IBM PRESS

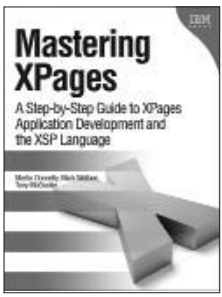
XPages

Portable Command Guide

A Compact Resource to XPages Application Development and the XSP Language

Martin Donnelly, Maire Kehoe,
Tony McGuckin, Dan O'Connor

Related Books of Interest



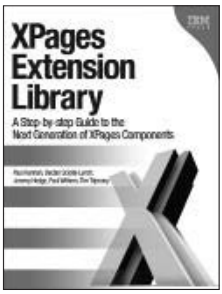
Mastering XPages

A Step-by-Step Guide to XPages Application Development and the XSP Language

By Martin Donnelly, Mark Wallace, Tony McGuckin

ISBN: 0-13-248631-8

The first complete, practical guide to XPages development—direct from members of the XPages development team at IBM Lotus. Martin Donnelly, Mark Wallace, and Tony McGuckin have written the definitive programmer's guide to utilizing this breakthrough technology. Packed with tips, tricks, and best practices from IBM's own XPages developers, *Mastering XPages* brings together all the information developers need to become experts—whether you're experienced with Notes/Domino development or not. The authors start from the very beginning, helping developers steadily build your expertise through practical code examples and clear, complete explanations. Readers will work through scores of real-world XPages examples, learning cutting-edge XPages and XSP language skills and gaining deep insight into the entire development process. Drawing on their own experience working directly with XPages users and customers, the authors illuminate both the technology and how it can be applied to solving real business problems.



XPages Extension Library

A Step-by-Step Guide to the Next Generation of XPages Components

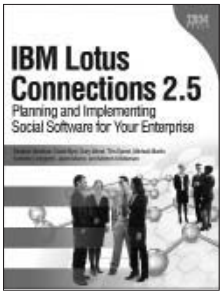
By Paul Hannan, Declan Sciolla-Lynch, Jeremy Hodge, Paul Withers, Tim Tripcony

ISBN: 0-13-290181-1

The XPages Extensibility Framework is one of the most powerful application development features found in IBM Lotus Notes Domino. It enables developers to build their own artifacts and move far beyond XPages' out-of-the-box features. The XPages Extension Library is the greatest manifestation of this framework. A team of all-star XPages experts from inside and outside IBM show developers how to take full advantage of the XPages Extensibility Library and the growing portfolio of components built with them. The authors walk through installing and configuring the XPages Extension Library, integrating it with Domino Designer, and using new XPages components to quickly build state-of-the-art applications for web, the Notes client and mobile devices.

Sign up for the monthly IBM Press newsletter at

Related Books of Interest



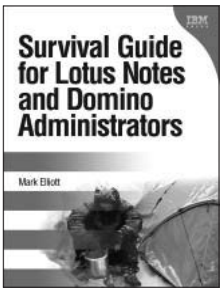
IBM Lotus Connections 2.5

Planning and Implementing Social Software for Your Enterprise

By Stephen Hardison, David Byrd, Gary Wood, Tim Speed, Michael Martin, Suzanne Livingston, Jason Moore, and Morten Kristiansen

ISBN: 0-13-700053-7

In *IBM Lotus Connections 2.5*, a team of IBM Lotus Connections 2.5 experts thoroughly introduces the newest product and covers every facet of planning, deploying, and using it successfully. The authors cover business and technical issues and present IBM's proven, best-practices methodology for successful implementation. The authors begin by helping managers and technical professionals identify opportunities to use social networking for competitive advantage–and by explaining how Lotus Connections 2.5 places full-fledged social networking tools at their fingertips. *IBM Lotus Connections 2.5* carefully describes each component of the product–including profiles, activities, blogs, communities, easy social bookmarking, personal home pages, and more.

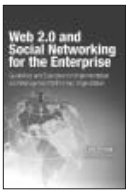


Survival Guide for Lotus Notes and Domino Administrators

By Mark Elliott

ISBN: 0-13-715331-7

Mark Elliott has created a true encyclopedia of proven resolutions to common problems and has streamlined processes for infrastructure support. Elliott systematically addresses support solutions for all recent Lotus Notes and Domino environments.



Web 2.0 and Social Networking for the Enterprise

Guidelines and Examples for Implementation and Management Within Your Organization

Bernal

ISBN: 0-13-700489-3

Related Books of Interest

The Social Factor

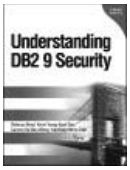
Innovate, Ignite, and Win through Mass Collaboration and Social Networking

By Maria Azua

ISBN: 0-13-701890-8

Business leaders and strategists can drive immense value from social networking "inside the firewall." Drawing on her unsurpassed experience deploying innovative social networking systems within IBM and for customers, Maria Azua demonstrates how to establish social networking communities, and then leverage those communities to drive extraordinary levels of innovation. *The Social Factor* offers specific techniques for promoting mass collaboration in the enterprise and strategies to monetize social networking to generate new business opportunities. Whatever your industry, *The Social Factor* will help you learn how to choose and implement the right social networking solutions for your unique challenges...how to avoid false starts and wasted time...and how to evaluate and make the most of today's most promising social technologies—from wikis and blogs to knowledge clouds.

Listen to the author's podcast at:
ibmpressbooks.com/podcasts



Understanding DB2 9 Security

Bond, See, Wong, Chan

ISBN: 0-13-134590-7

DB2 9 for Linux, UNIX, and Windows

DBA Guide, Reference, and Exam Prep, 6th Edition

Baklarz, Zikopoulos

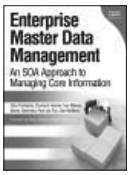
ISBN: 0-13-185514-X

The Art of Enterprise Information Architecture

A Systems-Based Approach for Unlocking Business Insight

Godinez, Hechler, Koening, Lockwood, Oberhofer, Schroeck

ISBN: 0-13-703571-3

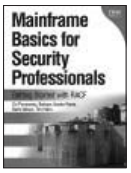


Enterprise Master Data Management

An SOA Approach to Managing Core Information

Dreibelbis, Hechler, Milman, Oberhofer, van Run, Wolfson

ISBN: 0-13-236625-8



Mainframe Basics for Security Professionals

Getting Started with RACF

Pomerantz, Vander Weele, Nelson, Hahn

ISBN: 0-13-173856-9

Sign up for the monthly IBM Press newsletter at

This page intentionally left blank

XPages Portable Command Guide

This page intentionally left blank

XPages Portable Command Guide

A Compact Resource to XPages Application Development and the XSP Language

Martin Donnelly, Maire Kehoe, Tony McGuckin, Dan O'Connor

IBM Press, Pearson plc

Upper Saddle River, NJ • Boston • Indianapolis • San Francisco
New York • Toronto • Montreal • London • Munich • Paris • Madrid
Cape Town • Sydney • Tokyo • Singapore • Mexico City
ibmpressbooks.com

The authors and publisher have taken care in the preparation of this book, but make no expressed or implied warranty of any kind and assume no responsibility for errors or omissions. No liability is assumed for incidental or consequential damages in connection with or arising out of the use of the information or programs contained herein.

© Copyright 2012 by International Business Machines Corporation. All rights reserved.

Note to U.S. Government Users: Documentation related to restricted right. Use, duplication, or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract with IBM Corporation.

IBM Press Program Managers: Steven M. Stansel, Ellice Uffer

Cover design: IBM Corporation

Published by Pearson plc

Publishing as IBM Press

IBM Press offers excellent discounts on this book when ordered in quantity for bulk purchases or special sales, which may include electronic versions and/or custom covers and content particular to your business, training goals, marketing focus, and branding interests. For more information, please contact

U.S. Corporate and Government Sales
1-800-382-3419
corpsales@pearsontechgroup.com

For sales outside the U.S., please contact

International Sales
international@pearson.com

The following terms are trademarks of International Business Machines Corporation in many jurisdictions worldwide: IBM Press, Notes, Domino, Java, IBM, Rational, WebSphere, LotusScript, developerWorks, and Sametime. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at www.ibm.com/legal/copytrade.shtml. Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates. Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both. Linux is a registered 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.

Library of Congress Cataloging-in-Publication Data

XPages portable command guide : a compact resource to XPages application development and the XSP language / Martin Donnelly ... [et al.].
p. cm.
Includes bibliographical references.
ISBN 978-0-13-294305-5 (pbk.)
1. XPages. 2. Application software--Development. 3. Web site development. I. Donnelly, Martin, 1963-
QA76.625.X63 2012
006.7'6--dc23

2011047429

All rights reserved. This publication is protected by copyright, and permission must be obtained from the publisher prior to any prohibited reproduction, storage in a retrieval system, or transmission in any form or by any means, electronic, mechanical, photocopying, recording, or likewise. For information regarding permissions, write to:

Pearson Education, Inc.
Rights and Contracts Department
501 Boylston Street, Suite 900
Boston, MA 02116
Fax (617) 671-3447

Text printed in the United States on recycled paper at R.R. Donnelley in Crawfordsville, Indiana.

First printing February 2012

ISBN-13: 978-0-13-294305-5
ISBN-10: 0-13-294305-0

Associate Publisher
Dave Dusthimer

Marketing Manager
Stephane Nakib

Executive Editor
Mary Beth Ray

Publicist
Heather Fox

Development Editor
Eleanor Bru

Managing Editor
Kristy Hart

Designer
Alan Clements

Project Editor
Anne Goebel

Copy Editor
Krista Hansing
Editorial Services, Inc.

Indexer
Lisa Stumpf

Compositor
Nonie Ratcliff

Proofreader
Debbie Williams

Manufacturing Buyer
Dan Uhrig

Dedications

To the memory of my parents, Betty and Paddy, whose love and support I will always cherish.

—Martin

To my parents and my husband, Nelius, for all their support.

—Maire

To Martin: Once again, you pulled us over the line! You deserve a medal.

To my parents and family: I love you all and hope you enjoy reading another great book about XPages!

For "my two girls," Paula and Anna-Rose: a beautiful wife and special daughter who mean absolutely everything to me!

—Tony

Dedicated to the memory of my parents, Peter and Rita—I miss you both.

To my family—in particular, my wife, Anne Marie, and daughter, Aileen—my contribution to this book would not have been possible without your support and encouragement. I love you both.

Finally, to my coauthors—thank you for putting faith in a "Designer developer" to contribute to this fine book!

—Dan

This page intentionally left blank

Contents

Introduction

Welcome to the XPages Portable Command Guide! This book is designed, for the most part, as a quick information guide for XPages developers with some real-world experience under their belts. It focuses on the road less traveled—**xsp.properties** parameters, **notes.ini** settings, XSP JS object functions, and such. In other words, it covers the little-known magic bullets that are not well documented but invariably help get you out of a programming bind. In that sense, it is an ideal companion for more holistic tomes such as *Mastering XPages,* which is designed to give broad coverage to the runtime and application development experience in general. Having said that, this book does dive into detail, when appropriate—after all, the authors are developers, so we just can't help ourselves!

XPages is a rich and powerful application development framework for Notes/Domino, first introduced in version 8.5 at Lotusphere 2009. Since that time, XPages has gone from strength to strength, with three further release updates, an open source XPages Extension Library, a dedicated IBM XWork server, a best-selling IBM Press book, and many other initiatives and innovations. We hope this Portable Command Guide helps add to the general success of XPages by bringing new information to the community and making application development a little bit easier for all concerned.

Reading Audience

This book is for XPages developers with some practical experience. Neophytes are advised to start with a more general book, such as *Mastering XPages,* or perhaps to use this book as its companion guide.

Conventions

Any programming code, markup, or XSP keywords are illustrated in numbered listings using a fixed width font.

User interface elements (menus, links, buttons, and so on) of the Notes client, Domino Designer, or any sample applications are referenced using a **bold** font. So too are file system paths, locations, and artifacts, such as the **notes.ini** and **xsp.properties** files.

