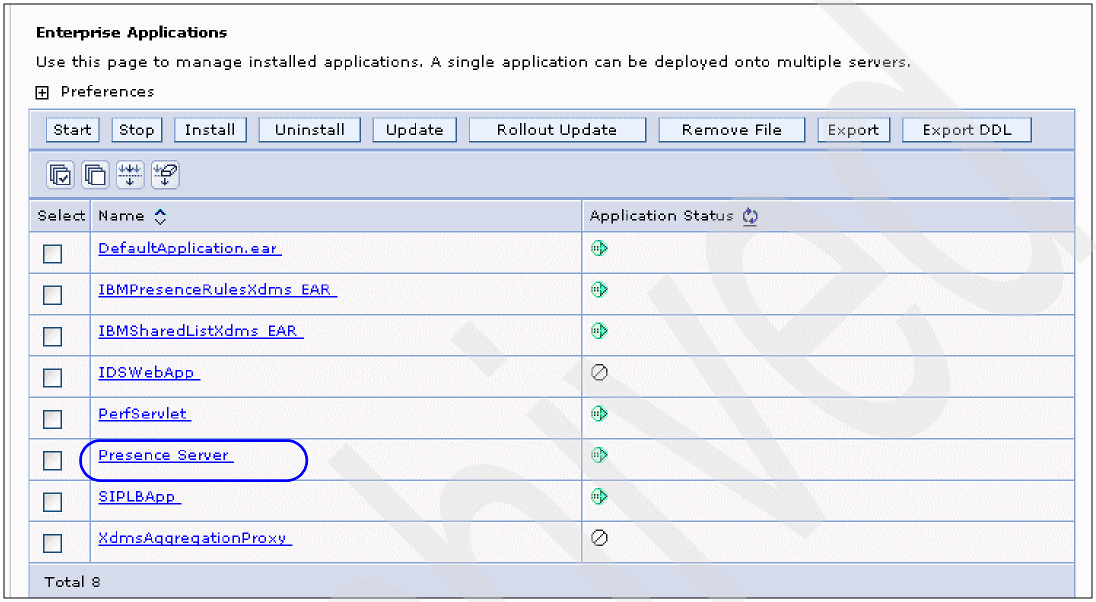


Figure 11-57 Check the virtual host for the presence application

3. In the Web Module Properties, at the left side menu, click the **Virtual hosts** link, as shown in Figure 11-58 on page 434.

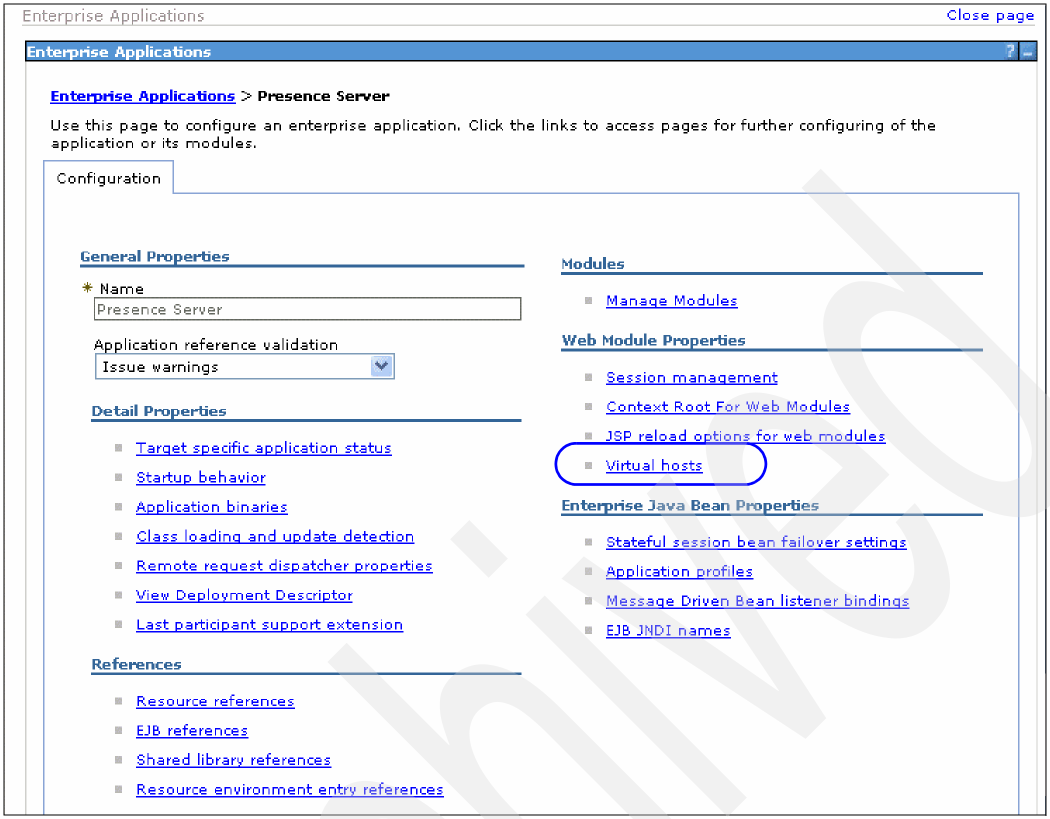


Figure 11-58 Click the Virtual hosts link

4. Make sure that the virtual host is `ServerSiplets`, as shown in Figure 11-59. For our example, it is default_host.

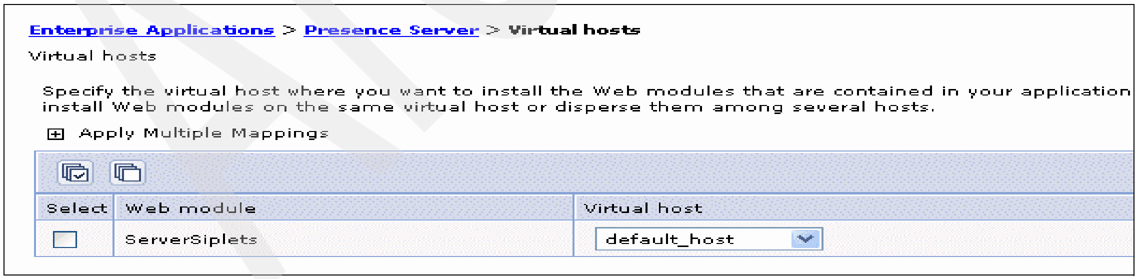


Figure 11-59 Ensure that the virtual host ServerSiplets is present

5. Add the new server's SIP_DEFAULTHOST port number to the list of default_host virtual host:

   a. From the left side of the menu, as shown in Figure 11-60, select **Environment** → **Virtual Hosts**.

   b. In the list to view the detailed information, choose **default host**.

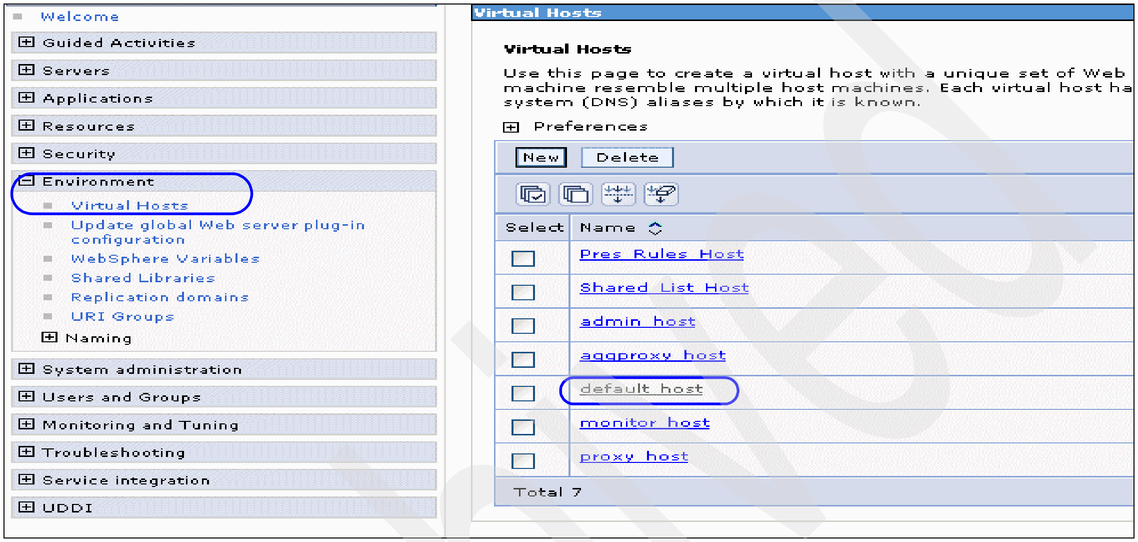


Figure 11-60 Choose the default host link

6. In Figure 11-61(detail info of default_host), at the right side of the page, click the **Host Aliases** link.

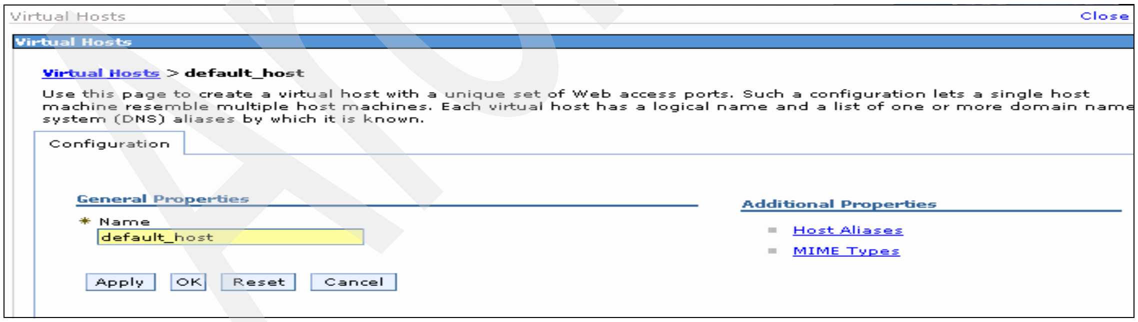


Figure 11-61 Default host

7. In Figure 11-62 (the list for Host Aliases), click the **New** button, and add the new Host Aliases. The Host Name is *, and the Port is 5062.

