



BEA WebLogic Adapter for ClarifyCRM®

User Guide

Copyright

Copyright © 2003 BEA Systems, Inc. All Rights Reserved.

Copyright © 2003 iWay Software. 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.

BEA WebLogic Adapter for ClarifyCRM User Guide

| Part Number | Date | Release |
|-------------|------------|---------|
| N/A | April 2003 | 7.0.3 |

Table of Contents

About This Document

| | |
|---------------------------------|------|
| What You Need to Know | viii |
| Related Information..... | viii |
| Contact Us! | viii |
| Documentation Conventions | ix |

1. Introducing the BEA WebLogic Adapter for ClarifyCRM

| | |
|---------------------------------------------------------------|------|
| Understanding ClarifyCRM | 1-2 |
| About ClarifyCRM Architecture..... | 1-2 |
| ClarifyCRM, Tuxedo, and the WebLogic Tuxedo Connector | 1-3 |
| About ClarifyCRM Business Objects | 1-5 |
| How ClarifyCRM Business Objects Access the Database..... | 1-5 |
| About Processing Services | 1-5 |
| About Processing Events..... | 1-7 |
| Resource Adapters..... | 1-8 |
| About the CBO Interface for the ClarifyCRM Event Adapter..... | 1-8 |
| About the ClarifyCRM service adapter..... | 1-9 |
| Application Interaction..... | 1-10 |
| Executing ClarifyCRM Business Functions..... | 1-10 |
| Accessing Data Stored in ClarifyCRM | 1-10 |

2. Customizing Your WebLogic Tuxedo Connector Environment

| | |
|-------------------------------------------------|-----|
| Configuring the WebLogic Tuxedo Connector | 2-2 |
| Creating a New WTC Server | 2-2 |
| Creating a Local WebLogic Domain | 2-5 |
| Creating a Remote Tuxedo Domain..... | 2-6 |
| Exporting a Local WTC Service | 2-7 |

| | |
|--------------------------------------------------------|------|
| Importing a Remote Tuxedo Service..... | 2-9 |
| Configuring the WebLogic Tuxedo Connector EJBs..... | 2-10 |
| Creating Domain Access Permissions for WTC Server..... | 2-14 |

3. Creating Application Views for the CBO interface

| | |
|-----------------------------------------------------|------|
| Creating an Application View Folder..... | 3-2 |
| Creating an Application View | 3-3 |
| Adding a Service to an Application View | 3-7 |
| Before Adding an Event to an Application View | 3-10 |
| Creating the Event Tables | 3-10 |
| Applying DB Triggers..... | 3-11 |
| Adding an Event to an Application View..... | 3-12 |
| Deploying an Application View | 3-16 |
| Deploying an Application View With a Service | 3-16 |
| Deploying an Application View With an Event..... | 3-19 |
| Undeploying an Application View | 3-22 |
| Testing a Deployed Service | 3-24 |
| Testing a Deployed Event..... | 3-27 |
| Using a Service or Event in a Workflow | 3-30 |
| Example: Using a Service in a Workflow | 3-30 |
| About the ClarifyCRM Request..... | 3-31 |
| Start the Workflow | 3-32 |
| The ClarifyCRM Response | 3-32 |

4. Creating Application Views for the ClearBasic Interface

| | |
|----------------------------------------------------|------|
| Creating an Application View Folder..... | 4-2 |
| Creating an Application View | 4-3 |
| Adding a Service to an Application View | 4-7 |
| Adding an Event to an Application View..... | 4-10 |
| Deploying an Application View | 4-13 |
| Deploying an Application View With a Service | 4-13 |
| Deploying an Application View With an Event..... | 4-17 |
| Undeploying an Application View | 4-20 |
| Testing a Deployed Service | 4-21 |
| Testing a Deployed Event..... | 4-24 |

| | |
|----------------------------------------------|------|
| Using a Service or Event in a Workflow | 4-27 |
| Example: Using a Service in a Workflow | 4-27 |
| The ClarifyCRM Request | 4-28 |
| Start the Workflow | 4-29 |
| The Clarify Response..... | 4-29 |

5. Creating Schema Repositories

| | |
|--------------------------------------------|-----|
| Understanding Metadata..... | 5-1 |
| Introducing Schemas and Repositories | 5-2 |
| Naming a Schema Repository | 5-3 |
| The Repository Manifest | 5-3 |
| Creating a Repository Manifest..... | 5-4 |
| Sample Files | 5-7 |

6. Using Tracing

| | |
|------------------------------------------------------------|-----|
| Levels and Categories of Tracing..... | 6-2 |
| Tracing and Performance | 6-3 |
| Creating Traces for Services and Events..... | 6-4 |
| Creating Traces for a Service | 6-4 |
| Creating or Modifying the Tracing Level for an Event | 6-6 |
| Creating Adapter Logs for an Event | 6-9 |

A. Customizing Tuxedo Field Tables

| | |
|-----------------------------------------------------------|-----|
| Working With Field Tables..... | A-1 |
| Creating a WTC FML32 Field Table | A-2 |
| Creating a Tuxedo Field Table Header File | A-2 |
| Using Your Field Table in WebLogic Tuxedo Connector | A-3 |
| Using Your Field Table in Tuxedo..... | A-4 |



About This Document

The *BEA WebLogic Adapter for ClarifyCRM User Guide* is organized as follows:

- [Chapter 1, “Introducing the BEA WebLogic Adapter for ClarifyCRM,”](#) introduces the BEA WebLogic Adapter for ClarifyCRM, describes its features, and provides an overview of how it works.
- [Chapter 2, “Customizing Your WebLogic Tuxedo Connector Environment,”](#) describes how to configure the WebLogic Tuxedo Connector and the Clarify ClearBasic Enterprise JavaBeans.
- [Chapter 3, “Creating Application Views for the CBO interface,”](#) describes how to define and deploy application views, how to test deployed services and events, and how to use services and events in a workflow.
- [Chapter 4, “Creating Application Views for the ClearBasic Interface,”](#) describes how to define and deploy application views, how to test deployed services and events, and how to use services and events in a workflow.
- [Chapter 5, “Creating Schema Repositories,”](#) addresses the schema repositories, manifests, and schemas that describe the documents entering and exiting a WebLogic Integration system.
- [Chapter 6, “Using Tracing,”](#) describes how to use tracing.
- [Appendix A, “Customizing Tuxedo Field Tables,”](#) describes how to customize the Tuxedo field definition file.

What You Need to Know

This document is written for system integrators with programming backgrounds and an understanding of ClarifyCRM™. Extensive knowledge of ClarifyCRM is not required, but may be helpful in learning about the adapter.

This document provides details on working with the adapter tools to develop online interconnections to ClarifyCRM using BEA WebLogic Integration.

Related Information

The following documents provide additional information for the associated software components:

- *BEA WebLogic Adapter for ClarifyCRM Installation and Configuration Guide*
- *BEA WebLogic Adapter for ClarifyCRM Release Notes*
- BEA WebLogic Server installation and user documentation, which is available at the following URL:

http://edocs.bea.com/more_wls.html

- BEA WebLogic Integration installation and user documentation, which is available at the following URL:

http://edocs.bea.com/more_wli.html

Contact Us!

Your feedback on the BEA Adapter for ClarifyCRM 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 adapter documentation.

In your e-mail message, please indicate which version of the adapter documentation you are using.

If you have any questions about this version of the adapter, or if you have problems using it, contact BEA Customer Support through BEA WebSupport at www.bea.com. You can also contact Customer Support by using the contact information provided on the Customer Support Card, which is included in the product package.

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 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. |
| <i>italics</i> | Indicates emphasis or book titles. |

| Convention | Item |
|----------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| monospace text | <p>Indicates code samples, commands and their options, data structures and their members, data types, directories, and file names and their extensions. Monospace text also indicates text that you must enter from the keyboard.</p> <p><i>Examples:</i></p> <pre>#include <iostream.h> void main () the pointer psz chmod u+w * \tux\data\ap .doc tux.doc BITMAP float</pre> |
| monospace boldface text | <p>Identifies significant words in code.</p> <p><i>Example:</i></p> <pre>void commit ()</pre> |
| <i>monospace italic text</i> | <p>Identifies variables in code.</p> <p><i>Example:</i></p> <pre>String <i>expr</i></pre> |
| UPPERCASE TEXT | <p>Indicates device names, environment variables, and logical operators.</p> <p><i>Examples:</i></p> <pre>LPT1 SIGNON OR</pre> |
| { } | <p>Indicates a set of choices in a syntax line. The braces themselves should never be typed.</p> |
| [] | <p>Indicates optional items in a syntax line. The brackets themselves should never be typed.</p> <p><i>Example:</i></p> <pre>buildobjclient [-v] [-o name] [-f <i>file-list</i>]... [-l <i>file-list</i>]...</pre> |
| | <p>Separates mutually exclusive choices in a syntax line. The symbol itself should never be typed.</p> |

| Convention | Item |
|---------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ... | <p>Indicates one of the following in a command line:</p> <ul style="list-style-type: none">■ 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 <p>The ellipsis itself should never be typed.</p> <p><i>Example:</i></p> <pre>buildobjclient [-v] [-o name] [-f file-list]... [-l file-list]...</pre> |
| | <p>Indicates the omission of items from a code example or from a syntax line.</p> <p>The vertical ellipsis itself should never be typed.</p> |



1 Introducing the BEA WebLogic Adapter for ClarifyCRM

The BEA WebLogic Adapter for ClarifyCRM provides a means to exchange real-time business data between ClarifyCRM systems and other application, database, or external business partner systems. The adapter supports this data exchange with the CBO and ClearBasic interfaces of ClarifyCRM. The adapter allows for both inbound and outbound processing.

This section provides information about the adapter that will help you accomplish your integration projects. It includes the following topics:

- [Understanding ClarifyCRM](#)
- [Resource Adapters](#)
- [Executing ClarifyCRM Business Functions](#)
- [Accessing Data Stored in ClarifyCRM](#)

Note: Some features of the WebLogic Adapter for ClarifyCRM are restricted to either the ClearBasic or the CBO interface. When that is the case, it is stated clearly. Where there is no such indication, the feature applies to both interfaces.

Understanding ClarifyCRM

ClarifyCRM eFrontOffice (CeFO) provides an integrated solution that allows you to manage your interactions with your customers. With CeFO, you can track sales, contracts, customers, product development, inventory, and repair operations. You can capture any customer interactions made over the telephone, email, or the Web. You can follow the interactions with a customer from the first request for information through the sales cycle and through subsequent requests for additional product or services. Some of the CeFO client applications include:

- ClearSupport, which allows you to manage incoming calls to customer support.
- ClearQuality, which allows you to record and monitor product defects, enhancement requests, and changes to products.
- ClearLogistics, which allows you to manage service-related inventory and on-site service calls.
- ClearSales, which allows you to collect and manage information for marketing campaigns and sales opportunities.
- ClearContracts, which allows you to manage service contracts.

About ClarifyCRM Architecture

CeFO is a client-server solution, which means that you and other employees can run applications on your computers (the clients) to work with information stored in a central database on another computer (the server).

When you run a CeFO application on the client, the application displays forms for your business processes, such as bills of material, shipping labels, inventory counts, and contract information. You use these forms to view and change information in a central database on the server.

For example, in ClearSupport, you can use a form to enter information about an incoming customer call. The information that you enter is stored in the main database. Other employees can use ClearSupport to view and update the information that you have entered.

CeFO applications are available in the following two types of clients:

- ClarifyCRM Desktop LAN/WAN Client
- ClarifyCRM Desktop Web Client

The ClarifyCRM Desktop LAN/WAN and Web Clients are virtually identical.

ClarifyCRM, Tuxedo, and the WebLogic Tuxedo Connector

This section applies only to the ClearBasic interface of ClarifyCRM.

ClarifyCRM runs under the control of the WebLogic Tuxedo Transaction Manager, and makes use of all the features that Tuxedo provides, such as transaction control, application clustering, or distributed invocation. The WebLogic Server itself provides support for Tuxedo services via the WebLogic Tuxedo Connector (WTC). With WTC, WebLogic Java-based modules (Enterprise Java Beans, or EJBs) may access the Application to Monitor Interface (ATMI) of Tuxedo from the WebLogic platform and make service calls of Tuxedo modules. Conversely, Tuxedo modules may make service calls of WTC-hosted EJB modules. The Java ATMI (JATMI) capability and the domain-to-domain communication capability of Tuxedo and the WTC are the foundation of the BEA WebLogic Adapter for ClarifyCRM.

Data passed between two Tuxedo modules, whether they be EJBs hosted on WebLogic or C-based programs hosted on a Tuxedo application server, is done in a typed buffer format called FML or FML32 (a 32-bit addressed typed buffer). The typed buffer is described in a field definition file and each Tuxedo application system may have its own unique list of fields applicable to the functions performed by the ATMI / JATMI modules installed.

The BEA WebLogic Adapter for ClarifyCRM allows WebLogic Integration workflows to make service calls of Clarify through JATMI service calls of remote Tuxedo services. The standard Clarify CB_EXESUB Tuxedo module routes these requests into any of the registered ClarifyCRM ClearBasic routines, marshalling parameters from the FML typed buffer and unmarshalling returned ClearBasic results.

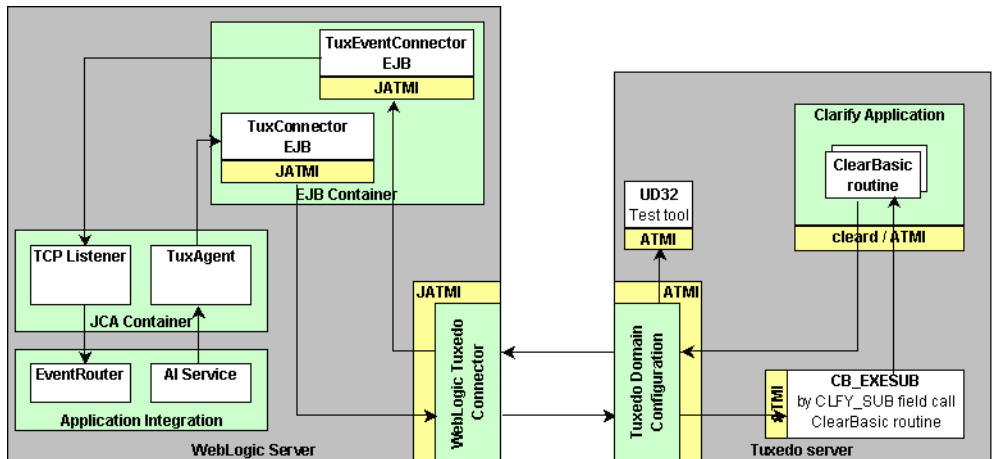
ClarifyCRM ClearBasic modules may use the standard ClearBasic call method to invoke a remote Tuxedo service. The adapter provides a WTC-hosted service, IWAYWTC_GEN_SVC, which marshals its event data from the FML typed buffer, creates a corresponding XML representation, and then posts the event XML document

into the WebLogic Integration EventRouter via a standard TCP event listener. Workflows registered to the event posted in by the IWAYWTC_GEN_SVC EJB JATMI module will be invoked.

Each Tuxedo and each WTC installation may define its own list of fields that are relevant to them. However, the fields passed between the two must correspond with matching field names, IDs, and data types. See [Appendix A, “Customizing Tuxedo Field Tables,”](#) for instructions about how to customize the WebLogic WTC environment for your custom Clarify application modules. Additionally, the BEA WebLogic Adapter for ClarifyCRM makes use of two specific fields in the FML typed buffers shared between the two platforms:

- The CLFY_SUB field is mandated by the Clarify CB_EXESUB module. The value contained in this field buffer is used to determine which ClearBasic routine to invoke.
- The second field, whose name defaults to HI_WORKFLOW_NAME, is configurable by the adapter, and controls the root element of the event schema being created. The contents of this field buffer will be used as the root element of the created XML document.

Figure 1-1 BEA WebLogic Adapter for ClarifyCRM Architecture



About ClarifyCRM Business Objects

ClarifyCRM Business Objects (CBO) are part of the Clarify eBusiness Framework for developing applications for the Clarify eFrontOffice (CeFO) database.

These business objects are C++ objects with Java layer that provide access to data in the CeFO database. Each type of business object implements part of the Clarify data model and encapsulates the application logic for working with that part of the model.

How ClarifyCRM Business Objects Access the Database

A business object has methods and properties that can be used to query and update data in a CeFO database table.

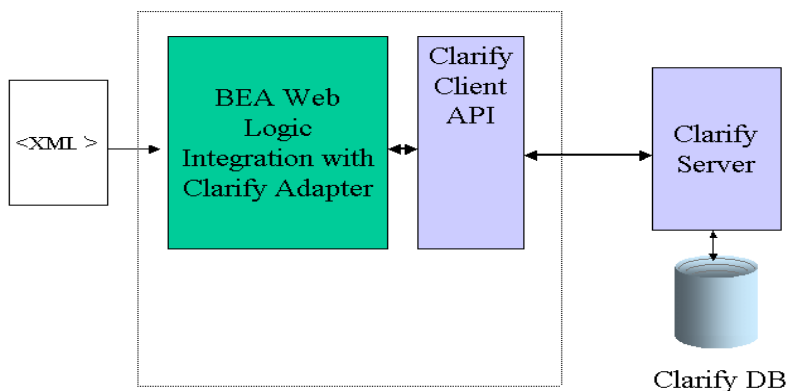
Each business object contains a rowset, which is an in-memory copy of a set of rows from a CeFO database table. When database table is queried, the business object holds the results of the query in its rowset. A row in the rowset can be selected and any of the values of fields in that row can be obtained.

The rowset can be used to make changes to data in the database table. The values of fields in rows can be modified and the changed rows committed back to the database. If a new row must be added to the database table, add the rows to the rowset in the business object and commit the new rows to the database.

About Processing Services

The following figure illustrates the WebLogic Integration service processing framework.

Figure 1-2 Service Architecture



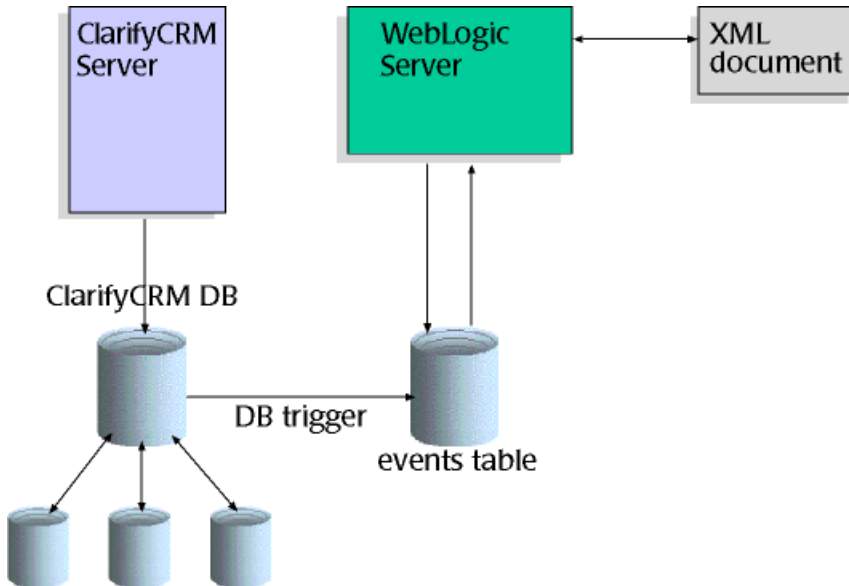
When the BEA WebLogic Adapter for ClarifyCRM processes a service:

1. The adapter receives an XML service request document from a WebLogic Server workflow and connects it, via the workflow, to a pre-configured WebLogic service.
2. The adapter validates the request to ensure that it conforms to the service's request schema.
3. The adapter interprets the document and issues the corresponding CBO method calls to the ClarifyCRM Server.
4. The ClarifyCRM Server processes the calls and returns the service response, if any, to the adapter.
5. From the response, the adapter generates an XML response document. The document conforms to the service's response schema.
6. The adapter returns the XML response document to WebLogic Server.
7. WebLogic Server, via a workflow, transforms the document and/or routes it to the service originator.

About Processing Events

The following figure illustrates the WebLogic Integration event processing framework.