Important words and phrases are emphasized using an *italic* font.

Visual representations of the design-time experience or runtime features are typically captured as screen shots and written up as numbered figures, using super-imposed callouts where appropriate.

In general, chapters feature a summary table of XPages commands, parameters, or properties near the beginning and seek to explain these in brief, concise terms. These items, or important subsets thereof, are typically then given more expansive

treatment in the rest of the chapter. Most chapters also have an accompanying NSF sample application containing practical examples that can be perused using Domino Designer and run in preview mode for the web or Notes client. These samples are available online for download at the following website: www.ibmpressbooks.com/title/0132943050

The samples are based on the latest release of XPages available at the time of writing (version 8.5.3), although many examples work with earlier releases. Visit this website to download the no-charge version of Domino Designer 8.5.3: www.ibm.com/developerworks/downloads/ls/dominodesigner/

How This Book Is Organized

This book is divided into six chapters, to separately address the many different aspects of XPages software development in as logical a manner as possible.

- **Chapter 1, "Working with XSP Properties,"** gives you all the details you need to locate, edit, and load the **xsp.properties** file, and thus configure the XPages runtime for your own specific requirements. An XSP property is a simple parameter definition that can modify the behavior of the XPages runtime in "magical" ways.
- **Chapter 2, "Working with Notes/Domino Configuration Files,"** concerns itself with the practical business of identifying the **notes.ini** settings that have particular relevance to XPages and explains their usage in detail.
- **Chapter 3, "Working with the Console,"** gives an overview of the many ways you can interact with the XPages runtime at the console level for runtime analysis, troubleshooting, or application debugging.
- **Chapter 4, "Working with the XSP Client Side JavaScript Object,"** examines the XSP Client Side JavaScript Object and lists simple examples of all the publically exposed functions that that can be used in an XPage. It also provides a general overview of Client Side JavaScript scripting techniques and other miscellaneous features relevant to XPages development.
- **Chapter 5, "Server-Side Scripting,"** gives an overview of Server Side JavaScript scripting objects and supporting libraries. This chapter also examines ways to integrate custom Java classes and create Managed Beans.
- **Chapter 6, "Server-Side Debugging Techniques,"** provides detail on setting up a debug and logging environment for your XPages applications. It also explains the details of stack traces and how you can analyze and decipher such information when troubleshooting an application.
- **Appendix A, "Definitive Resources,"** points to a collection of definitive reference sources that describe all the details of the XSP tags and Java and JavaScript classes. It also points to specification documents that define the technologies that XPages consumes or extends.

- **Appendix B, "Useful Online Resources,"** gives a snapshot of the authors' favorite XPages websites at the time of writing. This list of sites should help you find whatever you need to know about XPages that you cannot find in this book.
- **Appendix C, "Make Your Own Journal,"** provides blank pages for you to add your own specific notes on settings, markup, code fragments, or whatever else you need that might not be listed in this book.

Acknowledgments

We would like to start by thanking our two very thorough and knowledgeable technical reviewers, Mark Wallace and David Taieb. Thanks to you both for keeping us honest and for providing invaluable feedback—*most* of which we included here. ;-)

A big and sincere thank you to all those in the Notes/Domino application development leadership team for supporting this project—especially to Eamon Muldoon, Pete Janzen, Maureen Leland, Peter Rubinstein, and Philippe Riand.

Behind us are some very special teams of people—particularly the XPages runtime team in IBM Ireland and the Domino Designer team in Littleton, Massachusetts. Each member of these teams has unique strengths and skills, which we have completely exploited over the course of writing this book. The user experience and documentation teams also worked closely with us and helped bring clarity and objectivity to all we do. Our thanks to all: Andrejus Chaliapinas, Brian Gleeson, Darin Egan, Edel Gleeson, Graham O'Keeffe, Greg Grunwald, Jim Cooper, Jim Quill, Kathy Howard, Lisa Henry, Lorcan McDonald, Mark Vincenzes, Michael Blout, Mike Kerrigan, Padraic Edwards, Paul Hannan, Robert Harwood, Robert Perron, Simon McLoughlin, Teresa Monahan, and Vin Manduca.

It was once again a tremendous privilege for us to work with our friends at IBM Press, particularly Mary Beth Ray, Ellie Bru, Anne Goebel, Vanessa Evans, and Chris Cleveland. On the IBM side, Steven Stansel and Ellice Uffer worked tirelessly on getting the message out there for the *Mastering XPages* book and are already beating the drum for this one! Thanks for the help and the fun along the way.

Finally a great big thank you as always to our customers and business partners for continuing to explore new ground with XPages and driving further adoption of this most truly wonderful technology. Viva XPages!

About the Authors

The authors of this book have a few things in common. All four hail from Ireland, work for the IBM software group, and have made significant contributions to the development of both XPages and Domino Designer.

Martin Donnelly is a software architect and tech lead for the XPages runtime team in IBM Ireland. He graduated with a Bachelor of Commerce degree from University College Cork in 1984 and later completed a Master's degree in Computer Science at Boston University (2000). Martin has worked on all XPages releases, from Notes/Domino 8.5 through 8.5.3, and also worked on a precursor technology: XFaces for Lotus Component Designer. In the 1990s, while living and working in Massachusetts, he was a lead developer on Domino Designer. Now based once again in Ireland, Martin lives in Cork with his wife, three daughters, and two greyhounds. Despite the fact that he should have hung up his boots years ago, he still persists in playing soccer on a weekly basis and enjoys salmon angling during the summer when the opportunity presents itself.

Maire Kehoe is a senior software engineer in the IBM Ireland software lab. She completed an Honors Bachelor of Science degree in Computer Applications in Dublin City University (DCU) and began working for IBM in 2003. She worked on the Lotus Component Designer product from 2004 to 2007 and moved to IBM Lotus Domino to help develop the XPages runtime for the Domino server. Maire lives in Dublin with her husband and enjoys travel and musicals (and tea).

Tony McGuckin is a senior software engineer in the IBM Ireland software lab. After studying Software Engineering at the University of Ulster, he began his career with IBM in 2006, working in software product development on the Lotus Component Designer runtime. He then transitioned into the XPages core runtime team when XPages was born. When not directly contributing to the core runtime, Tony is kept busy with research and development of the next generation of IBM software development tools, as well as middleware, conferencing, and consultancy. Outside the lab, Tony enjoys food, wine, and cooking; recently acquired a curious taste for classical music; and likes to get off the beaten track to take in Irish scenery and wildlife.

Dan O'Connor is a senior software engineer in the Littleton, Massachusetts, software lab. He graduated with a Bachelor of Engineering degree in Computer Engineering from the University of Limerick, Ireland in 2000. He joined IBM through Lotus Software in Cambridge, Massachusetts, in 2000. Since then, Dan has worked on different projects, but most have focused on Eclipse and JavaServer Faces. In 2002, he joined the Rational Application Developer team to work on a "new" technology called JSF. In 2006, he rejoined the Lotus division to work on Lotus Component Designer and moved to Domino Designer in 2008 as the UI team lead. Dan lives in Milton, Massachusetts, with his wife and daughter. In his spare time, he spends too many hours following Gaelic football and occasionally dabbles in "home improvement," much to the profit of the local plumber!

CHAPTER 3

Working with the Console

When working with an application running on an application server, it is often necessary to interact with the server's console to analyze, troubleshoot, and debug any problems that might arise. This is also true for XPages applications and controls. The Domino server console provides the developer with a wide variety of commands, ranging from starting a server task to reporting the status of an OSGi bundle running on the server. At some point, the XPage developer inevitably will need to call on the console to analyze why an application is not working or functioning as desired.

The Domino server has a long history. Over time, the Domino server's console has served Domino administrators and developers alike as the first line of attack when troubleshooting problems. In an effort to maintain this level of service over the evolution of the server, and to enable administrators and developers to quickly get to the root of issues relating to XPages applications, the server's console has been instrumented with a large array of commands specifically built with the XPages runtime in mind.

About the XSP Command Manager

The XPages runtime is embedded within the Domino server's HTTP task. The XSP Command Manager serves as the common link for the Domino HTTP task, the server's JVM, and the XPages runtime. The XSP command manager is responsible for dispatching XPages' requests received from the HTTP task and the Domino console, and is also ultimately responsible for the XPages runtime's lifecycle. The XSP Command Manager has many useful commands built in that enable the administrator or developer to quickly analyze whether a particular XSP setting is causing an issue. It also can generate Java dumps that the development team can analyze.

How to Execute the XSP Command Manager Commands

Commands are executed via the XSP Command Manager similar to any other command on the Domino server. The XSP Command Manager is running within the HTTP task, so the commands it executes must be fed through the HTTP task, as in this example:

```
tell http xsp <<xsp command manager command>>
```

Table 3.1 lists all the XSP Command Manager Commands.

Table 3.1 XSP Command Manager Commands

Command Name	Description
show data directory	Shows the location of the Domino server's data directory.
show program directory	Shows the location of the Domino server's program directory.
show version	Displays the exact version of the XPages runtime that is installed and running on the Domino server.
show settings	Shows all the variables/properties that have been set on the server's **bootstrap.properties** file. If a **bootstrap.properties** file does not exist, the XPages runtime provides reasonable recommended defaults.
show modules	Displays the *modules* loaded in the system. The XPages runtime dynamically loads each Domino database as a web application *module*.
refresh	Causes the services in the XPages runtime to be refreshed. This is mainly reserved for future use.
heapdump	Performs a live dump of all objects on the Domino server's Java heap. Creates a dump file that must be read by other tools (such as the Eclipse Memory Analyzer); the file is not human readable.
javadump	Performs a Java dump, sometimes referred to as a thread dump or JavaCore dump, of the Domino server's JVM. The information collected during the dump operation is stored in human-readable format.
systemdump	Performs a full system dump, sometimes referred to as a core dump, of the Domino server's JVM. The dump information is platform specific and contains all the memory, process, and thread information for the JVM at the time the dump occurred.

These commands can greatly aid administrators and developers when trying to analyze particular issues. The dump commands are of particular importance because they perform diagnostic dumps on the server's JVM but do not cause the JVM or the server to stop operation.

show data directory

As the name suggests, this command simply tells the user where the Domino server's **data** directory resides on the operating system's file system. In a Domino server environment, the **data** directory stores all the databases that are available through the Domino server. The location of this directory is significant because all applications running on the server will reside in this directory or within a subdirectory of this directory.

Sample usage:

```
tell http xsp show data directory
```

Figure 3.1 shows the results of running the `show data directory` command on a Domino server.

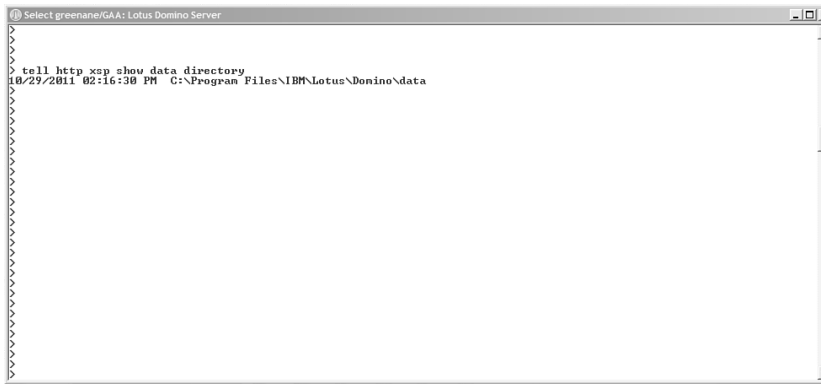


Figure 3.1 Result of running the show data directory command

show program directory

This command tells the user where the Domino server's program directory resides on the operating system's file system. This command can be convenient for developers who are not familiar with a particular setup of an individual server machine. The command enables developers or administrators to quickly identify the file system location of the Domino server's program directory.

Sample usage:

```
tell http xsp show program directory
```

Listing 3.1 shows the result of running the `show program directory` command in the Domino server console.