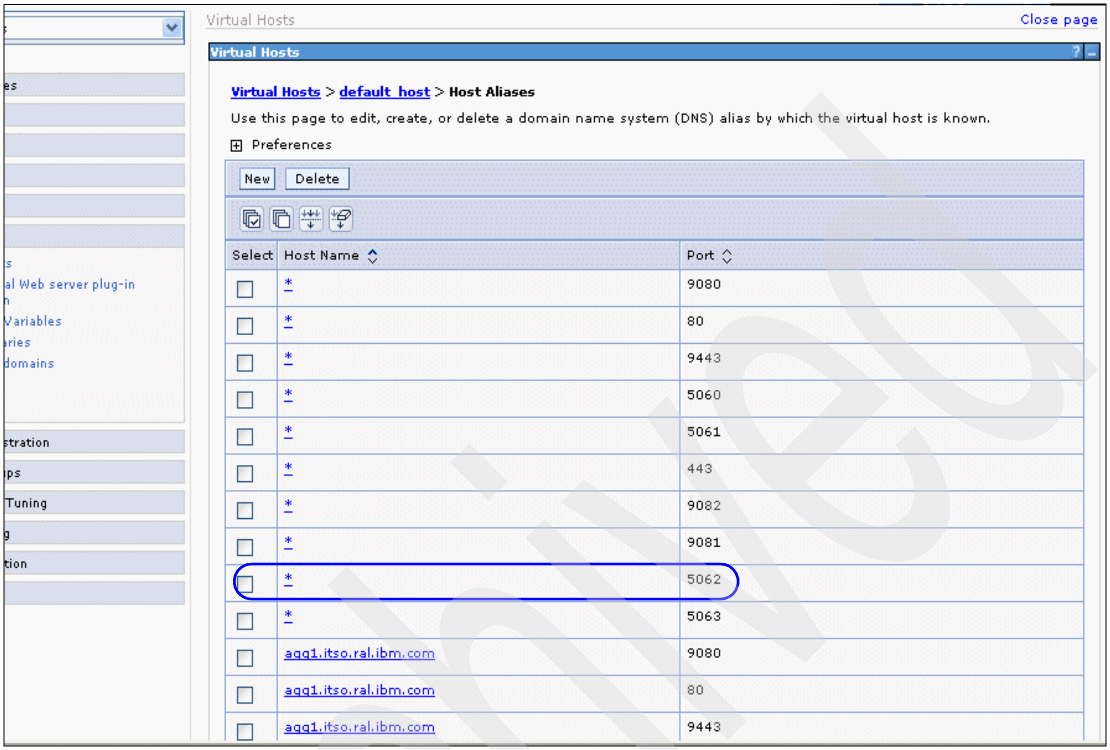


Figure 11-62 Add the new host aliases

12

Testing the scalable scenario

Having discussed the procedure to extend the Presence Server environment in terms of scalability in the previous chapter, in this chapter, we discuss how to extend and execute the scenario test cases in a scalable environment.

The objectives of this chapter are to:

- Demonstrate the distribution of requests among the Presence Cluster Server instances
- Demonstrate the distribution of requests that are processed by the Shared List Cluster server instances
- Demonstrate the distribution of requests that are processed by the Presence Rules Cluster Server instances
- Observe failures in processing requests and the possible causes

The underlying goal in this section is to ascertain that the environment demonstrates better throughput with no loss of calls, maintaining fail-over support, and without hitting any overload conditions in the environment. In addition, we discuss a couple of load test cases and analyze the results that are obtained from the scalable environment. Before continuing further, refer to the overload and tuning configurations in Appendix A.3, “Tuning” on page 457.

Though the scope of this book captures configurations and settings to Presence Server and XML Document Management Servers (XDMS), both of these components have inseparable dependency on underlying database mechanics and messaging resources. The functions that drive substantial utilization of database resources are:

- PUBLISH of Presentity information: A Presentity's presence information is an aggregation of one or more tuples comprising of presence also termed as aggregated presence information or full document. For an in-depth explanation on the format and semantics of a presence document, refer to IETF specifications RFC 3863 and RFC 4480. From the time Initial PUBLISH is received, the Presence Server is responsible for maintaining, aggregating and refreshing the presence information of a presentity by applying changes to presentity's state. This processing could be triggered more than once because of subsequent re-PUBLISH requests from client applications. The key aspect of this processing is persistence of presence information.
- SUBSCRIBE for presentity information: In the case of Subscription processing, Presence Server maintains the subscription request until the request's expiration time is reached. Presence Server might receive from watchers and client applications requests to refresh an existing subscription. In addition, the function of processing subscription to a group of presentities involves resolving and creating virtual subscription requests for each presentity on behalf of the subscriber. All of the above functions utilize data store as a persistence mechanism and messaging resources extensively.

From the requirements of Presence Server, in terms of database utilization and messaging resources utilization, performance tuning parameters to database and underlying messaging infrastructure is essential. To ensure that the Presence serving environment is handling the requests efficiently, configure the overload settings and performance parameters by referring to Appendix A.3.2, “Tuning the DB2 Database” on page 460.

12.1 Overview of the scalable scenario test cases

The scenario test cases for the scalability environment contain successive runs of scenario test scripts that generate substantial traffic. The goal is to validate that all of the server instances in the environment receive and process the client requests in a consistent manner. Some of the reference points that we use for validation later in this section are:

- The weights of all of the Presence Cluster members are identical. This setting ensures that the traffic that the SIP Proxy Servers fronting Presence Cluster forwards are evenly distributed.
- The weights of all of the Shared List Cluster members and Presence Rules Cluster members are identical, which ensures that the dialogs between Presence Server reaching Shared List and Presence Rules servers are evenly distributed for every SIP PUBLISH and SUBSCRIBE client request.
- The method employed to analyze the exact number of requests that each server processes is based on searching for SIP Dialog IDs.
- The scripts to create users in the Shared List server and Rules server to conduct SIPp load tests are in Appendix B, "Additional material" on page 479.

12.2 Executing the scalable scenario test cases

The scalability scenario test case is an extension to the base scenario test case. In this test, the SIPp script prepares scenario instances for 200 distinct users and submits a SIP SUBSCRIBE for every user's buddy list or group to the Load balancer that fronts the Presence Server environment at a call rate of one second. Post sending the SIP SUBSCRIBE, the script generates SIP PUBLISH for each of the five users in the group. A separate SIPp script is responsible for collecting the SIP NOTIFY that the Presence Server continuously returns. In the case of a load test that has 200 user scenario instances, the total number of SIP SUBSCRIBE and SIP PUBLISH requests that Presence Server services are shown in Figure 12-1 on page 440. As part of processing the above requests, the number of SIP SUBSCRIBE requests that are submitted to the Shared List server and Rules server are depicted in Figure 12-1 on page 440.

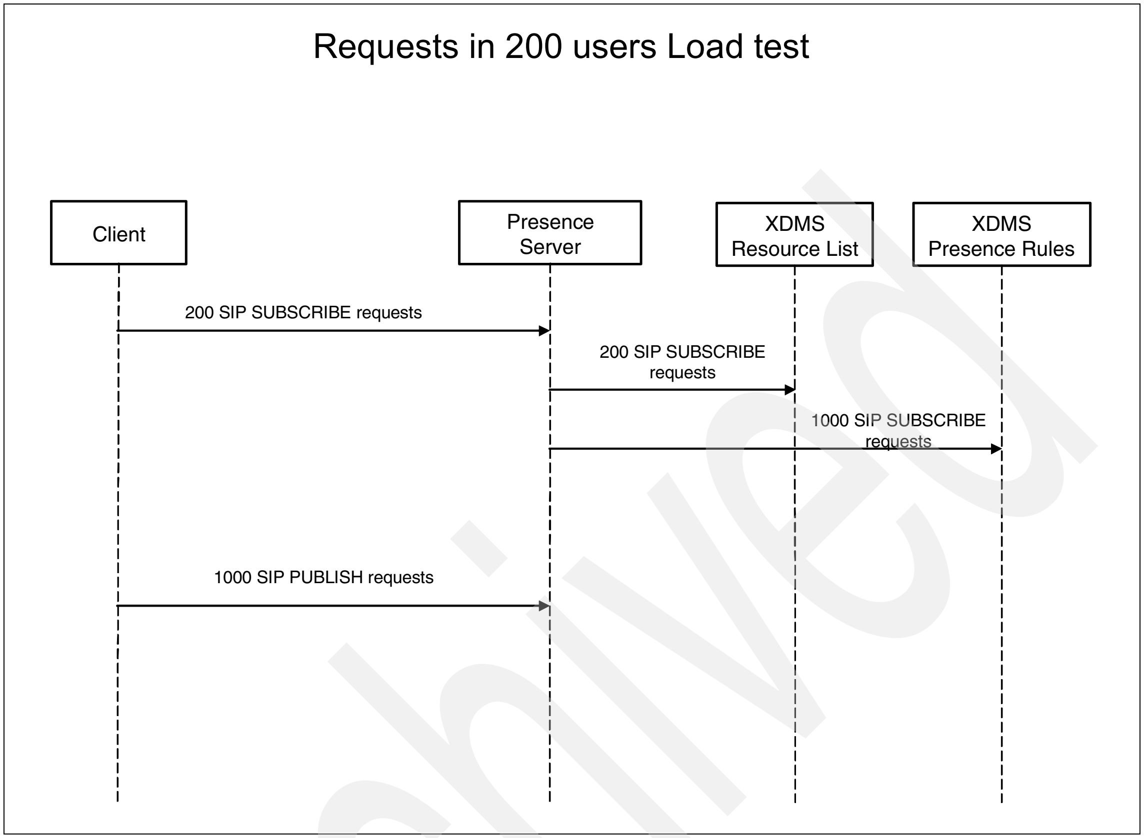


Figure 12-1 Number of requests serviced for a 200 user run

Post completion of the test execution, we provide the analysis for the following loads, based on the requests processed:

- Load distribution across Presence Cluster servers.
- Load distribution across XDMS Cluster Servers, namely the Shared List servers and Presence Rules Servers.

12.2.1 Load distribution in Presence Cluster

The basis of determining load distribution in Presence Cluster can be best described by understanding the actual sequence of messages that the SIPp scenario script generates and submits and corresponding messages that the Presence Cluster Servers serve. This analysis can be done by applying *'regular expression'* searches to the logs that are generated after a scenario test execution. Example 12-1 is a sample SIP SUBSCRIBE message that the script generates.