Figure 1-3 Event Architecture



When the BEA WebLogic Adapter for ClarifyCRM processes an event:

1. An update is run against the ClarifyCRM Server.
2. The update is written to the ClarifyCRM database, which triggers the relevant event data to be copied to the event table.
3. WebLogic Server (WebLogic Integration/Application Integration) periodically polls the events table with a specified SQL query. When the events table satisfies the query, the BEA WebLogic Adapter for ClarifyCRM generates an XML event document describing the event. The document conforms to the event's schema.
4. The adapter emits the event document to WebLogic Server, where it can be routed, or sent back through a workflow. The record in the event table is deleted upon successful generation and emission of the XML document.

5. WebLogic Server, via a workflow, transforms the document and/or routes it. The event's record is then deleted from the events table.

Resource Adapters

The BEA WebLogic Adapter for ClarifyCRM is a resource adapter: it connects one application to another when those applications are not originally designed to communicate with each other. The adapter sends requests to ClarifyCRM and returns data retrieved from ClarifyCRM. When the adapter makes requests of ClarifyCRM, it is referred to as a service adapter; when it detects an event in ClarifyCRM, it is referred to as an event adapter.

About the CBO Interface for the ClarifyCRM Event Adapter

This section applies only to the CBO interface for the adapter for ClarifyCRM.

The ClarifyCRM event adapter monitors a ClarifyCRM database, stored in either MS SQL Server or Oracle, that contains only the most recent changes made in a ClarifyCRM system. That information, when retrieved by the ClarifyCRM event adapter, is formatted into an XML document and sent to the WebLogic Server for further processing, transforming, and/or routing. Once the event is processed successfully, the event data is deleted from the event database.

Event adapters perform the following functions:

- Supplied database triggers move only relevant data about an action that was performed by ClarifyCRM into an events database.
- Poll the ClarifyCRM events database at user-configured time intervals.
- Translate the event information to XML.
- Support transformation of XML documents that contain event information.

About the ClarifyCRM service adapter

The ClarifyCRM service adapter processes requests for ClarifyCRM functions embedded in XML documents and forwards them to a back-end ClarifyCRM system. The resulting response information is then brought back and processed by the service adapter for further routing.

Service adapters receive an XML request document from a client and call a specific function in ClarifyCRM. They are consumers of request messages and, depending on the request, may or may not provide a response. There are two kinds of services:

- Asynchronous, in which the client application issues a service request and then proceeds with its processing. It does not wait for the response.
- Synchronous, in which the client application waits for the response before proceeding with further processing.

BEA WebLogic Integration supports both of these methods, so you do not have to provide this functionality in your own application code.

Service adapters perform the following functions:

- Receive service requests from an external client.
- Transform the XML request document into the ClarifyCRM-specific format. The request document conforms to the request XML schema for the service. The schema is based on Clarify/CRM metadata.
- Call the underlying function in ClarifyCRM and wait for its response.
- Transform the response from the ClarifyCRM-specific data format to an XML document that conforms to the response XML schema for the service. The schema is based on ClarifyCRM metadata.

Key features of the BEA WebLogic Adapter for ClarifyCRM include support for:

- Bi-directional message interaction between WebLogic Integration and ClarifyCRM.
- Service (ClarifyCRM inbound) and event (ClarifyCRM outbound) adapter integration operations with ClarifyCRM presenting XML schemas to WebLogic Integration Studio.

Application Interaction

The adapter enables non-ClarifyCRM systems to communicate and exchange transactions with ClarifyCRM using WebLogic Integration and XML messages.

Applications that interact with ClarifyCRM to cause a new ClarifyCRM event use WebLogic Integration application views, services, and WebLogic Integration Studio to send request messages to ClarifyCRM via the adapter. If the request retrieves data from ClarifyCRM, the adapter sends the application a response message with the data.

Executing ClarifyCRM Business Functions

This section applies only to the CBO interface of the adapter for ClarifyCRM.

You can use the BEA WebLogic Adapter for ClarifyCRM to perform inserts, updates, and deletes against ClarifyCRM objects stored in the ClarifyCRM database. The adapter supports the Contact, PartMaster, ItemPricesQty, ItemPrice, Quote, QuoteSchedule, QuoteLine QuotePONumber, QuoteScheduleBilltoCust, and QuoteScheduleShiptoCust objects. You can also use the adapter to integrate ClarifyCRM with non-ClarifyCRM systems.

Accessing Data Stored in ClarifyCRM

The BEA WebLogic Adapter for ClarifyCRM provides controlled access to the data contained in the ClarifyCRM database. The GET method is issued via an XML document the same way as Insert, Update, and Delete. The results of the GET are contained in an XML response document. W3C schemas for all the Request and Response documents are supplied, but must be specified in the `manifest.xml` file used when creating the application view.

2 Customizing Your WebLogic Tuxedo Connector Environment

This chapter applies to the ClearBasic interface of the adapter for ClarifyCRM only.

The ClearBasic interface of the BEA WebLogic Adapter for ClarifyCRM requires the WebLogic Tuxedo Connector. This section describes how to configure the WebLogic Tuxedo Connector environment for use with the adapter, and includes the following topics:

- [Configuring the WebLogic Tuxedo Connector](#)
- [Configuring the WebLogic Tuxedo Connector EJBs](#)
- [Creating Domain Access Permissions for WTC Server](#)

Configuring the WebLogic Tuxedo Connector

A WebLogic Tuxedo Connector (WTC) Server is a running instance of the WebLogic Tuxedo Connector. This section describes the basic tasks required to configure a WebLogic Tuxedo Connector Server for use with the adapter, and includes the following topics:

- Creating a New WTC Server
- Creating a Local WebLogic Domain
- Creating a Remote Tuxedo Domain
- Exporting a Local WTC Service
- Importing a Remote Tuxedo Service

Creating a New WTC Server

To create a new WTC Server:

1. Log on to the BEA WebLogic Server Console.

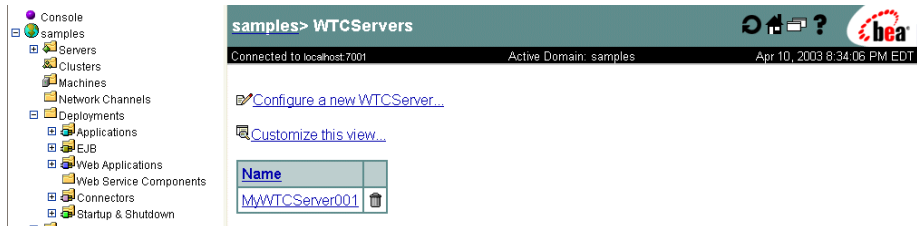
Figure 2-1 WebLogic Server Console Main Window



2. Click WebLogic Tuxedo Connector.

The WTCServers window opens.

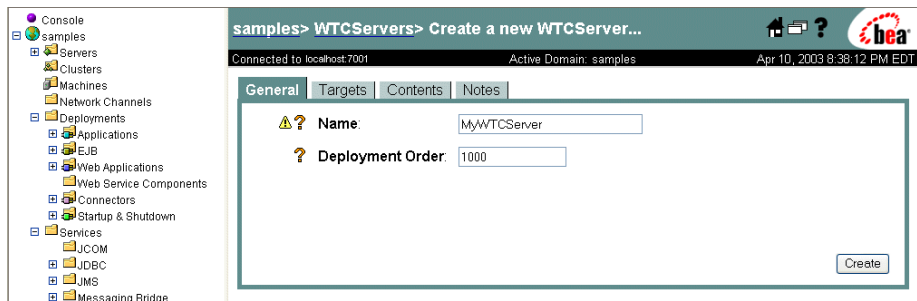
Figure 2-2 WTCServers Window



3. Click Configure a new WTCServer...

The Create a new WTCServer... window opens.

Figure 2-3 Create a New WTCServer General Tab

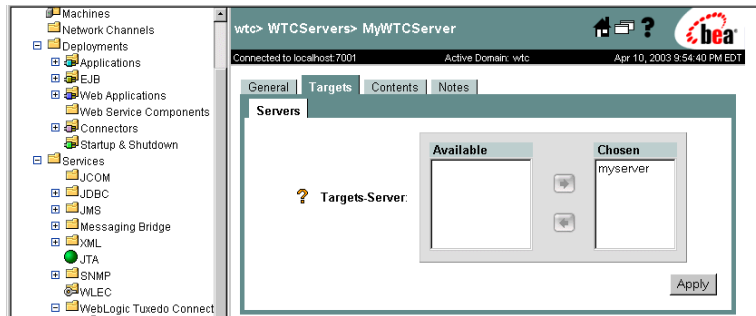


4. Enter a name and deployment order for this WebLogic Tuxedo Connector Server (that is, for this instance of the WebLogic Tuxedo Connector).

You can provide any descriptive name for the server: it serves only to contain the configuration of domains and services.

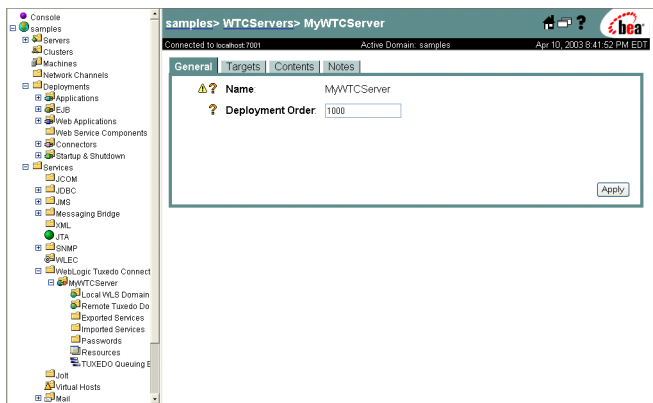
5. Click Create to create your new WTC Server.
6. Select the Targets tab.
7. Move the WebLogic Server on which you want the WTC server to be deployed to the Chosen list, if it is not already there.

Figure 2-4 New WTC Server Targets Tab



8. In the left pane, expand the node corresponding to your new WTC Server.
The WTC Server's components are displayed in the left pane.

Figure 2-5 Expanded WTC Server Node



You have successfully created the WTC Server.

To create:

- A local WebLogic domain, see “Creating a Local WebLogic Domain.”
- A remote Tuxedo domain, see “Creating a Remote Tuxedo Domain.”

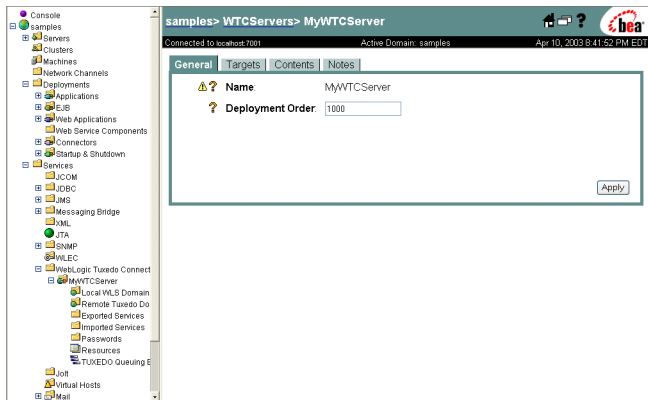
You can create the domains in any order.

Creating a Local WebLogic Domain

To create a local WebLogic domain:

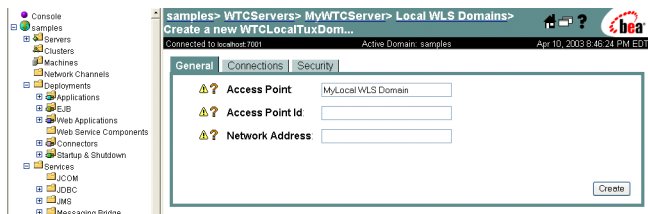
1. Logon to the WebLogic Server Console and expand your WebLogic Tuxedo Connector (WTC) Server's node in the left pane, if you have not yet done so.

Figure 2-6 Expanded WTC Server Node



2. Select the WTC Server's Local WLS Domains node in the left pane.
The Local WLS Domains window opens.
3. Click Configure a new Local WLS Domain in the right pane.
The Create a new WTCLocalTuxDom... window opens.

Figure 2-7 Create a New WTC Local Tuxedo Domain Window



You have successfully created the local WebLogic domain. To create a remote Tuxedo domain, see [“Creating a Remote Tuxedo Domain.”](#)

If you have already created a remote Tuxedo domain, you can now create services. To:

- Export a local WTC service, see “Exporting a Local WTC Service.”
- Import a remote Tuxedo service, see “Importing a Remote Tuxedo Service.”

You can export and import services in any order.

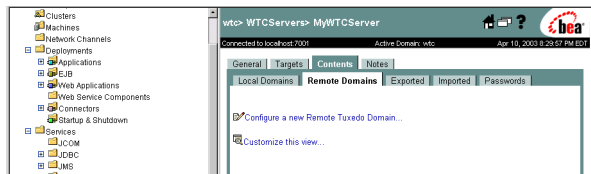
Creating a Remote Tuxedo Domain

This configuration corresponds to the remote Tuxedo domain that will be running ClarifyCRM and to which the local WebLogic Server WTC domain will be communicating.

To create a remote WebLogic domain:

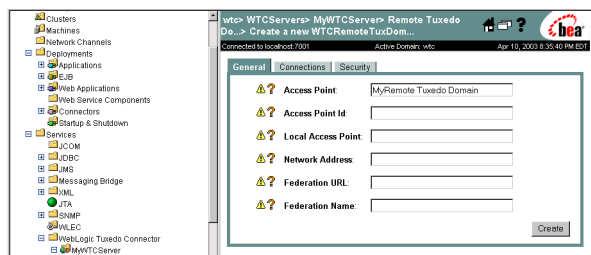
1. Logon to the WebLogic Server Console and expand your WebLogic Tuxedo Connector (WTC) Server's node in the left pane, if you have not yet done so.
2. Select the WTC Server's Remote Tuxedo Domains node in the left pane.

Figure 2-8 WTC Server Remote Domains Tab



3. Click **Configure a new Remote WLS Domain**.

Figure 2-9 Create a New Remote Tuxedo Domain Window



4. Enter the domain's properties:
 - Access Point: the name of the remote Tuxedo domain.
 - Access Point Id: the ID of the Tuxedo domain as listed in the DMCONFIG configuration on Tuxedo.
 - Local Access Point: the name of the local WLS domain that will be connected to this remote Tuxedo domain.
 - Network Address: the TCP/IP //host:port of the remote Tuxedo domain.
 - Federation URL: the URL for a foreign name service that is federated into JNDI. You are not required to provide this value.
 - Federation Name: the context at which to federate to a foreign name service. You are not required to provide this value.
5. Click Create to activate the configuration you have defined.

You may also want to modify default values in the Connections and Security tabs. Please see your WebLogic Tuxedo Connector Administration manual for help with modifying these values.

You have successfully created a remote Tuxedo domain. To create a local WebLogic domain, see [“Creating a Local WebLogic Domain.”](#)

If you have already created a local WebLogic domain, you can now create services. To:

- Export a local WTC service, see [“Exporting a Local WTC Service.”](#)
- Import a remote Tuxedo service, see [“Importing a Remote Tuxedo Service.”](#)

You can export and import services in any order.

Exporting a Local WTC Service

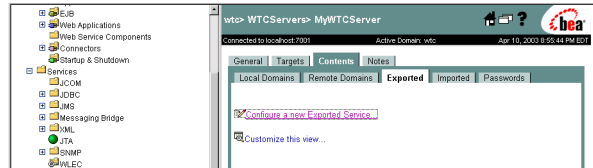
Exported local WebLogic Tuxedo Connector (WTC) services are services hosted on the WebLogic Server, implemented through Enterprise JavaBeans (EJBs) which implement the `weblogic.wtc.jatmi.TuxedoService` interface. The ClearBasic interface for the BEA WebLogic Adapter for ClarifyCRM supplies an EJB to support the inbound invocation of service requests and to post the constructed XML event document into WebLogic Integration. These local WTC services support the event adapter.

2 Customizing Your WebLogic Tuxedo Connector Environment

To export a local WTC service:

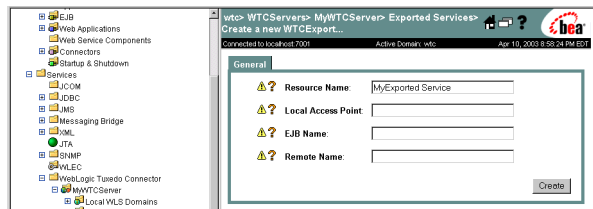
1. Logon to the WebLogic Server Console and expand your WTC Server's node in the left pane, if you have not yet done so.
2. Select the WTC Server's Exported Services node in the left pane.

Figure 2-10 WTC Server Exported Tab



3. Click Configure a new Exported Service.

Figure 2-11 Exported Service General Tab



4. Enter the service's properties:
 - Resource Name: the locally known name of the service.
 - Local Access Point: the name of the local WebLogic WTC domain access point containing the service.
 - EJB Name: the JNDI name of the EJB implementing the Tuxedo service locally on the WebLogic Server.
 - Remote Name: the name the local service will be known as when accessed in the remote Tuxedo domains.
5. Click Create to activate the configuration entries.

You have successfully exported the local WTC service. To import a remote Tuxedo service, see [“Importing a Remote Tuxedo Service.”](#)

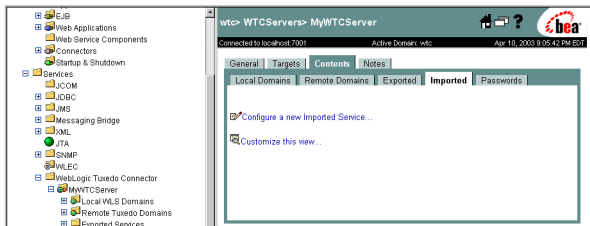
Importing a Remote Tuxedo Service

Imported remote Tuxedo services are services hosted on the remote Tuxedo server. ClarifyCRM supplies a routing service, CB_EXESUB, that takes Tuxedo requests and calls into ClarifyCRM ClearBasic routines.

To export a remote WTC service:

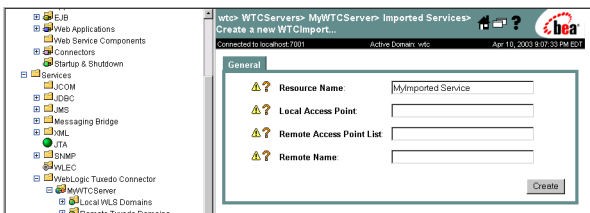
1. Logon to the WebLogic Server Console and expand your WTC Server's node in the left pane, if you have not yet done so.
2. Select the WTC Server's Imported Services node in the left pane.

Figure 2-12 WTC Server Imported Tab



3. Click Configure a new Imported Service.

Figure 2-13 Exported Service General Tab



4. Enter the service's properties:
 - Resource Name: the locally-known name of the service.
 - Local Access Point: the name of the local WebLogic WTC domain access point in which the service will be made available.
 - Remote Access Point List: the list of all defined remote Tuxedo domains that can be called on to execute the imported service.

- Remote Name: the name the remote service as it exists in the Tuxedo domains.
5. Click Create to activate the configuration entries.

You have successfully imported the remote Tuxedo service. To export a local Tuxedo service, see [“Exporting a Local WTC Service.”](#)

Configuring the WebLogic Tuxedo Connector EJBs

The ClearBasic interface for the BEA WebLogic Adapter for ClarifyCRM requires Enterprise JavaBeans (EJBs) to access the Java-based Application to Monitor Interface (JATMI) of the WebLogic Tuxedo Connector (WTC). Two EJBs are supplied:

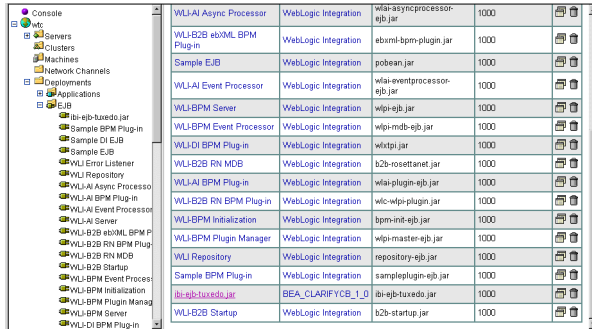
- TuxConnectorEJB for WLAI service invocation.
- TuxEventConnectorEJB for Tuxedo service invocation and WLAI event posting.

The EJBs are automatically installed and deployed when the adapter's Web Application .ear file is installed. This section describes the configuration options available to the Tuxedo Connector EJBs.

1. Logon to the WebLogic Server Console if you have not already done so.
2. Select Deployments→EJB in the left pane.

The list of all EJBs deployed on your WebLogic Server is displayed.