Listing 3.1 Result of Running the show program directory Command in the Console

```
> tell http xsp show program directory
09/20/2011 10:52:33 PM  C:\Program Files\IBM\Lotus\Domino
```

show version

This command shows the exact version of the XPages runtime that is installed and running on the Domino server. The version number is updated only when upgrading from one release to another of XPages core runtime. Adding or upgrading extensions such as

the XPages Extension library does not update the version number. This command typically is used when a developer or administrator needs to confirm which version of the XPages runtime is running on a particular server. New features are added to the XPages runtime with each release. These features can range from new properties on existing controls to entirely new controls. Over time, a developer or administrator must confirm that the version of XPages runtime is at the appropriate release level for the applications running on the server. This command enables the developer or administrator to quickly confirm the XPages runtime version.

Sample usage:

```
tell http xsp show version
```

Listing 3.2 shows how to determine the version of the XPages runtime.

Listing 3.2 Result of Running the show version Command in the Console

```
> tell http xsp show version
09/20/2011 04:34:21 PM  XSP Runtime Version: [DSI8.5.3] 20110629.1645
```

In the previous example, the version number can be broken down as follows:

- The `DSI` prefix is a constant, which does not vary from release to release.
- `8.5.3` represents the *Major.Minor.Maintenance* version number. The first digit is updated with each major feature release, the second digit is updated with each minor feature release, and the last digit is updated with each maintenance release.
- The final number (`20110629.1645`) represents the time stamp (yyyyMMdd.hhmm) at which the build in question occurred.

show settings

This command makes a request to the XPages runtime to print all the settings in use by the runtime. By default, the XPages runtime is configured with a host of default settings. These settings can be overwritten by adding a **bootstrap.properties** file to the **xsp** directory, which resides in the Domino server's program directory (for example **C:\domino\xsp**). As a result of being able to override the default settings in the XPages runtime (via **bootstrap.properties**), it is not guaranteed that the XPages runtime defaults will apply from server to server. This command enables developers and administrators to quickly list all the current settings without needing to manually access various file system locations to determine which properties are being applied.

Sample usage:

```
tell http xsp show settings
```

Listing 3.3 shows the XPages runtime default settings being output to the Domino server console.

Listing 3.3 Result of Running the show settings Command in the Console (Default Case)

```
> tell http xsp show settings
09/16/2011 11:24:26 AM  xsp.commas.not.delimiters.in.cookie=false
09/16/2011 11:24:26 AM  com.ibm.faces.USE_UNENCODED_CONTEXT_PATH=/xsp
09/16/2011 11:24:26 AM  xsp.gc.on.shutdown=false
09/16/2011 11:24:26 AM  xsp.sessionid.name=SessionID
09/16/2011 11:24:26 AM  xsp.default.charset=UTF-8
09/16/2011 11:24:26 AM  xsp.log.severe.stack.trace=false
09/16/2011 11:24:26 AM  xsp.default.post.buffer.size=1024
09/16/2011 11:24:26 AM  xsp.allow.cookie.sessionid=true
09/16/2011 11:24:26 AM  xsp.global.context.path=/xsp
09/16/2011 11:24:26 AM  xsp.send.set.cookie2.header=true
09/16/2011 11:24:26 AM  xsp.max.cookies.per.session=50
09/16/2011 11:24:26 AM  xsp.allow.packagenames=false
09/16/2011 11:24:26 AM  xsp.allow.url.sessionid=true
09/16/2011 11:24:26 AM  xsp.default.chunk.post.buffer.size=8
```

In some cases, it is necessary to set extra system settings or even overwrite existing settings. Being able to quickly analyze which settings have changed can be invaluable. Listing 3.4 shows a case in which some settings (`xsp.sessionid.name`) have been overwritten by **bootstrap.properties** and some new logging settings (`log_configuration` and `logdir`) have been added. Chapter 6, "Server-Side Debugging Techniques," explains these settings

Listing 3.4 Result of Running the show settings Command in the Console

```
> tell http xsp show settings
09/16/2011 11:01:47 PM  xsp.commas.not.delimiters.in.cookie=false
09/16/2011 11:01:47 PM  com.ibm.faces.USE_UNENCODED_CONTEXT_PATH=/xsp
09/16/2011 11:01:47 PM  xsp.gc.on.shutdown=false
09/16/2011 11:01:47 PM  log_configuration=xsp/log.properties
09/16/2011 11:01:47 PM  xsp.sessionid.name=FOOID
09/16/2011 11:01:47 PM  xsp.default.charset=UTF-8
09/16/2011 11:01:47 PM  xsp.log.severe.stack.trace=false
09/16/2011 11:01:47 PM  xsp.default.post.buffer.size=1024
09/16/2011 11:01:47 PM  xsp.allow.cookie.sessionid=true
09/16/2011 11:01:47 PM  xsp.global.context.path=/xsp
09/16/2011 11:01:47 PM  xsp.send.set.cookie2.header=true
09/16/2011 11:01:47 PM  xsp.max.cookies.per.session=50
09/16/2011 11:01:47 PM  xsp.allow.packagenames=false
09/16/2011 11:01:47 PM  xsp.allow.url.sessionid=true
09/16/2011 11:01:47 PM  logdir=c:/Domino/log
09/16/2011 11:01:47 PM  xsp.default.chunk.post.buffer.size=8
```

show modules

Each Domino database (.NSF) that is running within the XPages runtime is loaded by the XPages runtime as an application *module*. The `show modules` command shows all the databases (NSF modules) that are currently running within the XPages runtime. This command also shows registered system service modules that the XPages runtime automatically loads. This command is convenient for server administrators who need to know which XPages applications are being served by the XPages runtime at any point in time.

Sample usage:

```
tell http xsp show modules
```

Listing 3.5 shows all the active modules running within a Domino server that has sessions open for three XPages applications.

Listing 3.5 Result of Running the show modules Command in the Console

```
> tell http xsp show modules
09/16/2011 11:47:36 AM  XSP Resources
09/16/2011 11:47:36 AM  Default Http Registry Module
09/16/2011 11:47:36 AM  OSGI WebContainer Bridge Service
09/16/2011 11:47:36 AM  oauthtokenstore.nsf
09/16/2011 11:47:36 AM  lsdemo2011.nsf
09/16/2011 11:47:36 AM  xpagessbt.nsf
```

In Listing 3.5, six modules are listed. Three of these modules are XPages runtime system modules; the other three modules represent XPages applications that are currently running on the server.

- `xpagessbt.nsf`, `lsdemo2011.nsf`, and `oauthtokenstore.nsf` are all XPages applications that were running on the server when the command was executed.
- XSP Resources is a module loaded by the XPages runtime; it is not configurable.
- Default Http Registry Module is a module loaded by the Domino web container; it is not configurable.
- OSGI WebContainer Bridge Service is a module loaded by the Domino to OSGi bridge; it is not configurable.

The core runtime modules are not configurable and can be removed or added to in future releases.

refresh

This command was implemented with future extensions of the XSP Command Manager's HTTP service in mind. As of release 8.5.3 of the Domino server, this command

does nothing. It is intended to be used with HTTP services and will enable services to be refreshed as necessary without restarting the HTTP task or the XPages runtime.

Sample usage:

```
tell http xsp refresh
```

heapdump

The `heapdump` command performs a live dump of all objects on the Domino server's Java heap. The operation creates a dump file that must be read by third-party tools; the file is not human readable. The dump file can be read using tools such as the Eclipse Memory Analyser Tool (www.eclipse.org/mat). Because the dump file is written in the IBM JVM heap dump format, it is necessary to install further add-ons to the Eclipse Memory Analyser Tool to read the heap dump information. You can download the add-on for the Eclipse Memory Analyzer tool from www.ibm.com/developerworks/java/jdk/tools/dtfj.html. The `heapdump` command causes a dump file to be generated in the server's program directory, as demonstrated in Figure 3.2.

Sample usage:

```
tell http xsp heapdump
```

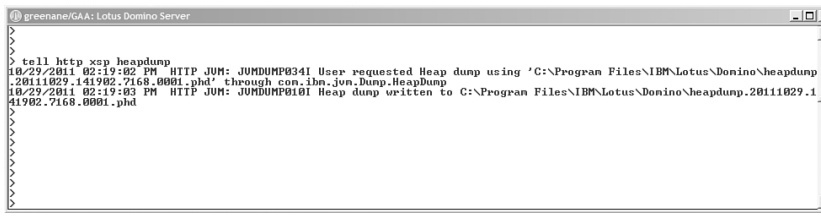


Figure 3.2 Result of running the `heapdump` command in the console

When configured, the Eclipse Memory Analyzer tool enables the user to read the content of the dump file and provide information on where memory is potentially being leaked and which objects are in use when the dump occurs. Figure 3.3 shows sample output from the Eclipse Memory Analyzer Tool.

javadump

Running the `javadump` command causes the server's JVM to create a Java Dump file. Sometimes referred to as a thread dump or a Javacore dump, the dump information is written to disk in a human-readable format—the contents of the dump file can be opened with applications such as Microsoft Notepad. The information stored as a result of a `javadump` is generally diagnostic information relating to the threads, stacks, locks, and memory that were in use by the JVM when the dump occurred. Javadump files are of particular use where the developer or administrator needs to quickly obtain system information (such as operating system version, JVM version, and loaded threads).

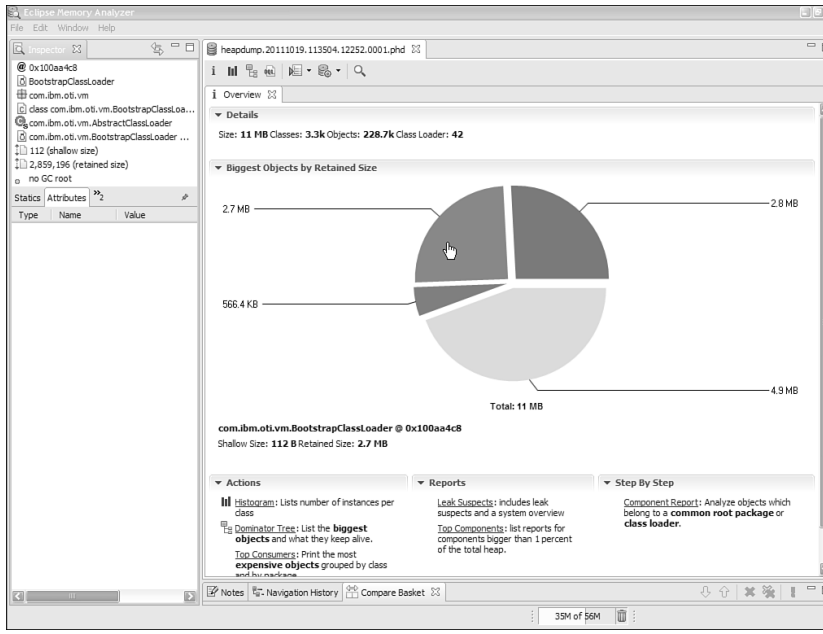


Figure 3.3 Eclipse Memory Analyzer Tool

The Javadump file is lightweight by nature and can help to quickly identify which threads are hung in the system.

Sample usage:

```
tell http xsp javadump
```

Listing 3.6 shows the console output when the `javadump` command is executed.

Listing 3.6 Result of Running the javadump Command in the Console

```
> tell http xsp javadump
10/18/2011 11:40:00 PM  HTTP JVM: JVMDUMP034I User requested
Java dump using 'C:\Program Files\IBM\Lotus\Domino\
javacore.20111018.233959.8220.0001.txt' through com.ibm.jvm.Dump.
➥JavaDump
10/18/2011 11:40:01 PM  HTTP JVM: JVMDUMP010I Java dump written to C:\
Program Files\IBM\Lotus\Domino\javacore.20111018.233959.8220.0001.txt
```

In Listing 3.6, you can see the result of executing the `javadump` command. A Javadump file is written to the location specified in the console output. It is beyond the scope of this book to go into the details of reading the contents of dump files. However, in the case of Javadump files, a few tips can easily be bestowed upon the reader to make

reading the contents of the Javadump file easier. The dump file can essentially be broken down into different sections:

- Date and time of the javadump.
- Operating system signal information (who initiated the javadump and how it was initiated). The signal information tells the reader whether the user initiated the dump or whether the operating system did so due to a program fault. The signal information is operating system specific.
- Java (JVM) version.
- Information about threads running when the javadump occurred.
- Operating system and processor details.
- Native libraries loaded by the JVM.
- Full command line, including arguments, that the Domino server used to launch the JVM.
- JVM monitor information.
- Current stack for each thread running within the JVM.