Example 12-1 Initial SIP SUBSCRIBE from a SIP User Agent

```
SUBSCRIBE sip:alice11friends@9.42.170.211:5060;transport=TCP SIP/2.0
Max-Forwards: 70
Via: SIP/2.0/TCP 9.42.170.229:5060
From: <sip:alice11@itso.ral.ibm.com>;tag=11
To: <sip:alice11friends@itso.ral.ibm.com>
Call-ID: 1-14690@9.42.170.229
CSeq: 1 SUBSCRIBE
Expires: 120
Contact: sip:alice11@9.42.170.229:5070;transport=TCP
P-Asserted-Identity: <sip:alice11@itso.ral.ibm.com>
Event: presence
Content-Length: 0
Accept: application/pidf+xml
Accept: application/rlmi+xml
Accept: multipart/related
Accept: multipart/signed
Accept: application/pkcs7-mime
Supported:
eventlist
```

The corresponding SIP response message for the above request is a ‘200 Ok’, which is illustrated in Example 12-2.

Example 12-2 Success response with dialog id as received by User Agent

```
SIP/2.0 200 OK
Require: eventlist
Call-ID: 1-14690@9.42.170.229
CSeq: 1 SUBSCRIBE
From: <sip:alice11@itso.ral.ibm.com>;tag=11
To:
<sip:alice11friends@itso.ral.ibm.com>;tag=020782146768034915_PresSrv01.
1193064061388.0_19_17
Via: SIP/2.0/TCP 9.42.170.229:5060
Contact: <sip:9.42.170.211:5060;transport=tcp>
Expires: 120
Content-Length:
0
```

From Example 12-2 on page 441, the tag that is associated with the To: header in the message, comprises of a unique transaction ID and a SIP dialog ID that the SIP container assigns. The dialog ID in Example 12-2 on page 441 has the signature of the server that handled the request 'PresSrv01'. From this test case point-of-view, the SIPp logs are searched using 'regular expressions' and analyzed for server signatures of each request type, for instance SIP SUBSCRIBE, SIP PUBLISH, and so on. Figure 12-2 gives the percentage of SIP SUBSCRIBE requests served by each server in the Presence Server Cluster.

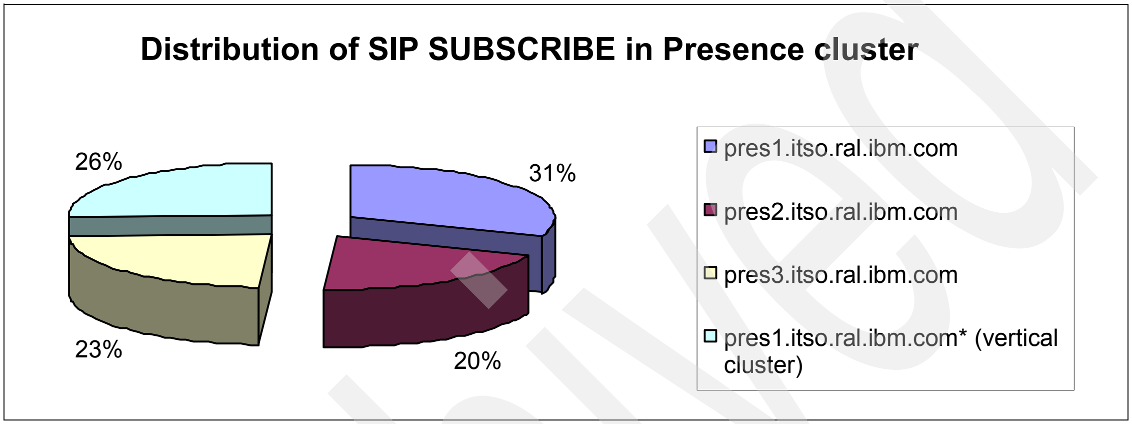


Figure 12-2 Percentage of SIP SUBSCRIBE requests handled by Presence Servers

Following the scenario test case, the script generates SIP PUBLISH requests for five users in the user group. The distribution of requests can vary from the SIP SUBSCRIBE as these are processed as independent SIP sessions. The method of analyzing the request distribution still remains same. Figure 12-3 on page 443, illustrates the SIP PUBLISH distribution in the Presence Server environment.

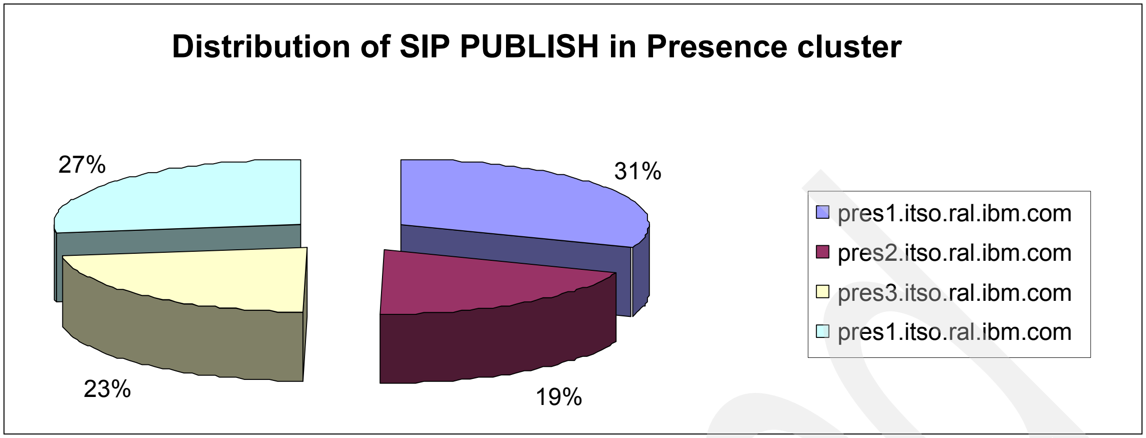


Figure 12-3 Percentage of SIP PUBLISH requests handled by Presence Servers

12.2.2 Load distribution in the XDMS Cluster

For every SIP SUBSCRIBE that a SIP User Agent submits to Presence Server, the processing in Presence Server involves generating a sequence of SIP and XCAP requests that are targeted to the Shared List server and Presence Rules Server respectively. In the context of the scenario test case, the sequence of requests that the Presence Server generates are:

1. SIP SUBSCRIBE to Shared List servers: Post receipt of a SIP SUBSCRIBE for a user group, Presence Server generates a SIP SUBSCRIBE with a specific Event package of type ‘ua-profile’ that is targeted to the Shared List Server to be notified about the group URI data. With a successful processing, the Shared List Server generates a SIP NOTIFY with the Event package comprising of ‘auid’ values ‘resource-lists’ and payload with document URI of the group. The next step in the processing involves retrieving the ‘resource-list’ document that contains the list of URIs of users, by submitting an XCAP request to the Shared List Server. The user list thus obtained is used in further processing of subscription by the Presence Server.
2. SIP SUBSCRIBE to Presence Rules servers: As part of request processing to a SIP SUBSCRIBE, Presence Server checks authorization rules of each individual presentity in the group. This check determines whether the subscriber is allowed to view presence information of a presentity or not. After evaluating the rules of all the users in the group, presence information of all the presentities pertaining to the user group is aggregated and composed into a single document. This processing involves generating a SIP SUBSCRIBE with a specific ‘Event’ package of type ‘ua-profile’ with ‘auid’ value ‘org.openmobilealliance.pres-rules’ along with a document URI, which is obtained from the group URI data received in the earlier step. The

corresponding SIP NOTIFY from the Presence Rules Server comprises of the authorization rules document URI in the payload of the message. The Presence Server uses this information to retrieve authorization rules of presentities by submitting an XCAP request to the Presence Rules Server for each presentity in the group.

Example 12-3 is a sample SIP SUBSCRIBE message that originated from the Presence Server and destined to the Shared List Server.

Example 12-3 SIP SUBSCRIBE from Presence Server to Shared List Server

```
CSeq: 2 SUBSCRIBE
To: <sip:alice11friends@itso.ral.ibm.com>
Event:ua-profile;profile-type="application";vendor="IBM";model="PresenceServer";versi
on="1"
Max-Forwards: 70
Expires: 3780
Call-ID: 372262659881O519@9.42.170.214
From:<sip:superadmin@itso.ral.ibm.com>;tag=5453366822970352_PresSrv01.1193064061388.0
_20_18
P-Asserted-Identity: <sip:superadmin@itso.ral.ibm.com>
Accept: application/xcap-diff+xml
```

Example 12-4 illustrates the corresponding SIP NOTIFY from the Shared List Server to the Presence Server.

Example 12-4 SIP NOTIFY from Shared List Server as received by Presence Server

```
CSeq: 2 NOTIFY
To:<sip:superadmin@itso.ral.ibm.com>;tag=5453366822970352_PresSrv01.1193064061388.0_2
0_18
Event:ua-profile;auid=resource-lists;profile-type=application;document=resource-lists
/users/alice11%40itso.ral.ibm.com/alice11friends.xml;model=PresenceServer;version=1;v
endor=IBM
Content-Length: 283
Max-Forwards: 68
Content-Type: application/xcap-diff+xml
Via: SIP/2.0/UDP
prox2.itso.ral.ibm.com:32842;branch=z9hG4bK2ab5430e3f5eaa89cae53ef032f5ee44;received=
9.42.170.213
From:<sip:alice11friends@itso.ral.ibm.com>;tag=47493068674444683_ListSrv03.1192922136
555.0_254_253
Call-ID: 3722626598810519@9.42.170.214
Subscription-State: active;expires=3780
Contact: <sip:9.42.170.225:5060;transport=udp>
l
```

From the load test execution of scenario test cases and analysis of logs in the Presence Server environment, Figure 12-4 shows the distribution SIP SUBSCRIBE requests that the Shared List Cluster servers handle to process the request to resolve resource-lists document URI of the group that is being subscribed.