Figure 2-14 Deployed EJB Window



| | | | | | |
|---------------------------|----------------------|----------------------------|------|--|--|
| WLJAI Async Processor | WebLogic Integration | wla-asyncprocessor-ejb.jar | 1000 | | |
| WLJ-E2B ebXML BPM Plug-in | WebLogic Integration | ebxml-bpm-plugin.jar | 1000 | | |
| Sample EJB | WebLogic Integration | pokean.jar | 1000 | | |
| WLJAI Event Processor | WebLogic Integration | wla-eventprocessor-ejb.jar | 1000 | | |
| WLJBPM Server | WebLogic Integration | wlp-ejb.jar | 1000 | | |
| WLJ-BPM Event Processor | WebLogic Integration | wlp-mdb-ejb.jar | 1000 | | |
| WLJDI BPM Plug-in | WebLogic Integration | wlxtpi.jar | 1000 | | |
| WLJ-E2B RN MCB | WebLogic Integration | b2b-rosettanet.jar | 1000 | | |
| WLJAI BPM Plug-in | WebLogic Integration | wla-plugin-ejb.jar | 1000 | | |
| WLJ-E2B RN BPM Plug-in | WebLogic Integration | wlc-wlp-plugin.jar | 1000 | | |
| WLJBPM Initialization | WebLogic Integration | bpm-init-ejb.jar | 1000 | | |
| WLJ-BPM Plugin Manager | WebLogic Integration | wlp-master-ejb.jar | 1000 | | |
| WLJ Repository | WebLogic Integration | repository-ejb.jar | 1000 | | |
| Sample BPM Plug-in | WebLogic Integration | sampleplugin-ejb.jar | 1000 | | |
| ibi-ejb-tuxedo.jar | BEA_CLARIFYCB_1_0 | ibi-ejb-tuxedo.jar | 1000 | | |
| WLJ-E2B Startup | WebLogic Integration | b2b-startup.jar | 1000 | | |

- Click ibi-ejb-tuxedo.jar, the EJB .jar file that is supplied with the adapter. Both EJBs are contained in this file.

The ibi-ejb-tuxedo.jar summary window opens.

Figure 2-15 ibi-ejb-tuxedo.jar Summary Window



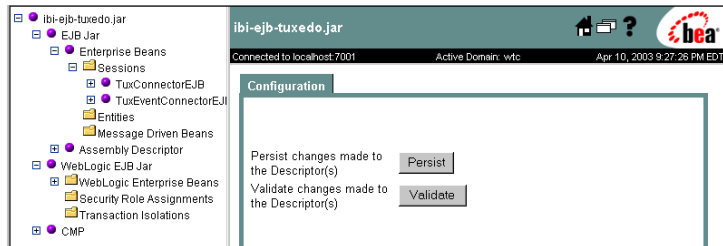
- Click Edit EJB Descriptor...

A separate WebLogic Server Console window opens, with ibi-ejb-tuxedo.jar the root node in the left pane.

- Expand ibi-ejb-tuxedo.jar→EJB Jar→Enterprise Beans→Sessions in the left pane.

The nodes for the two EJBs are displayed in the left pane.

Figure 2-16 EJB Nodes Displayed in Left Pane



6. Expand the two EJB nodes in the left pane.

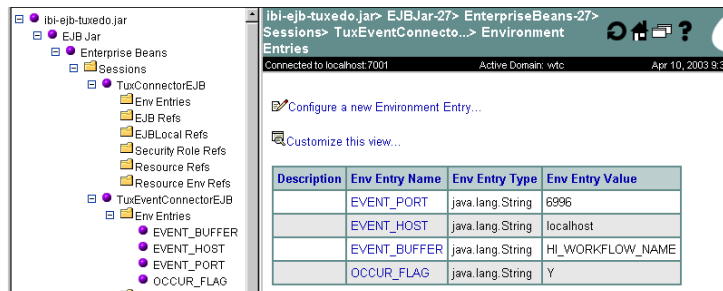
TuxEventConnectorEJB is responsible for receiving Tuxedo service requests, translating the Tuxedo data into XML, and posting into the WebLogic Integration event adapter. TuxEventConnectorEJB has environment entries (run-time properties that control the behavior of the EJB). You will configure its properties.

TuxConnectorEJB receives XML request documents and invokes Tuxedo service calls via the JATMI layer of WTC, and receives the corresponding response data and creates response documents from it. It has no properties to be configured.

7. Unfold TuxEventConnectorEJB's Env Entries folder.

The EJB's environment entries are displayed.

Figure 2-17 EJB Environment Entries Window

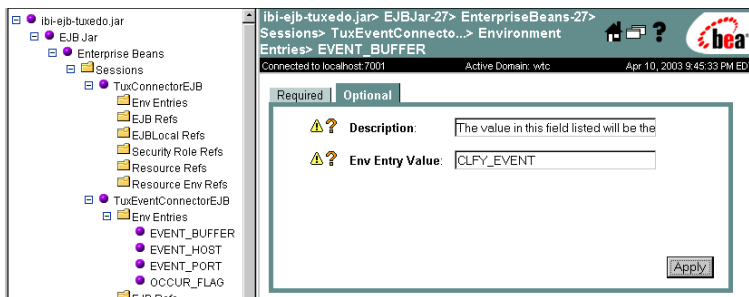


This EJB has the following environment entries:

- **EVENT_PORT**, which specifies the port to which the event XML file will be posted by the EJB.

- EVENT_HOST, which specifies the host to which the event XML file will be posted by the EJB.
 - EVENT_BUFFER, which specifies the Tuxedo FML field from which the root XML element will be obtained.
 - OCCUR_FLAG, which indicates whether OCCURRENCE attributes will be added to the created XML document elements to ensure addressability of repeating XML elements. (Y indicates that OCCURRENCE attributes will be created for every field; N indicates that no OCCURRENCE attributes will be created.)
8. To change the EJB's environment entry:
- a. Select the entry.
 - b. Select the Optional tab.

Figure 2-18 Event Buffer Environment Entry Optional Tab



- c. Enter a new value.
- d. Click Apply.

You have finished configuring the WTC EJBs.

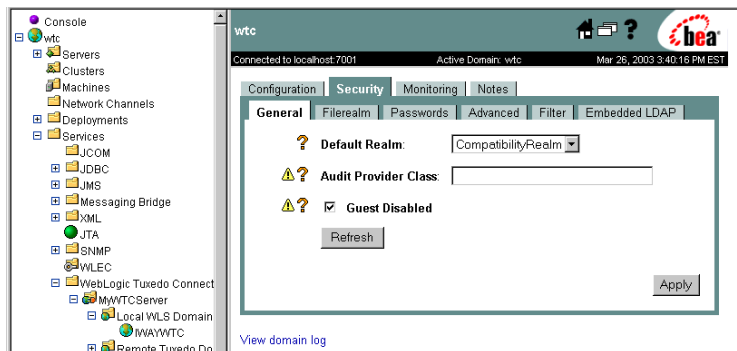
Creating Domain Access Permissions for WTC Server

The Local WLS-based WTCdomain and the remote Tuxedo domain have several security settings. By default, there is no domain-to-domain security configured for a WebLogic Server WebLogic Tuxedo Connector (WTC) to Tuxedo domain setup. For information about other security options that restrict access at the domain level, consult your WTC manual.

Regardless of the security levels chosen (“none”, “application password,” or “domain password”) on the local and remote domains, a connection principle will be required for the remote Tuxedo domain. It will be under this account that the remote domain will invoke local WTC-based service requests. By default, the connection principle or user ID under which a remote domain connects is its Access Point ID. An account must be created in the active security realm under which the remote domain request can be executed.

WebLogic Interface adapters must run in a domain that supports application integration, and these domains run under CompatibilityRealm security.

Figure 2-19 Domain Security Settings



For this reason, you must create a user account for the remote Tuxedo domain:

1. Create a new account for the remote Tuxedo domain.

From the WebLogic Console, expand the Compatibility Security branch, and select Users.

Fill in the requisite fields to create a new user:

- name, which must match the remote Tuxedo Access Point ID.
- password, which can be anything as it is not checked when domain security is set to “none.”

Click Create and then click Click here to save these changes to the realm implementation.

Figure 2-20 New Account Configuration

The screenshot displays the 'Users' configuration page in the BEA WebLogic console. The left sidebar shows a tree view of the configuration hierarchy, with 'Users' selected under 'Security'. The main panel is titled 'Users' and shows the 'Create a New User' form. The form has two columns: 'Create a New User' and 'Change a User's Password'. The 'Create a New User' column contains fields for 'Name' (with the value 'clarly'), 'Password', and 'Confirm Password', along with a 'Create' button. The 'Change a User's Password' column contains fields for 'Name', 'Old Password', 'New Password', and 'Confirm Password'. Below these forms is a 'Delete a User' section with a 'Name' field and a 'Delete' button. At the bottom, there is an 'Unlock Users...' link and a list of users: 'admin, wlsSamplesUser, joe, hub, wlsystem, system, mary, guest'.

3 Creating Application Views for the CBO interface

The information in this chapter applies only to the CBO interface for the adapter for ClarifyCRM. For information on creating application views for the ClearBasic interface for the adapter for ClarifyCRM, see [Chapter 4, “Creating Application Views for the ClearBasic Interface.”](#)

When you define an application view, you create an XML-based interface between WebLogic Server and a particular Enterprise Information System (EIS) application within your enterprise. Once you create an application view, a business analyst can use it to create business processes that use the application. With the BEA WebLogic Adapter for ClarifyCRM, you can create any number of application views, each with any number of services and events.

This section provides information on application views and deployed services and events, and includes the following topics:

- [Creating an Application View Folder](#)
- [Creating an Application View](#)
- [Adding a Service to an Application View](#)
- [Before Adding an Event to an Application View](#)
- [Adding an Event to an Application View](#)
- [Deploying an Application View](#)

- [Undeploying an Application View](#)
- [Testing a Deployed Service](#)
- [Testing a Deployed Event](#)
- [Using a Service or Event in a Workflow](#)

Creating an Application View Folder

Application views reside within WebLogic Integration. WebLogic Integration provides you with a root folder in which you can store all of your application views; if you wish, you can create additional folders to organize related application views into groups.

To create an application view folder:

1. Log on to the WebLogic Integration Application View Console at `//appserver-host:port/wlai`.

Here, *appserver-host* is the IP address or host name where the WebLogic Integration Server is installed, and *port* is the socket on which the server is listening. The port, if not changed during installation, defaults to 7001.

2. If prompted, enter a user name and password.

Note: If the user name is not `system`, it must be included in the adapter group. For more information on adding the administrative server user name to the adapter group, see the *BEA WebLogic Adapter for ClarifyCRM Installation and Configuration Guide*.

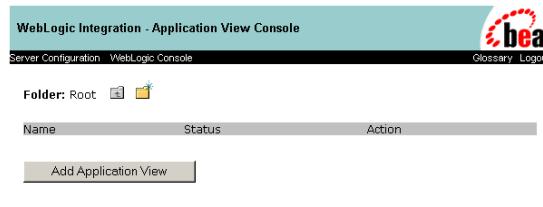
For more information, see “Logging On to the WebLogic Integration Application View Console” in “Defining an Application View” in *Using Application Integration*:

- For WebLogic Integration 7.0, see <http://edocs.bea.com/wli/docs70/aiuser/2usrdef.htm>

3. Click Login.

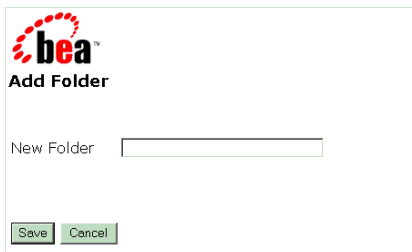
The WebLogic Integration Application View Console opens.

Figure 3-1 Application View Console Main Window



4. Double-click the new folder icon. The Add Folder window opens.

Figure 3-2 Add Folder Window



5. Supply a name for the folder, and then click Save.

You have finished creating the application view folder. To create an application view, see [“Creating an Application View” on page 3-3](#).

Creating an Application View

To create an application view:

1. Log on to the WebLogic Application View Console at `//appserver-host:port/wlai`.

Here, *appserver-host* is the IP address or host name where the WebLogic Integration Server is installed, and *port* is the socket on which the server is listening. The port, if not changed during installation, defaults to 7001.

2. If prompted, enter a user name and password.

3 Creating Application Views for the CBO interface

Note: If the user name is not `system`, it must be included in the adapter group. For more information on adding the administrative server user name to the adapter group, see the *BEA WebLogic Adapter for ClarifyCRM Installation and Configuration Guide*.

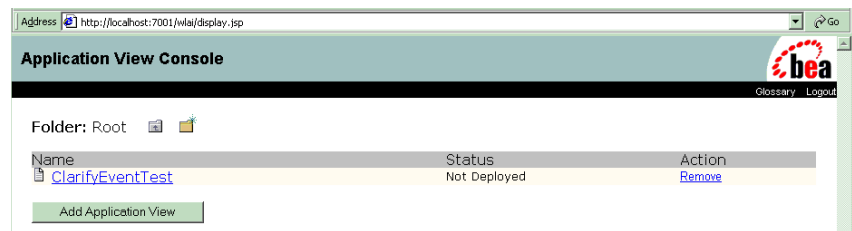
For more information, see “Logging On to the WebLogic Integration Application View Console” in “Defining an Application View” in *Using Application Integration*:

- For WebLogic Integration 7.0, see <http://edocs.bea.com/wli/docs70/aiuser/2usrdef.htm>

3. Click Login.

The WebLogic Integration Application View Console opens.

Figure 3-3 Selecting a Folder in the Main Window

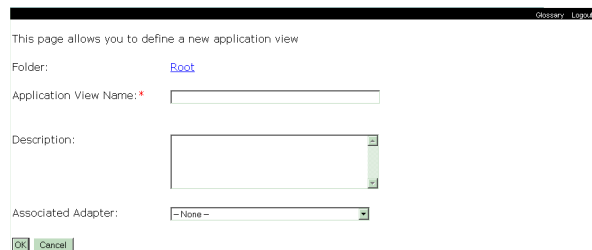


4. Select the desired application view folder.

5. Click Add Application View.

The Define New Application View window opens.

Figure 3-4 Define New Application View Window



An application view enables a set of functions for the adapter's target EIS application. For detailed information, see “Defining an Application View” in *Using Application Integration*:

- For WebLogic Integration 7.0, see <http://edocs.bea.com/wli/docs70/aiuser/2usrdef.htm>
6. Enter a name and description for the application view.
 The name should describe the set of functions performed by this application. Each application view name must be unique to its adapter. Valid characters are a-z, A-Z, 0-9, and _ (underscore).
 The description is seen by users when they use this application view in workflows.
 7. Select BEA_CLARIFY_1_0 from the Associated Adapter drop-down list.
 The following is an example of a completed window.

Figure 3-5 Completed Define New Application View Window

This page allows you to define a new application view

Folder: [Root](#)

Application View Name: *

Description:

Associated Adapter:

8. Click OK.

The Configure Connection Parameters window opens.

Figure 3-6 Configure Connection Parameters Window

Configure Connection Parameters

Application View Console WebLogic Console

Configure Connection

Administration
Add Service
Add Event
Deploy Application View

On this page, you supply parameters to connect to your EIS

The BEA Application Explorer generates schema information for a session stored at a location that must be known to the general adapter. Enter this session location here. A session can support multiple connections.

Once you have entered the **session path** location, click on the pulldown arrow for the **connection name**, which will display a selection list of valid connections.

Session Path *

Connection Name *

9. Enter the name of the BEA WebLogic Adapter for ClarifyCRM session path (sometimes known as the session base directory).

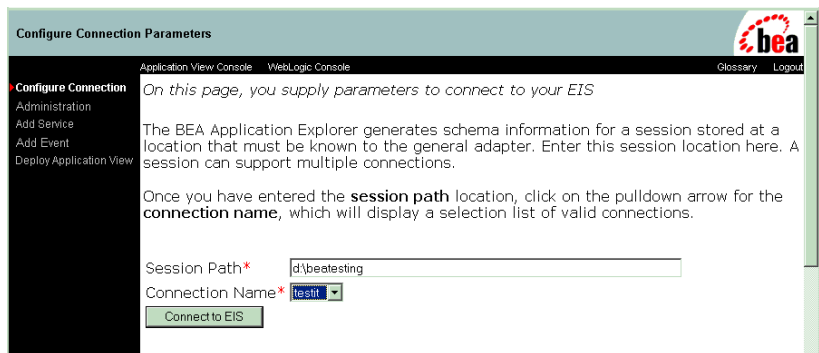
3 Creating Application Views for the CBO interface

This path holds your ClarifyCRM schema and connection information.

For more information about the session path and schemas, see [Chapter 5, “Creating Schema Repositories.”](#)

10. Select the connection name (sometimes known as the session name) from the Connection Name drop-down list.

Figure 3-7 Completed Configure Connection Parameters Window



11. Click Connect to EIS. The Application View Administration window opens.
12. Click Save. You have finished creating the application view.

You can now add a service or event to the application view, as described in [“Adding a Service to an Application View” on page 3-7](#) and [“Before Adding an Event to an Application View” on page 3-10](#). (You must add a service or event before you can deploy the application view.)

Note that you can access the Configure Connection Parameters window (displayed in a previous step) when the application view is not deployed, simply by clicking the Reconfigure connection parameters link. If the application view is deployed, you can access the window by first undeploying the application view.

Adding a Service to an Application View

You can add a service to an application view to support the application's functions. (For information about creating an application view, see [“Creating an Application View” on page 3-3.](#))

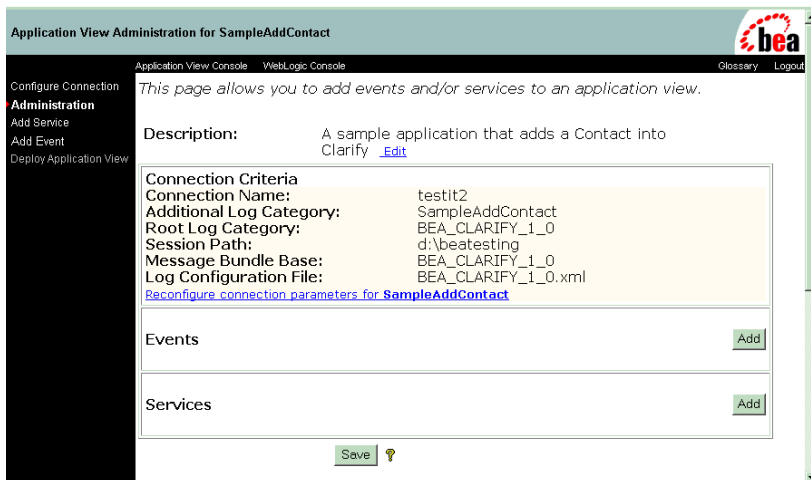
Note: If the application view is deployed, you must undeploy it before adding the service. For information about undeploying, see [“Undeploying an Application View” on page 3-22.](#)

To add a service:

1. Log on to the WebLogic Application View Console as described in [“Creating an Application View” on page 3-3.](#)
2. Select the folder in which this application view resides, and then select the application view.

The Application View Administration window opens.

Figure 3-8 Application View Administration Window (Active Section)



3. Select Add Service in the left pane, or click Add in the Services section.

The Add Service window opens.

Figure 3-9 Add Service Window

4. Specify the service's properties, which are described in the following table. Required values are indicated on the screen with an asterisk (*).

Table 3-1 Service Properties

| Property | Description |
|-----------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Unique Service Name | The name of the service that you are adding. The name must be unique within the application view. Valid characters include a-z, A-Z, 0-9, and underscore (_). |
| Clarify Server Name | Network id name for computer hosting ClarifyCRM Server |
| Clarify Database Name | Name of the database used by the ClarifyCRM Server |
| User Name | The user ID required to access the ClarifyCRM service. |
| Password | The password associated with the specified user ID. |
| minimum connections in pool | The minimum number of connections to the ClarifyCRM Server the adapter should open. |
| maximum connections in pool | The maximum number of connections to the ClarifyCRM Server the adapter should open. |

Table 3-1 Service Properties (Continued)

| Property | Description |
|-----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| schema | The name of the schema that describes the service. |
| Trace on/off | Generates a basic trace that displays the input XML (up to 300 bytes) before parsing, and shows the request being processed. For more information about tracing, see Chapter 6, “Using Tracing.” |
| Verbose Trace on/off | Generates a trace that displays configuration parameters used by the adapter. For more information about tracing, see Chapter 6, “Using Tracing.” |
| Document Trace on/off | Generates a trace that displays the input document after it was analyzed and the response document being returned. For more information about tracing, see Chapter 6, “Using Tracing.” |

5. Click Add.

The Application View Administration window opens.