For further in-depth information on how to read the contents of the Javadump file, see the following article from IBM support:

```
www-01.ibm.com/support/docview.wss?uid=swg21181068
```

Alternatively, you can search for information on how to read a javacore dump file in your favorite Internet search engine.

systemdump

The `systemdump` command is the most intensive of the three dump commands available through the XSP Command Manager. As a result, the footprint of the resulting systemdump file can be quite large. The systemdump file contains detailed information on the JVM's threads, memory, and active processes. When a Java application crashes as a result of general protection fault failure or as a result of a major JVM error, a systemdump file is generated by default.

Sample usage:

```
tell http xsp systemdump
```

Listing 3.7 shows the console output when the `systemdump` command is executed.

Listing 3.7 Result of Running the systemdump Command in the Console

```
09/20/2011 12:36:30 AM  HTTP JVM: JVMDUMP034I User requested
System dump using 'C:\Program Files\IBM\Lotus\Domino\
core.20110920.003630.8220.0002.dmp' through com.ibm.jvm.Dump.SystemDump
```

```
09/20/2011 12:38:26 AM  HTTP JVM: JVMDUMP010I System dump written to
C:\Program Files\IBM\Lotus\Domino\core.20110920.003630.8220.0002.dmp
```

The dump file is stored in a platform-specific format and, as a result, must be read by tools specific to the platform on which the dump was created. The IBM Dump Analyzer enables you to read and analyze the contents of a system dump that is performed on the Domino server. For more information on the IBM Dump Analyzer tool, refer to the following websites:

- "Java Diagnostics, IBM Style, Part 1: Introducing the IBM Diagnostic and Monitoring Tools for Java—Dump Analyzer," at IBM.com: www.ibm.com/developerworks/java/library/j-ibmtools1/
- "Installing the IBM Monitoring and Diagnostic Tools for Java—Dump Analyzer," at IBM.com: www.tinyurl.com/IBMJavaDumpAnalyzer

The information generated by a system dump is extremely granular in nature. An XPage developer rarely will need to create a system dump because the information the dump generates details information about every process executing on the system, not just the information pertinent to the JVM. A system dump generally is needed only when the failure is as a result of complex interactions with programs running outside the Domino server.

Working with the OSGi Console

Before delving into the inner workings of the OSGi console, it is best to briefly explain OSGi. OSGi stands for Open Services Gateway initiative framework. This framework allows software to be written and executed as independent components. In OSGi-speak, these components are referred to as *bundles*. OSGi is used in a wide range of applications, from client programs such as Eclipse and IBM Lotus Notes, to mobile phones, to server applications such as IBM Lotus Domino. As a result of their modular nature, OSGi bundles can be started, stopped, and debugged on an individual basis, without the need for stopping or restarting the entire platform. Both the Domino server and the Notes client use Eclipse's implementation of OSGi (Equinox) as their OSGi runtime platform.

OSGi was added to the Domino platform in release 8.5.2. As a result, in Domino 8.5.2, the XPages runtime was repackaged to run as OSGi bundles (instead of just a regular collection of Java JARS), also referred to as *Eclipse plug-ins*.

The OSGi console allows for the input of commands that the OSGi platform then performs. The platform posts the results of such commands back to the console. The OSGi platform itself has a whole host of commands that can simplify the troubleshooting of problems. The OSGi console can assist developers in developing XPages controls and applications, as well as assisting support personnel in diagnosing runtime issues. Developers who extend the XPages runtime by creating libraries will find the OSGi console commands to be a particularly powerful tool in analyzing problems. The OSGi console

is of particular use when the developer/administrator needs to know whether individual plug-ins (or sets of plug-ins) are loading correctly or which version of a plug-in is in use.

As mentioned earlier, OSGi is embedded within both the Notes client and the Domino server. Depending on where your XPages application is running (whether on the client or the server), your method of accessing the OSGi console will vary. We start by explaining how to access the OSGi console on the Domino server.

OSGi is embedded within the HTTP task on the Domino server, as a result, the OSGi console is started automatically whenever the HTTP task is started. OSGi console commands are routed to the OSGi console via the HTTP task. That is, when entering an OSGi console command on the Domino server, the user must tell the HTTP task to route the specified command to the OSGi console—for example:

```
tell http osgi <<command>>
```

Here, `<<command>>` is the name of the OSGi console command. Any OSGi command can be executed using the preceding syntax.

When it comes to OSGi commands, every developer and administrator should know several rudimentary commands. These commands can be your "go to" commands when problems arise—say, when you suspect bundle loading might be a factor. Even when you do not *think* that bundle loading is the problem, it is often best to first confirm that the bundle is actually loaded before proceeding with other debugging techniques.

Table 3.2 lists some of the more commonly used OSGi commands that are available to use for diagnosing plug-in issues on the Domino server (and also the Notes client). In Table 3.2, *bundle-symbolic-name* is referenced extensively. This is the name by which the OSGi platform references bundles. *bundle-symbolic-name* correlates directly to the *Bundle-SymbolicName* manifest header, often referred to as the *plug-in name*.

Table 3.2 OSGi Console Commands

Command Syntax	Description
tell http osgi diag <bundle-symbolic-name>	Diagnoses the status of the bundle whose name is provided. Determines whether the bundle is resolved and, if not, states why the bundle is not resolved.
tell http osgi ss <bundle-symbolic-name>	Lists the status of all bundles in the system. Optionally, a symbolic name or a symbolic name prefix can be provided to obtain the status of a particular bundle or a subset of bundles.
tell http osgi start <bundle-symbolic-name>	Starts the bundle with the specified symbolic name.
tell http osgi stop <bundle-symbolic-name>	Stops the bundle with the specified symbolic name.
tell http osgi b <bundle-symbolic-name>	Prints metadata relating to the specified bundle.

Command Syntax	Description
tell http osgi headers <bundle-symbolic-name>	Lists the OSGi headers for the specified bundle.
tell http osgi help	Lists all the OSGi command available on the server, along with some text describing each command.

All the commands listed in Table 3.2 can be entered via the Domino server console, with the results of such commands being printed back to the console, as illustrated in Figure 3.4.

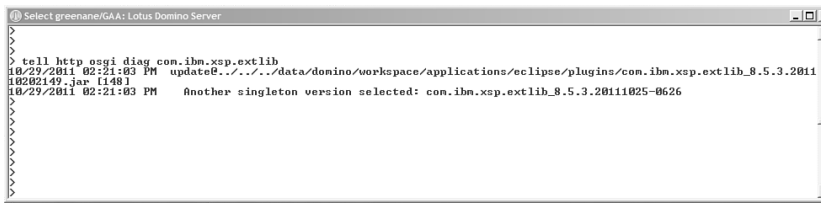


Figure 3.4 Running an OSGi command on the Domino Server Console

All the commands referenced in Table 3.2 can alternatively take the *bundle id* as a parameter (instead of the *bundle-symbolic-name)*. The bundle id is a numeric ID that the OSGi runtime assigns to the bundle during platform initialization. The ID might vary from instance to instance of the platform, but users might find it easier to input than having to enter the entire bundle symbolic name. Examples of how to determine and use the bundle id are given later in this chapter.

Rarely does a single OSGi console command answer all the questions on why a plug-in is not loading or operating as expected. The following sections explain in greater detail how you can use each of these commands and the results you can expect to see from executing such commands.

diag <bundle-symbolic-name>

This is one of the most valuable commands in your arsenal and will likely be the one you'll use most frequently when diagnosing issues. You can use this command to determine whether a bundle is resolved within the OSGi platform. The status returned by this command will be one of the following:

- `No unresolved constraints`
- `Unresolved constraint`

If `No unresolved constraints` is the returned status, it suggests that the system has recognized the bundle and that all dependencies of the bundle are satisfied. When an `Unresolved constraint` status is returned, it suggests that one or more bundles

or packages that the bundle requires are missing or cannot be loaded. It is worth noting here that a bundle might still fail to *start* even though the OSGi console reports that the bundle has been *resolved*. If a bundle fails to start and is resolved, some code in the bundle's activator likely is failing (throwing an exception).

Sample usage:

```
tell http osgi diag com.ibm.xsp.core
```

Listing 3.8 shows the typical output of running the `diag` command against the com.ibm.xsp.core plug-in.

Listing 3.8 Result of Running the diag Command Against a Specific Bundle—Successful Case

```
> tell http osgi diag com.ibm.xsp.core
10/17/2011 09:43:14 PM
  initial@reference:file:../../shared/eclipse/plugins/com.ibm.xsp.
core_8.5.3.20110629-1645/[119]
10/17/2011 09:43:14 PM    No unresolved constraints.
```

In this case, the `diag` command reports that there were `No unresolved constraints` against the entered bundle symbolic name—in other words, the system recognizes the given bundle. Upon closer examination, the user can obtain further information about the bundle in question. It can determine where the bundle being used by the platform is installed, and the platform-assigned *bundle id* can also be obtained.

From reading the console output, the user can see that the bundle is installed to **../../shared/eclipse/plugins/com.ibm.xsp.core_8.5.3.20110629-1645.** The location specified is relative to the **osgi/rcp/eclipse** directory, which is a child of the Domino program directory. In this case, the console output indicates that the plug-in is installed at: **<domino program directory>/osgi/shared/eclipse/plugins**.

Finally, the output states the platform-assigned bundle id for the specified bundle. `119` is the id assigned to this bundle in this example. As stated previously, the OSGi commands listed here can use the bundle id interchangeably. In this example, executing the following command has identical output to that in Listing 3.8.

Sample usage:

```
tell http osgi diag 119
```

Listing 3.9 shows sample output of running the `diag` command in an unsuccessful scenario.

Listing 3.9 Result of Running the diag Command Against a Specific Bundle—Error Case

```
> tell http osgi diag com.ibm.xsp.extlib.sbt 09/09/2011 04:05:51 PM
update@../../../data/domino/workspace/applications/eclipse/plugins/com.
ibm.xsp.extlib.sbt_8.5.3.201108111413.jar [116]      09/09/2011 04:05:51
```

```
PM   Direct constraints which are unresolved: 09/09/2011 04:05:51 PM
Missing host com.ibm.xsp.extlib_0.0.0.
```

In Listing 3.9, you can see that the OSGi platform reports that the bundle in question is not resolved as a result of a missing dependency. We can see from the console output that the OSGi platform has actually found the bundle that we are looking for (`com.ibm.xsp.extlib.sbt`), but as one of the bundles that `com.ibm.xsp.extlib.sbt` depends on is not resolved, the `com.ibm.xsp.extlib.sbt` bundle does not get resolved itself. Looking a little more closely at the console output, we can determine the following:

The bundle `com.ibm.xsp.extlib.sbt` is installed at **../../../data/domino/workspace/applications/eclipse/plugins/com.ibm.xsp.extlib.sbt_8.5.3.201108111413.jar**. We now know that this path is relative to the **<domino program directory>/osgi/rcp/eclipse** directory. Hence, we can deduce that `com.ibm.extlib.sbt` is installed at the **<domino program directory>/data/domino/workspace/applications/eclipse/plugins/** directory.

The OSGi platform–assigned *bundle id* for this bundle is `116`.