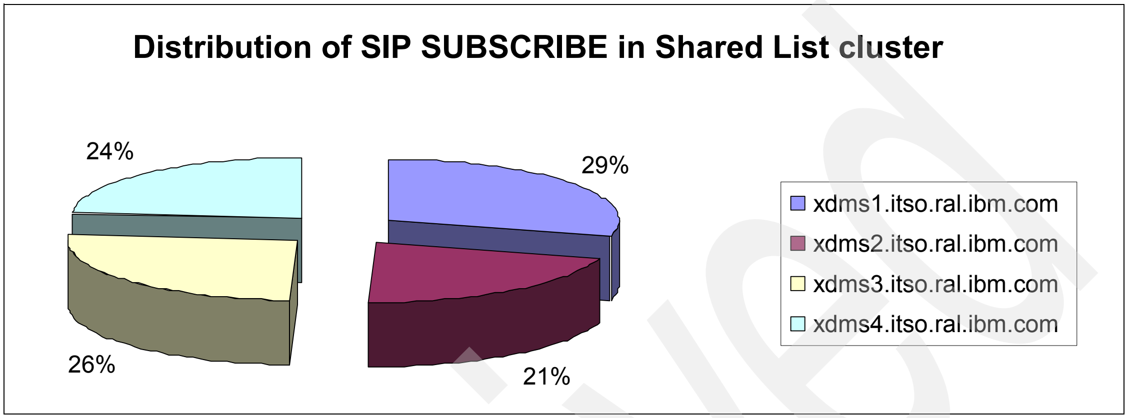


Figure 12-4 Percentage of SIP SUBSCRIBE requests handled by Shared List servers

Following the Presence Server processing sequence in managing the subscription to a group, Example 12-5 illustrates a sample SIP SUBSCRIBE from the Presence Server to the Presence Rules Server.

Example 12-5 SIP SUBSCRIBE from Presence Server to Presence Rules Server

```
CSeq: 2 SUBSCRIBE
To: <sip:bill11@itso.ral.ibm.com>
Event:ua-profile;profile-type="application";vendor="IBM";model="PresenceServer";versi
on="1";auid="org.openmobilealliance.pres-rules";document="users/bill11@itso.ral.ibm.c
om/pres-rules"
Max-Forwards: 70
Expires: 3780
Call-ID: 2144096418415397709.42.170.214
From:<sip:superadmin@itso.ral.ibm.com>;tag=23224830801727814_PresSrv01.1193064061388.
0_21_19
P-Asserted-Identity: <sip:superadmin@itso.ral.ibm.com>
Accept: application/xcap-diff+xml
```

The corresponding SIP NOTIFY from the Presence Rules Server to the Presence Server is shown in Example 12-6 on page 446.

Example 12-6 SIP NOTIFY from the Presence Rules Server as received by Presence Server

```
CSeq: 2 NOTIFY
To:<sip:superadmin@itso.ral.ibm.com>;tag=23224830801727814_PresSrv01.1193064061388.0_
21_19
Event:ua-profile;auid=org.openmobilealliance.pres-rules;profile-type=application;docu
ment=org.openmobilealliance.pres-rules/users/bill11%40itso.ral.ibm.com/pres-rules;mod
el=PresenceServer;version=1;vendor=IBM
Content-Length: 293
Max-Forwards: 68
Content-Type: application/xcap-diff+xml
Via: SIP/2.0/UDP
prox2.itso.ral.ibm.com:32842;branch=z9hG4bKcd2c0625238e4d4c4117a368d17aaebf;received=
9.42.170.213
From:<sip:bill11@itso.ral.ibm.com>;tag=35173117090612305_RuleSrv04.1192912283968.0_13
22_1321
Call-ID: 21440964184153977@9.42.170.214
Subscription-State: active;expires=3780
Contact: <sip:9.42.170.225:5060;transport=udp>
```

Post execution of the load tests based on the scenario test cases and analyzing the server logs on the Presence Server, a view of a load distribution is depicted in Figure 12-5.

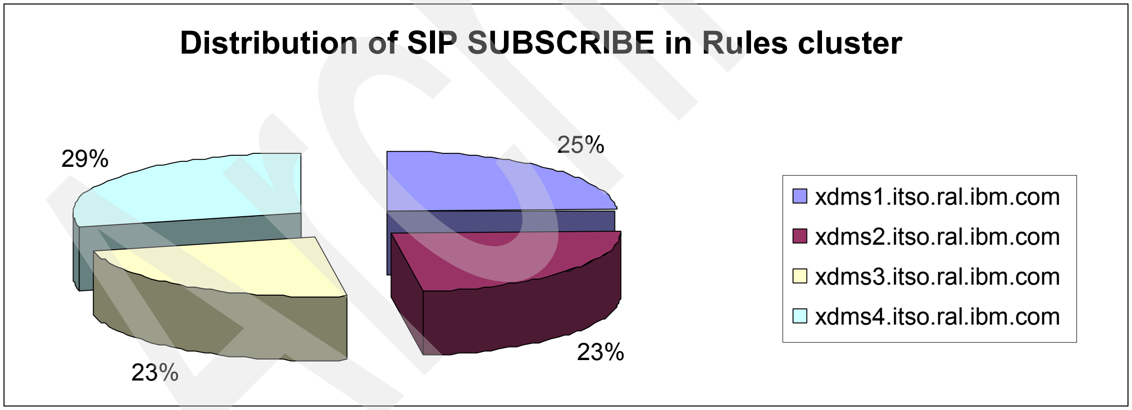


Figure 12-5 Percentage of SIP SUBSCRIBE requests handled by Rules servers

12.2.3 Possible errors and causes in Presence Server environment

In this section, we discuss some of the erroneous situations experienced in the highly-available environment during the execution of scenario test cases. Most of these errors surface when the servers are subject to near overload conditions.

Although the scenarios focus on high availability and scalability of Presence Server and XML Document Management Servers (XDMS), some of the reasons and possible remedies point to components of the infrastructure that are outside of the scope of this book.

Possible errors and causes are:

- Erroneous responses from Presence Server due to latency in data access: Under overload conditions, in the event of handling a SIP PUBLISH, Presence Server often responds with unexpected results. One of the reasons for this behavior leads to the inability of Presence Server to complete a database call in a set period of time. This might result in SIP requests expiring and eventually, Presence Server responds with an unexpected response, such as, a `412 Bad Request`. Much of the Presence Server functionality relies on the underlying database. Data and storage organization is an essential step in deployment architecture. For a check list of tuning parameters that are specific to the DB2 database, refer to Appendix A.3.2, "Tuning the DB2 Database" on page 460.
- Failure to process requests in the event of an abrupt shutdown of the server instance in the cluster: In the event of an unexpected shutdown of one of the redundant Presence Servers in cluster, often this problem can be experienced. One of the probable reasons is that the SIP session was not established between the client application and the SIP container before the recipient server process is terminated. As a result of this behavior, the client application does not receive any response from Presence Server. If this situation occurs frequently, consider applying performance tuning and overload settings to the SIP container and relevant Application Server resources. For a check list of tuning parameters that are specific to the SIP container, refer to Appendix A.3.2, "Tuning the DB2 Database" on page 460.

Part 5

Appendixes

A

Performance, monitoring, and tuning

In this appendix, we provide some performance tips and tuning configuration procedures for the IBM WebSphere IMS Component.

This appendix contains the following:

A.1 Performance

To ensure that IMS can perform under load conditions, you must properly configure Linux, the DB2 Database, and the WebSphere Application Servers where IMS is deployed. Follow the instructions from the Tuning section to obtain better performance for the IBM WebSphere IMS products.

A.2 Monitoring

In this section, we describe how to configure the Performance Monitoring Infrastructure (PMI) for IBM WebSphere components.

A.2.1 Before you begin

The Performance Monitoring Infrastructure uses a client-server architecture. The server collects performance data from various WebSphere Application Server components. A client retrieves performance data from one or more servers and processes the data.

In the WebSphere Application Server, version 6 and later, PMI counters are enabled, based on a monitoring or instrumentation level. The levels are None, Basic, Extended, All, and Custom. These levels are specified in the PMI module XML file. Enabling the module at a given level includes all of the counters at the given level, plus counters from levels below the given level. So, enabling the module at the extended level enables all of the counters at that level and all of the basic level counters as well.

We strongly recommend that PMI is tightly scoped. PMI should be enabled before the server starts. If PMI is enabled after the server is started, the server needs to be restarted to start the PMI.

A.2.2 Enabling PMI

To enable PMI:

1. Open the administrative console, and in the console navigation tree, click **Servers** → **Application Servers**, and click a server.
2. Click the **Configuration** tab.
3. Under Performance, click **Performance Monitoring Infrastructure (PMI)**.
4. Select the **Enable Performance Monitoring Infrastructure (PMI)** option.

5. Optionally, select the **Use sequential counter updates** option to enable precise statistic update.
6. Optionally, under the Currently Monitored Statistic Set section, choose a statistic set that needs to be monitored.
7. Optionally, click **Custom** to selectively enable or disable statistics. From the left side of the tree, choose a component, and enable or disable statistics that are relevant to your business.
8. Click **Apply** or **OK**.
9. Click **Save**.
10. Restart the Application Server. The changes you make do not take effect until you restart the Application Server.

For more information about PMI, reference:

http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.zseries.doc/info/zseries/ae/cprf_pmi_arch.html?resultof=%22%50%4d%49%22%

A.2.3 Presence Server monitoring

WebSphere Presence Server provides Key Performance Indicators (KPI) in the form of Performance MBean integration using the WebSphere Application Server Performance Monitor Infrastructure. The indicators can also be written to trace files and to be viewed with the Presence Server Monitor.

