BEA WebLogic Integration™

Programming Management Applications for B2B Integration

Release 7.0
Document Date: June 2002

Copyright

Copyright © 2002 BEA Systems, Inc. All Rights Reserved.

Restricted Rights Legend

This software and documentation is subject to and made available only pursuant to the terms of the BEA Systems License Agreement and may be used or copied only in accordance with the terms of that agreement. It is against the law to copy the software except as specifically allowed in the agreement. This document may not, in whole or in part, be copied photocopied, reproduced, translated, or reduced to any electronic medium or machine readable form without prior consent, in writing, from BEA Systems, Inc.

Use, duplication or disclosure by the U.S. Government is subject to restrictions set forth in the BEA Systems License Agreement and in subparagraph (c)(1) of the Commercial Computer Software-Restricted Rights Clause at FAR 52.227-19; subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013, subparagraph (d) of the Commercial Computer Software--Licensing clause at NASA FAR supplement 16-52.227-86; or their equivalent.

Information in this document is subject to change without notice and does not represent a commitment on the part of BEA Systems. THE SOFTWARE AND DOCUMENTATION ARE PROVIDED “AS IS” WITHOUT WARRANTY OF ANY KIND INCLUDING WITHOUT LIMITATION, ANY WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. FURTHER, BEA Systems DOES NOT WARRANT, GUARANTEE, OR MAKE ANY REPRESENTATIONS REGARDING THE USE, OR THE RESULTS OF THE USE, OF THE SOFTWARE OR WRITTEN MATERIAL IN TERMS OF CORRECTNESS, ACCURACY, RELIABILITY, OR OTHERWISE.

Trademarks or Service Marks

BEA, Jolt, Tuxedo, and WebLogic are registered trademarks of BEA Systems, Inc. BEA Builder, BEA Campaign Manager for WebLogic, BEA eLink, BEA Manager, BEA WebLogic Commerce Server, BEA WebLogic Enterprise, BEA WebLogic Enterprise Platform, BEA WebLogic Express, BEA WebLogic Integration, BEA WebLogic Personalization Server, BEA WebLogic Platform, BEA WebLogic Portal, BEA WebLogic Server, BEA WebLogic Workshop and How Business Becomes E-Business are trademarks of BEA Systems, Inc.

All other trademarks are the property of their respective companies.

Programming Management Applications for B2B Integration

Part Number	Date	Software Version
N/A	June 2002	7.0

Contents

About This Document

This document describes how to develop applications to monitor run-time activities for the BEA WebLogic Integration™ B2B integration system.

This document is organized as follows:

- Chapter 1, "Introduction," provides an overview to developing applications for the WebLogic Integration environment.
- Chapter 2, "Developing Management Applications," describes how to create applications that monitor run-time B2B integration activities using WebLogic Integration Managed Beans (MBeans).

What You Need to Know

This document is intended primarily for:

- Business process designers who use the WebLogic Integrator Studio to design workflows that can be used with a B2B integration environment.
- Application developers who write Java applications that monitor run-time statistics in a B2B integration environment.
- System administrators who set up and administer B2B integration applications.

For an overview of the B2B integration architecture, see "Overview" in *Introducing B2B Integration*.

e-docs Web Site

BEA product documentation is available at the following location:

http://e-docs.bea.com

How to Print this Document

You are reading the PDF version of this document, either online or a printout. You can print the entire document or any portion of the document from Adobe Acrobat Reader. If you do not have the Adobe Acrobat Reader, you can get it for free from the Adobe Web site at the following location:

http://www.adobe.com

Alternatively, you can print a copy of the HTML version of this document from a Web browser, one file at a time, by using the File→Print option on your Web browser.

Related Information

For more information about Java 2 Enterprise Edition (J2EE), Extended Markup Language (XML), and Java programming, see the Javasoft Web site at the following URL:

http://java.sun.com

Contact Us!