One other tidbit of information can be extracted from the console output, in this case. The final line of the output tells us that the *host* is missing:

```
Missing host com.ibm.xsp.extlib_0.0.0
```

This tells us that the bundle we are looking for (`com.ibm.xsp.extlib.sbt`) is, in fact, a plug-in fragment, and the unresolved constraint (`com.ibm.xsp.extlib`) is the host plug-in.

ss, ss <bundle-symbolic-name>, or ss <bundle-name-prefix>

Similar to the `diag` command, this command quickly determines the status of a particular bundle—or all the bundles installed in the platform. Users can optionally specify a bundle name or a bundle name prefix to get the status of specific bundles. The returned status shows the bundle id, state, and bundle name of all bundles. In many situations, this command is just as useful as the `diag` command because it also reports the status of a bundle. This command does not tell the user why a particular bundle is not loading, but it does tell the user the state of a bundle.

Sample usage:

```
tell http osgi ss
```

Listing 3.10 shows the result of running the `ss` command without any parameters.

Listing 3.10 Result of Running the ss Command Without Any Bundle Name Parameter

```
> tell http osgi ss
09/09/2011 01:46:07 PM  Framework is launched.
09/09/2011 01:46:07 PM  id      State      Bundle
09/09/2011 01:46:07 PM  0       ACTIVE    org.eclipse.
osgi_3.4.3.R34x_v20081215-1030-RCP20110624-1648
09/09/2011 01:46:07 PM                    Fragments=57, 76, 88, 89, 235
09/09/2011 01:46:07 PM  1       RESOLVED  org.eclipse.equinox.
event_1.1.0.v20080225
09/09/2011 01:46:07 PM                    Fragments=32
09/09/2011 01:46:07 PM  2       RESOLVED  com.ibm.pvc.jndi.provider.
java.nl_6.2.3.20110625-0109
09/09/2011 01:46:07 PM                    Master=71
09/09/2011 01:46:07 PM  3       RESOLVED  com.ibm.eclipse.equinox.
preferences.nl_6.2.3.20110624-1648
09/09/2011 01:46:07 PM                    Master=85
09/09/2011 01:46:07 PM  4       <<LAZY>>  com.ibm.icu.
base_3.8.1.v20080530
09/09/2011 01:46:07 PM  5       RESOLVED  com.ibm.pvc.servlet.
jsp_2.1.0.20110625-0109
09/09/2011 01:46:07 PM  6       RESOLVED  org.apache.commons.
logging_1.0.4.20110625-0109
```

Listing 3.10 lists a subset of the information that displays when this command is run in a normal server environment. However, the listing does show all the information needed to understand the output of the command.

The command outputs several important pieces of information about each bundle:

- `Bundle-id`—for example, `2`, which is the OSGi platform–assigned ID of the bundle.
- `Bundle state`—for example, `RESOLVED`, which is the state of the bundle within the OSGi platform. A bundle can have one of seven states. Table 3.3 explains all of these.
- `Bundle name`—for example `com.ibm.eclipse.equinox.preferences.nl_6.2.3.20110624-1648`, which is the bundle symbolic name with its version information appended to the name.
- `Master` or `Fragments`—for example, `Master=71`. This data tells whether the bundle in question is a plug-in or a fragment. If the bundle specifies neither `Master` nor `Fragments`, it is automatically implied that the bundle is a plug-in bundle. The digits corresponding to the fragments or plug-ins are the OSGi platform–assigned bundle ids of the fragments or the master plug-in of the bundle in question.

Sample usage:

```
tell http osgi ss com.ibm.xsp.extlib
```

Listing 3.11 shows the result of running the `ss` command with a bundle prefix specified.

Listing 3.11 Result of Running the ss Command, Specifying a Bundle Prefix

```
> tell http osgi ss com.ibm.xsp.extlib
09/09/2011 02:25:36 PM  Framework is launched.
09/09/2011 02:25:36 PM  id      State        Bundle
09/09/2011 02:25:36 PM  108     RESOLVED com.ibm.xsp.extlib.
conns_8.5.2.20110724
09/09/2011 02:25:36 PM                       Master=117
09/09/2011 02:25:36 PM  109     RESOLVED     com.ibm.xsp.extlib.
domino_8.5.2.201107241628
09/09/2011 02:25:36 PM                       Master=117
09/09/2011 02:25:36 PM  112     RESOLVED     com.ibm.xsp.extlib.
oneui_8.5.2.201107241628
09/09/2011 02:25:36 PM                       Master=117
09/09/2011 02:25:36 PM  115     RESOLVED     com.ibm.xsp.extlib.
stime_8.5.2.201107241628
09/09/2011 02:25:36 PM                       Master=117
09/09/2011 02:25:36 PM  117     ACTIVE       com.ibm.xsp.
extlib_8.5.2.201107241628
```

Similar to Listing 3.10, Listing 3.11 shows the results of executing the `ss` command, only this time the command is passed a bundle symbolic name as a parameter. The `ss` command finds all bundles on the system that either start with the parameter or have a bundle symbolic name that is the same as the parameter. Listing 3.11 lists all the bundles, along with their bundle id and state.

Table 3.3 lists all the possible states of an OSGi bundle.

Table 3.3 OSGi Bundle States

State	Description
UNINSTALLED	The bundle is uninstalled and is unusable.
INSTALLED	The bundle has been installed, but the platform has not yet resolved it.
RESOLVED	The bundle has been resolved and is in a position to be started. Note that it is still possible for the bundle to fail to start, even though it has been resolved by the environment.
<<LAZY>>	Similar to `RESOLVED`, the platform has resolved the bundle and is in a position to be started. The bundle is not yet `ACTIVE` because it has been configured (via its bundle manifest) to be initialized lazily—that is, only when another `ACTIVE` bundle references the bundle will it be activated.

State	Description
STARTING	The bundle is in the process of starting. Either another bundle has specifically caused the bundle to start (by referring to a class within the bundle) or the user has manually started the bundle via the console. Rarely is a bundle in this state because it is transient.
STOPPING	The bundle is in the process of shutting down. Similar to `STARTING`, a bundle rarely is in this state.
ACTIVE	The bundle is running within the OSGi platform.

Developers and administrators should be aware that, on the Domino server, the state of a bundle is not persisted from one session to the next—that is, after the HTTP task is restarted, any bundles that were started manually in the previous session must be started again. Luckily, the `ss` command has an argument for filtering all bundles in a given state. The `ss` command can filter the bundles based on their state, by appending `-s [state]` to the command syntax.

Sample usage:

```
tell http osgi ss -s active
```

Figure 3.5 shows the output of running the `ss` command with the `-s` active argument.

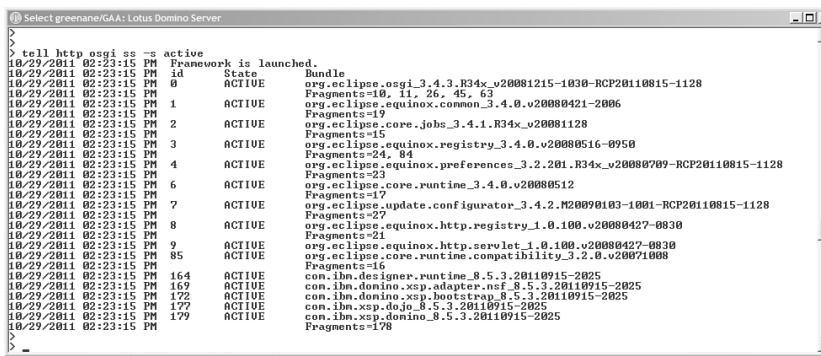


Figure 3.5 Result of running the ss command in the Domino server console

start <bundle-symbolic-name>

This command requests that the platform manually start the specified bundle. Calling this command does not guarantee that the specified bundle will be started. An exception can still occur during bundle initialization that would cause the bundle initialization to fail. Performing an `ss` command after the `start` command reports the status of the bundle. This command is helpful when a new bundle has been installed on the server,

but the administrator or developer is not in a position to restart the HTTP task to start the new bundle.

Sample usage:

```
tell http osgi start com.ibm.xsp.extlib.sbt
```

Figure 3.6 shows that, by running a combination of the `ss` and `start` commands, a bundle can be started and its state can be verified.

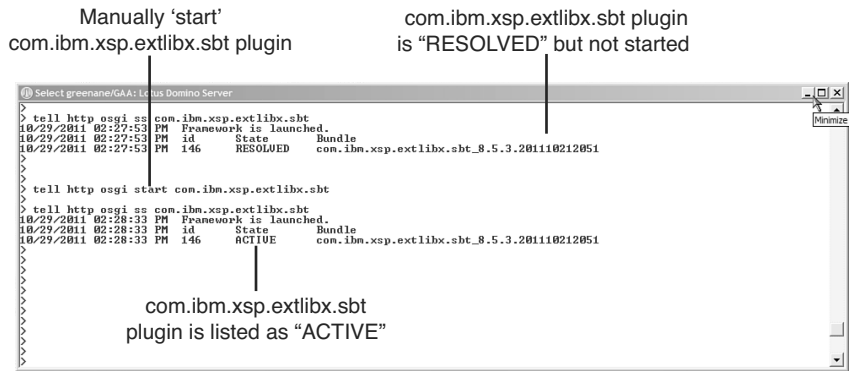


Figure 3.6 Result of running the start and ss commands in the console

stop <bundle-symbolic-name>

This command tells the platform to stop the specified bundle. Users should be careful when calling this on a production environment. In some cases, it might not be possible for the platform to stop the bundle. If this is the case, the reason will be printed to the console.

Sample usage:

```
tell http osgi stop com.ibm.xsp.extlib.sbt
```

Figure 3.7 shows how running a combination of the `ss` and `stop` command stops a bundle and verifies its state.

b <bundle-symbolic-name>

This command prints all metadata relating to the specified bundle. The metadata includes imported packages, required bundles, exported packages, bundle location, and so on. This command is useful when the developer needs to quickly verify that the bundle loaded by the platform has the meta information that the developer believes it has.

Sample usage:

```
tell http osgi b com.ibm.xsp.extlib
```

Listing 3.12 shows a subset of the output from running the `b` command against a specified bundle.

Listing 3.12 Sample Result of Running the b Command Against a Specified Bundle

```
tell http osgi b com.ibm.xsp.extlib
09/09/2011 02:15:21 PM
update@../../../data/domino/workspace/applications/eclipse/plugins/com.
ibm.xsp.extlib_8.5.2.201107241628NTF.jar [117]
  09/09/2011 02:15:21 PM     Id=117, Status=<<LAZY>>     Data Root=C:\
Program Files\IBM\Lotus\Domino\data\domino\workspace\.config\
  org.eclipse.osgi\bundles\117\data
  09/09/2011 02:15:21 PM     No registered services.
  09/09/2011 02:15:21 PM     No services in use.
  09/09/2011 02:15:21 PM     Exported packages
  09/09/2011 02:15:21 PM       com.ibm.xsp.extlib.actions.client;
version="0.0.0"[exported]
09/09/2011 02:15:21 PM       com.ibm.xsp.extlib.actions.client.data;
version="0.0.0"[exported]
09/09/2011 02:15:21 PM       com.ibm.xsp.extlib.actions.client.dojo;
version="0.0.0"[exported]
09/09/2011 02:15:21 PM       com.ibm.xsp.extlib.actions.client.dojo.fx;
version="0.0.0"[exported]
09/09/2011 02:15:21 PM       com.ibm.xsp.extlib.actions.server;
version="0.0.0"[exported]
```

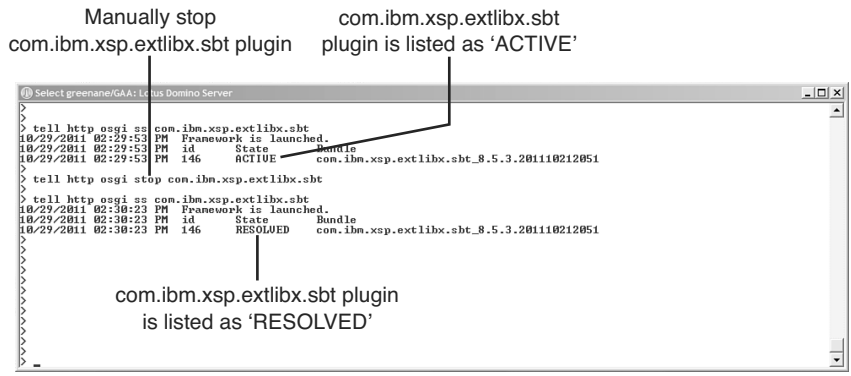


Figure 3.7 Result of running the start and ss commands in the console

headers <bundle-symbolic-name>

This command causes the OSGi header information for the specified bundle to be printed to the console. This command is convenient for checking information such as the packages that a specific bundle exports or the bundles that the specified bundle depends

upon. All the information stored in the bundle's `manifest.mf` file is printed to the console.