Figure 3-10 Service Added to Application View

Add Event

Application View / Console / WebLogic Console / Glossary / Log

On this page, you add events to your application view.

Configure Connection
Administration
Add Service
Add Event
Deploy Application View

Unique Event Name: *

Clarify

Character Set Encoding* UTF-8

Driver*

url*

User Name

Password

Format* field

Maximum Rows 1

SQL Query*

SQL Post Query

Delete Keys

Polling Interval 20

Data Source Name

schema: BEA_clarify_event_row

settings

Trace on/off ☐

Verbose Trace on/off ☐

Document Trace on/off ☐

Add

6. Click Continue.

The Deploy Application View window opens.

To deploy the application view, see [“Deploying an Application View” on page 3-16](#).

Before Adding an Event to an Application View

Before using the BEA WebLogic adapter for ClarifyCRM for event processing, some preliminary database maintenance must be performed, a process that must be carried out only once. Once done, you can add events to Application Views. The steps that follow describe how to use the provided programs and files to set up your ClarifyCRM database to work with the adapter for event processing.

Creating the Event Tables

The adapter polls an event table inside the ClarifyCRM database, which is populated with information about external updates, through database triggers applied to the tables used by the CBO objects supported by the adapter. This database must be created using the SQL script files provided with the BEA WebLogic Adapter for ClarifyCRM.

For ClarifyCRM Systems with a database stored in Oracle, the `WLI_CLARIFY_EVENTS.sql` will create the events database inside Oracle. Also, the `WLI_CLARIFY_EVENTS_SEQ.sql` will create a second table that keeps track of the sequences, to prevent the same event being processed twice. You can run these scripts inside the Oracle SQL. For more information on running Oracle SQL scripts, refer to your Oracle documentation).

For ClarifyCRM Systems with a database stored in Microsoft SQL Server, the `wli_clarify_events.sql` will create the events database inside Microsoft SQL Server. Also, the `wli_clarify_events_seq.sql` will create a second table that keeps track of the sequences to prevent the same event being processed twice. You can run these scripts inside the SQL Query Analyzer application. For more information on running MSSQL SQL scripts, refer to your MS SQL Server documentation).

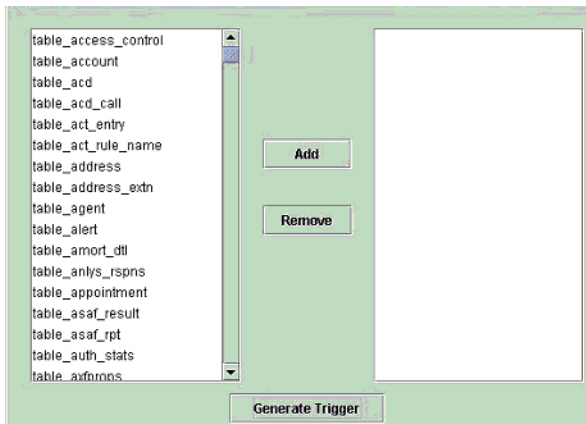
Applying DB Triggers

The ClarifyCRM trigger generator presents a list of all the ClarifyCRM tables that can be listened to. When used, the table will place Create, Delete, and Update triggers onto the tables you have selected. The trigger generator supports only Oracle 8i and Microsoft SQL Server 2000, both through the respective JDBC drivers, which must be downloaded from the RDBMS vendor's download site. For details on where to obtain these drivers, see the *BEA WebLogic Adapter for ClarifyCRM Installation and Configuration Guide*.

The ClarifyCRM Trigger Creation tool should be started using the provided batch files. The batch file you choose will depend on your ClarifyCRM installation. If you have ClarifyCRM installed on Oracle, run "Oracle_TriggerGen.bat"; if you use ClarifyCRM installed on MS SQL, run "MSSQL_TriggerGen.bat". You must modify the batch files to change the user name and password used for the logon and provide a DB Connection URL and database name for the ClarifyCRM database.

After launching the tool, a window similar to that shown in the following figure opens.

Figure 3-11 Trigger Creation Window



The left pane displays a list of tables that the SQL Listener will be able to listen to for events. To add triggers to a table or a set of tables:

1. Select the table(s) from the left pane and click Add.
2. Click Generate Triggers.

You can remove a table by selecting it in the right pane and clicking Remove.

Note: Because selecting Remove does not remove a table from the database, you must remove tables manually.

Adding an Event to an Application View

You can add an event to an application view to support the application's functions. (For information about creating an application view, see [“Creating an Application View” on page 3-3.](#))

Note: If the application view is deployed, you must undeploy it before adding the event. For information about undeploying, see [“Undeploying an Application View” on page 3-22.](#)

To add an event:

1. Log on to the WebLogic Application View Console as described in [“Creating an Application View” on page 3-3.](#)
2. Select the folder in which this application view resides, and then select the application view.

The Application View Administration window opens.

Figure 3-12 Application View Administration Window (Active Section)

Application View Administration for SampleAddContact

Application View Console WebLogic Console Glossary Logout

This page allows you to add events and/or services to an application view.

Description: A sample application that adds a Contact into Clarify [Edit](#)

Connection Criteria

Connection Name: testit2
 Additional Log Category: SampleAddContact
 Root Log Category: BEA_CLARIFY_1_0
 Session Path: d:\beatesting
 Message Bundle Base: BEA_CLARIFY_1_0
 Log Configuration File: BEA_CLARIFY_1_0.xml
[Reconfigure connection parameters for SampleAddContact](#)

Events [Add](#)

Services [Add](#)

[Save](#)

3. Select Add Event in the left pane, or click Add in the Events section.

The Add Event window opens.

Figure 3-13 Add Event Window

Add Event

Application View Console WebLogic Console Glossary Logout

On this page, you add events to your application view.

Unique Event Name:*

Clarify

| | |
|------------------|----------------------------------------|
| encoding | ISO-8859-1 |
| Driver* | <input type="text"/> |
| url* | <input type="text"/> |
| User Name | <input type="text"/> |
| Password | <input type="text"/> |
| Format* | field <input type="button" value="v"/> |
| Maximum Rows | 1 |
| SQL Query* | <input type="text"/> |
| SQL Post Query | <input type="text"/> |
| Delete Keys | <input type="text"/> |
| Polling Interval | 20 |
| Data Source Name | <input type="text"/> |

schema:

[Add](#)

3 Creating Application Views for the CBO interface

4. Specify the event's properties, which are described in the following table.
Required values are indicated on the screen with an asterisk (*).

Table 3-2 Event Properties

| Property | Description |
|-------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Unique Event Name | The name of the event that you are adding. The name must be unique within the application view. Valid characters include a-z, A-Z, 0-9, and underscore (_). |
| encoding | Sets the character set encoding to be used. (The default is UTF-8.) |
| Driver | Is the name of the JDBC driver that the adapter should use when querying the ClarifyCRM event table. Note that ClarifyCRM supports storing data in both Microsoft SQL Server and Oracle databases, so be sure to specify the JDBC driver for the DBMS your ClarifyCRM environment uses. |
| url | URL for the JDBC driver to access the database. |
| User Name | User name for the JDBC driver to access the database. |
| Password | Database password for user. |
| Format | Style of XML document containing results of the SQL query. The value is Column or Field. |
| Maximum Rows | Maximum number of rows to include in each document. |
| SQL_Query | The query that the BEA WebLogic Adapter for ClarifyCRM will issue against the event table. By default, the value is <code>SELECT * FROM event_tablename</code> , but you can specify a different query so that the event captures the data you desire. |
| SQL Post Query | SQL statement issued after the query. If omitted, <code>delete <fields> from table where <field values> or <keys></code> is used. |
| Delete Keys | Comma-separated list of keys to be used to build delete statement. If omitted, all fields are used. Do not use if a listener exit is provided. Case sensitive. |

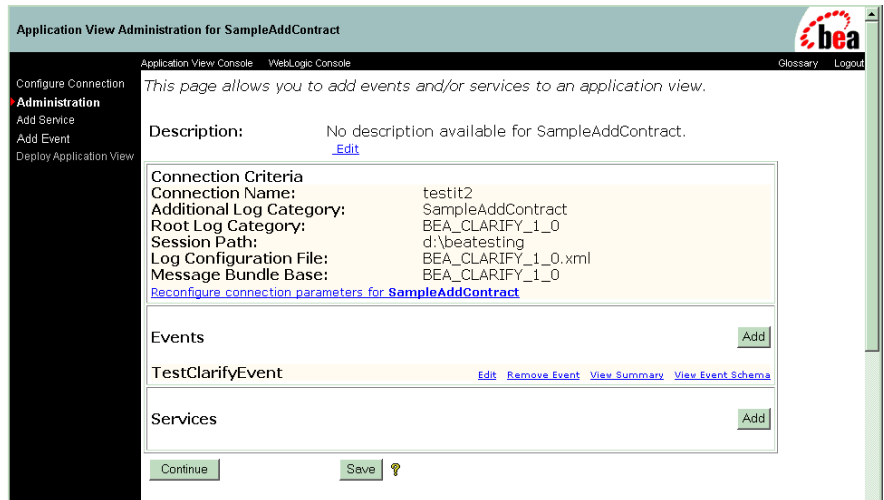
Table 3-2 Event Properties (Continued)

| Property | Description |
|-----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Polling Interval | Indicates how often, in seconds, the adapter should issue the SQL query. The higher this value, the longer the interval, and so the fewer system resources are used. The default value is 20 seconds. |
| Data Source Name | The name of the data source. |
| schema | The name of the schema that describes this event. |
| Trace on/off | Generates a basic trace that displays the input XML (up to 300 bytes) before parsing, and shows the request being processed. For more information about tracing, see Chapter 6, “Using Tracing.” |
| Verbose Trace on/off | Generates a trace that displays configuration parameters used by the adapter. For more information about tracing, see Chapter 6, “Using Tracing.” |
| Document Trace on/off | Generates a trace that displays the input document after it was analyzed and the response document being returned. For more information about tracing, see Chapter 6, “Using Tracing.” |

5. Click Add.

The Application View Administration window opens.

Figure 3-14 Event Added to Application View



To deploy the application view, see “Deploying an Application View” on page 3-16.

Deploying an Application View

You can deploy an application view when you have added at least one service or event to it. You must deploy an application view before you can test its services and events and before you can use it with WebLogic Server.

Deploying an application view places relevant metadata about its services and events in a run-time metadata repository. Deployment makes the application view available to other WebLogic Server clients, enabling it to interact with business processes.

Deploying an Application View With a Service

To deploy an application view with a service:

1. If you are not already in the Application View Console’s Administration window:

- a. Log on to the WebLogic Application View Console as described in [“Creating an Application View”](#) on page 3-3.
 - b. Select the folder in which this application view resides, and then select the application view. The Administration window opens.
2. Click Continue in the lower left corner of the window.

The Deploy Application View window opens.

Figure 3-15 Deploy Application View Window

Deploy Application View SampleAddContact to Server

Application View Console WebLogic Console Glossary Logout

Configure Connection Administration Add Service Add Event **Deploy Application View**

On this page you deploy your application view to the application server.

Required Service Parameters

Enable asynchronous service invocation? ☒

Connection Pool Parameters

Use these parameters to configure the connection pool used by this application view

Minimum Pool Size*

Maximum Pool Size*

Target Fraction of Maximum Pool Size*

Allow Pool to Shrink? ☒

Log Configuration

Set the log verbosity level for this application view.

Log warnings, errors, and audit messages

Configure Security

[Restrict Access to SampleAddContact using J2EE Security](#)

Deploy ☒ Deploy persistently? ☐ Save

3. Update service parameters, connection pool parameters, log configuration, and security as necessary. Parameters that appear on the screen with an asterisk (*) are required.

Table 3-3 Service Deployment Parameters

| Parameter | Description |
|----------------------------------------|-------------------------------------------------|
| Enable asynchronous service invocation | Enables you to run this service asynchronously. |
| Minimum Pool Size | The minimum number of threads to ClarifyCRM. |
| Maximum Pool Size | The maximum number of threads to ClarifyCRM. |

3 *Creating Application Views for the CBO interface*

Table 3-3 Service Deployment Parameters (Continued)

| Parameter | Description |
|--------------------------------------|---------------------------------------------------------|
| Target Fraction of Maximum Pool Size | The optimal thread level. |
| Allow Pool to Shrink | Enables the removal of threads that are no longer used. |
| Log verbosity level | The level of messages sent to the log. |
| Restrict Access using J2EE Security | Security parameter. |
| Deploy persistently | Persistence parameter. |

For more information about these parameters, see “Defining an Application View” in “Using Application Integration”:

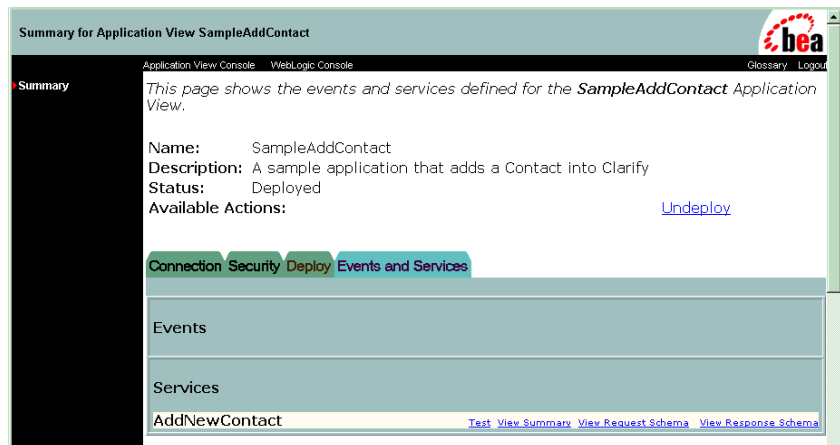
- For WebLogic Integration 7.0, see <http://edocs.bea.com/wli/docs70/aiuser/2usrdef.htm>

4. Click Deploy to deploy the application view.

Note: To save the parameters without first deploying the application view, click Save.

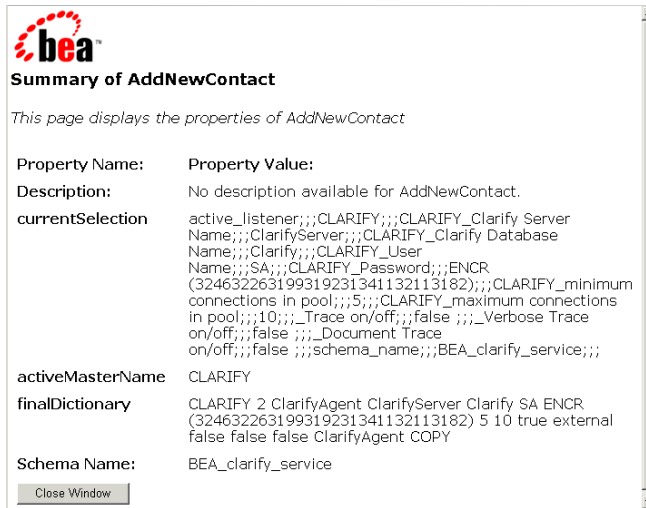
The Summary window for the application view opens.

Figure 3-16 A Deployed Application View With a Service



5. To view a summary of the service as deployed, select the service and click View Summary.

A summary of the service is displayed in a new window.



You are now ready to test your service, as described in [“Testing a Deployed Service” on page 3-24](#).

Deploying an Application View With an Event

To deploy an application view with an event:

1. If you are not already in the Application View Console’s Administration window:
 - a. Log on to the WebLogic Application View Console as described in [“Creating an Application View” on page 3-3](#).
 - b. Select the folder in which this application view resides, and then select the application view. The Administration window opens.
2. Click Continue in the lower left corner of the window.

The Deploy Application View window opens.

3 Creating Application Views for the CBO interface

Figure 3-17 Deploy Application View Window

Deploy Application View ClarifyEventTest to Server

Application View Console WebLogic Console

On this page you deploy your application view to the application server.

Required Event Parameters

Event Router URL*

Connection Pool Parameters

Use these parameters to configure the connection pool used by this application view

Minimum Pool Size*

Maximum Pool Size*

Target Fraction of Maximum Pool Size*

Allow Pool to Shrink? ☒

Log Configuration

Set the log verbosity level for this application view.

Configure Security

[Restrict Access to ClarifyEventTest using J2EE Security](#)

☒ Deploy persistently?

3. Update event parameters, connection pool parameters, log configuration, and security as necessary. Parameters that appear on the screen with an asterisk (*) are required.

Table 3-4 Event Deployment Parameters

| Parameter | Description |
|--------------------------------------|---------------------------------------------------------|
| Event Router URL | The location of the router for this event. |
| Minimum Pool Size | The minimum number of threads to ClarifyCRM. |
| Maximum Pool Size | The maximum number of threads to ClarifyCRM. |
| Target Fraction of Maximum Pool Size | The optimal thread level. |
| Allow Pool to Shrink | Enables the removal of threads that are no longer used. |
| Log verbosity level | The level of messages sent to the log. |
| Restrict Access using J2EE Security | Security parameter. |
| Deploy persistently | Persistence parameter. |

For more information about these parameters, see “Defining an Application View” in “Using Application Integration”:

- For WebLogic Integration 7.0, see <http://edocs.bea.com/wli/docs70/aiuser/2usrdef.htm>
4. Click Deploy to deploy the application view.
- Note:** To save the parameters without first deploying the application view, click Save.

The Summary window for the application view opens.

Figure 3-18 A Deployed Application View With an Event



5. To view a summary of the event as deployed, select the event and click View Summary.

A summary of the event is displayed in a new window.

| Summary of ClarifyEvent | |
|---------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| This page displays the properties of ClarifyEvent | |
| Property Name: | Property Value: |
| Description: | No description available for ClarifyEvent. |
| Clarify_Format | field |
| eisId | sql_encoding_UTF-8_Driver_com.microsoft.jdbc.sqlserver Name_sa_password_ENCR(3246322631993192313411321 Query__Delete Keys__Polling Interval_20_Data Source N |
| eisDictionary | Clarify 2 UTF-8 com.microsoft.jdbc.sqlserver.SQLServer (324632263199319231341132113182) field 1 select * fro |
| schema_name | clarify |
| active_listener | Clarify |
| Clarify_Driver | com.microsoft.jdbc.sqlserver.SQLServerDriver |
| Clarify_SQL | select * from ixte_events |

You are now ready to test your event, as described in [“Testing a Deployed Event” on page 3-27](#).

Undeploying an Application View

Once an application view is deployed, it cannot be modified. In order to modify it (for example, to add another service or event) you must first undeploy it.

To undeploy an application view:

1. If you are not already in the Application View Console’s Summary window for the application view you want to undeploy:
 - a. Log on to the WebLogic Application View Console as described in [“Creating an Application View” on page 3-3](#).
 - b. Select the folder in which this application view resides, and then select the application view. The Summary window opens.

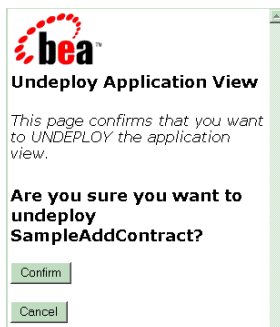
Figure 3-19 Undeploying an Application View



2. Click Undeploy.

A confirmation window asks if you are sure you want to undeploy the application view.