The following list contains the complete list of Presence Server PMI counters to enable:

- SIP_PROVIDER.SUBSCRIPTION.REQUEST: "XPROVIDER SUBSCRIBE rate"
- UN.PUBLISH.REQUEST: "Un-PUBLISH rate"
- MODIFY.PUBLISH.REQUEST: "Modify PUBLISH rate"
- INITIAL.PUBLISH.REQUEST: "Initial PUBLISH rate"
- UN.URI.SUBSCRIPTION.REQUEST: "Uri List Un-SUBSCRIBEs rate"
- POLL.URI.SUBSCRIPTION.REQUEST: "Uri List Poll-SUBSCRIBEs rate"
- RE.URI.SUBSCRIPTION.REQUEST: "Uri List Re-SUBSCRIBEs rate"
- NEW.URI.SUBSCRIPTION.REQUEST: "Uri List New SUBSCRIBEs rate"
- POLL.GROUP.SUBSCRIPTION.REQUEST: "Group Poll-SUBSCRIBEs rate"
- UN.GROUP.SUBSCRIPTION.REQUEST: "Group Un-SUBSCRIBEs rate"

- RE.GROUP.SUBSCRIPTION.REQUEST: "Group Re-SUBSCRIBEs rate"
- NEW.GROUP.SUBSCRIPTION.REQUEST: "Group New SUBSCRIBEs rate"
- POLL.SINGLE.SUBSCRIPTION.REQUEST: "Single Poll-SUBSCRIBEs rate"
- UN.SINGLE.SUBSCRIPTION.REQUEST: "Single Un-SUBSCRIBEs rate"
- RE.SINGLE.SUBSCRIPTION.REQUEST: "Single Re-SUBSCRIBEs rate"
- NEW.SINGLE.SUBSCRIPTION.REQUEST: "Single New SUBSCRIBEs rate"
- RE.PUBLISH.REQUEST: "Re-PUBLISH rate"
- NOTIFY.GROUP.REQUEST: "Group NOTIFY rate"
- NOTIFY.SINGLE.REQUEST: "Single NOTIFY rate"
- NOTIFY.URI.REQUEST: "URI List NOTIFY rate"
- SUCCESSFUL.SUBSCRIPTION.RESPPONSE: "Successful SUBSCRIBE responses rate"
- SUCCESSFUL.PUBLISH.RESPPONSE: "Successful PUBLISH responses rate"
- SUCCESSFUL.NOTIFY.RESPPONSE: "Successful NOTIFY responses rate"
- FAILED.SUBSCRIPTION.RESPPONSE: "Failed SUBSCRIBE responses rate"
- FAILED.PUBLISH.RESPPONSE: "Failed PUBLISH responses rate"
- FAILED.NOTIFY.RESPPONSE: "Failed NOTIFY responses rate"
- SIP_PROVIDER.SUBSCRIPTION.REQUEST.PER.SEC: "XPROVIDER SUBSCRIBE request per second"
- UN.PUBLISH.REQUEST.PER.SEC: "Un-PUBLISH request per second"
- MODIFY.PUBLISH.REQUEST.PER.SEC: "Modify PUBLISH request per second"
- INITIAL.PUBLISH.REQUEST.PER.SEC: "Initial PUBLISH request per second"
- UN.URI.SUBSCRIPTION.REQUEST.PER.SEC: "Uri List Un-SUBSCRIBEs request per second"
- POLL.URI.SUBSCRIPTION.REQUEST.PER.SEC: "Uri List Poll-SUBSCRIBEs request per second"
- RE.URI.SUBSCRIPTION.REQUEST.PER.SEC: "Uri List Re-SUBSCRIBEs request per second"
- NEW.URI.SUBSCRIPTION.REQUEST.PER.SEC: "Uri List New SUBSCRIBEs request per second"

- POLL.GROUP.SUBSCRIPTION.REQUEST.PER.SEC: "Group Poll-SUBSCRIBEs request per second"
- UN.GROUP.SUBSCRIPTION.REQUEST.PER.SEC: "Group Un-SUBSCRIBEs request per second"
- RE.GROUP.SUBSCRIPTION.REQUEST.PER.SEC: "Group Re-SUBSCRIBEs request per second"
- NEW.GROUP.SUBSCRIPTION.REQUEST.PER.SEC: "Group New SUBSCRIBEs request per second"
- POLL.SINGLE.SUBSCRIPTION.REQUEST.PER.SEC: "Single Poll-SUBSCRIBEs request per second"
- UN.SINGLE.SUBSCRIPTION.REQUEST.PER.SEC: "Single Un-SUBSCRIBEs request per second"
- RE.SINGLE.SUBSCRIPTION.REQUEST.PER.SEC: "Single Re-SUBSCRIBEs request per second"
- NEW.SINGLE.SUBSCRIPTION.REQUEST.PER.SEC: "Single New SUBSCRIBEs request per second"
- RE.PUBLISH.REQUEST.PER.SEC: "Re-PUBLISH request per second"
- NOTIFY.GROUP.REQUEST.PER.SEC: "Group NOTIFY request per second"
- NOTIFY.SINGLE.REQUEST.PER.SEC: "Single NOTIFY request per second"
- NOTIFY.URI.REQUEST.PER.SEC: "URI List NOTIFY request per second"
- SUCCESSFUL.SUBSCRIPTION.RESPPONSE.PER.SEC: "Successful SUBSCRIBE responses per second"
- SUCCESSFUL.PUBLISH.RESPPONSE.PER.SEC: "Successful PUBLISH responses per second"
- SUCCESSFUL.NOTIFY.RESPPONSE.PER.SEC: "Successful NOTIFY responses per second"
- FAILED.SUBSCRIPTION.RESPPONSE.PER.SEC: "Failed SUBSCRIBE responses per second"
- FAILED.PUBLISH.RESPPONSE.PER.SEC: "Failed PUBLISH responses per second"
- FAILED.NOTIFY.RESPPONSE.PER.SEC: "Failed NOTIFY responses per second"
- TOTAL.NOTIFY.SIP.REQUESTS: "Total number of sent SIP NOTIFY messages"
- TOTAL.SUBSCRIPTION.SIP.REQUESTS: "Total number of received SIP SUBSCRIBE messages"

- TOTAL.PUBLISH.SIP.REQUESTS: "Total number of received SIP PUBLISH messages"
- TOTAL.SIP_PROVIDERS.SUBSCRIPTION.OBJECTS: "Current number of outgoing XProvidres subscription objects"
- TOTAL.URI.SUBSCRIPTION.OBJECTS: "Current number of URI lists subscription objects"
- TOTAL.GROUP.SUBSCRIPTION.OBJECTS: "Current number of group subscription objects"
- TOTAL.SINGLE.SUBSCRIPTION.OBJECTS: "Current number of single subscription objects"
- TOTAL.FAILED.SUBSCRIPTION.RESPPONSE: "Total Failed SUBSCRIBE responses"
- TOTAL.FAILED.PUBLISH.RESPPONSE: "Total Failed PUBLISH responses"
- TOTAL.FAILED.NOTIFY.RESPPONSE: "Total Failed NOTIFY responses"
- TOTAL.CACHE.HITS: "Total Hits of Document Cache"
- TOTAL.CACHE.MISS: "Total Misses of Document Cache"

A.2.4 XDMS monitoring

XDMS provides Key Performance Indicators in the form of Performance MBean integration using the WebSphere Application Server Performance Monitor Infrastructure.

The following list contains the complete list of XDMS PMI counters to enable:

PUT Requests	Number of PUT requests
GET Requests	Number of GET requests
DELETE Requests	Number of DELETE requests
POST Requests	Number of POST requests
SUBSCRIBE Requests	Number of SUBSCRIBE requests
Success rate	Percentage of successful responses
PUT Latency	The average latency time for PUT requests
GET Latency	The average latency time for GET requests
DELETE Latency	The average latency time for DELETE requests
POST Latency	The average latency time for POST requests
SUBSCRIBE Latency	The average latency time for SUBSCRIBE requests

A.3 Tuning

In this section, we provide tuning parameter recommendations to improve performance and scalability for IBM WebSphere components.

A.3.1 Tuning the Linux operating system

To tune the Linux operating system

1. Tune the Linux operating system to optimize the performance of your WebSphere Application Server. Access the following Web site:

   http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.express.doc/info/exp/ae/tsip_tunelinux.html

2. Tune SIP Servlets for Linux.

   A Session Initiation Protocol (SIP) servlet under load might retransmit messages or drop calls. The UDP socket queues might fill. A review of the verbose garbage collection output might show that there are fairly long garbage collection times, for example, 0.5 to 1.5 seconds. The cause of this problem is that the Ethernet driver, Linux operating system, WebSphere Application Server, or any combination of the items are not tuned for SIP applications. Access the following Web site for more information:

   http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.express.doc/info/exp/ae/tsip_tunelinux.html?resultof=%22%6b%65%72%6e%65%6c%22%20

3. Change the clock selection from `pmtimer` to `tsc`:

   > **Note:** Intel® RHEL Linux earlier then 2.6.18 only.

   a. Edit the /etc/grub.conf file, and add `clock=tsc` to the /kernel line, as shown in Figure A-1 on page 458.

```
[root@perfsips15 ~] # cat /etc/ grub.conf
# grub.conf generated by anaconda
#
# Note that you do not have to rerun grub after making
     changes to this file
# NOTICE: You have a /boot partition. This means that
#          all kernel and initrd paths are relative to
     /boot/, eg.
#            root (hd0,0)
#            kernel /vmlinuz-version ro
     root=/dev/VolGroup00/LogVo100
#            initrd /initrd-version.img
#boot=/dev/sda
              default=0
timeout=5
splashimage=(hd0,0)/grub/splash.xpm.gz
hiddenmenu
title Red Hat Enterprise Linux AS (2.6.9-42.ELsmp)
root (hd0,0)
kernel /vmlinuz-2.6.9-42.ELsmp ro
     root=/dev/VolGroup00/LogVo100 rhgb quiet clock=tsc
initrd /initrd-2.6.9-42.ELsmp.img
title Red Hat Enterprise Linux AS-up (2.6.9-42.EL)
root (hd0,0)
kernel /vmlinuz-2.6.9-42.EL ro root=/dev/VolGroup00/Log'
```

Figure A-1 Add clock=tsc to the /kernel line

b. After you reboot, you can check to see that the change took effect by looking in the dmesg log, as shown in Figure A-2.

```
[root@perfsips15 ~] # cat /var/log/dmesg | grep tsc
Kernel command line: ro root=/dev/VolGroup00/LogVol
```

Figure A-2 The dmesg log

This configuration might not be optimum for a given application, and you might need to adjust the configuration to achieve the best performance; however, you can use these values as a starting point.

A.3.1.1 SIP for WebSphere Application Server

To tune SIP for WebSphere Application Server:

1. Create a separate thread pool for the SIP servlet container, using the following path in the administrative console:
   a. Click **Server** → **Application servers** → *server_name*.
   b. Under Additional properties, click **Thread Pools** → **New**.
   c. In the Name field, type `SipContainer`.
   d. In the Minimum Size and Maximum Size fields, enter `15`. These values should be adequate for most applications.
   e. Click **OK**.
2. Create custom properties for the SIP Servlet container using the following path in the administrative console:
   a. Click **Server** → **Application servers** → *server_name*.
   b. Click **SIP container**.
   c. Under Additional properties, click **Custom Properties** → **New**.
   d. In the Name field, type `javax.sip.max.object.pool.size`.
   e. In the Value field, enter `1000`.
   f. Click **OK**.
   g. In the Name field, enter `max.tu.pool.size`.
   h. In the Value field, enter `1000`.
   i. Click **OK**.
3. Create custom properties for the SIPUDP channel, if the User Datagram Protocol (UDP) is the primary transport for SIP traffic. Follow this path in the administrative console:
   a. Click **Server** → **Application servers** → *server_name*.
   b. Click **SIP container** → **Transport Chain** → **SIPCInboundDefaultUDP** → **UDP Inbound channel (UDP1)**.
   c. Under Additional Properties, click **Custom Properties** → **New**.
   d. In the Name field, type `receiveBufferSocketSize`.
   e. In the Value field, type `3000000`.
   f. Click **OK**.
   g. In the Name field, type `sendBufferSocketSize`.
   h. In the Value field, type `3000000`.
4. Specify the SIP servlet container general properties. Follow this path in the administrative console:
   a. Click **Servers** → **Application Servers** → **server_name** → **SIP container**.

   b. Enter the Maximum application sessions value.