Sample usage:

```
tell http osgi headers com.ibm.xsp.extlib.sbt
```

Listing 3.13 shows the result of running the `headers` command on the Domino server console.

Listing 3.13 Sample Result of Running the headers Command with a Specified Bundle Name

```
tell http osgi headers com.ibm.xsp.extlib.sbt
  09/09/2011 04:34:52 PM  Bundle headers:
  09/09/2011 04:34:52 PM   Bundle-ClassPath = .,lib/httpclient-
4.0.1.jar,lib/httpcore-4.0.1.jar,lib/commons-codec-1.3.jar,lib/
oauth-20100527.jar,lib/
oauth-consumer-
20090617.jar,lib/oauth-consumer-20100527.jar,lib/oauth-httpclient4-
20090913.jar,lib/oauth-provider-20100527.jar
  09/09/2011 04:34:52 PM   Bundle-ManifestVersion = 2
  09/09/2011 04:34:52 PM   Bundle-Name = IBM Social Business Toolkit
09/09/2011 04:34:52 PM   Bundle-SymbolicName = com.ibm.xsp.extlib.
sbt;singleton:=true
  09/09/2011 04:34:52 PM   Bundle-Vendor = IBM
  09/09/2011 04:34:52 PM   Bundle-Version = 8.5.3.201108111413
  09/09/2011 04:34:52 PM   Export-Package =
com.ibm.xsp.extlib.fragment,com.ibm.xsp.extlib.model,com.ibm.xsp.extlib.
resources,com.ibm.xsp.extlib.sbt.activitystreams,com.ibm.xsp.extlib.sbt.
activitystreams.entry,com.ibm.xsp.
extlib.sbt.activitystreams.queue,com.ibm.xsp.extlib.sbt.connections,com.
ibm.xsp.extlib.sbt.connections.meta,com.ibm.xsp.extlib.security.
authorization,com.ibm.xsp.extlib.security
.authorization.beans,com.ibm.xsp.extlib.security.oauth_10a,com.ibm.xsp.
extlib.security.oauth_10a.servlet
  09/09/2011 04:34:52 PM   Fragment-Host = com.ibm.xsp.extlib
  09/09/2011 04:34:52 PM   Manifest-Version = 1.0
```

Listing 3.13 lists many different OSGi headers. You can find a full list of OSGi headers and their descriptions in the official OSGi specification: www.osgi.org/download/r4v43/r4.core.pdf.

help

This command tells the OSGi platform to print all commands that it supports, along with a short description of each command.

Sample usage:

```
tell http osgi help
```

Figure 3.8 shows the sample output from running the `help` OSGi command on the Domino server console.

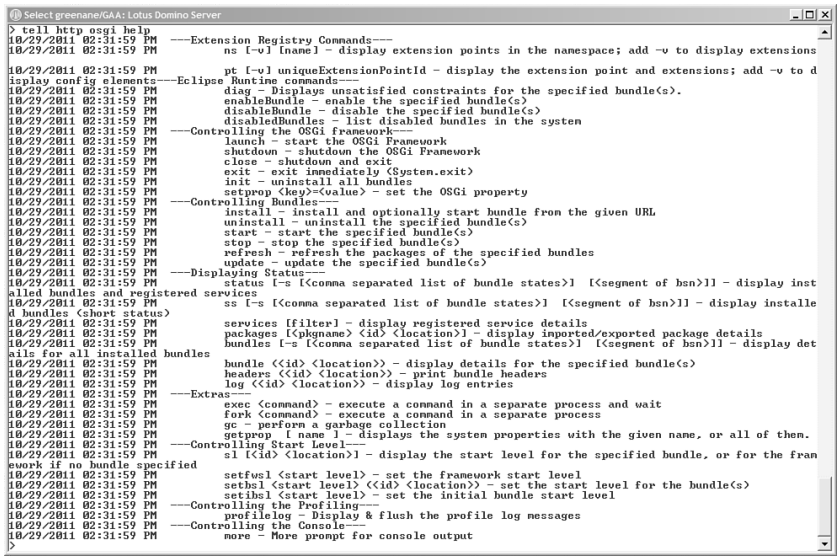

Select greenane/GAA: Lotus Domino Server
> tell http osgi help
10/29/2011 02:31:59 PM ---Extension Registry Commands---
10/29/2011 02:31:59 PM ns [-v] [name] - display extension points in the namespace; add -v to display extensions
10/29/2011 02:31:59 PM pt [-v] uniqueExtensionPointId - display the extension point and extensions; add -v to d
isplay config elements---Eclipse Runtime commands---
10/29/2011 02:31:59 PM diag - Displays unsatisfied constraints for the specified bundle(s).
10/29/2011 02:31:59 PM enableBundle - enable the specified bundle(s)
10/29/2011 02:31:59 PM disableBundle - disable the specified bundle(s)
10/29/2011 02:31:59 PM disabledBundles - list disabled bundles in the system
10/29/2011 02:31:59 PM ---Controlling the OSGi Framework---
10/29/2011 02:31:59 PM launch - start the OSGi Framework
10/29/2011 02:31:59 PM shutdown - shutdown the OSGi Framework
10/29/2011 02:31:59 PM close - shutdown and exit
10/29/2011 02:31:59 PM exit - exit immediately (System.exit)
10/29/2011 02:31:59 PM init - uninstall all bundles
10/29/2011 02:31:59 PM setprop <key>=<value> - set the OSGi property
10/29/2011 02:31:59 PM ---Controlling Bundles---
10/29/2011 02:31:59 PM install - install and optionally start bundle from the given URL
10/29/2011 02:31:59 PM uninstall - uninstall the specified bundle(s)
10/29/2011 02:31:59 PM start - start the specified bundle(s)
10/29/2011 02:31:59 PM stop - stop the specified bundle(s)
10/29/2011 02:31:59 PM refresh - refresh the packages of the specified bundles
10/29/2011 02:31:59 PM update - update the specified bundle(s)
10/29/2011 02:31:59 PM ---Displaying Status---
10/29/2011 02:31:59 PM status [-s [<comma separated list of bundle states>] [<segment of bsn>]] - display inst
alled bundles and registered services
10/29/2011 02:31:59 PM ss [-s [<comma separated list of bundle states>] [<segment of bsn>]] - display installe
d bundles (short status)
10/29/2011 02:31:59 PM services [filter] - display registered service details
10/29/2011 02:31:59 PM packages [<pkgname> <id> <location>] - display imported/exported package details
10/29/2011 02:31:59 PM bundles [-s [<comma separated list of bundle states>] [<segment of bsn>]] - display det
ails for all installed bundles
10/29/2011 02:31:59 PM bundle <<id> <location>> - display details for the specified bundle(s)
10/29/2011 02:31:59 PM headers <<id> <location>> - print bundle headers
10/29/2011 02:31:59 PM log <<id> <location>> - display log entries
10/29/2011 02:31:59 PM ---Extras---
10/29/2011 02:31:59 PM exec <command> - execute a command in a separate process and wait
10/29/2011 02:31:59 PM fork <command> - execute a command in a separate process
10/29/2011 02:31:59 PM gc - perform a garbage collection
10/29/2011 02:31:59 PM getprop [name] - displays the system properties with the given name, or all of them.
10/29/2011 02:31:59 PM ---Controlling Start Level---
10/29/2011 02:31:59 PM sl [<id> <location>] - display the start level for the specified bundle, or for the fram
ework if no bundle specified
10/29/2011 02:31:59 PM setfwsl <start level> - set the framework start level
10/29/2011 02:31:59 PM setbsl <start level> <<id> <location>> - set the start level for the bundle(s)
10/29/2011 02:31:59 PM setibsl <start level> - set the initial bundle start level
10/29/2011 02:31:59 PM ---Controlling the Profiling---
10/29/2011 02:31:59 PM profilelog - Display & flush the profile log messages
10/29/2011 02:31:59 PM ---Controlling the Console---
10/29/2011 02:31:59 PM more - More prompt for console output
>

Figure 3.8 Result of running the help command in the console

How to Launch Notes/Designer Along with the OSGi Console

As of release 8.0 of Lotus Notes and release 8.5.0 of Domino Designer, both applications have been built upon the Eclipse platform. Eclipse itself is built upon the OSGi platform. As of Notes 8.5.1, it is now possible to run XPages applications within the Notes client.

With the emergence of the official XPages extension APIs in Notes/Domino 8.5.2 and the powerful functionality delivered as extensions to XPages (such as the XPages Extension Library), it is becoming more likely that, over time, end users will have Extension Library plug-ins installed into the Notes client platform. Either this can occur directly as a result of the user manually installing them or the plug-ins may be autoprovisioned to the platform via policy directives. Undoubtedly, at some point, XPages developers will need to debug the Notes client to figure out why certain XPages applications or functionality is not working as expected. The first step in such debugging should almost always

be analysis to determine whether the extended plug-ins in question are actually installed and running on the Notes client. The most accurate way to determine whether a plug-in is installed and running within the Notes client (or Domino Designer) is through the use of the OSGi console.

All the commands previously discussed and documented are available both on the Domino server and on the Notes client (and Domino Designer). However, the OSGi console that runs with the Notes client is a pure OSGi console, so it is not necessary to enter the HTTP task prefix required on the Domino server console. In the case of the Notes client OSGi console, it is necessary only to type the actual OSGi command—for example:

```
diag com.ibm.xsp.core
```

as opposed to

```
tell http osgi diag com.ibm.xsp.core.
```

To display the OSGi console for the Notes client or Domino Designer, the user must launch Notes with some additional arguments that tell the core Notes code to launch the console in a separate window when the Notes client is launching.

To do this, the user must navigate to the Notes program directory in a DOS prompt and enter the following DOS command:

```
notes.exe –RPARAMS –console
```

The `RPARAMS` argument for Notes and Domino Designer signals to both programs that the user is entering arguments that are to be redirected to the Eclipse and OSGi runtime. It may be useful to create a new shortcut on your desktop that enables you to easily launch the OSGi Console with Notes or Domino Designer. To do this, simply copy your existing Notes or Domino Designer launch shortcut and modify the `Target` information as follows:

```
C:\Notes85\notes.exe -RPARAMS -console "=C:\Notes85\notes.ini"
```

Here, **C:\Notes85** is the location of your Notes program directory. All the remaining shortcut information should be the same as your existing Notes or Domino Designer shortcut, as shown in Figure 3.9.

Arguments after the –`RPARAMS` parameter are sent to the Eclipse and OSGi runtimes for processing. Users should be aware that closing the Notes OSGi console window directly is not supported and can cause undesired behavior, such as causing the Notes program to hang. All instances of Notes, Domino Designer, and Domino Administrator should be shut down before running this command. Figure 3.10 shows the OSGi console running with Notes.