Figure 3-20 Undeployment Confirmation

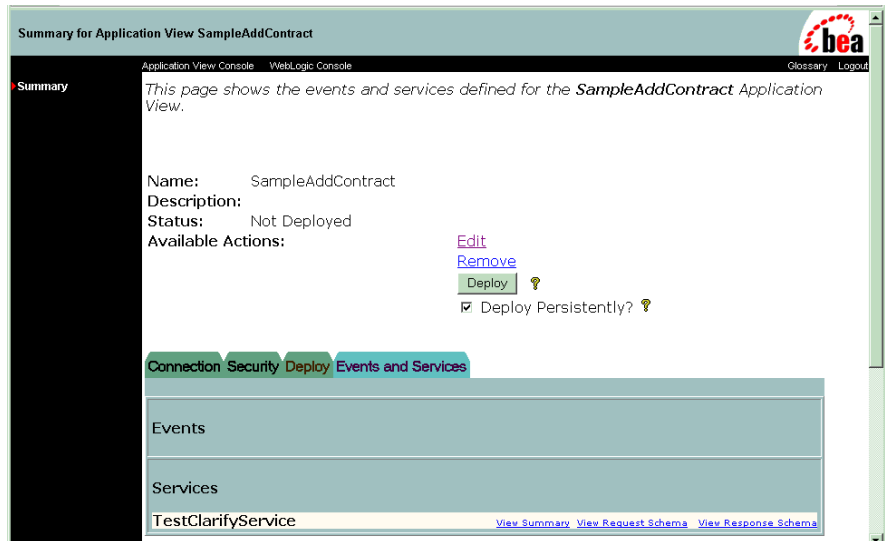


3. Click Confirm to undeploy the application view.

For more information see “Optional Step: Undeploying an Application View” in “Defining an Application View” in *Using Application Integration*:

- For WebLogic Integration 7.0, see <http://edocs.bea.com/wli/docs70/aiuser/2usrdef.htm>

Figure 3-21 Undeployed Application View



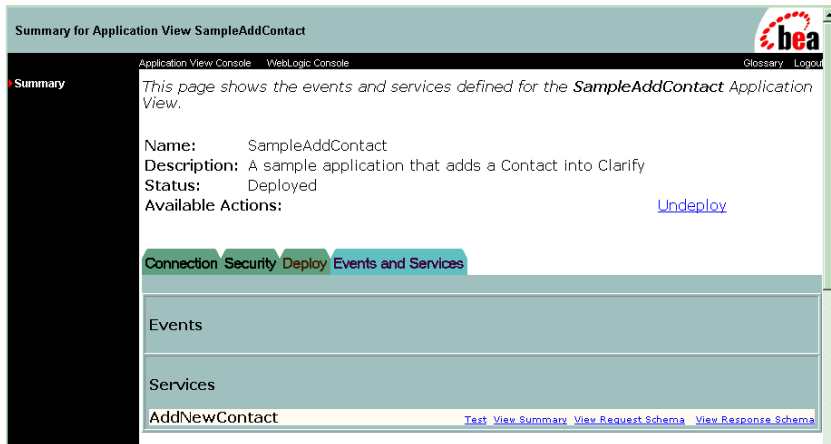
Testing a Deployed Service

After you create and deploy an application view that contains a service, you can test the application view service to evaluate whether it interacts properly with the BEA WebLogic Adapter for ClarifyCRM.

To test a deployed service:

1. If you are not already in the Application View Console's Summary window for the application view service you want to test:
 - a. Log on to the WebLogic Application View Console as described in [“Creating an Application View” on page 3-3](#).
 - b. Select the folder in which this application view resides, and then select the application view. The Summary window opens.

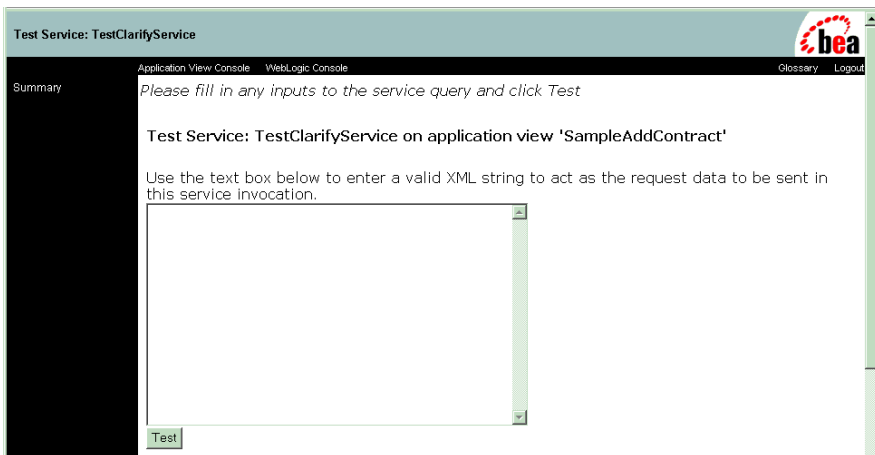
Figure 3-22 Summary Window for a Service to Be Tested



2. For the service you want to test, click Test.

The Test Service window opens.

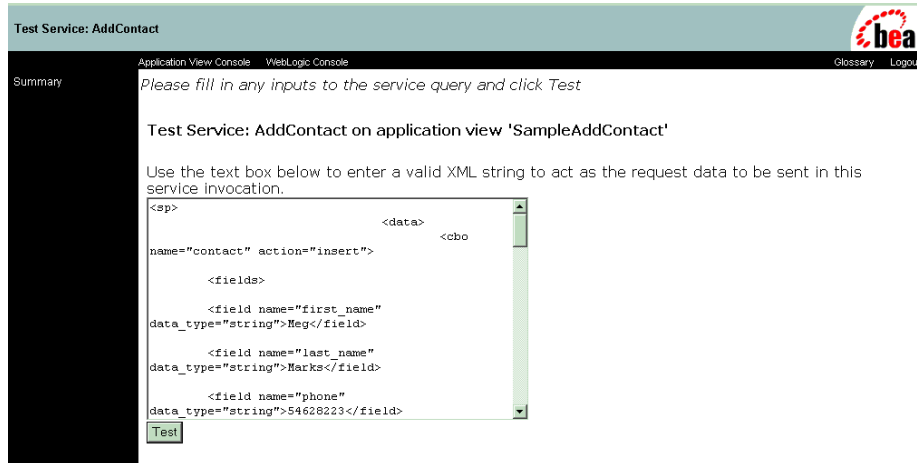
Figure 3-23 Test Service Window



3. Enter the XML that invokes the service request.

In the following figure, the request that has been entered adds a new contact to the ClarifyCRM database.

Figure 3-24 Entering Request XML



4. Click Test to test the service.

The request and response documents are then displayed.

Figure 3-25 Test Service Results



If the test fails, a response document displays the appropriate error messages. You can correct the error and resubmit the request.

If the test succeeds, the response document that corresponds to the request is displayed in the output field.

You have successfully deployed and tested the application view service.

If you wish, you can write custom code to create a workflow. For more information, see “Using Application Views in the Studio” in *Using Application Integration*:

- For WebLogic Integration 7.0, see
<http://edocs.bea.com/wli/docs70/aiuser/3usruse.htm>

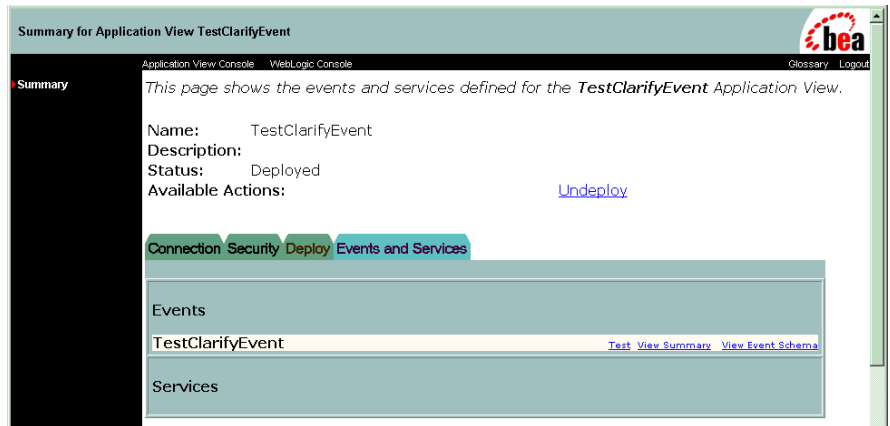
Testing a Deployed Event

After you create and deploy an application view that contains an event, you can test the application view event to evaluate whether it interacts properly with the BEA WebLogic Adapter for ClarifyCRM.

To test a deployed event:

1. If you are not already in the Application View Console’s Summary window for the application view event you want to test:
 - a. Log on to the WebLogic Application View Console as described in “[Creating an Application View](#)” on page 3-3.
 - b. Select the folder in which this application view resides, and then select the application view. The Summary window opens.

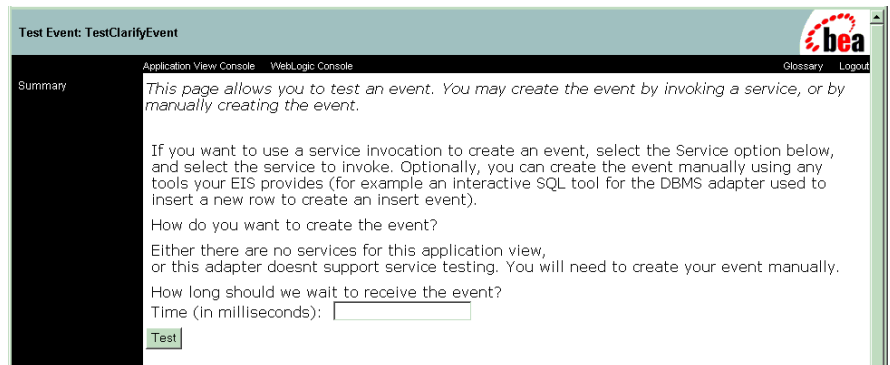
Figure 3-26 Summary Window for an Event to Be Tested



2. For the event you want to test, click Test.

The Test Event window opens.

Figure 3-27 Test Event Window



3. Enter a time, in milliseconds, that indicates how long to wait to receive the event. Make sure to specify sufficient time to initiate the event.

Figure 3-28 Testing an Event

Test Event: TestClarifyEvent

Application View Console WebLogic Console Glossary Logout

Summary

This page allows you to test an event. You may create the event by invoking a service, or by manually creating the event.

If you want to use a service invocation to create an event, select the Service option below, and select the service to invoke. Optionally, you can create the event manually using any tools your EIS provides (for example an interactive SQL tool for the DBMS adapter used to insert a new row to create an insert event).

How do you want to create the event?

Either there are no services for this application view, or this adapter doesn't support service testing. You will need to create your event manually.

How long should we wait to receive the event?

Time (in milliseconds):

Test

4. Click Test to test the event.

When the event occurs, the event test result document is displayed.

Figure 3-29 Event Test Results

Test Result for ClarifyEvent

Application View Console WebLogic Console Glossary Logout

Summary

This page shows the results from testing an event.

Generated event of type ClarifyEvent on application view SampleAddContact

```
<?xml version="1.0" encoding="UTF-8"?>
<ClarifyEvents table="ixte_events">
  <row>
    <eventId type="4">119</eventId>
    <objectId type="4">9</objectId>
    <object_type type="1">site</object_type>
    <datestamp type="93">2002-11-11
18:44:17.733</datestamp>
    <username type="1">sa</username>
    <sent_flag type="1">n</sent_flag>
    <event_type type="1">UPDATE</event_type>
  </row>
</ClarifyEvents>
```

Execution time: 1042 (ms)

If the test fails, a message is displayed indicating that the event timed out. You can correct the error and retest the event.

If the test succeeds, the event result document is displayed.

You have successfully deployed and tested the application view event.

If you wish, you can write custom code to create a workflow. For more information, see “Using Application Views in the Studio” in *Using Application Integration*:

- For WebLogic Integration 7.0, see <http://edocs.bea.com/wli/docs70/aiuser/3usruse.htm>

Using a Service or Event in a Workflow

This section contains specific examples of the use of a service and an event in a workflow. A workflow is the process that links an event in one Enterprise Integration System (EIS), such as a ClarifyCRM system, to the services in another EIS. You can build complex workflows that involve multiple events and services, transformations, and other business logic.

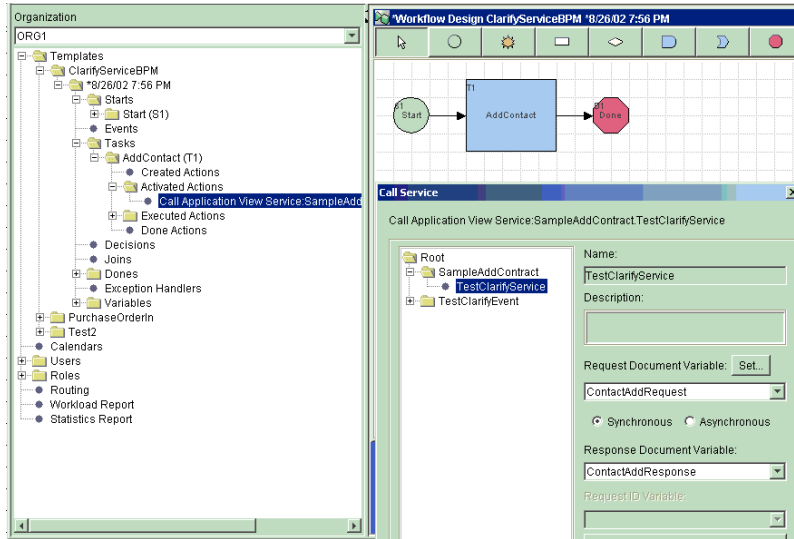
This section does not provide a comprehensive description of workflows. For more information, see “Using Application Views in the Studio” in *Using Application Integration*:

- For WebLogic Integration 7.0, see <http://edocs.bea.com/wli/docs70/aiuser/3usruse.htm>

Example: Using a Service in a Workflow

In this example, a workflow is built using a static document called `ContactAddRequest`. This document is then used as input to the ClarifyCRM service defined in “[Adding a Service to an Application View](#)” on page 3-7. The resulting document is placed in the `ContactResponse`.

The following figures illustrate the service request in a workflow. In this workflow, the input document is defined as an action of the Start node. The `AddContact` action defines the use of the XML document passed to it from the Start node and sends it to the ClarifyCRM service.

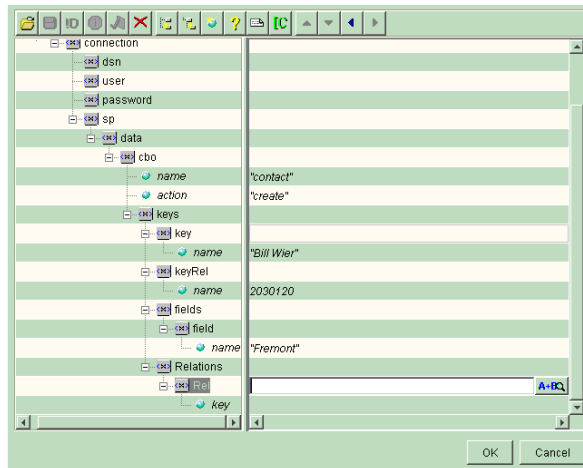
Figure 3-30 Application View Service in a Workflow

About the ClarifyCRM Request

The following figure shows the static document created for the workflow. In this case, the document is a `ContactRequest` intended to be added into the ClarifyCRM application. In a real-world workflow, the document would typically be the result of an event in another EIS, with transformations and enrichment steps performed. For testing, however, this Service Request Template allows you to fill in test values.

3 Creating Application Views for the CBO interface

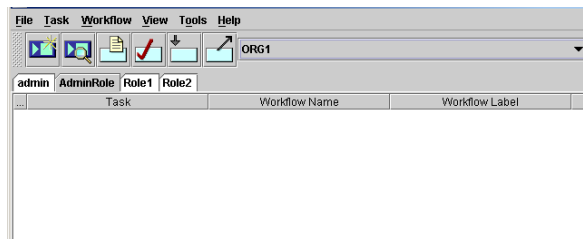
Figure 3-31 Static Document



Start the Workflow

Once the workflow is built, Worklist is used to start the process manually. Worklist is required since the input document is static and not the result of an event from another EIS.

Figure 3-32 Starting the Workflow



The ClarifyCRM Response

When the workflow is started, the Task1 step takes the document and sends it to the ClarifyCRM application. Upon completion of the task, the workflow places the response XML document in the ContactAddResponse variable. By pressing the View Response Definition from the Call Service window, the XML Schema is displayed.

Figure 3-33 Response Document



3 *Creating Application Views for the CBO interface*

4 Creating Application Views for the ClearBasic Interface

The information in this chapter applies only to the ClearBasic interface of the adapter for ClarifyCRM. For information on creating application views for the CBO interface of the adapter for ClarifyCRM, see [Chapter 3, “Creating Application Views for the CBO interface.”](#)

When you define an application view, you create an XML-based interface between WebLogic Server and a particular Enterprise Information System (EIS) application within your enterprise. Once you create an application view, a business analyst can use it to create business processes that use the application. With the BEA WebLogic Adapter for ClarifyCRM, you can create any number of application views, each with any number of services and events.

This section provides information about application views and deployed services and events, and includes the following topics:

- [Creating an Application View Folder](#)
- [Creating an Application View](#)
- [Adding a Service to an Application View](#)
- [Adding an Event to an Application View](#)
- [Deploying an Application View](#)
- [Undeploying an Application View](#)

- [Testing a Deployed Service](#)
- [Testing a Deployed Event](#)
- [Using a Service or Event in a Workflow](#)

Creating an Application View Folder

Application views reside within WebLogic Integration. WebLogic Integration provides you with a root folder in which you can store all of your application views; if you wish, you can create additional folders to organize related application views into groups.

To create an application view folder:

1. Log on to the WebLogic Integration Application View Console at `//appserver-host:port/wlai`.

Here, *appserver-host* is the IP address or host name where the WebLogic Integration Server is installed, and *port* is the socket on which the server is listening. The port, if not changed during installation, defaults to 7001.

2. If prompted, enter a user name and password.

Note: If the user name is not `system`, it must be included in the adapter group. For more information on adding the administrative server user name to the adapter group, see the *BEA WebLogic Adapter for ClarifyCRM Installation and Configuration Guide*.

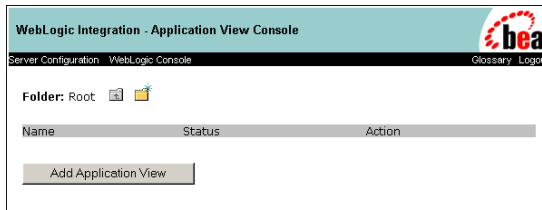
For more information, see “Logging On to the WebLogic Integration Application View Console” in “Defining an Application View” in *Using Application Integration*:

`http://edocs.bea.com/wli/docs70/aiuser/2usrdef.htm`

3. Click Login.

The WebLogic Integration Application View Console opens.

Figure 4-1 Application View Console Main Window



4. Double-click the new folder icon. The Add Folder window opens.

Figure 4-2 Add Folder Window



5. Supply a name for the folder, and then click Save.

You have finished creating the application view folder. To create an application view, see [“Creating an Application View” on page 4-3](#).

Creating an Application View

To create an application view:

1. Log on to the WebLogic Application View Console at `//appserver-host:port/wlai`.

Here, *appserver-host* is the IP address or host name where the WebLogic Integration Server is installed, and *port* is the socket on which the server is listening. The port, if not changed during installation, defaults to 7001.

2. If prompted, enter a user name and password.

Note: If the user name is not `system`, it must be included in the `adapter` group. For more information on adding the administrative server user name to the

4 Creating Application Views for the ClearBasic Interface

adapter group, see the *BEA WebLogic Adapter for ClarifyCRM Installation and Configuration Guide*.

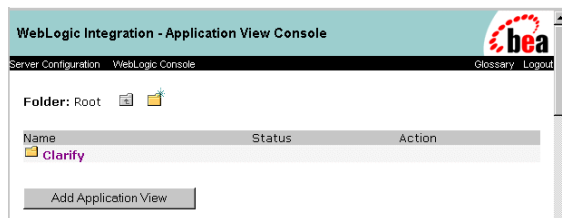
For more information, see “Logging On to the WebLogic Integration Application View Console” in “Defining an Application View” in *Using Application Integration*:

<http://edocs.bea.com/wli/docs70/aiuser/2usrdef.htm>

3. Click Login.

The WebLogic Integration Application View Console opens.

Figure 4-3 Selecting a Folder in the Main Window

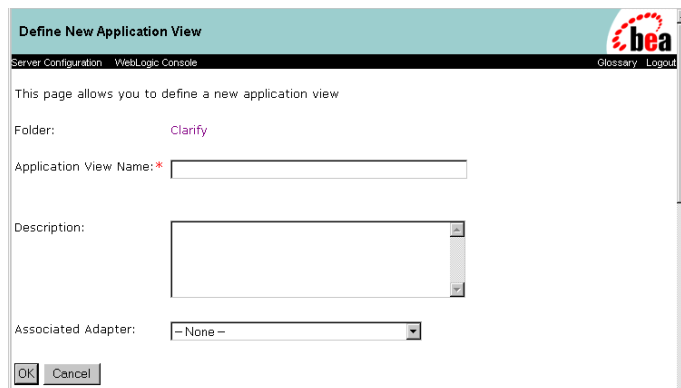


4. Select the desired application view folder.

5. Click Add Application View.

The Define New Application View window opens.