Your feedback about the WebLogic Integration documentation is important to us. Send us e-mail at docsupport@bea.com if you have questions or comments. Your comments will be reviewed directly by the BEA professionals who create and update the WebLogic Integration documentation.

In your e-mail message, please indicate that you are using the documentation for BEA WebLogic Integration Release 7.0.

If you have any questions about this release of WebLogic Integration, or if you have problems installing and running WebLogic Integration , contact BEA Customer Support through BEA WebSUPPORT at the following location:

`http://www.bea.com`

When contacting Customer Support, be prepared to provide the following information:

- Your name, e-mail address, phone number, and fax number
- Your company name and company address
- Your machine type and authorization codes
- The name and version of the product you are using
- A description of the problem and the content of pertinent error messages

Documentation Conventions

The following documentation conventions are used throughout this document.

Convention	Item
boldface text	Indicates terms defined in the glossary.
Ctrl+Tab	Indicates that you must press two or more keys simultaneously.

Convention	Item
italics	Indicates emphasis or book titles.

`monospace text`	Indicates code samples, commands and their options, data structures and their members, data types, directories, and filenames and their extensions. Monospace text also indicates text that you must enter from the keyboard.
	Examples:
	`#include <iostream.h> void main ( ) the pointer psz`
	`chmod u+w *`
	`\tux\data\ap`
	`.doc`
	`tux.doc`
	`BITMAP`
	`float`

`monospace boldface text`	Identifies significant words in code. *Example*: `void` **`commit`** `( )`
`monospace italic text`	Identifies variables in code. *Example*: `String` *`expr`*
UPPERCASE TEXT	Indicates device names, environment variables, and logical operators. *Example*s: LPT1 SIGNON OR
{ }	Indicates a set of choices in a syntax line. The braces themselves should never be typed.
[ ]	Indicates optional items in a syntax line. The brackets themselves should never be typed. *Example*: `buildobjclient [-v] [-o name] [-f` *`file-list`*`]... [-l` *`file-list`*`]...`
|	Separates mutually exclusive choices in a syntax line. The symbol itself should never be typed.

Convention	Item
. . .	Indicates one of the following in a command line: ■ That an argument can be repeated several times in a command line ■ That the statement omits additional optional arguments ■ That you can enter additional parameters, values, or other information The ellipsis itself should never be typed. *Example*: `buildobjclient [-v] [-o name] [-f file-list]... [-l file-list]...`
. . .	Indicates the omission of items from a code example or from a syntax line. The vertical ellipsis itself should never be typed.

CHAPTER

1 Introduction

The following sections provide an introduction to B2B integration management applications:

WebLogic Integration Applications

This document introduces the following topics, which are related to WebLogic Integration applications:

- The use of MBeans to monitor B2B integration activities
- Management applications

MBeans are one of three types of component applications available in WebLogic Integration. In addition to MBeans, WebLogic Integration allows you to use the following kinds of applications:

- Logic plug-ins, for customized routing, filtering, and information processing, as described in *Programming Logic Plug-Ins for B2B Integration*.
- Messaging applications, as described in *Programming Messaging Applications for B2B Integration*.

For an introduction to the B2B integration system, see *Introducing B2B Integration*.

Management Applications

WebLogic Integration management applications monitor run-time activities that support B2B integration, such as message traffic and conversation statistics. WebLogic Integration provides a unified administrative tool, the WebLogic Integration B2B Console, that monitors the B2B engine at run time. In addition to the system tools provided by BEA, developers can create custom management applications that provide comparable monitoring functionality.

Developers can implement a variety of management applications to:

- Monitor activities on a WebLogic Server instance
- Provide run-time statistics for server instances, delivery channels, business transaction definitions, trading partners, and business messages

For all management applications, WebLogic Integration provides a set of Managed Beans, or *MBeans*, which are special JavaBeans with attributes and methods for management operations. These MBeans are BEA implementations of the Java Management Extensions (JMX) Managed Beans API, which is defined in the Java Management Extensions Specification published by Sun Microsystems, Inc.