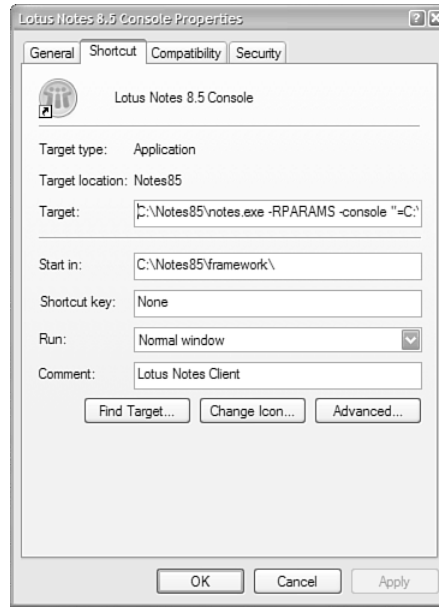


Figure 3.9 Shortcut to launch Notes with the OSGi console

Figure 3.10 Notes client running with the OSGi console

You can find more information on specific OSGi commands at these sites:

http://eclipse.org/equinox/

http://fusesource.com/docs/esb/4.1/command_ref/ESBosgi.html

Common Console Commands You Should Know

Beyond the realm of OSGi and the XSP command manager, the Domino server has a rich set of commands. Knowing at least a subset of them will greatly benefit any budding XPages developer or administrator. Table 3.4 lists some of the more commonly used commands.

Table 3.4 Common Domino Server Commands

Command	Description
help	Displays a list of server commands, with a brief description
load [task name]	Loads the named Domino server task
load [task name] -?	Gets help for the specified command
quit	Tells the Domino server to shut down
restart server	Tells the Domino server to shut down completely and restart
tell [task name] quit	Tells the named Domino server task to shut down
restart task [task name]	Tells the name Domino server task to restart
show server	Prints all basic statistics relating to the server to the console
show conf [notes.ini variable]	Prints the value of the server's **notes.ini** variable to the console
set conf [notes.ini variable=value]	Sets the value of the server's **notes.ini** variable to the specified value
tell adminp [options]	Performs various administrative tasks on the Domino server
load chronos [options]	Updates full-text indexes that are marked to be updated hourly or daily
load updall [path] [options]	Updates the view indexes and the full-text index for the specified database (or for all databases, if one is not provided)
load design [source] [target] [options]	Updates all databases with design updates from their master templates
load fixup [path] [options]	Locates and fixes corrupted databases on the server
show allports	Shows all enabled and disabled ports on the server
show diskspace	Displays the amount of free disk space on the server
show heartbeat	Displays a value if the server is responding

Command	Description
show memory	Displays the amount of RAM available on the server
show tasks	Displays the names of all the Domino server tasks running

You can obtain a much more extensive list of server commands by reading the Domino Administrator help, which is installed on the Domino server under the **help** directory.

help

This command displays a list of server console commands, with a brief description of each command, the command's arguments, and a sample of the syntax of each command.

Sample usage:

```
help
```

Figure 3.11 shows a subset of the sample output from running the `help` command on the Domino server console.

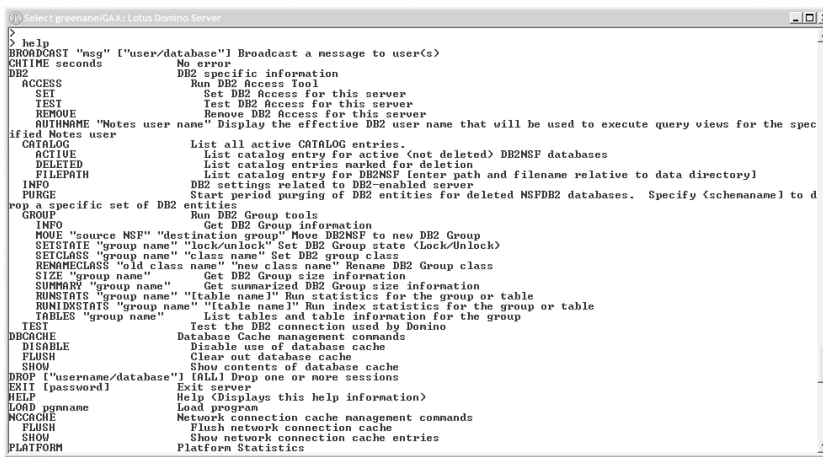


Figure 3.11 Result of running the help command on the Domino server console

load [task-name]

This command loads and starts the specified server task. It loads tasks that run continually until the server is stopped or loads a task that runs until complete. Further task arguments can be passed to the task as needed. This command is convenient because it

enables developers and administrators to dynamically start server tasks without needing to restart the entire server. For example, the HTTP task can be started without affecting other tasks running on the Domino server.

Sample usage:

```
load http
```

In this example, the HTTP task is loaded, allowing the Domino server to act as a HTTP server.

Listing 3.14 shows the console output of running the previous command.

Listing 3.14 Result of Running the load http Command on the Domino Server Console

```
> load http
09/19/2011 08:05:03 PM  HTTP Server: Using Web Configuration View
09/19/2011 08:05:07 PM  JVM: Java Virtual Machine initialized.
09/19/2011 08:05:07 PM  HTTP Server: Java Virtual Machine loaded
09/19/2011 08:05:07 PM  HTTP Server: DSAPI Domino Off-Line Services
HTTP extension Loaded successfully
09/19/2011 08:05:12 PM  XSP Command Manager initialized
09/19/2011 08:05:12 PM  HTTP Server: Started
```

load [task-name] -?

This command displays help information that relates to the task specified. In general, the help information lists any options or flags that can or should be passed to the task.

Sample usage:

```
load chronos -?
```

Listing 3.15 shows the sample output from running the help command against a specific task name.

Listing 3.15 Sample Output from Running the help Command Against the Chronos Task

```
> load chronos -?
>
Purpose:   Performs automatic hourly and daily full text indexing.
Usage:     Load CHRONOS [options]...
[options]:
hourly     Update all hourly full text indexes.
daily      Update all daily  full text indexes.
```

quit

This command stops the server. The server shuts down completely after running this command.

Sample usage:

```
quit
```

Figure 3.12 shows output from running the `quit` command on the Domino server console.

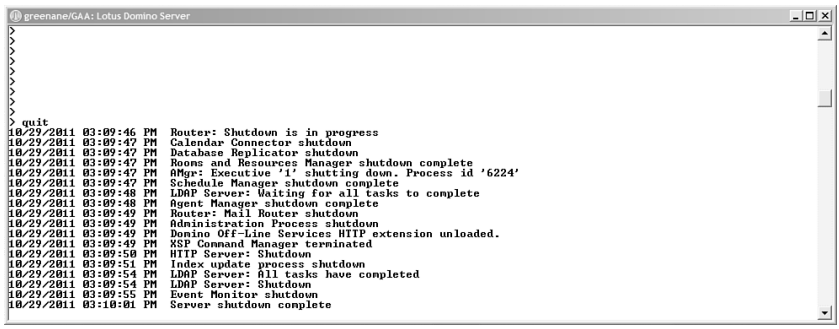


Figure 3.12 Result of running the quit command on the Domino server console

restart server

This command stops the server completely and then restarts the server after a brief delay.

Sample usage:

```
restart server
```

Figure 3.13 shows output from running the `restart server` command on the Domino server console.

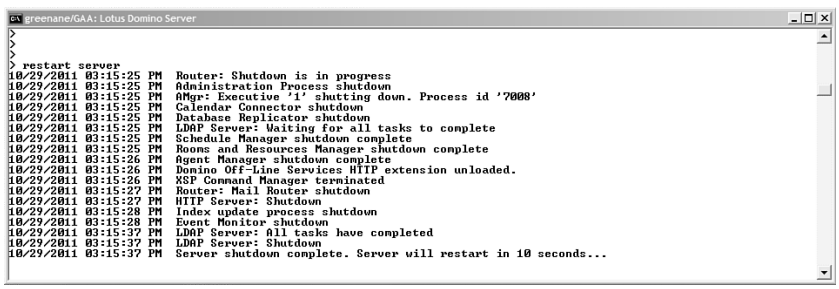


Figure 3.13 Result of running the restart server command on the Domino server console

tell [task-name] quit

This command stops the named task. All other server tasks remain in their current state.

Sample usage:

```
tell http quit
```

Listing 3.16 shows the sample console output after executing the quit command on a specific task.

Listing 3.16 Domino Server Console Output from Running the quit Command on the HTTP Task

```
> tell http quit
10/19/2011 08:50:21 PM  Domino Off-Line Services HTTP extension
unloaded.
10/19/2011 08:50:21 PM  XSP Command Manager terminated
10/19/2011 08:50:22 PM  HTTP Server: Shutdown
```

This sample terminates the HTTP task so that the Domino web server and all other HTTP functions are shut down. XPages developers might find this useful if the web server needs to be quickly and independently recycled—say, to reread and apply new XSP runtime settings.

restart task [task-name]

This command stops and restarts the named task. All other server tasks remain in their current state. XPages developers will find this to be a particularly powerful command because it enables them to completely and quickly restart the XPages runtime. This is of particular importance when debugging OSGi bundles running on the server. Chapter 6 discusses this in greater detail.

Sample usage:

```
restart task http
```

Listing 3.17 shows the Domino server output that results from restarting a specific task.

Listing 3.17 Sample Output from Running the restart task http Command

```
> restart task http
10/19/2011 09:03:10 PM  Domino Off-Line Services HTTP extension
unloaded.
10/19/2011 09:03:10 PM  XSP Command Manager terminated
10/19/2011 09:03:11 PM  HTTP Server: Shutdown
10/19/2011 09:03:13 PM  HTTP Server: Using Web Configuration View
10/19/2011 09:03:16 PM  JVM: Java Virtual Machine initialized.
10/19/2011 09:03:16 PM  HTTP Server: Java Virtual Machine loaded
10/19/2011 09:03:16 PM  HTTP Server: DSAPI Domino Off-Line Services
```

```
HTTP extension Loaded successfully
10/19/2011 09:03:19 PM  XSP Command Manager initialized
10/19/2011 09:03:19 PM  HTTP Server: Started
```

show server

This command prints all the basic information to the server's console, including (but not limited to) the server's name, data directory location, amount of time since the server was started, and total number of transactions completed by the server since it was started.

Sample usage:

```
show server
```

Listing 3.18 shows sample output from executing the `show server` command on the Domino server console.

Listing 3.18 Sample Output from the show server Command

```
> show server
  Lotus Domino (r) Server (Build V853_06302011 for Windows/32)
09/14/2011 07:28:42 PM
Server name:              greenane/GAA - Greenane
Domain name:              ibm
Server directory:         C:\Program Files\IBM\Lotus\Domino\data
Partition:                C.Program Files.IBM.Lotus.Domino.data
Elapsed time:             1 day 01:38:37
Transactions/minute:      Last minute: 10; Last hour: 200; Peak: 997
Peak # of sessions:       60 at 09/14/2011 06:50:06 PM
Transactions: 4524        Max. concurrent: 40
ThreadPool Threads:       40  (TCPIP Port)
Availability Index:       100 (state: AVAILABLE)
Mail Tracking:            Not Enabled
Mail Journalling:         Not Enabled
Number of Mailboxes:      10
Pending mail: 0           Dead mail: 0
Waiting Tasks:            0
DAOS:                     Not Enabled
Transactional Logging:    Not Enabled
Fault Recovery:           Not Enabled
Activity Logging:         Not Enabled
Server Controller:        Not Enabled
Diagnostic Directory:     C:\Program Files\IBM\Lotus\Domino\data\
➥IBM_TECHNICAL_SUPPORT
Console Logging:          Enabled (10240K)
Console Log File:         C:\Program Files\IBM\Lotus\Domino\data\
➥IBM_TECHNICAL_SUPPORT\console.log
DB2 Server:              Not Enabled
```

show conf [notes.ini variable]

This command enables the developer or administrator to examine the value of any given **notes.ini** variable without needing to physically open the **notes.ini** file residing in the Domino server's program directory. This is a powerful command because it allows developers and administrators alike to view the values of **notes.ini** variables that the runtime is using without needing to wade through the array of variables present in the Domino server's **notes.ini** file.

Sample usage:

```
show conf HTTPJVMMaxHeapSize
```

Listing 3.19 shows sample output as a result of executing the `show conf` command on the Domino server console.