Figure 4-4 Define New Application View Window



An application view enables a set of functions for the adapter's target EIS application. For detailed information, see “Defining an Application View” in *Using Application Integration*:

<http://edocs.bea.com/wli/docs70/aiuser/2usrdef.htm>

6. Enter a name and description for the application view.

The name should describe the set of functions performed by this application. Each application view name must be unique to its adapter. Valid characters are a-z, A-Z, 0-9, and _ (underscore).

The description is seen by users when they use this application view in workflows.

7. Select BEA_CLARIFYCB_1_0 from the Associated Adapter drop-down list.

The following is an example of a completed window.

Figure 4-5 Completed Define New Application View Window

Define New Application View

Server Configuration > WebLogic Console

This page allows you to define a new application view

Folder: Clarify

Application View Name: * CLARIFYCB

Description: BEA Adapter for Clarify ClearBasic

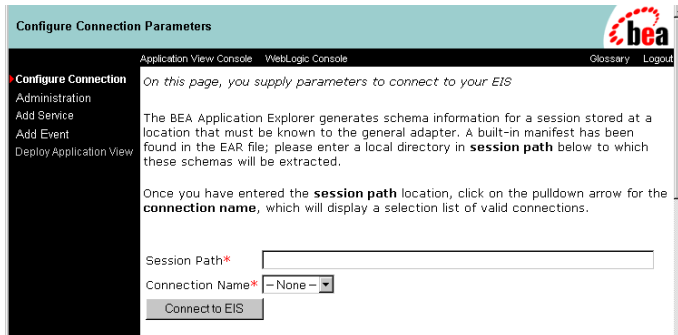
Associated Adapter: BEA_CLARIFYCB_1_0

OK Cancel

8. Click OK.

The Configure Connection Parameters window opens.

Figure 4-6 Configure Connection Parameters Window



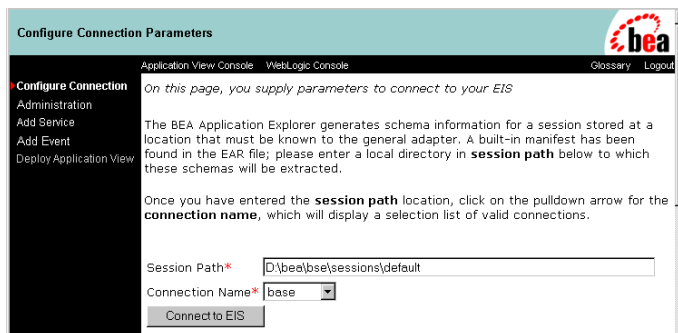
9. Enter the name of the BEA WebLogic Adapter for ClarifyCRM session path (sometimes known as the session base directory).

This path holds your ClarifyCRM schema and connection information.

For more information about the session path and schemas, see [Chapter 5, “Creating Schema Repositories.”](#)

10. Select the connection name (sometimes known as the session name) from the Connection Name drop-down list.

Figure 4-7 Completed Configure Connection Parameters Window



11. Click Connect to EIS. The Application View Administration window opens.

12. Click Save. You have finished creating the application view.

You can now add a service or event to the application view, as described in [“Adding a Service to an Application View”](#) on page 4-7 and [“Adding an Event](#)

to an Application View” on page 4-10. (You must add a service or event before you can deploy the application view.)

Note that you can access the Configure Connection Parameters window (displayed in a previous step) when the application view is not deployed, simply by clicking the Reconfigure connection parameters link. If the application view is deployed, you can access the window by first undeploying the application view.

Adding a Service to an Application View

You can add a service to an application view to support the application's functions. (For information about creating an application view, see “[Creating an Application View](#)” on page 4-3.)

Note: If the application view is deployed, you must undeploy it before adding the service. For information about undeploying, see “[Undeploying an Application View](#)” on page 4-20.

To add a service:

1. Log on to the WebLogic Application View Console as described in “[Creating an Application View](#)” on page 4-3.
2. Select the folder in which this application view resides, and then select the application view.

The Application View Administration window opens.

Figure 4-8 Application View Administration Window (Active Section)

Application View Administration for Clarify.CLARIFYCB

Application View Console WebLogic Console Glossary Logout

Configure Connection
Administration
Add Service
Add Event
Deploy Application View

This page allows you to add events and/or services to an application view.

Description: BEA Adapter for Clarify ClearBasic [Edit](#)

Connection Criteria

| | |
|--------------------------|------------------------------|
| Connection Name: | base |
| Additional Log Category: | CLARIFYCB |
| NOT_VALID_000: | true |
| Root Log Category: | BEA_CLARIFYCB_1_0 |
| Session Path: | D:\bea\bpse\sessions\default |
| Message Bundle Base: | BEA_CLARIFYCB_1_0 |
| Log Configuration File: | BEA_CLARIFYCB_1_0.xml |

[Reconfigure connection parameters for CLARIFYCB](#)

Events [Add](#)

Services [Add](#)

[Save](#) ?

3. Select Add Service in the left pane, or click Add in the Services section.

The Add Service window opens.

Figure 4-9 Add Service Window

Add Service

Application View Console WebLogic Console Glossary Logout

Configure Connection
Administration
Add Service
Add Event
Deploy Application View

On this page, you add services to your application view.

Unique Service Name:*

CLARIFYCB_IWAYWTC_3ATMI

| | |
|------------------------------------|---------------------------------------------------------------|
| Host_URL_for_WTC* | <input type="text" value="t3://localhost:7001"/> |
| JNDI_Name_of_WTC-based_EJB* | <input type="text" value="tbi.TuxConnectorEJB.TuxConnector"/> |
| TP_Call_is_Asynchronous_(TPACALL)* | <input type="checkbox"/> |

schema:

settings

| | |
|-----------------------|--------------------------|
| Trace_on/off | <input type="checkbox"/> |
| Verbose_Trace_on/off | <input type="checkbox"/> |
| Document_Trace_on/off | <input type="checkbox"/> |

[Add](#)

4. Specify the service's properties, which are described in the following table.
Required values are indicated on the screen with an asterisk (*).

Table 4-1 Service Properties

| Property | Description |
|----------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Unique Service Name | The name of the service that you are adding. The name must be unique within the application view. Valid characters include a-z, A-Z, 0-9, and underscore (_). |
| Host URL for WTC | URL of the WebLogic system hosting the WebLogic Tuxedo Connector (WTC). |
| JNDI Name of WTC-based EJB | Java Naming Service entry for WTC-based ATMI Enterprise Java Bean. |
| TPCall is Asynchronous (TPACALL) | A flag to determine whether the call to Tuxedo is done synchronously (the default) or asynchronously. |
| schema | The name of the schema that describes the service. |
| Trace_on/off | Generates a basic trace that displays the input XML (up to 300 bytes) before parsing, and shows the request being processed. For more information about tracing, see Chapter 6, “Using Tracing.” |
| Verbose_Trace_on/off | Generates a trace that displays configuration parameters used by the adapter. For more information about tracing, see Chapter 6, “Using Tracing.” |
| Document_Trace_on/off | Generates a trace that displays the input document after it was analyzed and the response document being returned. For more information about tracing, see Chapter 6, “Using Tracing.” |

5. Click Add.

The Application View Administration window opens.

Figure 4-10 Service Added to Application View

The screenshot shows the 'Application View Administration for Clarify.CLEARIFYCB' web interface. The left sidebar contains navigation links: 'Configure Connection', 'Administration' (selected), 'Add Service', 'Add Event', and 'Deploy Application View'. The main content area has a title bar with 'Application View Console' and 'WebLogic Console', and a 'Glossary Logout' link. Below the title bar is a message: 'This page allows you to add events and/or services to an application view.' The 'Description' field is 'BEA Adapter for Clarify ClearBasic' with an 'Edit' link. The 'Connection Criteria' section lists: 'Connection Name: base', 'Additional Log Category: CLARIFYCB', 'NOT_VALID_000: true', 'Root Log Category: BEA_CLARIFYCB_1_0', 'Session Path: d:\bea\bea\sessions\default', 'Message Bundle Base: BEA_CLARIFYCB_1_0', and 'Log Configuration File: BEA_CLARIFYCB_1_0.xml'. A link 'Reconfigure connection parameters for CLARIFYCB' is below. There are 'Events' and 'Services' sections, each with an 'Add' button. The 'Services' section shows 'CB_EXESUB' with links 'Edit', 'Remove Service', 'View Summary', 'View Request Schema', and 'View Response Schema'. At the bottom are 'Continue' and 'Save' buttons with a key icon.

6. Click Continue.

The Deploy Application View window opens.

To deploy the application view, see [“Deploying an Application View” on page 4-13](#).

Adding an Event to an Application View

You can add an event to an application view to support the application's functions. (For information about creating an application view, see [“Creating an Application View” on page 4-3](#).)

Note: If the application view is deployed, you must undeploy it before adding the event. For information about undeploying, see [“Undeploying an Application View” on page 4-20](#).

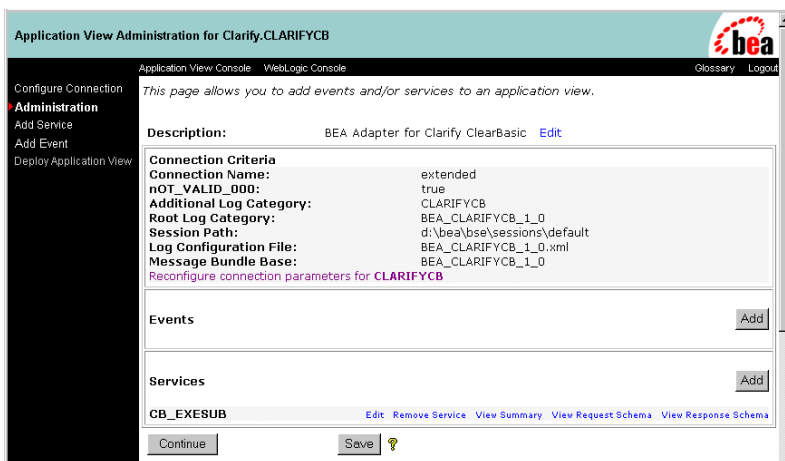
To add an event:

1. Log on to the WebLogic Application View Console as described in [“Creating an Application View” on page 4-3](#).

2. Select the folder in which this application view resides, and then select the application view.

The Application View Administration window opens.

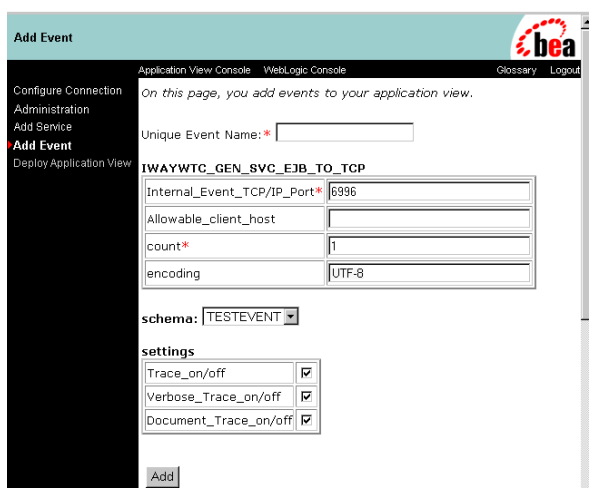
Figure 4-11 Application View Administration Window (Active Section)



3. Select Add Event in the left pane, or click Add in the Events section.

The Add Event window opens.

Figure 4-12 Add Event Window



4 *Creating Application Views for the ClearBasic Interface*

4. Specify the event's properties, which are described in the following table.
Required values are indicated on the screen with an asterisk (*).

Table 4-2 Event Properties

| Property | Description |
|----------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Unique Event Name | The name of the event that you are adding. The name must be unique within the application view. Valid characters include a-z, A-Z, 0-9, and underscore (_). |
| Internal Event TCP/IP Port | The TCP port to which Tuxedo events are posted into WebLogic Integration. |
| Allowable_client_host | The TCP/IP host name or address of WTC ATMI EJB clients sending events into WebLogic Integration. |
| count | The number of simultaneous threads created to service Tuxedo event requests. |
| encoding | The character encoding system to use. It defaults to UTF-8. |
| schema | The name of the schema that describes this event. |
| Trace_on/off | Generates a basic trace that displays the input XML (up to 300 bytes) before parsing, and shows the request being processed. For more information about tracing, see Chapter 6, "Using Tracing." |
| Verbose_Trace_on/off | Generates a trace that displays configuration parameters used by the adapter. For more information about tracing, see Chapter 6, "Using Tracing." |
| Document_Trace_on/off | Generates a trace that displays the input document after it was analyzed and the response document being returned. For more information about tracing, see Chapter 6, "Using Tracing." |

5. Click Add.

The Application View Administration window opens.

Figure 4-13 Event Added to Application View

To deploy the application view, see [“Deploying an Application View”](#) on page 4-13.

Deploying an Application View

You can deploy an application view when you have added at least one service or event to it. You must deploy an application view before you can test its services and events and before you can use it with WebLogic Server.

Deploying an application view places relevant metadata about its services and events in a run-time metadata repository. Deployment makes the application view available to other WebLogic Server clients, enabling it to interact with business processes.

Deploying an Application View With a Service

To deploy an application view with a service:

1. If you are not already in the Application View Console’s Administration window:

4 Creating Application Views for the ClearBasic Interface

- a. Log on to the WebLogic Application View Console as described in “[Creating an Application View](#)” on page 4-3.
 - b. Select the folder in which this application view resides, and then select the application view. The Administration window opens.
2. Click Continue in the lower left corner of the window.
- The Deploy Application View window opens.

Figure 4-14 Deploy Application View Window

Deploy Application View Clarify.CLARIFYCB to Server

Application View Console WebLogic Console Glossary Logout

Configure Connection
Administration
Add Service
Add Event
Deploy Application View

On this page you deploy your application view to the application server.

Required Service Parameters

Enable asynchronous service invocation? ☒

Required Event Parameters

Event Router URL*

Connection Pool Parameters

Use these parameters to configure the connection pool used by this application view

Minimum Pool Size*

Maximum Pool Size*

Target Fraction of Maximum Pool Size*

Allow Pool to Shrink? ☒

Log Configuration

Set the log verbosity level for this application view.

Configure Security

[Restrict Access to CLARIFYCB using J2EE Security](#)

☒ Deploy persistently?

3. Update service parameters, connection pool parameters, log configuration, and security as necessary. Parameters that appear on the screen with an asterisk (*) are required.

Table 4-3 Service Deployment Parameters

| Parameter | Description |
|----------------------------------------|--------------------------------------------------------------------|
| Enable asynchronous service invocation | Enables you to run this service asynchronously. |
| Minimum Pool Size | The minimum number of threads to connect to the ClarifyCRM Server. |

Table 4-3 Service Deployment Parameters

| Parameter | Description |
|--------------------------------------|--------------------------------------------------------------------|
| Maximum Pool Size | The maximum number of threads to connect to the ClarifyCRM Server. |
| Target Fraction of Maximum Pool Size | The optimal thread level. |
| Allow Pool to Shrink | Enables the removal of threads that are no longer used. |
| Log verbosity level | The level of messages sent to the log. |
| Restrict Access using J2EE Security | Security parameter. |
| Deploy persistently | Persistence parameter. |

For more information about these parameters, see “Defining an Application View” in “Using Application Integration”:

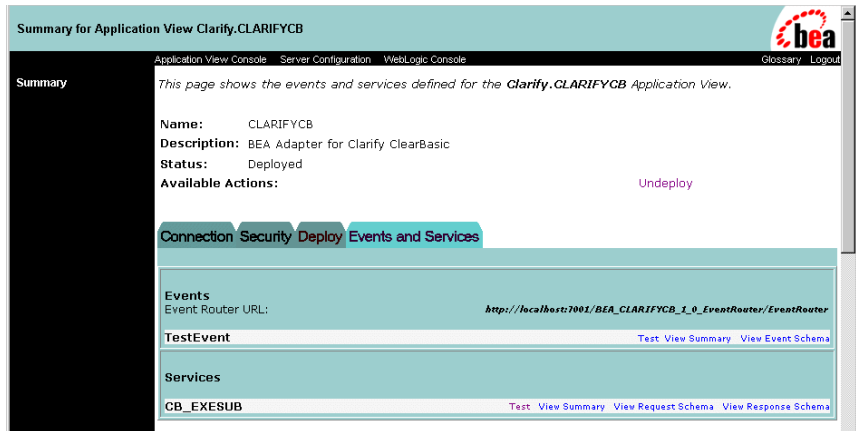
<http://edocs.bea.com/wli/docs70/aiuser/2usrdef.htm>

4. Click Deploy to deploy the application view.

Note: To save the parameters without first deploying the application view, click Save.

The Summary window for the application view opens.

Figure 4-15 A Deployed Application View With a Service and Event



5. To view a summary of the service as deployed, select the service and click View Summary.

Figure 4-16 Summary of Service Window



You are now ready to test your service, as described in “Testing a Deployed Service” on page 4-21.

Deploying an Application View With an Event

To deploy an application view with an event:

1. If you are not already in the Application View Console's Administration window:
 - a. Log on to the WebLogic Application View Console as described in [“Creating an Application View” on page 4-3](#).
 - b. Select the folder in which this application view resides, and then select the application view. The Administration window opens.
2. Click Continue in the lower left corner of the window.

The Deploy Application View window opens.

Figure 4-17 Deploy Application View Window

Deploy Application View Clarify.CLARIFYCB to Server

Application View Console WebLogic Console Glossary Logout

On this page you deploy your application view to the application server.

Required Service Parameters

Enable asynchronous service invocation? ☒

Required Event Parameters

Event Router URL*

Connection Pool Parameters

Use these parameters to configure the connection pool used by this application view

Minimum Pool Size*

Maximum Pool Size*

Target Fraction of Maximum Pool Size*

Allow Pool to Shrink? ☒

Log Configuration

Set the log verbosity level for this application view.

Configure Security

[Restrict Access to CLARIFYCB using J2EE Security](#)

☒ Deploy persistently?

3. Update event parameters, connection pool parameters, log configuration, and security as necessary. Parameters that appear on the screen with an asterisk (*) are required.

Table 4-4 Event Deployment Parameters

| Parameter | Description |
|--------------------------------------|---------------------------------------------------------------------|
| Event Router URL | The location of the router for this event. |
| Minimum Pool Size | The minimum number of threads to connect to the Clarify CRM Server. |
| Maximum Pool Size | The maximum number of threads to connect to the ClarifyCRM Server. |
| Target Fraction of Maximum Pool Size | The optimal thread level. |
| Allow Pool to Shrink | Enables the removal of threads that are no longer used. |
| Log verbosity level | The level of messages sent to the log. |
| Restrict Access using J2EE Security | Security parameter. |
| Deploy persistently | Persistence parameter. |

For more information about these parameters, see “Defining an Application View” in “Using Application Integration”:

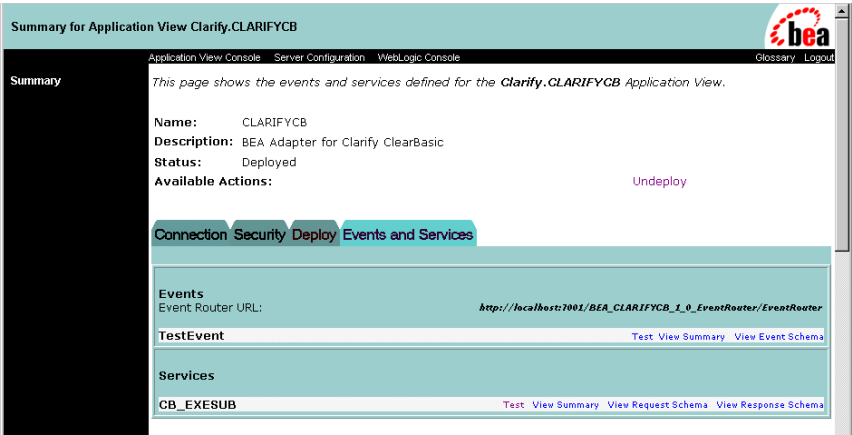
<http://edocs.bea.com/wli/docs70/aiuser/2usrdef.htm>

4. Click Deploy to deploy the application view.

Note: To save the parameters without first deploying the application view, click Save.

The Summary window for the application view opens.

Figure 4-18 A Deployed Application View With an Event



5. To view a summary of the event as deployed, select the event and click View Summary.

Figure 4-19 Summary of Event Window



You are now ready to test your event, as described in “Testing a Deployed Event” on page 4-24.