CHAPTER

2 Developing Management Applications

The following sections describe how to create a management application that monitors run-time activity on a WebLogic Integration B2B engine:

- Working with Management Tools
- Working with MBeans and the MBean Server
- Programming Management Applications

Working with Management Tools

The WebLogic Integration B2B Console provides run-time monitoring of B2B integration activities. If you want additional tools, you can create custom management applications that provide the same monitoring information displayed by the B2B Console.

For example, custom management applications may be desirable for the following reasons:

- Read-only access to real-time statistics, such as the number of messages exchanged in a particular conversation or the number of messages received by B2B integration

- Some administrative tasks, such as shutting down a particular delivery channel or WebLogic Server, or leaving or terminating a particular conversation

Note: Custom management applications cannot update the WebLogic Integration repository. To update the repository, use one of the following tools:

- WebLogic Integration B2B Console, as described in the *Online Help for the WebLogic Integration B2B Console*
- Bulk Loader, as described in "Working with the Bulk Loader" in *Administering B2B Integration*

Working with MBeans and the MBean Server

WebLogic Integration provides the application programming interfaces (APIs) needed to create custom management applications that monitor run-time activity on B2B engines. The B2B Console also use these APIs to provide real-time monitoring information.

These APIs consist of sets of Java Management Extensions (JMX) Managed Beans, or *MBeans*, which are special JavaBeans with attributes and methods for management operations. For more information about JMX, particularly the use of the JMX API (including the MBean Server and MBeans), see the Java Management Extensions Specification published by Sun Microsystems, Inc., at the following URL:

`http://www.java.sun.com/products/JavaManagement/index.html`

MBean Package

WebLogic Integration provides the `com.bea.b2b.management` package for creating custom management applications. This package provides the following:

- MBeans that enable you to monitor run-time activity on B2B engines

- The `ManagementException` class for handling errors that occur in a run-time management application

For detailed information about this package, see the *BEA WebLogic Integration Javadoc*.

In this release, all MBeans are implemented as Standard MBeans, which make up a class that implements its own MBean interface. WebLogic Integration MBeans are not implemented as remote MBeans. Therefore all management applications must reside on the B2B engine being monitored.

MBeans

The following table describes the WebLogic Integration MBeans.

Table 2-1 WebLogic Integration MBeans

Label	Description
`WLCMBean`	Represents an instance of B2B integration. Used for monitoring that instance at run time.
`DeliveryChannelMBean`	Represents a delivery channel. Used for monitoring delivery channels on a B2B integration system at run time.
`ConversationMBean`	Represents a business conversation managed by the Transaction Manager on the B2B integration instance. Used for monitoring active transactions within a delivery channel.
`TradingPartnerSessionMBean`	Represents a session with a trading partner. Used for monitoring trading partners.
`MessageMBean`	Represents a message in a conversation. Used for monitoring messages.
`CollaborationAgreementMBean`	Represents a collaboration agreement. A collaboration agreement represents a technical agreement between two parties on how they plan to communicate with each other using a specific protocol.

Note: For WebLogic Integration Releases 2.1 and 7.0, and BEA WebLogic Collaborate 2.0, all MBeans are centralized in this package. In previous versions of WebLogic Collaborate, MBeans are split between the c-hub and c-enabler packages. If you are upgrading from WebLogic Collaborate 1.0 or 1.0.1, you must modify any management applications you have written to make use of the new MBeans. The following table outlines changes between Release 1.x MBeans and Release 2.x MBeans.