Listing 3.19 Result of Executing the show conf Command Using the HTTPJVMMaxHeapSize Variable

```
> show conf HTTPJVMMAxHeapSize
HTTPJVMMAXHEAPSIZE=256M
```

set conf [notes.ini variable=value]

This command enables developers and administrators to quickly and easily set a **notes.ini** variable in the Domino server's **notes.ini** without actually physically opening the file and editing the value. This command is particularly useful because it enables users to set the **notes.ini** variable while the server is running. A typical use case for this command is one in which the administrator wants to increase the minimum Java heap size of the HTTP task's JVM without worrying about accidentally overwriting any other server settings that may have been written to **notes.ini** in the time the file was open for editing.

Sample usage:

```
set conf JavaMinHeapSize=64M
```

Figure 3.14 shows how the `JavaMinHeapSize notes.ini` variable can be reset using the `set conf` command and displays how the setting is applied by restarting the HTTP task.

tell adminp [options]

This command performs various automated administration tasks on the server. A wide range of options can be passed to this task; you can obtain the complete listing of `adminp` options from the Lotus Domino Administrator help, installed in the **help** directory of the Domino server.

Sample usage:

```
tell adminp show databases
```

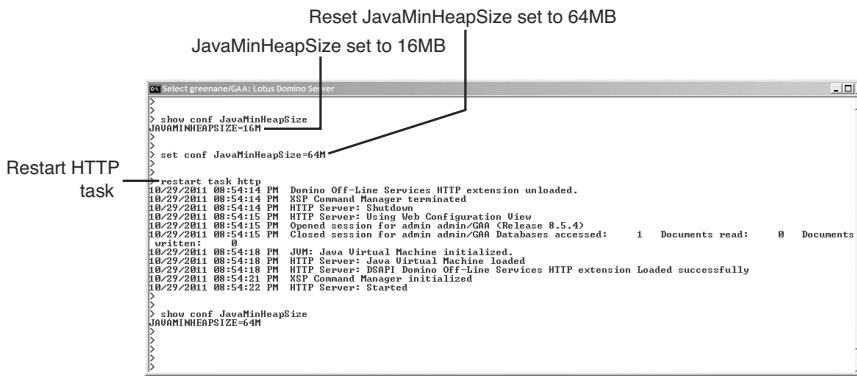


Figure 3.14 Result of running the set conf command on the JavaMinHeapSize notes.ini variable

Listing 3.20 shows the output from executing `adminp` with the `show databases` option specified.

Listing 3.20 Result of Executing the adminp Task on the Domino Server Console

```
> tell adminp show databases
10/20/2011 04:11:32 PM  Admin Process: These databases have greenane/
➥GAA designated as their Administration Server.
10/20/2011 04:11:32 PM  Title: Administration Requests Path: admin4.nsf
10/20/2011 04:11:32 PM  Title: CPP FreeBusy WebService Path:
➥cppfbws.nsf
10/20/2011 04:11:32 PM  Title: Domino Directory Cache (6) Path:
➥dbdirman.nsf
10/20/2011 04:11:32 PM  Title: Offline Services Path: doladmin.nsf
10/20/2011 04:11:32 PM  Title: greenane's Log Path: log.nsf
10/20/2011 04:11:32 PM  Title: admin admin Path: mail\aadmin.nsf
10/20/2011 04:11:32 PM  Title: Eileen Leonard Path: mail\eleonard.nsf
10/20/2011 04:11:32 PM  Title: Frank Adams Path: mail\fadams.nsf
```

load chronos [options]

This command loads the `chronos` task on the Domino server. The task is responsible for updating the full-text indexes of databases that are marked to be updated daily or hourly. This is useful to XPage developers when the full-text index of a database is needed to test particular functionality. This task enables developers to force the creation or update of the index without needing to modify the indexing schedule.

Sample usage:

```
load chronos hourly
```

Listing 3.21 shows the sample output from running the chronos task.

Listing 3.21 Sample Console Output from Running the chronos Task

```
>load chronos hourly
09/14/2011 08:35:06 PM  Chronos: Performing hourly full text indexing
09/14/2011 08:35:09 PM  Chronos: Full text indexer terminating
```

load updall [path] [options]

This command updates all changed views and/or all full-text indexes within the given database or all databases on the server. Obviously, this is quite useful if you are working with `FTSearch` features in your XPages application because testing and debugging requires an up-to-date full-text index.

You can pass a wide range of options to this task. The Lotus Domino Administrator help, installed in the **help** directory of the Domino Server, includes a complete listing of `adminp` options.

Sample usage:

```
load updall XPagesSBT.nsf –f
```

Listing 3.22 shows the output received on the Domino server console from running the `updall` task on the Domino server.

Listing 3.22 Sample Console Output from Running the updall Task to Update Full-Text Indexes on a Specified Application

```
> load updall XPagesSBT.nsf -f
09/14/2011 08:44:39 PM  Index update process started:  XPagesSBT.nsf -f
09/14/2011 08:44:39 PM  Updating views in C:\Program
➥Files\IBM\Lotus\Domino\data\XPagesSBT.nsf
09/14/2011 08:44:39 PM  Index update process shutdown
```

load design [source] [target] [options]

This command updates all databases on the server with design updates from their master template. This command can be quite useful when an administrator has accidentally modified the design of a particular database and needs to update the design of that database from the master template outside the regular design update schedule.

Sample usage:

```
load design rossacussane.swg.myco.com greenane.swg.myco.com –f
➥XPagesSBT.nsf
```

Listing 3.23 shows the Domino server console output received from executing the design task on the Domino server.

Listing 3.23 Sample Console Output from Running the Design Task

```
> load design rossacussane.swg.myco.com greenane.swg.myco.com
➥-f XPagesSBT.nsf
09/14/2011 08:54:52 PM  Database Designer started
09/14/2011 08:54:52 PM  Opened session for rossacussane/GAA (Release
➥8.5.3)
09/14/2011 08:54:55 PM  Closed session for rossacussane/GAA Databases
accessed: 3   Documents read: 0   Documents written:  0
09/14/2011 08:54:55 PM  Opened session for greenane/GAA (Release 8.5.3)
➥09/14/2011
08:54:55 PM  Closed session for greenane/GAA Databases accessed:
1   Documents read: 0
Documents written:  0
09/14/2011 08:54:55 PM  Opened session for greenane/GAA (Release 8.5.3)
09/14/2011 08:54:55 PM  Database Designer shutdown
09/14/2011 08:54:55 PM  Closed session for greenane/GAA Databases
➥accessed: 1
Documents read: 0   Documents written:  0
```

load fixup [path] [options]

This command runs the `fixup` task on the specified database or on all databases on the server. The `fixup` task scans for databases that contain inconsistencies from partially written operations that may have occurred during a previous failure, such as a hardware failure or a crash. You can pass a wide range of options to this task. The complete listing of `adminp` options is available from the Lotus Domino Administrator help, installed in the **help** directory of the Domino server.

Sample usage:

```
load fixup XPagesSBT.nsf -l
```

Listing 3.24 shows the result of running the `fixup` command against a particular database on the Domino server.

Listing 3.24 Sample Console Output from Running the fixup Command

```
> load fixup XPagesSBT.nsf -l
 09/14/2011 09:08:55 PM  Database Fixup: Started:  XPagesSBT.nsf -l
 09/14/2011 09:08:55 PM  Checking database C:\Program Files\IBM\
➥Lotus\Domino\data\XPagesSBT.nsf
 09/14/2011 09:08:55 PM  Performing consistency check on
➥XPagesSBT.nsf...
 09/14/2011 09:08:56 PM  Completed consistency check on XPagesSBT.nsf
 09/14/2011 09:08:56 PM  Performing consistency check on views in
➥database XPagesSBT.nsf
```

```
 09/14/2011 09:08:56 PM  Completed consistency check on views in
➥database XPagesSBT.nsf
 09/14/2011 09:08:56 PM  Database Fixup: Shutdown
```

show tasks

This command shows the names of all the Domino Server tasks that are running on the server. Administrators will find this useful for determining which tasks are running on any given server.

Sample usage:

```
show tasks
```

Figure 3.15 shows the sample output received when running the `show tasks` command on a Domino server.

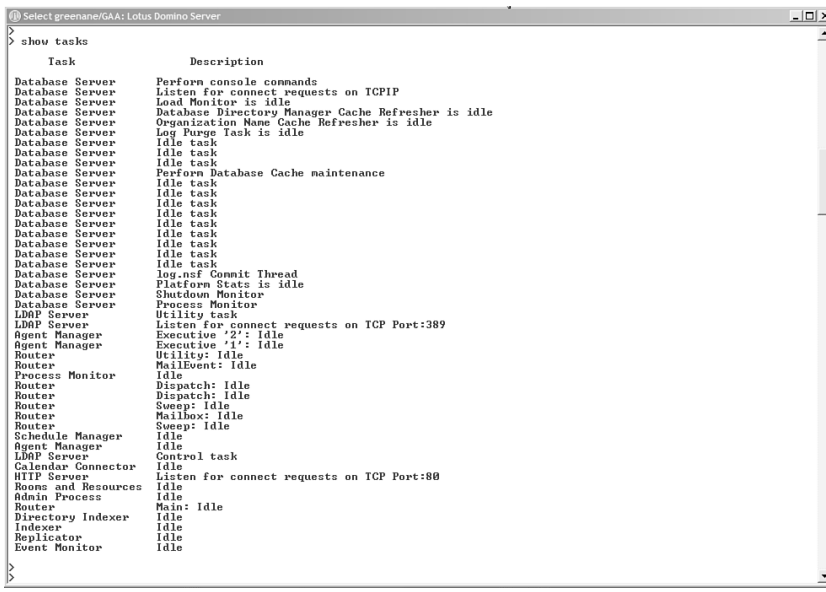


Figure 3.15 Result of running the show tasks command on the Domino server console

show allports

This command prints the configuration of all enabled and disabled ports on the server.

Sample usage:

```
show allports
```

Listing 3.25 shows the result of executing the `show allports` command on the Domino server console.

Listing 3.25 Sample Console Output from Running the show allports Command

```
> show allports
Enabled Ports:
TCPIP=TCP, 0, 15, 0
Disabled Ports:
LAN0=NETBIOS, 0, 15, 0
LAN1=NETBIOS, 1, 15, 0
LAN2=NETBIOS, 2, 15, 0
LAN3=NETBIOS, 3, 15, 0
LAN4=NETBIOS, 4, 15, 0
LAN5=NETBIOS, 5, 15, 0
LAN6=NETBIOS, 6, 15, 0
LAN7=NETBIOS, 7, 15, 0
LAN8=NETBIOS, 8, 15, 0
```

show diskspace

This command prints the amount of disk space available on the server.

Sample usage:

```
show diskspace
```

Listing 3.26 displays the results from executing the `show diskspace` command.

Listing 3.26 Sample Console Output from the show diskspace Command

```
> show diskspace
Available disk space 83,342,319,616 bytes
```

show heartbeat

This command prints a value to the console if the server is still responding.

Sample usage:

```
show heartbeat
```

Listing 3.27 shows the result of running the show `heartbeat` command on the Domino server console.

Listing 3.27 Sample Console Output for the show heartbeat Command

```
> show heartbeat
greenane/GAA's elapsed time: 100827 seconds
```

Conclusion

This chapter outlined the most relevant commands available to you as an XPages developer via the Domino server console and the Notes OSGi console. Over time, these commands will undoubtedly prove to be powerful tools in the resolution of issues. Although executing the commands is a relatively simple exercise, the result they yield will often lead you directly to the source of a problem. These commands will also improve productivity by reducing the amount of time needed to test an application. For example, scheduled tasks, such as indexing operations, can be run on demand using these commands, without having to wait for tasks to execute on schedule. Make the most of them!

Index

D

E

F

G

H

I

J-K

L

M

N

O

P

Q–R

S

T

U

V-W

X-Y-Z