Undeploying an Application View

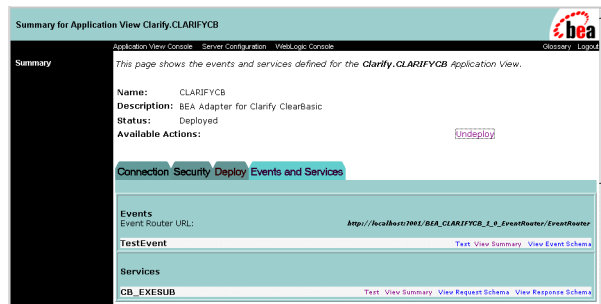
Once an application view is deployed, it cannot be modified. In order to modify it (for example, to add another service or event) you must first undeploy it.

To undeploy an application view:

1. If you are not already in the Application View Console's Summary window for the application view you want to undeploy:
 - a. Log on to the WebLogic Application View Console as described in [“Creating an Application View”](#) on page 4-3.
 - b. Select the folder in which this application view resides, and then select the application view.

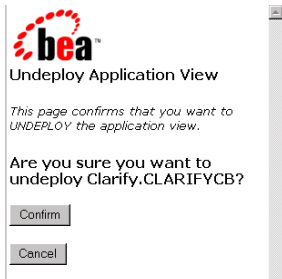
The Summary window opens.

Figure 4-20 Undeploying an Application View



2. Click Undeploy.

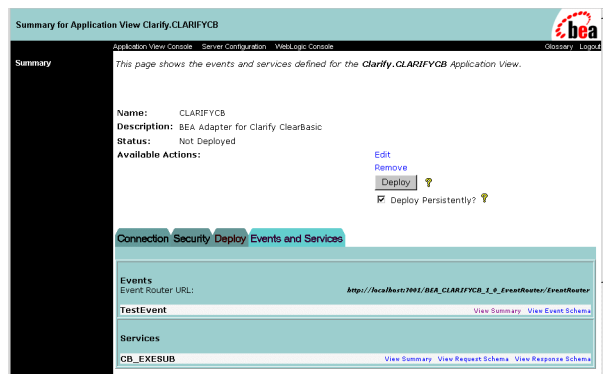
A confirmation window asks if you are sure you want to undeploy the application view.

Figure 4-21 Undeployment Confirmation

3. Click Confirm to undeploy the application view.

For more information see “Optional Step: Undeploying an Application View” in “Defining an Application View” in *Using Application Integration*:

<http://edocs.bea.com/wli/docs70/aiuser/2usrdef.htm>

Figure 4-22 Undeployed Application View

Testing a Deployed Service

After you create and deploy an application view that contains a service, you can test the application view service to evaluate whether it interacts properly with the BEA WebLogic Adapter for ClarifyCRM.

To test a deployed service:

4 Creating Application Views for the ClearBasic Interface

1. If you are not already in the Application View Console's Summary window for the application view service you want to test:
 - a. Log on to the WebLogic Application View Console as described in [“Creating an Application View” on page 4-3](#).
 - b. Select the folder in which this application view resides, and then select the application view.

The Summary window opens.

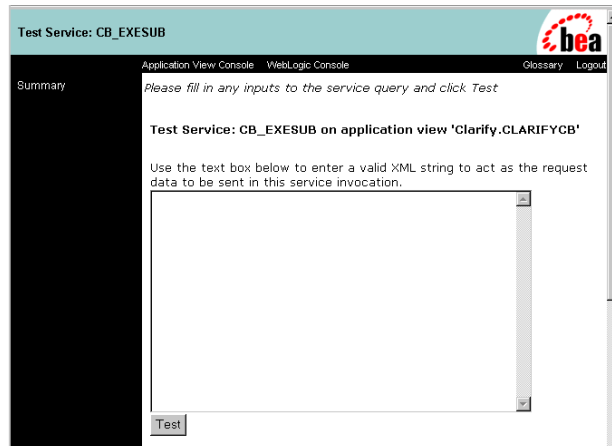
Figure 4-23 Summary Window for a Service to Be Tested



2. For the service you want to test, click Test.

The Test Service window opens.

Figure 4-24 Test Service Window



3. Enter the XML that invokes the service request.

In the following figure, the request that has been entered adds a new contact to the ClarifyCRM database.

Figure 4-25 Entering Request XML

The screenshot shows the BEA WebLogic console interface. The title bar reads "Test Service: CB_EXESUB". Below the title bar, there are tabs for "Application View Console" and "WebLogic Console", along with links for "Glossary" and "Logout". The main content area is titled "Test Service: CB_EXESUB on application view 'Clarify.CLARIFYCB'". It contains a text box with the following XML code:

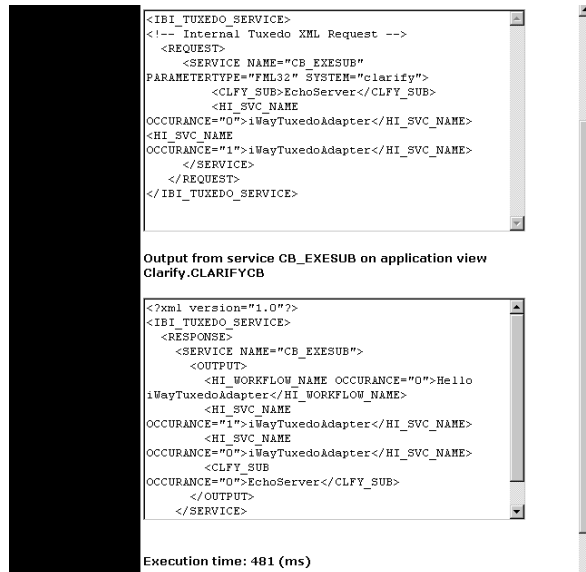
```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<IBI_TUXEDO_SERVICE>
  <!-- Internal Tuxedo XML Request -->
  <REQUEST>
    <SERVICE NAME="CB_EXESUB"
    PARAMETERTYPE="FML32" SYSTEM="clarify">
      <CLFY_SUB>EchoServer</CLFY_SUB>
      <HI_SVC_NAME
      OCCURANCE="0">iWayTuxedoAdapter</HI_SVC_NAME>
      <HI_SVC_NAME
      OCCURANCE="1">iWayTuxedoAdapter</HI_SVC_NAME>
    </SERVICE>
  </REQUEST>
</IBI_TUXEDO_SERVICE>
```

At the bottom of the text box is a "Test" button.

4. Click Test to test the service.

The request and response documents are then displayed.

Figure 4-26 Test Service Results



If the test fails, a response document displays the appropriate error messages. You can correct the error and resubmit the request.

If the test succeeds, the response document that corresponds to the request is displayed in the output field.

You have successfully deployed and tested the application view service.

If you wish, you can write custom code to create a workflow. For more information, see “Using Application Views in the Studio” in *Using Application Integration*:

- <http://edocs.bea.com/wli/docs70/aiuser/3usruse.htm>

Testing a Deployed Event

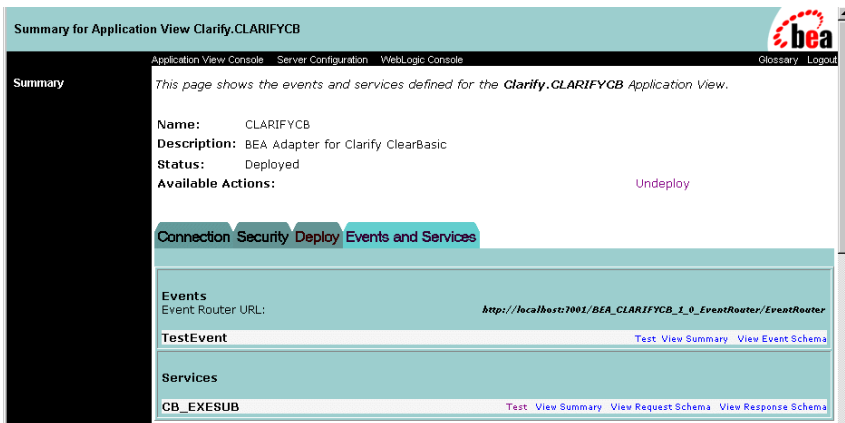
After you create and deploy an application view that contains an event, you can test the application view event to evaluate whether it interacts properly with the BEA WebLogic Adapter for ClarifyCRM.

To test a deployed event:

1. If you are not already in the Application View Console's Summary window for the application view event you want to test:
 - a. Log on to the WebLogic Application View Console as described in [“Creating an Application View”](#) on page 4-3.
 - b. Select the folder in which this application view resides, and then select the application view.

The Summary window opens.

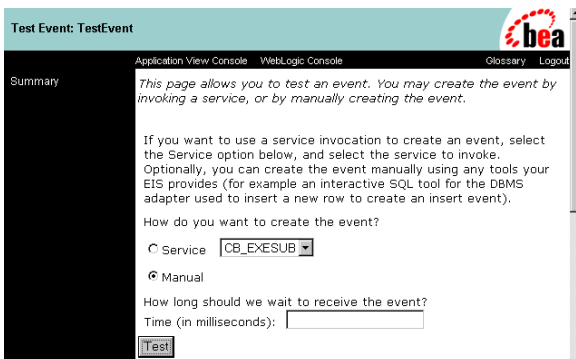
Figure 4-27 Summary Window for an Event to Be Tested



2. For the event you want to test, click Test.

The Test Event window opens.

Figure 4-28 Test Event Window



4 Creating Application Views for the ClearBasic Interface

3. Enter a time, in milliseconds, that indicates how long to wait to receive the event. Make sure to specify sufficient time to initiate the event.

Figure 4-29 Testing an Event

Test Event: TestEvent

Application View Console WebLogic Console Glossary Logout

Summary

This page allows you to test an event. You may create the event by invoking a service, or by manually creating the event.

If you want to use a service invocation to create an event, select the Service option below, and select the service to invoke. Optionally, you can create the event manually using any tools your EIS provides (for example an interactive SQL tool for the DBMS adapter used to insert a new row to create an insert event).

How do you want to create the event?

☐ Service **CB_EXESUB**

☒ Manual

How long should we wait to receive the event?

Time (in milliseconds):

Test

4. Click Test to test the event.

When the event occurs, the event test result document opens.

Figure 4-30 Event Test Results

Test Result for TestEvent

Application View Console WebLogic Console Glossary Logout

Summary

This page shows the results from testing a event.

Generated event of type TestEvent on application view Clarify.CLARIFYCB

```
<TESTEVENT>
<IWAYUTC_APP_ERR>IWAYUTC_APP_ERR_value</IWAYUTC_APP_ERR>
<IWAYUTC_ADAPTER_ERR>IWAYUTC_ADAPTER_ERR_value</IWAYUTC_ADAPTER_ERR>
<IWAYUTC_ADAPTER_ERR_CODE>IWAYUTC_ADAPTER_ERR_CODE_value</IWAYUTC_ADAPTER_ERR_CODE>
<HI_SVC_NAME>HI_SVC_NAME_value</HI_SVC_NAME>
<HI_PCK_EYE>HI_PCK_EYE_value</HI_PCK_EYE>
<HI_PCK_VER>HI_PCK_VER_value</HI_PCK_VER>
<HI_PCK_LEN>HI_PCK_LEN_value</HI_PCK_LEN>
<HI_PCK_SYS>HI_PCK_SYS_value</HI_PCK_SYS>
```

Execution time: 601 (ms)

If the test fails, a message is displayed indicating that the event timed out. You can correct the error and retest the event.

If the test succeeds, the event result document is displayed.

You have successfully deployed and tested the application view event.

If you wish, you can write custom code to create a workflow. For more information, see “Using Application Views in the Studio” in *Using Application Integration*:

<http://edocs.bea.com/wli/docs70/aiuser/3usruse.htm>

Using a Service or Event in a Workflow

This section contains examples of the use of a service and an event in a workflow. A workflow is the process that links an event in one Enterprise Integration System (EIS), such as a ClarifyCRM system, to the services in another EIS. You can build complex workflows that involve multiple events and services, transformations, and other business logic.

The procedure for using a service or event in a workflow is almost identical for the CBO and ClearBasic interfaces; the windows in the figures show data for the CBO interface. This section does not provide a comprehensive description of workflows. For more information, see “Using Application Views in the Studio” in *Using Application Integration*:

■ <http://edocs.bea.com/wli/docs70/aiuser/3usruse.htm>

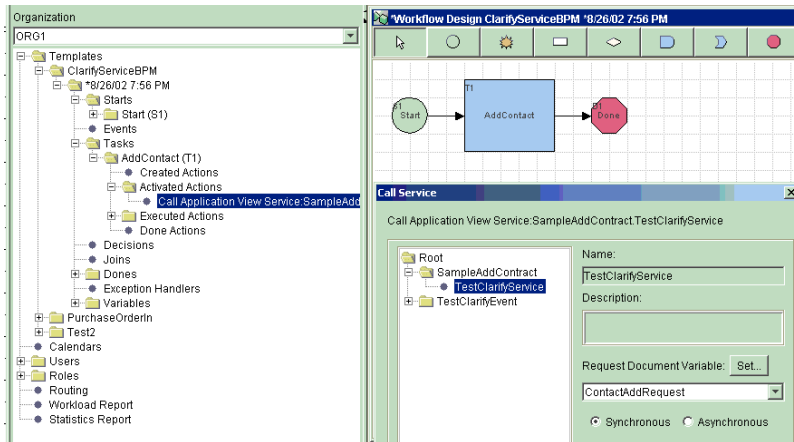
Example: Using a Service in a Workflow

In this example, a workflow is built using a static document called ContactAddRequest. This document is then used as input to the ClarifyCRM service defined in “[Adding a Service to an Application View](#)” on page 4-7. The resulting document is placed in the ContactResponse.

The following figures illustrate the service request in a workflow. In this workflow, the input document is defined as an action of the Start node. The AddContact action defines the use of the XML document passed to it from the Start node and sends it to the ClarifyCRM service.

4 Creating Application Views for the ClearBasic Interface

Figure 4-31 Application View Service in a Workflow



The ClarifyCRM Request

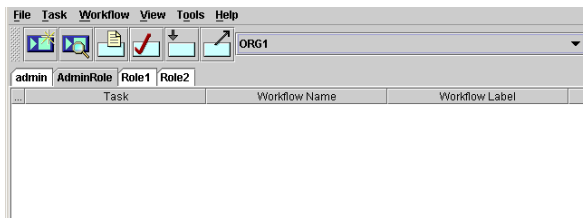
The following figure shows the static document created for the workflow. In this case, the document is a ContactRequest intended to be added into the ClarifyCRM application. In a real-world workflow, the document would typically be the result of an event in another EIS, with transformations and enrichment steps performed. For testing, however, this Service Request Template allows you to fill in test values.

Figure 4-32 Static Document

Start the Workflow

Once the workflow is built, Worklist is used to start the process manually. Worklist is required since the input document is static and not the result of an event from another EIS.

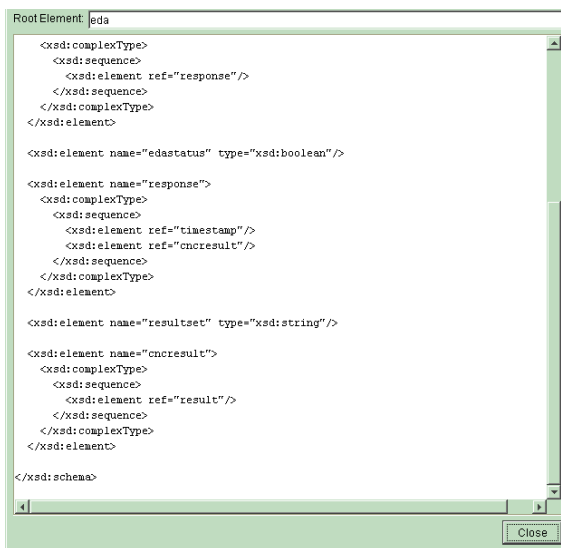
Figure 4-33 Starting the Workflow



The Clarify Response

When the workflow is started, the Task1 step takes the document and sends it to the ClarifyCRM application. Upon completion of the task, the workflow places the response XML document in the ContactAddResponse variable. By pressing the View Response Definition from the Call Service window, the XML Schema is displayed.

Figure 4-34 Response Document



5 Creating Schema Repositories

This section explains how to describe metadata for ClarifyCRM to WebLogic Integration. Normally, end-system metadata is used by the BAE to generate XML schemas that the adapter uses for events and services. Without this metadata, the Application Explorer can't generate these schemas for you. Because ClarifyCRM doesn't expose any of the metadata the BEA Application Explorer uses to generate the XML schemas, you must describe this metadata. After you have described the metadata, you can create and deploy application views using the WebLogic Application View Console.

This section includes the following topics:

- [Understanding Metadata](#)
- [Introducing Schemas and Repositories](#)
- [Naming a Schema Repository](#)
- [The Repository Manifest](#)

Understanding Metadata

When you define an application view, you are creating an XML-based interface between WebLogic Integration and ClarifyCRM or another enterprise information system (EIS) or application within your enterprise. The BEA WebLogic Adapter for

ClarifyCRM provides schemas for services and events that will handle any of the supported ClarifyCRM operations. These schemas must be specified in the `manifest.xml` file for the application view which uses the adapter.

Introducing Schemas and Repositories

You describe all the documents entering and exiting your WebLogic Integration system using W3C XML schemas. These schemas describe each event arriving to and propagating out of an event, and each request sent to and each response received from a service. There is one schema for each event and two for each service (one for the request, one for the response). The schemas are usually stored in files with an `.xsd` extension.

Use the WebLogic Integration Application View Console to access events and services, and to assign a schema to each event, request, and response. Assign each application view to a schema repository; you can assign several application views to the same repository.

BEA WebLogic adapters all make use of a schema repository to store their schema information and present it to the WebLogic Application View Console. The schema repository is a directory containing:

- A manifest file that describes the event and service schemas.
- The corresponding schema descriptions.

To work with schemas, you must know how to:

- Name a schema repository.
- Create a manifest.
- Create a schema.

Naming a Schema Repository

The schema repository has a three-part naming convention:

session_path\clarify\connection_name

Here,

- *session_path* is the schema's session base path (also known as the session base directory), beneath which you can store multiple sessions of schemas.
- *clarify* is the type of adapter.
- *connection_name* is a name representing a particular instance of the adapter.

For example, if the session path is `/usr/opt/bea/bse`, the adapter type is ClarifyCRM, and the connection name is TestClarify, then the schema repository is the directory:

`/usr/opt/bea/bse/ClarifyCRM/TestClarify`

The Repository Manifest

Each schema repository has a manifest that describes the repository and its schemas. This repository manifest is stored as an XML file named `manifest.xml`.

The manifest file relates documents (through their schemas) to services and events. The manifest exposes schema references to the event, relating the required document (via the root tag) to the corresponding schema. Schemas and manifests are stored in the same schema repository.

The following is an example of a manifest file. It illustrates the relationships between events and services and their related schemas:

Listing 5-1 Sample Manifest File

```
<manifest>
<connection/>
  <schemaref name="clarify">
    <request root="sp" file="bea_clarify_request.xsd" />
    <response root="response" file="bea_clarify_response.xsd" />
    <event root="ClarifyEvents" file="bea_clarify_events_row.xsd" />
  </schemaref>
</manifest>
```

The manifest has a schema reference section, named `schemaref`. The schema reference name is displayed in the schema drop-down list on the Add Service and Add Event windows in the WebLogic Application View Console.

This sample manifest has two schema references:

- One for events.

Events require only one schema, defined by the `event` tag. This relates the root tag of an XML document produced when a ClarifyCRM event occurs.

- One for services.

Services require two schemas: one for the document being passed to the service, represented by the `request` tag, and one for the expected response document received from the service operation, represented by the `response` tag.

You may also create a schema reference that comprises both services and events.

Creating a Repository Manifest

The repository manifest is an XML file with the root element `manifest` and two sub-elements:

- `connection`, which appears once, and which you can ignore because it is not used by the BEA WebLogic Adapter for ClarifyCRM at runtime, but only by the BEA Application Explorer when generating schemas from ClarifyCRM metadata.

Note: At the time this document was created, BEA Application Explorer did not support the dynamic creation of service and event schemas for ClarifyCRM. Check with BEA Systems for the latest BEA Application Explorer adapter support.

- `schemaref`, which appears multiple times, once for each schema name, and which contains all three schemas—request, response, and event.