Table 2-2 MBean Labels

Release 1.x Label	Release 2.x Label
EnablerMBean	WLCMBean
HubMBean	WLCMBean
CSpaceMBean	DeliveryChannelMBean
ConversationMBean	ConversationMBean
GlobalConversationMBean	ConversationMBean
EnablerSessionMBean	TradingPartnerSessionMBean
CollaboratorMBean	TradingPartnerSessionMBean
MessageMBean	MessageMBean

MBean Server Implementation

When your applications uses an MBean, WebLogic Integration registers the MBean with the MBean Server instance that is created when a WebLogic Server instance starts. For more information about the WebLogic Server MBean Server, see *Using WebLogic Server JMX Services* at the following URL:

http://e-docs.bea.com/wls/docs70/jmx/index.html

Programming Management Applications

To access WebLogic Integration MBeans using the JMX API, a Java application must perform the following steps:

- Step 1: Import the Necessary Packages
- Step 2: Get a Reference to the MBeanServer Object
- Step 3: Construct an ObjectName Object
- Step 4: Query the MBean Server
- Step 5: Read the Attributes of the MBean
- Step 6: Navigate Across MBeans
- Step 7: Handle Exceptions

If you If you want to find info, you need an app that loops through all the nodes in the cluster. Each indiv node should function as standalone node. You have to do iterating

Note: If you are deploying your management application in a multinode cluster, you application needs logic that loops through all the nodes individually. Regarding management applications, each machine continues to function as a a standalone node.

Step 1: Import the Necessary Packages

To work with MBeans, a management application must import the necessary packages. At a minimum, the application must import the packages described in the following table.

Table 2-3 Packages that Must Be Imported

Label	Description
`javax.management.*;`	Required for JMX MBeans, as mandated in the Java Management Extensions Specification published by Sun Microsystems, Inc.
`javax.naming.*;`	Required for retrieving the MBean server object using JNDI lookup. Only the following packages are required: ■ `javax.naming.Context` ■ `javax.naming.InitialContext`
`com.bea.b2b.management.*`	Required for all management applications.
`weblogic.management.*`	Required for all management implementations. Gets `MBeanServer` and `MBeanHome`.

The code in the following listing imports the necessary packages for a management application.

Listing 2-1 Importing Packages for a Management Application

```
import javax.management.*;
import javax.naming.Context;
import javax.naming.InitialContext;
import com.bea.b2b.management.*;
import weblogic.management.*;
```

Step 2: Get a Reference to the MBeanServer Object

WebLogic Integration uses the MBeanS erver that is instantiated when an instance of WebLogic Server is started. To get a reference to the MBean Server, the `MBeanHome` of that server is required. The `MBeanHome` of the MBean Server is available from the server's JNDI tree at:

`weblogic.management.MBeanHome.JNDI_NAME.serverName`

An administration server publishes an `MBeanHome` for each server in the domain on its JNDI tree. The administration `MBeanHome` is available only from the JNDI tree of the administration server at:

`weblogic.manageme nt.MBeanHome.ADMIN_JNDI_NAME`

The underlying MBean Server for any `MBeanHome` can be obtained by invoking the `getMBeanServer()` method on that `MBeanHome`.

The following code shows an example of a JNDI lookup for the administration server `MBeanHome`.

Listing 2-2 Getting a Reference to the MBean Server Object

```
import javax.naming.Context;
import javax.naming.NamingException;
import javax.naming.AuthenticationException;
import javax.naming.CommunicationException;
import weblogic.jndi.Environment;          ...
import weblogic.management.MBeanHome;
        ...
    MbeanHome home = null;
    try {
      Environment env = new Environment();
      ctx = env.getInitialContext();
      home = (MBeanHome) ctx.lookup(MBeanHome.ADMIN_JNDI_NAME);
      RemoteMBeanServer server = home.getMBeanServer();
    }
    catch (AuthenticationException e) {
        ... //Error handling
    } catch (CommunicationException e) {
        ... //Error handling
    } catch (NamingException e) {
```

```
        ... //Error handling
    }
```

Step 3: Construct an ObjectName Object

MBeans are identified by unique object names inside the MBean Server. The `ObjectName` class represents an object name.