      The Maximum application sessions value can be calculated as: Maximum call hold time or session time out x Call rate x Safety factor.

   c. Enter the Maximum messages per averaging period value.

      The Maximum messages per averaging period value can be calculated as: Maximum call hold time or session time out x Maximum rate of SIP messages x Safety factor.

   d. Enter the Maximum dispatch queue size value.

      The Maximum dispatch queue size value can be calculated as: Maximum rate of SIP messages x Maximum latency in SIP processing x Safety factor.

   e. Set the thread pool to the newly created SIP container thread pool (to the drop down name "SipContainer").

5. Tune the Java virtual machine (JVM) garbage collection policy. Follow this path in the administrative console:

   a. Click **Server** → **Application servers** → `server_name`.

   b. Under Server Infrastructure, click **Java and Process Management** → **Process Definition**.

   c. Under Additional Properties, click **Java Virtual Machine**.

   d. In the Generic JVM arguments field, enter the following value as one continuous line: `1"-Xgcpolicy:gencon -Xgc:scvNoAdaptiveTenure,scvTenureAge=8, stdGlobalCompactToSatisfyAllocate"`.

      Optional: You also might add a value of `768` to the Initial Heap Size and Maximum Heap Size fields. It is also a good practice to enable the **Verbose garbage collection** option during performance testing or tuning operations.

A.3.2 Tuning the DB2 Database

To ensure that IMS can perform under load conditions, you must properly configure the DB2 Database:

1. The organization of data on the disks is an important factor for benchmark performance. The DB2 Database must be suitably divided into separate RAID arrays and properly partitioned. To gain performance improvements, set the

log files for each database on a different hard drive from the database files. An example of an array configuration is 3 RAID arrays:

– a is one array disk for the OS
– b is three array disks for the transaction log
– c is two array disks for the database and tablespace

All arrays are partitioned to one partition.

a. Create RAID arrays using the RAID configuration tool. Set up as RAID 0. If redundancy is required, use RAID 10.

b. Partition arrays using the **fdisk** command:

   i. To add a partition, where <x> is the variable used to name the partition:

      fdisk /dev/sd<x> -a

   ii. To write out the partition, where <x> is the variable used to name the partition:

      fdisk /dev/sd<x> -w

      After partitioning is complete, there are entries in the /proc/partitions file.

c. Create the log files. For performance enhancement, use a different array than the database files:

   i. Make a file system:

      mkfs –t ext3 /dev/sdb1

   ii. Assign a label:

      e2label /dev/sdb1 /tranlog

   iii. Add a new file system line to /etc/fstab:

      mkdir /tranlog

   iv. /dev/sdb1 /tranlog ext3 defaults 1 2

   v. Reboot.

d. Create your databases. For performance enhancement, use a different array than the log files:

   i. Perform the following DB2 commands:

```
db2 "create database ${db_name} on '/db/sib'"

db2 "UPDATE DB CFG FOR ${db_name} USING
     NEWLOGPATH /tranlog/sib"

db2stop force

db2start
```

2. DB2 Asynchronous I/O (AIO) for Linux allows processes to perform other work while waiting for an I/O request to return. DB2 page cleaners can make use of AIO to more effectively use system resources:

   a. Enable AIO on Linux by setting the DB2 registry variable DB2LINUXAIO to db2set DB2LINUXAIO=true

   For more information about DB2 AIO on Linux, refer to:

   `http://www.ibm.com/developerworks/db2/library/techarticle/dm-0509wright/`

3. DB2 buffer pools and the Linux file cache perform largely the same function, to cache a copy of data that is read from disk. The result is that system cycles are used to make the copy and maintain two areas of memory that largely perform the same function. Direct I/O solves this issue by directing reads to be made directly from disk into DB2 memory areas, bypassing the file cache entirely. This feature is available on a tablespace-by-tablespace basis, which effectively allows some data in DB2 to use file caching and other data to bypass it. The tablespace option NO FILE SYSTEM CACHING can be specified during the creation of the tablespace or altered to include this option.

   a. Create or alter your tablespace with the NO FILE SYSTEM CACHING option:

      i. Perform the following DB2 commands to create the tablespace:

         CONNECT TO <DB>;

         CREATE REGULAR TABLESPACE <TABLESPACE_NAME>

         MANAGED BY SYSTEM

         USING ('/data/')

         NO FILE SYSTEM CACHING;

      ii. Perform the following to alter the tablespace:

         CONNECT TO <DB>;

         ALTER TABLESPACE <TABLESPACE>

         NO FILE SYSTEM CACHING;

      For more information about DB2 Direct I/O on Linux, refer to:

      `http://www.ibm.com/developerworks/db2/library/techarticle/dm-0509wright/`

A.3.3 Tuning DB2 for XDMS

To ensure that XDMS can perform under load conditions, you must configure the DB2 server:

1. Update and run the XDMSDBcreate.sh script. The script is located at /imscenter/IMS/XDMS/dbscripts:
   a. Make the highlighted changes in Figure A-3 on page 464 to the script.
   b. Run the script on the database server using the following command:

```
./XDMSDBcreate.sh <db_name (xdms)> <db_user (db2inst1)>
<db_password (password)>
```

```
#!/bin/sh
#   begin_generated_IBM_prolog
#
#   prolog.pro.ocol.telcosp
#*********************************************************************
#   IBM Licensed Source
#
#   OCO Source Materials
#
#   Product(s): 5724-O05
#
#   Copyright IBM Corp. 2007, 2007
#
#   The source code for this program is restricted to the terms
#   specified by the End User License Agreement (EULA) that
#   accompanied this product.
#*********************************************************************
#
#   end_generated_IBM_prolog
.
.

# Create the DB
db2 "create database ${db_name} on '/db/db2' USING CODESET UTF-8 TERRITORY US
COLLATE USING SYSTEM DFT_EXTENT_SZ 64"
.
.

#db2 "create bufferpool BP_4K all nodes size 2000 AUTOMATIC pagesize 4K"
#db2 "create bufferpool BP_4K all nodes size 2000 AUTOMATIC pagesize 4K"
db2 "create bufferpool BP_32K all DBPARTITIONNUMS size 2000 AUTOMATIC pagesize 32K"
db2 "create bufferpool BP_4K all DBPARTITIONNUMS size 2000 AUTOMATIC pagesize 4K"

db2 "create tablespace IDTS pagesize 4K managed by database USING (FILE 'IDTSFILE'
1000000) EXTENTSIZE 32 OVERHEAD 24.1 PREFETCHSIZE 16 TRANSFERRATE 0.9 bufferpool
BP_4K NO FILE SYSTEM CACHING dropped table recovery off"
db2 "create tablespace DOCUMENTSTS pagesize 32K managed by database USING (FILE
'DOCSFILE' 125000) EXTENTSIZE 32 OVERHEAD 24.1 PREFETCHSIZE 16 TRANSFERRATE 0.9
bufferpool BP_32K NO FILE SYSTEM CACHING dropped table recovery off"
.
.

db2stop force
db2start
```

Figure A-3 Changing the XDMSDBcreate.sh script

2. Update and run the PRXDMSDBcreate.sh script. The script is located at /imscenter/IMS/XDMS/dbscripts:

   a. Make the highlighted changes in Figure A-4 on page 465 to the script.

b. Run the script on the database server using the following command:

```
./PRXDMSDBcreate.sh <db_name (prxdms)> <db_user (db2inst1)>
<db_password (password)>
```

```
#!/bin/sh
#   begin_generated_IBM_prolog
#
#   prolog.pro.ocol.telcosp
#*******************************************************************
#   IBM Licensed Source
#
#   OCO Source Materials
#
#   Product(s): 5724-O05
#
#   Copyright IBM Corp. 2007, 2007
#
#   The source code for this program is restricted to the terms
#   specified by the End User License Agreement (EULA) that
#   accompanied this product.
#*******************************************************************
#
#   end_generated_IBM_prolog
.
.

# Create the DB
db2 "create database ${db_name} on '/db/db2' USING CODESET UTF-8 TERRITORY US
COLLATE USING SYSTEM DFT_EXTENT_SZ 64"
.
.

#db2 "create bufferpool BP_4K all nodes size 2000 AUTOMATIC pagesize 4K"
#db2 "create bufferpool BP_4K all nodes size 2000 AUTOMATIC pagesize 4K"
db2 "create bufferpool BP_32K all DBPARTITIONNUMS size 2000 AUTOMATIC pagesize 32K"
db2 "create bufferpool BP_4K all DBPARTITIONNUMS size 2000 AUTOMATIC pagesize 4K"

db2 "create tablespace IDTS pagesize 4K managed by database USING (FILE 'IDTSFILE'
1000000) EXTENTSIZE 32 OVERHEAD 24.1 PREFETCHSIZE 16 TRANSFERRATE 0.9 bufferpool
BP_4K NO FILE SYSTEM CACHING dropped table recovery off"
db2 "create tablespace DOCUMENTSTS pagesize 32K managed by database USING (FILE
'DOCSFILE' 125000) EXTENTSIZE 32 OVERHEAD 24.1 PREFETCHSIZE 16 TRANSFERRATE 0.9
bufferpool BP_32K NO FILE SYSTEM CACHING dropped table recovery off"
.
.

db2stop force
db2start
```

Figure A-4 Changing the PRXDMSDBcreate.sh script

3. Update and run the SIBDBcreate.sh script. The script is located at /imscenter/IMS/XDMS/dbscripts:
   a. Make the highlighted changes in Figure A-5 to the script.
   b. Run the script on the database server using the following command:

      ./SIBDBcreate.sh <db_name (xdmssib)> <db_user (db2inst1)> <db_password (password)>

```
#!/bin/sh
#   begin_generated_IBM_prolog
#
#   prolog.pro.ocol.telcosp
#*********************************************************************
#   IBM Licensed Source
#
#   OCO Source Materials
#
#   Product(s): 5724-O05
#
#   Copyright IBM Corp. 2007, 2007
#
#   The source code for this program is restricted to the terms
#   specified by the End User License Agreement (EULA) that
#   accompanied this product.
#*********************************************************************
#
#   end_generated_IBM_prolog
.
.


# Create the DB
db2 "create database ${db_name} on '/db/db2'"
#db2 "UPDATE DB CFG FOR ${db_name} USING NEWLOGPATH /tranlog/sib"
'
'

db2stop force
db2start
```

Figure A-5 Change the SIBDBcreate.sh script

4. Update and run the URDBupdate.sh script. The script is located at /imscenter/IMS/XDMS/dbscripts:
   a. Make the highlighted changes to the script
   b. Run the script on the database server using the following command:

      ./URDBupdate.sh <db_name (xdmurdb)>

5. Update and run the createXdmsSIBus.sh script. On the Application Server, the script is located at: <WAS_HOME>/AppServer/installableApps/xdms/scripts/dbScripts/xdms/ createXdmsSIBus.sh:
   a. Move createXdmsSIBus.sh from the Application Server to the database server.
   b. Make the highlighted changes to the script.
   c. Run the script on the database server, as shown in Figure A-6 on page 468.

```
#!/bin/sh
#   begin_generated_IBM_prolog
#
#   prolog.pro.ocol.telcosp
#*********************************************************************
#   IBM Licensed Source
#
#   OCO Source Materials
#
#   Product(s): 5724-O05
#
#   Copyright IBM Corp. 2007, 2007
#
#   The source code for this program is restricted to the terms
#   specified by the End User License Agreement (EULA) that
#   accompanied this product.
#*********************************************************************
#
#   end_generated_IBM_prolog

# Usage function
usage ( ) (
        echo "Usage:"
        echo "    $0 <db_name> <db_user> <db_password>"
        echo ""
        echo "Example:"
        echo "    $0 xdmssibus db2inst1 dbpassword"
        echo ""
)

# Check the variables
if [ $# != 3 ]
then
        usage
        exit -1
fi

# Variables
db_name=$1
db_user=$2
db_password=$3
```

Figure A-6   createXdmsSIBus.sh script

6. Update and run the IBMSharedListXdms.sh script. On the Application Server, the script is located at: <WAS_HOME>/AppServer/installableApps/xdms/scripts/dbScripts/xdms/IBMSharedListXdms.sh
   a. Move the IBMSharedListXdms.sh from the Application Server to the database server.
   b. Make the changes in Figure A-7 on page 469 to the script.

```
.

# Create the DB
db2 "create database ${db_name} on '/db' USING CODESET UTF-8 TERRITORY US COLLATE
USING SYSTEM DFT_EXTENT_SZ 64"
#DBM config updates
db2 "UPDATE DATABASE MANAGER CONFIGURATION USING MAXAGENTS 1000"

.
.
.

db2 "UPDATE DATABASE CONFIGURATION FOR ${db_name} USING AUTO_RUNSTATS ON"

db2 "UPDATE DB CFG FOR xdmssl USING NEWLOGPATH /tranlog"

# Create the bufferpool, tablespace, and table
db2 "CONNECT TO ${db_name} USER ${db_user} USING ${db_password}"
.
.
.
.

db2 "create tablespace IDTS pagesize 4K managed by database USING (FILE 'IDTSFILE'
1000000) EXTENTSIZE 32 OVERHEAD 24.1 PREFETCHSIZE 16 TRANSFERRATE 0.9 bufferpool
BP_4K NO FILE SYSTEM CACHING dropped table recovery off"
db2 "create tablespace DOCUMENTSTS pagesize 32K managed by database USING (FILE
'DOCSFILE' 125000) EXTENTSIZE 32 OVERHEAD 24.1 PREFETCHSIZE 16 TRANSFERRATE 0.9
bufferpool BP_32K NO FILE SYSTEM CACHING dropped table recovery off"
.
```

Figure A-7 Changing the IBMSharedListXdms.sh script

7. Update and run the IBMPresenceRulesXdms.sh script. On the Application Server, the script is located at:
<WAS_HOME>/AppServer/installableApps/xdms/scripts/dbScripts/xdms/IBMSharedListXdms.sh

   a. Move the IBMPresenceRulesXdms.sh from the Application Server to the database server.

   b. Make the changes in Figure A-8 on page 470 to the script.

```
db2 "create database ${db_name} on '/db' USING CODESET UTF-8 TERRITORY US COLLATE
USING SYSTEM DFT_EXTENT_SZ 64"
#DBM config updates
db2 "UPDATE DATABASE MANAGER CONFIGURATION USING MAXAGENTS 1000"

.
.
.

db2 "UPDATE DATABASE CONFIGURATION FOR ${db_name} USING AUTO_RUNSTATS ON"

db2 "UPDATE DB CFG FOR xdmssl USING NEWLOGPATH /tranlog"

# Create the bufferpool, tablespace, and table
db2 "CONNECT TO ${db_name} USER ${db_user} USING ${db_password}"
.
.
.
.

db2 "create tablespace IDTS pagesize 4K managed by database USING (FILE 'IDTSFILE'
1000000) EXTENTSIZE 32 OVERHEAD 24.1 PREFETCHSIZE 16 TRANSFERRATE 0.9 bufferpool
BP_4K NO FILE SYSTEM CACHING dropped table recovery off"
db2 "create tablespace DOCUMENTSTS pagesize 32K managed by database USING (FILE
'DOCSFILE' 125000) EXTENTSIZE 32 OVERHEAD 24.1 PREFETCHSIZE 16 TRANSFERRATE 0.9
bufferpool BP_32K NO FILE SYSTEM CACHING dropped table recovery off"
.
```

Figure A-8 Changing the IBMPresenceRulesXdms.sh script

8. Configure the Data Sources.

   Data sources allow you to manage a pool of connections to a database. Using connection pools provides you with many advantages. It improves performance. Creating connections is expensive. A data source creates a connection as soon as it is instantiated. It simplifies resource allocation. Resources are only allocated from the data sources, and not at arbitrary places in the code.

   For more information about Data Sources:

   http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.ws.ast.st.common.ui.doc/topics/cwdatsrc.html?resultof=%22%64%61%74%61%22%20%22%73%6f%75%72%63%65%73%22%20%22%73%6f%75%72%63%22%20

   a. In the navigation panel, click **Resources** → **JDBC** → **Data Sources**.
   b. Select the JDBC provider that the XDMS Server is using.
   c. Under Additional Properties, click **Connection pool properties**.
   d. In the Maximum connections field, type `50`.
   e. Click **OK**, and click **Save** to save changes to the master configuration.

A.3.4 Tuning DB2 for the Presence Server

To tune DB2 for the Presence Server:

1. Change the following parameters at the database level if you want to turn on the Usage Record database. Be careful changing these because some of them rely on the amount of memory that the DBHEAP variable allocates:
   - UPDATE DATABASE CONFIGURATION FOR dbname USING DBHEAP 8192 - This is one of the main shared memory tunables.
   - UPDATE DATABASE CONFIGURATION FOR dbname USING LOGBUFSZ 2048 - Increases the size of the log buffer.
   - UPDATE DATABASE CONFIGURATION FOR dbname USING APP_CTL_HEAP_SZ 4096
   - UPDATE DATABASE CONFIGURATION FOR dbname USING SORTHEAP 512 - Increases the size of the sort area.
   - UPDATE DATABASE CONFIGURATION FOR dbname USING APPLHEAPSZ 2048
   - UPDATE DATABASE CONFIGURATION FOR dbname USING NUM_IOCLEANERS 3
   - UPDATE DATABASE CONFIGURATION FOR dbname USING LOGFILSIZ 5000 - Increases the size of the log files that are created.
   - UPDATE DATABASE CONFIGURATION FOR dbname USING LOGPRIMARY 20 - Increases the number of primary log files that are created.
   - UPDATE DATABASE CONFIGURATION FOR dbname USING LOGSECOND 20 - Increases the number of secondary log files that are created.
   - UPDATE DATABASE CONFIGURATION FOR dbname USING AUTO_MAINT on
   - UPDATE DATABASE CONFIGURATION FOR dbname USING AUTO_TBL_MAINT on
   - UPDATE DATABASE CONFIGURATION FOR dbname USING AUTO_RUNSTATS on
   - UPDATE DATABASE CONFIGURATION FOR dbname USING MAXLOCKS 80Allows for the percentage of locks to be higher.
   - UPDATE DATABASE CONFIGURATION FOR dbname USING LOCKTIMEOUT 60
   - UPDATE DATABASE CONFIGURATION FOR dbname USING LOCKLIST 10000 - Allows for more locks to occur.