To create a manifest:

1. Create an XML file with the following structure:

```
<manifest>
  <connection>
  </connection>
</manifest>
```

2. For each new event or service schema you define, create a `schemaref` section using this model:

```
<schemaref name="Product">
  <request root="sp" file="bea_clarify_request.xsd"/>
  <response root="response" file="bea_clarify_response.xsd"/>
  <event root="clarify" file="bea_clarify_event_row.xsd"/>
</schemaref>
```

Here, the value you assign to:

- `file` is the name of the file in the schema repository.
- `root` is the name of the root element in the actual instance documents that will arrive at, or be sent to, the event or service.

The following is an example of an instance document for the Product event referred to in [“Creating a Repository Manifest” on page 5-4](#).

Listing 5-2 Request Document for ClarifyCRM Contact_Insert Service

```
<?xml version="1.0"?>
  <sp>
    <data>
      <cbo name="contact" action="create">
        <keys>
          <key name="first_name">Bill</key>
          <key name="last_name">Wier</key>
          <key name="phone">408-777-9000</key>
          <keyRel name="contact2contact_role">2030120</keyRel>
```

```
<fields>
  <field name="address_1">235 darwin drive</field>
  <field name="address_2">APT 235</field>
  <field name="City">fremont</field>
  <field name="e_mail">bwier@bww.com</field>
  <field name="fax_number">408-999-9898</field>
  <field name="mail_stop">M123</field>
  <field name="salutation">Mr.</field>
  <field name="status">0</field>
  <field name="title">CEO</field>
</fields>
<Relations>
  <Rel name="contact2contact_role">26589412</Rel>
</Relations>
</keys>
</cbo>
</data>
</sp>
```

The following is a schema matching this instance document; it may be manually coded or generated from any XML editor:

Listing 5-3 Schema Matching Contact_Add Service Request Document

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  elementFormDefault="qualified">
  <xsd:complexType name="cboType">
    <xsd:sequence>
      <xsd:element name="keys" type="keysType"
minOccurs="0"/>
      <xsd:element name="fields" type="fieldsType"
minOccurs="0"/>
    </xsd:sequence>
    <xsd:attribute name="name" type="xsd:string"
use="required"/>
    <xsd:attribute name="action" type="xsd:string"/>
    <xsd:attribute name="relation" type="xsd:string"/>
  </xsd:complexType>
  <xsd:complexType name="dataType">
    <xsd:sequence>
      <xsd:element name="cbo" type="cboType"/>
    </xsd:sequence>
```

```
</xsd:complexType>
<xsd:element name="sp" type="spType"/>
<xsd:complexType name="fieldType">
  <xsd:simpleContent>
    <xsd:extension base="xsd:string">
      <xsd:attribute name="name"
type="xsd:string" use="required"/>
      <xsd:attribute name="data_type"
type="xsd:string"/>
    </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>
<xsd:complexType name="fieldsType">
  <xsd:sequence>
    <xsd:element name="cbo" type="cboType"
minOccurs="0" maxOccurs="unbounded"/>
    <xsd:element name="field" type="fieldType"
minOccurs="0" maxOccurs="unbounded"/>
    <xsd:element name="cbo" type="cboType"
minOccurs="0" maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="keysType">
  <xsd:sequence>
    <xsd:element name="field" type="fieldType"/>
  </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="spType">
  <xsd:sequence>
    <xsd:element name="data" type="dataType"/>
  </xsd:sequence>
</xsd:complexType>
</xsd:schema>
```

Sample Files

Supplied with the BEA WebLogic Adapter for ClarifyCRM are XSD schema files for service requests and responses and for events. With these schemas, all of the ClarifyCRM methods and objects that the adapter supports can be used in application views. These files are supplied in a zip file called `bea_clarify_samples.zip`. These schemas can be used in the `manifest.xml` files you build.

For services, use the schemas as defined in the table that follows.

Table 5-1 Service Schemas

| For . . . | Use the following schema . . . |
|-------------------|---------------------------------------|
| Service requests | <code>bea_clarify_request.xsd</code> |
| Service responses | <code>bea_clarify_response.xsd</code> |

For events, use one of three files in the `manifest.xml` file, depending on the format you select when you build the event, as defined in the table that follows.

Table 5-2 Event Schemas

| Format | Use the following schema . . . |
|--------|------------------------------------------|
| Row | <code>bea_clarify_event_row.xsd</code> |
| Column | <code>bea_clarify_event_col.xsd</code> |
| Field | <code>bea_clarify_event_field.xsd</code> |

The `bea_clarify_samples.zip` file also includes sample request files that work with the above-mentioned schemas. These instances contain the object and operation inside ClarifyCRM to be executed, in addition to sample data.

6 Using Tracing

Tracing is an essential feature of an adapter. Most adapters integrate different applications and do not interact with end users while processing data. Unlike a front-end component, when an adapter encounters an error or a warning condition, the adapter cannot stop processing and wait for an end user to respond.

Moreover, many business applications that are connected by adapters are mission-critical. For example, an adapter might maintain an audit report of every transaction with an EIS. Consequently, adapter components must provide both accurate logging and auditing information. The adapter tracing and logging framework is designed to accommodate both logging and auditing.

This section describes tracing for services and events. It contains the following topics:

- [Levels and Categories of Tracing](#)
- [Tracing and Performance](#)
- [Creating Traces for Services and Events](#)

Levels and Categories of Tracing

Tracing is provided by both the BEA adapter framework and by the BEA WebLogic Adapter for ClarifyCRM. The BEA WebLogic Integration framework provides five distinct levels of tracing:

Table 6-1 Tracing Levels

| Level | Indicates |
|-------|-----------------------------------------------------------------------------------------------------------------------------------------------------------|
| AUDIT | An extremely important log message related to the business processing performed by an adapter. Messages with this priority are always written to the log. |
| ERROR | An error in the adapter. Error messages are internationalized and localized for the user. |
| WARN | A situation that is not an error, but that could cause problems in the adapter. Warning messages are internationalized and localized for the user. |
| INFO | An informational message that is internationalized and localized for the user. |
| DEBUG | A debug message, that is, information used to determine how the internals of a component are working. Debug messages usually are not internationalized. |

The adapter framework provides three specialized categories of tracing:

Table 6-2 Tracing Categories

| Level | Indicates |
|---------------|-------------------------------------------------------------------------------------------------------------------------------------------|
| Basic Trace | Basic traces. Displays the input XML (up to 300 bytes) before parsing, and shows the request being processed. The default setting is off. |
| Verbose Trace | More extensive traces. Displays configuration parameters used by the adapter. The default setting is off. |

Table 6-2 Tracing Categories

| Level | Indicates |
|----------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Document Trace | Displays the input document after it was analyzed and the response document being returned. Because some documents are very large, this trace category can severely affect performance and memory use. The default setting is off. |

Note: To obtain the appropriate trace, both the level and the category must be declared. In a debug situation, BEA Customer Support will request (minimally) a Basic and a Verbose trace.

Tracing and Performance

The additional trace capabilities provided by the adapter are not strictly hierarchic; rather they are categorized. These traces are designed to provide debugging help with minimum effect on performance. All internal adapter traces are controlled through the additional tracing settings, and all additional settings route their output to the standard debug setting.

If you configure the adapter for additional settings and do not configure standard trace settings, the traces are generated but never appear in output. This affects performance, as the production of the trace continues even though you receive no benefit of the additional trace information.

Creating Traces for Services and Events

The following topics discuss the steps required to create traces to diagnose adapter problems.

Creating Traces for a Service

To create traces for a service:

1. Create or modify the service.
2. Ensure that all of the adapter parameters are entered correctly.

Figure 6-1 Add Service window

3. Select the appropriate schema from the drop-down list.
4. Select the appropriate trace levels as described in [Table 6-2](#): Trace, Verbose trace, and Document trace.
5. Click Add to continue to the next configuration pane.

- Click Continue to move to the next configuration pane.

The Deploy Application View window opens.

- Navigate to the Log Configuration area and select the desired trace level.

This pane enables you to select the trace level for the BEA WebLogic Integration framework.

Figure 6-2 Deploy Application View window

Deploy Application View clarify to Server

Application View Console WebLogic Console Glossary Logout

Configure Connection
Administration
Add Service
Add Event
Deploy Application View

On this page you deploy your application view to the application server.

Required Service Parameters

Enable asynchronous service invocation? ☒

Connection Pool Parameters

Use these parameters to configure the connection pool used by this application view

Minimum Pool Size*

Maximum Pool Size*

Target Fraction of Maximum Pool Size*

Allow Pool to Shrink? ☒

Log Configuration

Set the log verbosity level for this application view.

Log all messages
Log errors and audit messages
Log warnings, errors, and audit messages
Log informationals, warnings, errors, and audit messages
Log all messages

Deploy ☒ Deploy persistently? ☐ Save

For maximum tracing, select Log all Messages.

This is recommended to obtain optimum debugging information for BEA support personnel.

Note: This causes all generated messages to be written to the log. You must select the desired category as defined in [Table 6-2](#) in the adapter to generate the required messages.

- Click Deploy (or Save) to set the trace settings and deploy the application view.

Traces are created the next time the service is invoked.

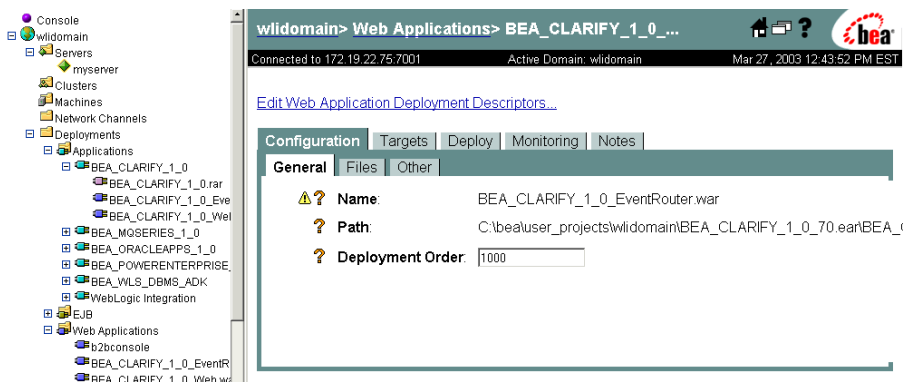
Traces are output to a file named BEA_CLARIFY_1_0.log in the WebLogic Domain home directory.

Creating or Modifying the Tracing Level for an Event

To create or modify the tracing level for an event:

1. Logon to the BEA WebLogic Server Console.

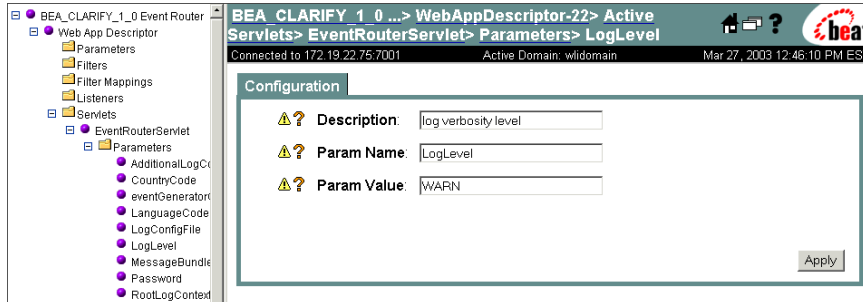
Figure 6-3 WebLogic Server Console



2. Select Web Applications.
3. Select BEA_CLARIFY_1_0_EventRouter.war.
4. Click Edit Web Application Deployment Descriptors.
5. When the following window opens, select Servlets.
6. In the folder below Servlets, select EventRouterServlet.
7. Select Parameters.

8. Select LogLevel.

Figure 6-4 WebLogic Server Console: Configuration



This pane enables you to select the trace level for the BEA WebLogic Integration framework.

For maximum tracing, enter DEBUG. This is recommended to obtain optimum debugging information for BEA support personnel

The following levels are valid:

Table 6-3 Trace Levels

| Level | Indicates |
|-------|-----------------------------------------------------------------------------------------------------------------------------------------------------------|
| AUDIT | An extremely important log message related to the business processing performed by an adapter. Messages with this priority are always written to the log. |
| ERROR | An error in the adapter. Error messages are internationalized and localized for the user. |
| WARN | A situation that is not an error, but that could cause problems in the adapter. Warning messages are internationalized and localized for the user. |
| INFO | An informational message that is internationalized and localized for the user. |
| DEBUG | A debug message, that is, information used to determine how the internals of a component are working. Debug messages usually are not internationalized. |

9. Click Apply to save the newly entered trace level.

10. Click BEA_Clarify_1_0 EventRouter.
11. Click Persist to apply the logging changes.
This change need only be made once.
It is set for all events associated with a given adapter.
12. Return to the WebLogic Server Console.
13. Select Applications from the WebLogic Server Console.
14. Select the adapter whose EventRouter you have modified in the previous steps.
15. Select the Deploy tab in the right pane.

The right pane displays the following adapter components:

- BEA_CLARIFY_1_0.rar
- BEA_CLARIFY_1_0.web.rar
- BEA_CLARIFY_1_0_EventRouter.war.

Figure 6-5 WebLogic Server Console: Redeploy

Deployment Status by Target:

| Component | Component Type | Target | Target Type | Deployed | |
|-------------------------------------------------|---------------------|--------------------------|-------------|----------|---------------------------------------------------|
| BEA_CLARIFY_1_0_Web.war | Web Application | myserver | Server | true | Undeploy Redeploy |
| BEA_CLARIFY_1_0_EventRouter.war | Web Application | myserver | Server | true | Undeploy Redeploy |
| BEA_CLARIFY_1_0.rar | Connector Component | myserver | Server | true | Undeploy Redeploy |

[Undeploy Application](#) Undeploy the entire application
[Deploy Application](#) Deploy or redeploy the entire application

Deployment Activity:

| Description | Status | BeginTime | EndTime |
|-------------|--------|-----------|---------|
|-------------|--------|-----------|---------|

16. Redeploy the EventRouter by clicking the Redeploy button to the right of BEA_CLARIFY_1_0_EventRouter.war.

Creating Adapter Logs for an Event

To create adapter logs for an event:

1. Create or modify the event.
2. Ensure that all of the adapter parameters are entered correctly.

Figure 6-6 Add Event window

Add Event

Application View Console WebLogic Console Glossary Logout

On this page, you add events to your application view.

Unique Event Name: *

Clarify

Character Set Encoding* UTF-8

Driver* oracle.jdbc.driver.OracleDriver

url* jdbc:Oracle:thin@{host:port}/{database}

User Name

Password

Format* field

Maximum Rows 1

SQL Query* select * from {db name}

SQL Post Query

Delete Keys

Polling Interval 20

Data Source Name

schema: BEA_clarify_event_row

settings

Trace on/off ☒

Verbose Trace on/off ☒

Document Trace on/off ☒

Add

3. Select the appropriate schema from the drop-down list.
4. Select the appropriate trace levels as described in [Table 6-2](#): Trace, Verbose trace, and Document trace.
5. Click Add to continue to the next configuration pane.
6. Click Continue to move to the next configuration pane.

The Deploy Application View window opens.

7. Navigate to the Log Configuration area and select the desired trace level.

This pane enables you to select the trace level for the BEA WebLogic Integration framework.

Figure 6-7 Deploy Application View window

For maximum tracing, select Log all Messages. This is recommended to obtain optimum debugging information for BEA support personnel.

8. Click Deploy (or Save) to set the trace settings and deploy the application view.

Traces are created the next time the event occurs.

Traces are output to a file named BEA_CLARIFY_1_0.log in the WebLogic Domain home directory.

A Customizing Tuxedo Field Tables

This appendix applies to the ClearBasic interface of ClarifyCRM only.

Field tables are required to describe all the fields that the BEA WebLogic Adapter for ClarifyCRM will handle in service requests and responses. This section describes how to customize Tuxedo field tables, and includes the following topics:

- [Working With Field Tables](#)
- [Creating a WTC FML32 Field Table](#)
- [Creating a Tuxedo Field Table Header File](#)
- [Using Your Field Table in WebLogic Tuxedo Connector](#)
- [Using Your Field Table in Tuxedo](#)

Working With Field Tables

Field Manipulation Language (FML) field tables and their field definition files describe the fields available in any Tuxedo typed buffer exchanged between a service requestor and service responder. All fields to be exchanged between ClarifyCRM and WebLogic Integration, either as an inbound parameter, or as an outbound response field, must be previously defined in the field definition file. The field definition file defines the field identifier, the field name, and the field type to the Tuxedo system. The BEA WebLogic Adapter for ClarifyCRM ships with a field definition file, but you will want to customize the field definition file for your purposes.

The field definition files need to match on both the Tuxedo and WebLogic Tuxedo Connector (WTC) side of the domain communication channel. The field definitions need not be identical, but the fields used between the two systems must exist in both field definition files and match in ID and type. The field definition file is used at design and configuration time and is converted into an FML field table class for execution in the WTC.

The format of the field definition file is the same on both the Tuxedo and WTC platforms. There are different utilities to use the FML field table from the source field definition files. On the Tuxedo platform, the command line utility `mkfldhdr` can be used to create the header file for a user's C-based ATMI application. On the WTC platform, the `mkfieldclass32` utility produces Java source for the field table that must be compiled for use on WebLogic WTC.

Creating a WTC FML32 Field Table

By default, your adapter makes use of a field table named `_SrvFieldsTable`. This will be the name used for the field definition file and the associated `.java` and `.class` files. To produce a field class source Java file from your field definition file, run the following Java program:

```
java weblogic/wtc/jatmi/mkfldclass32 _SrvFieldsTable
```

The output from this program will be a Java definition of the field class. This Java file must be compiled using the `javac` compile command in the Java Development Kit (JDK). For example:

```
javac _SrvFieldsTable.java
```

The class path must include the `weblogic.jar` library.

Creating a Tuxedo Field Table Header File

To produce a Tuxedo FML field table header file from a field definition file, run the following command line utility:

```
mkfldhdr [-d targetDirectory] [field_table ...]
```

The resulting field table header file can be found in the current directory, or (if you include the `-d` parameter) in the specified target directory.

Here:

- *targetDirectory* specifies that the output header files are to be created in a directory other than the present working directory.
- *field_table* is a field table name.

If you omit field table names, the utility uses the FIELDTBLS environment variable as the list of field tables to be converted, and FLDTBLDIR environment variable as a list of directories to be searched for the files.

FIELDTBLS specifies a comma-separated list of field table filenames. If FIELDTBLS has no value, fld.tbl is used as the name of the (only) field table file (in this case, the resulting header file will be fld.tbl.h). The FLDTBLDIR environment variable is a colon-separated list of directories in which to look for each field table whose name is not an absolute path name; the search for field tables is very similar to the search for executable commands using the UNIX System PATH variable. If FLDTBLDIR is not defined, only the current directory is searched. Thus, if no field table names are specified on the command line and FIELDTBLS and FLDTBLDIR are not set, mkfldhdr will convert the field table fld.tbl in the current directory into the header file fld.tbl.h.

Using Your Field Table in WebLogic Tuxedo Connector

Once your field class is compiled, you may override the `_SrvFieldsTable.class` file included in the supplied BEA WebLogic Adapter for ClarifyCRM Web application (.ear file) by moving or copying the class file into the WEB-INF/classes directory of your WebLogic domain. For example, placing the `_SrvFieldsTable.class` file in the directory:

```
/usr/local/bean/user_projects/yourdomain/applications/DefaultWebApp  
p_myserver/WEB-INF/classes
```

For more information on using field tables in your WebLogic Tuxedo Connector (WTC) environment, see the *WebLogic Tuxedo Connector Programmer's Guide*.

Using Your Field Table in Tuxedo

In your Tuxedo environment, the following variables are used throughout Field Manipulation Language (FML) to access field table files:

- **FIELDTBLS**, which should contain a comma-separated list of field table files for the application. Files given as full path names are used as is; files listed as relative path names are searched for through the list of directories specified by the **FLDTBLDIR** variable. **FIELDTBLS32** is used for **FML32**. If **FIELDTBLS** is not set, then the single file name **fld.tbl** is used. (**FLDTBLDIR** still applies.)
- **FLDTBLDIR**, which specifies a colon-separated list of directories to be used to find field table files with relative filenames. Its usage is similar to the **PATH** environment variable. If **FLDTBLDIR** is not set or is null, then its value is taken to be the current directory. **FLDTBLDIR32** is used for **FML32**.

For more information, see the *Programming a BEA Tuxedo Application Using FML* manual.