For WebLogic Integration Releases 7.0 and 2.1, and WebLogic Collaborate 2.0, only `WLCMBean` is registered with the MBean Server; all other MBeans can be retrieved from `WLCMBean`. `WLCMBean` has three attributes: a name, a type, and a domain. These attributes are reflected in the MBean's JMX Object Name. The Object Name is the unique identifier for a given MBean across all domains, and has the following structure:

`domain name``:Name=`*`name`*`,Type=`*`type`*`[,`*`attr`*`=`*`value`*`]...`

The value of *`name`* is unique for a given domain and a given type. For example:

```
mydomain:Name=WLC,Type=WLC
ObjectName objectName = new ObjectName("WLC", "WLC", "mydomain");
```

For MBeans, object names can also be used for query operations in which object name expressions are used. The MBean Server uses pattern matching on the object names of the registered MBeans. The matching syntax is consistent with file globing, which is described in the Java Management Extensions Specification published by Sun Microsystems, Inc.:

- An asterisk (`*`) matches any character sequence.
- A question mark (`?`) matches a single character.

Step 4: Query the MBean Server

After constructing an object name expression, an application queries the MBean Server by passing the `ObjectName` object corresponding to the expression. To retrieve the set of registered MBeans, the names of which satisfy an object name expression, use the following method:

`javax.management.MBeanServer.queryNames()`

The MBean Server returns a set of objects that satisfy the query criteria. Note that these are `ObjectName` objects that *represent* MBeans; they are *not* direct references to the MBeans themselves.

Step 5: Read the Attributes of the MBean

Use the `ObjectName` instance, obtained in the previous step, to access other MBeans, provided that the `ObjectName` has one or more attributes of the MBean type. To read the attributes of an MBean, use the following method, passing the `ObjectName` object as a parameter:

`javax.management.MBeanServer.getAttribute()`

After you call the `getAttribute` method by passing the `ObjectName` object for the first MBean, you can get direct references to other MBean instances.

The code in the following listing retrieves a set of attributes associated with WLCMBean.

Listing 2-3 Retrieving Conversation Attributes

```
MBeanHome home = Admin.getMBeanHome();
server = home.getMBeanServer();
ObjectName objectName = new ObjectName("WLC", "WLC", "mydomain");
beans = server.queryNames(objectName, null);
Iterator it = beans.iterator();
while (it != null && it.hasNext())
{
 //Should be only one
  obj = (ObjectName)it.next();
  break;
```

```
}
if (obj != null)
{
  Date startTime = (Date)server.getAttribute(obj, "ActiveSince");
  Date lastTime = (Date)server.getAttribute(obj, "LastMessageSentTime");
  ConversationMBean[] convs = (Conversation[]) server.getAttribute(obj,
     "ActiveConversations");
   if (convs != null)
   {
     for (int ii=0; ii< convs.length; ii++)
     {
        String protocol = convs[ii].getBusinessProtocolName();
      }
    }
 }
...
```

All the attributes shown in the preceding code listing can be retrieved by calling `getAttribute`. To invoke a method such as `shutDown` on WLCMBean, call the MBean Server. For more information see the JMX specification at the following URL:

`http://java.sun.com`

Step 6: Navigate Across MBeans

MBeans that are logically related have accessor methods to retrieve references to each other. These methods are strongly typed and return exact MBean types. For example, the `WLCMBean.getActiveDeliveryChannels()` method returns an array of type `DeliveryChannelMBean` that represents all the active delivery channels in the system. Similarly, the `TradingPartnerSessionMBean.getActiveConversations()` method returns an array of type `ConversationMBean` that represents all the active conversations in this session.

For detailed information about these methods, see the *BEA WebLogic Integration Javadoc*.

Step 7: Handle Exceptions

If an error occurs while a B2B integration management application is running, a `com.bea.b2b.management.ManagementException` is thrown. Management applications can catch this exception and process it appropriately, as shown in the following listing.

Listing 2-4 Handling ManagementExceptions in Management Applications

```
catch (ManagementException me){
      String msg = "Exception in Management Application: " + me;
      debug(msg);
      throw new Exception(msg);
```

Index