– UPDATE DATABASE CONFIGURATION FOR dbname USING MINCOMMIT 1 - Increases the number of applications to be connected.
– UPDATE DATABASE CONFIGURATION FOR dbname USING NUM_IOCLEANERS 11
– UPDATE DATABASE CONFIGURATION FOR dbname USING NUM_IOSERVERS 10

2. Create and run following script in Figure A-9 from a db2 prompt:

```
"update db cfg for $dbAlias using dbheap 8192"
"update db cfg for $dbAlias using logbufsz 2048"
"update db cfg for $dbAlias using app_ctl_heap_sz 4096"
"update db cfg for $dbAlias using sortheap 512"
"update db cfg for $dbAlias using applheapsz 2048"
"update db cfg for $dbAlias using num_iocleaners 3"
"update db cfg for $dbAlias using logfilsiz 5000"
"update db cfg for $dbAlias using logprimary 20"
"update db cfg for $dbAlias using logsecond20"
"update db cfg for $dbAlias using auto_maint on"
"update db cfg for $dbAlias using auto_tbl_maint on"
"update db cfg for $dbAlias using auto_runstats on"

"update db cfg for $dbAlias using MAXLOCKS 80"
"update db cfg for $dbAlias using LOCKTIMEOUT 60"
"update db cfg for $dbAlias using LOCKLIST 10000"
"update db cfg for $dbAlias using MINCOMMIT 1"

"update db cfg for $dbAlias using NUM_IOCLEANERS 11"
"update db cfg for $dbAlias using NUM_IOSERVERS 10"
"connect reset"
"terminate"
```

Figure A-9 Run script from a db2 prompt

3. Update and run the ConfigDB2.sh script. On the Application Server, the script is located at: <WAS_HOME>/AppServer/installableApps/presence/scripts/dbScripts/presence/ConfigDB2.sh:
   a. Move the CongigDb2.sh from the Application Server to the database server.
   b. Make the changes in Figure A-10 on page 473 to the script. For brevity, the entire script is not shown, only the changes.

```
db2 "create tablespace st8kts pagesize 8k managed by system using
('$dbDir/db2tbs/$dbName/st8kts') extentsize 256 prefetchsize 64 bufferpool st8kbp
NO FILE SYSTEM CACHING dropped table recovery off"
.
.
db2 "create temporary tablespace st8ktempts pagesize 8k managed by system using
('$dbDir/db2tbs/$dbName/st8ktemp') bufferpool st8ktempbp NO FILE SYSTEM
CACHING dropped table recovery off"
.
.
db2 "create tablespace st4kts pagesize 4k managed by system using
('$dbDir/db2tbs/$dbName/st4kts') extentsize 256 prefetchsize 64 bufferpool
ibmdefaultbp NO FILE SYSTEM CACHING dropped table recovery off"
.
.
db2 "create tablespace st4kts2 pagesize 4k managed by system using
('$dbDir/db2tbs/$dbName/st4kts2') extentsize 256 prefetchsize 64 bufferpool
ibmdefaultbp NO FILE SYSTEM CACHING dropped table recovery off"
```

Figure A-10 Changing the ConfigDB2.sh script

A.3.5 Tuning WebSphere Application Server

To ensure that the applications can perform under load conditions, you must properly configure the WebSphere Application Server where IMS is deployed.

WebSphere Application Server provides tunable settings for its major components to enable you to make adjustments to better match the runtime environment to the characteristics of your application.

For more information about tuning WebSphere Application Server:

http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.nd.multiplatform.doc/info/ae/ae/welc6toptuning.html

To tune WebSphere Application Server:

1. Turn Off Java 2 Security.

   Java 2 security provides a policy-based, fine-grain access control mechanism that increases overall system integrity by checking for permissions before allowing access to certain protected system resources.

   For more information about Java 2 Security:

   http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.nd.multiplatform.doc/info/ae/ae/csec_rsecmgr2.html?resultof=%22%6a%61%76%61%22%20%22%32%22%20%22%73%65%63%75%72%69%74%79%22%20%22%73%65%63%75%72%22%20

   a. Turn off the Java 2 Security in the WebSphere Application Server Admin Console:

      i. In the navigation panel, click **Security** → **Secure administration, applications, and infrastructure**.

      ii. Under Java 2 security, clear the **Use Java 2 security to restrict application access to local resources** option.

2. Tune the Connection Pools:

   Using connection pools helps to both alleviate connection management overhead and decrease development tasks for data access.

   Each time an application attempts to access a back end store, such as a database, it requires resources to create, maintain, and release a connection to that data store. To mitigate the strain that this process can place on overall application resources, the Application Server enables administrators to establish a pool of back end connections that applications can share.

   Connection pooling can improve the response time of any application that requires connections, especially Web-based applications. When a user makes a request over the Web to a resource, the resource accesses a data source. Because users connect and disconnect frequently with applications on the Internet, the application requests for data access can surge to considerable volume. Consequently, the total data store overhead quickly becomes high for Web-based applications and performance deteriorates. When connection-pooling capabilities are used, however, Web applications can realize performance improvements of up to 20 times the normal results.

   With connection pooling, most user requests do not incur the overhead of creating a new connection because the data source can locate and use an existing connection from the pool of connections. When the request is satisfied and the response is returned to the user, the resource returns the connection to the connection pool for reuse. The overhead of a disconnection is avoided. Each user request incurs a fraction of the cost for connecting or disconnecting. After the initial resources are used to produce the connections in the pool, additional overhead is insignificant because the existing connections are reused. For more information about Connection Pools:

http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.base.doc/info/aes/ae/cdat_conpool.html?resultof=%22%63%6f%6e%6e%65%63%74%69%6f%6e%22%20%22%63%6f%6e%6e%65%63%74%22%20%22%70%6f%6f%6c%22%20

For more information about Connection Pool Settings:

http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.base.doc/info/aes/ae/udat_conpoolset.html

3. Configure the SIP Container:

   a. Configure a thread pool for the SIP Container:

      i. In the navigation panel, click **Servers** → **Application Servers.**

      ii. Click an XDMS Server.

      iii. Under Additional Properties, click **Thread pools**.

      iv. Click **New**.

      v. In the Name field, type `SipContainer`.

      vi. In the Minimum Size field, type `15`.

      vii. In the Maximum Size field, type 20.

      viii. In the Thread inactivity timeout field, keep the default value of 50000.

      ix. Click **OK**, and click **Save** to save the changes to the master configuration.

      Note: Presence Server uses different numbers for these fields.

   b. Configure the SIP Container:

      i. In the navigation panel, click **Servers** → **Application Servers**.

      ii. Click an XDMS Server.

      iii. Under Container Settings, click **SIP Container Settings** → **SIP container**.

      iv. In the Maximum application sessions field, type `200000`.

      v. In the Maximum messages per second field, type `200000`.

      vi. In the Maximum dispatch queue size field, type `5000`.

      vii. From the Thread pool list, select **SipContainer**.

      viii. Click **OK**, and click **Save** to save the changes to the master configuration.

c. Configure the SIP routing rule:

> **Note:** If you have multiple clusters on the same Proxy Server, create a routing rule for each cluster.

i. Select **Proxy Servers** → **<proxy server name>** → **Proxy Settings** → **SIP Proxy Server Settings** → **Routing Rules**.

ii. Click **New**.

iii. From the Cluster pull-down menu, select **<cluster name>**.

iv. Click **OK**, and click **Save** to save changes to the master configuration.

v. In the Routing Rules list, click the **<cluster name>** that you just created.

vi. Click **Conditions**.

vii. Click **New**.

viii. Click **Condition type: Other**.

ix. From the Type pull-down menu, select **From**.

x. Type `sharedlist` for a Shared List Cluster, and type `presrules` for Presence Rules Cluster.

> **Note:** With this routing rules, you need to put the exact sharedlist or presrules string in the From: header of your SIP request that is sent to the Proxy Server to direct the request to either the Shared List Cluster or the Presence Rule Cluster.

A.3.6 Tuning the WebSphere Application Server for XDMS

The procedures that facilitate the performance of IBM WebSphere XML Document Management Server under load conditions through configuring databases and WebSphere Application Server, where WebSphere XML Document Management Server is deployed are at the following Web site:

http://www-1.ibm.com/support/docview.wss?rs=3248&context=SSWJLZ&q1=tuning&uid=swg27011071&loc=en_US&cs=utf-8&lang=en

A.3.7 Tuning WebSphere Application Server for Presence Server

The procedures that improve the performance of IBM WebSphere Presence Server Component under load conditions through configuring WebSphere

Application Server, where WebSphere Presence Server is deployed are at the following Web site:

http://www-1.ibm.com/support/docview.wss?rs=3074&context=SS2PTT&q1=tuning&uid=swg27011069&loc=en_US&cs=utf-8&lang=en

B

Additional material

In this section, we refer you to additional material that you can download from the Internet.

Locating the Web material

The Web material that is associated with this book is available in softcopy on the Internet from the IBM Redbooks publications Web server. Point your Web browser to:

`ftp://www.redbooks.ibm.com/redbooks/SG247370`

Alternatively, you can go to the IBM Redbooks publications Web site at:

`ibm.com/redbooks`

Select **Additional materials**, and open the directory that corresponds with the IBM Redbooks publication form number, SG247370.

Using the Web material

The additional Web material that accompanies this book includes the following files:

File name	*Description*
report.20071006.zip	Basic validation test - discussion report
report.20071008.zip	HA validation test - discussion report

How to use the Web material

Create a subdirectory (folder) on your workstation, and decompress the contents of the Web material compressed file into this folder.

Related publications

The publications that we list in this section are considered particularly suitable for a more detailed discussion of the topics that we covered in this book.

IBM Redbooks

For information about ordering these publications, see "How to get Redbooks" on page 481. Note that some of the documents referenced here might be available in softcopy only.

- *WebSphere Application Server Network Deployment V6: High Availability Solutions*, SG24-6688
- *WebSphere Application Server V6.1: Technical Overview*, REDP-4191

Online resources

This Web site is also relevant as an additional information source:

- About DB2 AIO on Linux

  http://www.ibm.com/developerworks/db2/library/techarticle/dm-0509wright/

How to get Redbooks

You can search for, view, or download IBM Redbooks publications, Redpapers, Technotes, draft publications, and Additional materials, as well as order hardcopy Redbooks, at this Web site:

ibm.com/redbooks

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services

Index

Redbooks

High Availability and Scalability for WebSphere Presence and XML Document Management Servers

(1.0” spine)
0.875”<->1.498”
460 <-> 788 pages

High Availability and Scalability for WebSphere Presence and XML Document Management Servers

Go from base to highly available and scalable environments

Architecture, topology and configuration

Sample implementation

In this IBM® Redbooks publication, we describe how to create scalable and highly-available solutions that incorporate only two of the WebSphere products for Telcom: the IBM WebSphere® Presence and the WebSphere XML Document Management Servers (XDMS). IBM WebSphere products for Telcom is a set of service enablers that extend the IBM WebSphere Application Server to deliver full IP Multimedia Subsystem (IMS) standards-compliant platform for creation and delivery of next-generation telecommunications services.

This book is structured into five parts:

Part 1, "Introduction" - overview of the IBM WebSphere Presence Server and XDMS Document Management Servers and description of the sample scenario that we use throughout this book.

Part 2, "The base scenario" - overview of the base environment for our sample scenario, which includes the architecture, and topology, describes how to set up and configure the different components in the base environment, and concludes with the execution of test cases that validate the base environment.

Part 3, "The High Availability (HA) scenario" - describes the enhancements to the base environment to make it highly available and demonstrates fail-over processing following the failure of servers in the environment.

Part 4, "The scalable scenario" - describes how to configure additional servers to make the highly available environment scalable and demonstrates the distribution of messages to additional servers that are introduced to the environment.

Part 5, "Appendices" provides additional information and pointers to related publications and other useful information.

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

SG24-7370-00 ISBN 0738431001